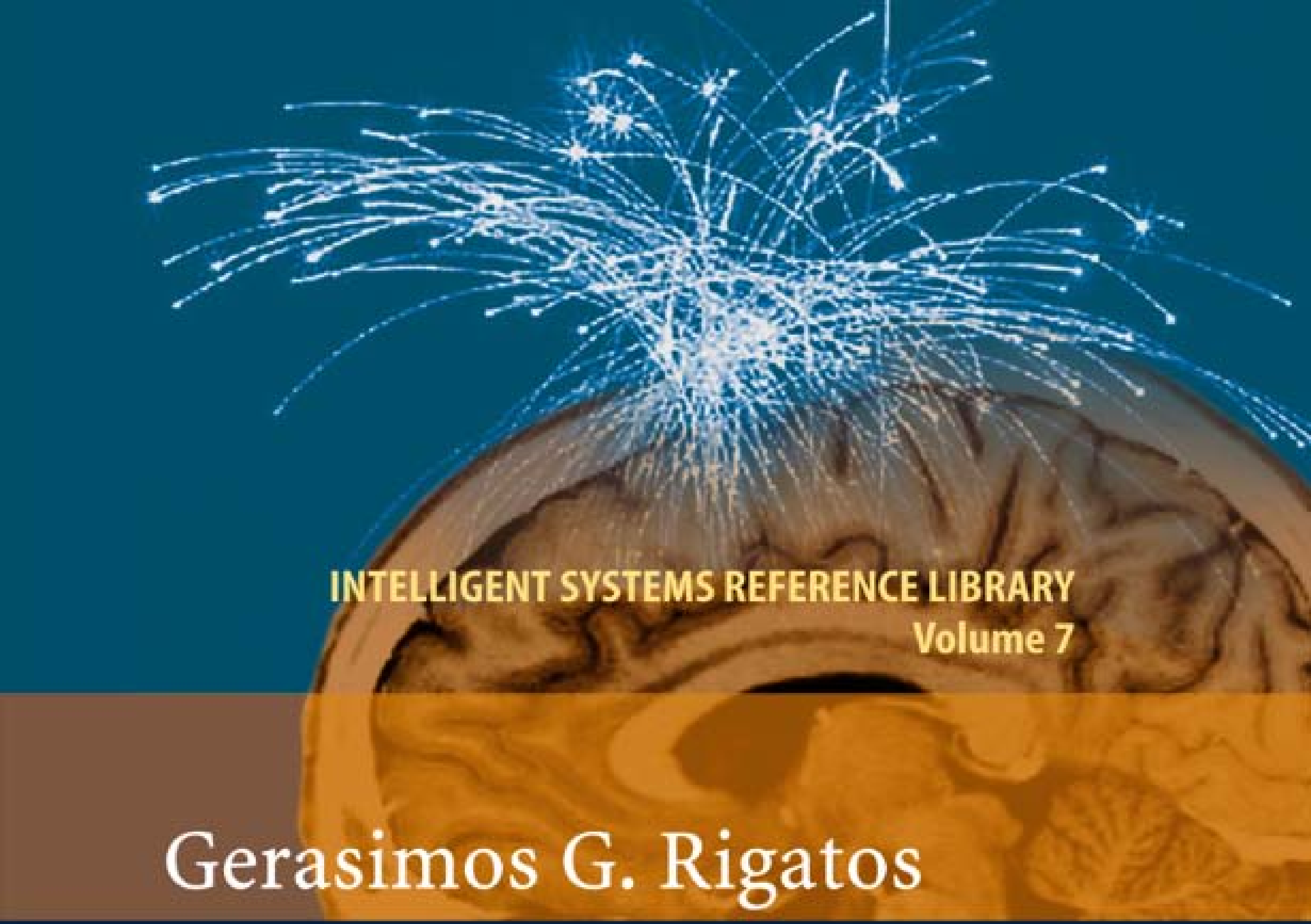


INTELLIGENT SYSTEMS REFERENCE LIBRARY
Volume 7

Gerasimos G. Rigatos

Modelling and Control for Intelligent Industrial Systems

Adaptive Algorithms in Robotics and Industrial Engineering

Springer

Gerasimos G. Rigatos

Modelling and Control for Intelligent Industrial Systems

Intelligent Systems Reference Library, Volume 7

Editors-in-Chief

Prof. Janusz Kacprzyk
Systems Research Institute
Polish Academy of Sciences
ul. Newelska 6
01-447 Warsaw
Poland
E-mail: kacprzyk@ibspan.waw.pl

Prof. Lakhmi C. Jain
University of South Australia
Adelaide
Mawson Lakes Campus
South Australia 5095
Australia
E-mail: Lakhmi.jain@unisa.edu.au

Gerasimos G. Rigatos

Modelling and Control for Intelligent Industrial Systems

Adaptive Algorithms in Robotics and Industrial Engineering

Springer

Gerasimos G. Rigatos
Unit of Industrial Automation,
Industrial Systems Institute,
Rion Patras,
Greece 26504
E-mail: ger9711@ath.forthnet.gr

ISBN 978-3-642-17874-0 e-ISBN 978-3-642-17875-7

DOI 10.1007/978-3-642-17875-7

Intelligent Systems Reference Library ISSN 1868-4394

Typeset & *Cover Design:* Scientific Publishing Services Pvt. Ltd., Chennai, India.

Printed on acid-free paper

9 8 7 6 5 4 3 2 1

springer.com

To Elektra

Foreword

There are two main requirements for the development of intelligent industrial systems: (i) learning and adaptation in unknown environments, (ii) compensation of model uncertainties as well as of unknown or stochastic external disturbances. Learning can be performed with the use of gradient-type algorithms (also applied to nonlinear modeling techniques) or with the use of derivative-free stochastic algorithms. The compensation of uncertainties in the model's parameters as well as of external disturbances can be performed through stochastic estimation algorithms (usually applied to filtering and identification problems), and through the design of adaptive and robust control schemes. The book aims at providing a thorough analysis of the aforementioned issues.

Dr. Gerasimos G. Rigatos
Senior Researcher
Unit of Industrial Automation
Industrial Systems Institute
Greece

Preface

Incorporating intelligence in industrial systems can help to increase productivity, cut-off production costs, and to improve working conditions and safety in industrial environments. This need has resulted in the rapid development of modeling and control methods for industrial systems and robots, of fault detection and isolation methods for the prevention of critical situations in industrial work-cells and production plants, of optimization methods aiming at a more profitable functioning of industrial installations and robotic devices and of machine intelligence methods aiming at reducing human intervention in industrial systems operation.

To this end, the book defines and analyzes some main directions of research in modeling and control for industrial systems. These are: (i) industrial robots, (ii) mobile robots and autonomous vehicles, (iii) adaptive and robust control of electromechanical systems, (iv) filtering and stochastic estimation for multi-sensor fusion and sensorless control of industrial systems (iv) fault detection and isolation in robotic and industrial systems, (v) optimization in industrial automation and robotic systems design, (vi) machine intelligence for robots autonomy, and (vii) vision-based industrial systems.

In the area of industrial robots one can distinguish between two main problems: (i) robots operating in a free working space, as in the case of robotic welding, painting, or laser and plasma cutting and (ii) robots performing compliance tasks, as in the case of assembling, finishing of metal surfaces and polishing. When the robotic manipulator operates in a free environment then kinematic and dynamic analysis provide the means for designing a control law that will move appropriately the robot's end effector and will enable the completion of the scheduled tasks. In the case of compliance tasks, the objective is not only to control the end effector's position but also to regulate the force developed due to contact with the processed surface. There are established approaches for simultaneous position and force control of robotic manipulators which were initially designed for rigid-link robots and which were subsequently extended to flexible-link robots.

In the area of mobile robots and autonomous vehicles one has to handle nonholonomic constraints and to avoid potential singularities in the design of the control law. Again the kinematic and dynamic model of the mobile robots provide the basis for deriving a control law that will enable tracking of desirable trajectories. Several applications can be noted such as path tracking by autonomous mobile robots and automatic ground vehicles (AGVs), trajectory tracking and dynamic positioning of surface and underwater vessels and flight control of unmanned aerial vehicles (UAVs). Apart from controller's design, path planning and motion planning are among the problems the robotics/industrial systems engineer have to solve. These problems become particularly complicated when the mobile robot operates in an unknown environment with moving obstacles and stochastic uncertainties in the measurements provided by its sensors.

In the area of adaptive control for electromechanical systems it is necessary to design controllers for the non-ideal but more realistic case in which the system dynamics is not completely known and the system's state vector is not completely measurable. Thus, one has finally to consider the problem of joint nonlinear estimation and control for dynamical systems. Most nonlinear control schemes are based on the assumptions that the state vector of the system is completely measurable and that the system's dynamical model is known (or at least there are known bounds of parametric uncertainties and external disturbances). However, in several cases measurement of the complete state vector is infeasible due to technical difficulties or due to high cost. Additionally, knowledge about the structure of the system's dynamical model and the values of its parameters can be only locally valid, therefore model-based control techniques may prove to be inadequate. To handle these cases control schemes can be implemented through the design of adaptive observers, and adaptive controllers where the state vector is reconstructed by processing output measurements with the use of a state observer or filter.

In the area of robust control for electromechanical systems one has to consider controllers capable of maintaining the desirable performance of the industrial or robotic system despite unmodeled dynamics and external disturbances. The design of such controllers can take place either in the time domain, as in the case of sliding mode control or H-infinity control, or in the frequency domain as in the case of robust control based on Kharitonov's theory. In the latter case one can provide the industrial system with the desirable robustness using a low-order controller and only feedback of the system's output. Whilst sliding-mode and H-infinity robust control can be particularly useful for robotic and motion transmission systems, Kharitonov's theory can provide reliable and easy to implement robust controllers for the electric power transmission and distribution system.

In the area of filtering and stochastic estimation one can see several applications to autonomous robots and to the development of control systems over communication networks. The need for robots capable of operating autonomously in unknown environments imposes the use of nonlinear estimation for reconstructing missing information and for providing the robots control loop with robustness to uncertain of ambiguous information. Additionally, the development of control systems over communication networks requires the application of nonlinear filtering for fusing distributed sensor measurements so as to obtain a global and fault-free estimate of the state of large-scale and spatially distributed systems. Filtering and estimation methods for industrial systems comprise nonlinear state observers, Kalman filtering approaches for nonlinear systems and its variants (Extended Kalman Filter, Sigma-Point Kalman Filters, etc.), and nonparametric estimators such as Particle Filters. Of primary importance is sensor-fusion based control for industrial systems, with particular applications to industrial robotic manipulators, as well as to mobile robots and autonomous vehicles (land vehicles, surface and underwater vessels or unmanned aerial vehicles). Moreover, the need for distributed filtering and estimation for industrial systems becomes apparent for networked control systems as well as for the autonomous navigation of unmanned vehicles.

In the area of fault detection and isolation one can note several examples of faults taking place in robotic and industrial systems. Robotic systems components, such as sensors, actuators, joints and motors, undergo changes with time due to prolonged functioning or a harsh operating environment and their performance may degrade to an unacceptable level. Moreover, in electric power systems, there is need for early diagnosis of cascading events, which finally lead to the collapse of the electricity network. The need for a systematic method that will permit preventive maintenance through the diagnosis of incipient faults is obvious. At the same time it is desirable to reduce the false alarms rate so as to avoid unnecessary and costly interruptions of industrial processes and robotic tasks. In the design of fault diagnosis tools the industrial systems engineer comes against two problems: (i) development of accurate models of the system in the fault-free condition, through system identification methods and filtering/ stochastic estimation methods (ii) optimal selection of the fault threshold so as to detect slight changes of the system's condition and at the same time to avoid false alarms. Additionally one can consider the problems of fault diagnosis in the frequency domain and fault diagnosis with parity equations and pattern recognition methods.

In the area of optimization for industrial and robotic systems one can find several applications of nonlinear programming-based optimization as well as of evolutionary optimization. There has been extensive research on nonlinear programming methods, such as gradient methods, while their convergence to optimum has been established through stochastic approximations theory. Robotics is a promising application field for nonlinear programming-based optimization, e.g. for problems of motion planning and adaptation to unknown environments, target tracking and collective behavior of multi-robot systems. On the other-hand evolutionary algorithms are very efficient for performing global optimization in cases that real-time constraints are not restrictive, e.g. in several production planning and resource management problems. Industrial and robotic systems engineers have to be well acquainted with optimization methods, so as to design industrial systems that will excel in performance metrics and at the same time will operate at minimum cost.

In the area of machine intelligence for robots autonomy one can note several applications both in control and in fault diagnosis tasks. Machine intelligence methods are particularly useful when analytical models of the robotic system are hard to obtain due to inherent complexity or due to infinite dimensionality of the robot's model. In such cases it is preferable to develop a model-free controller of the robotic system, exploiting machine learning tools (e.g. neural and wavelet networks, fuzzy models or automata models) instead of pursuing the design of a model-based controller through analytical techniques. Additionally, to perform fault diagnosis in robotic and industrial systems with event-driven dynamics it is recommended again to apply machine intelligence tools such as automata, while to handle the uncertainty associated with such systems probabilistic or possibilistic state machines can be used as fault diagnosers.

In the area of vision-based industrial systems one can note robotic visual servoing as an application where machine vision provides the necessary information for the functioning of the associated control loop. Visual servoing-based robotic systems are rapidly expanding due to the increase in computer processing power and low prices of cameras, image grabbers, CPUs and computer memory. In order to satisfy strict accuracy constraints imposed by demanding manufacturing specifications, visual servoing systems must be fault tolerant. This means that in the presence of temporary of permanent failures of the robotic system components, the system must continue to provide valid control outputs which will allow the robot to complete its assigned tasks. Nowadays, visual servoing-based robotic manipulators have been used in several industrial automation tasks, e.g. in the automotive industry, in warehouse management, or in vision-based navigation of autonomous vehicles. Moreover, visual servoing over networks of cameras can provide the robot's control loop with robust state estimation in case that visual measurements are occluded by noise sources, as it usually happens in harsh industrial environments (e.g. in robot welding and cutting applications).

It is noted that several existing publications in the areas of robotic and industrial systems focus exclusively on control problems. In some cases, issues which are significant for the successful operation of industrial systems, such as modelling and state estimation, sensorless control, or optimization, fault diagnosis, machine intelligence for robots autonomy, and vision-based industrial systems operation are omitted. Thus engineers and researchers have to address to different sources to obtain this information and this fragmentation of knowledge leads to an incomplete presentation of this research field. Unlike many books that treat separately each one of the previous topics, this book follows an interdisciplinary approach in the design of intelligent industrial systems and uses in a complementary way results and methods from the above research fields. The book is organized in 16 chapters:

In Chapter 1, a study of industrial robotic systems is provided, for the case of contact-free operation. This part of the book includes the dynamic and kinematic analysis of rigid-link robotic manipulators, and advances to more specialized topics, such as dynamic and kinematic analysis of flexible-link robots, and control of rigid-link and flexible-link robots in contact-free operation.

In Chapter 2, an analysis of industrial robot control is given, for the case of compliance tasks. First, rigid-link robotic models are considered and the impedance control and hybrid position-force control methods are analyzed. Next, force control methods are generalized in the case of flexible-link robots performing compliance tasks.

In Chapter 3, an analysis of the kinematic model of autonomous land vehicles is given and nonlinear control for this type of vehicles is analyzed. Moreover, the kinematic and dynamic model of surface vessels is studied and nonlinear control for the dynamic ship positioning problem is also analyzed.

In Chapter 4, a method for the design of stable adaptive control schemes for a class of industrial systems is first studied. The considered adaptive controllers can be based either on feedback of the complete state vector or on feedback of the system's output. In the latter case the objective is to succeed simultaneous estimation of the system's state vector and identification of the unknown system dynamics. Lyapunov analysis provides necessary and sufficient conditions in the controller's design that assure the stability of the control loop. Examples of adaptive control applications to industrial systems are presented.

In Chapter 5, robust control approaches for industrial systems are studied. Such methods are based on sliding-mode control theory where the controller's design is performed in the time domain and Kharitonov's stability theory where the controller's design is performed in the frequency domain. Applications to the problem of robust electric power system stabilization are given.

In Chapter 6, filtering and stochastic estimation methods are proposed for the control of linear and nonlinear dynamical systems. Starting from the theory of linear state observers the chapter proceeds to the standard Kalman filter and its generalization to the nonlinear case which is the Extended Kalman Filter. Additionally, Sigma-Point Kalman Filters are proposed as an improved nonlinear state estimation approach. Finally, to circumvent the restrictive assumption of Gaussian noise used in Kalman Filtering and its variants, the Particle Filter is proposed. Applications of filtering and estimation methods to industrial systems control when using a reduced number of sensors are presented.

In Chapter 7, sensor fusion with the use of filtering methods is studied and state estimation of nonlinear systems based on the fusion of measurements from distributed sources is proposed for the implementation of stochastic control loops for industrial systems. The Extended Kalman and Particle Filtering are first proposed for estimating, through multi-sensor fusion, the state vector of an industrial robotic manipulator and the state vector of a mobile robot. Moreover, sensor fusion with the use of Kalman and Particle Filtering is proposed for the reconstruction from output measurements the state vector of a ship which performs dynamic positioning.

In Chapter 8, distributed filtering and estimation methods for industrial systems are studied. Such methods are particularly useful in case that measurements about the industrial system are collected and processed by different monitoring stations. The overall concept is that at each monitoring station a filter tracks the state of the system by fusing measurements which are provided by various sensors, while by fusing the state estimates from the distributed local filters an aggregate state estimate for the industrial system is obtained. In particular, the chapter proposes first the Extended Information Filter (EIF) and the Unscented Information Filter (UIF) as possible approaches for fusing the state estimates provided by the local monitoring stations, under the assumption of Gaussian noises. The EIF and UIF estimated state vector can, in turn, be used by nonlinear controllers which can make the system's state vector track desirable setpoints. Moreover, the Distributed Particle Filter (DPF) is proposed for fusing the state estimates provided by the local monitoring stations (local filters). The motivation for using DPF is that it is well-suited to accommodate non-Gaussian measurements. The DPF estimated state vector is again used by nonlinear controller to make the system converge to desirable setpoints. The performance of the Extended Information Filter, of the Unscented Information Filter and of the Distributed Particle Filter is evaluated through simulation experiments in the case of a 2-UAV (unmanned aerial vehicles) model which is monitored and remotely navigated by two local stations.

In Chapter 9, fault detection and isolation theory for efficient condition monitoring of industrial systems is analyzed. Two main issues in statistical methods for fault diagnosis are residuals generation and fault threshold selection. For residuals generation, an accurate model of the system in the fault-free condition is needed. Such models can be obtained through nonlinear identification techniques or through nonlinear state estimation and filtering methods. On the other hand the fault threshold should enable both diagnosis of incipient faults and minimization of the false alarms rate.

In Chapter 10, applications of statistical methods for fault diagnosis are presented. In the first case the problem of early diagnosis of cascading events in the electric power grid is considered. Residuals are generated with the use of a nonlinear model of the distributed electric power system and the fault threshold is determined with the use of the generalized likelihood ratio, assuming that the residuals follow a Gaussian distribution. In the second case, the problem of fault detection and isolation in electric motors is analyzed. It is proposed to use nonlinear filters for the generation of residuals and to derive a fault threshold from the generalized likelihood ratio without prior knowledge of the residuals statistical distribution.

In Chapter 11, it is shown that optimization through nonlinear programming techniques, such as gradient algorithms, can be an efficient approach for solving various problems in the design of intelligent robots, e.g. motion planning for multi-robot systems. A distributed gradient algorithm is proposed for coordinated navigation of an ensemble of mobile robots towards a goal state, and for assuring avoidance of collisions between the robots as well as avoidance of collisions with obstacles. The stability of the multi-robot system is proved with Lyapunov's theory and particularly with LaSalle's theorem. Motion planning with the use of distributed gradient is compared to motion planning based on particle swarm optimization.

In Chapter 12, the two-fold optimization problem of distributed motion planning and distributed filtering for multi-robot systems is studied. Tracking of a target by a multi-robot system is pursued assuming that the target's state vector is not directly measurable and has to be estimated by distributed filtering based on the target's cartesian coordinates and bearing measurements obtained by the individual mobile robots. The robots have to converge in a synchronized manner towards the target, while avoiding collisions between them and avoiding collisions with obstacles in their motion plane. To solve the overall problem, the following steps are followed: (i) distributed filtering, so as to obtain an accurate estimation of the target's state vector. This estimate provides the desirable state vector to be tracked by each one of the mobile robots, (ii) motion planning and control that enables convergence of the vehicles to the goal position and also maintains the cohesion of the vehicles swarm. The efficiency of the proposed distributed filtering and distributed motion planning scheme is tested through simulation experiments.

In Chapter 13, it is shown that evolutionary algorithms are powerful optimization methods which complement the nonlinear programming optimization techniques. In this chapter, a genetic algorithm with a new crossover operator is developed to solve the warehouse replenishment problem. The automated warehouse management is a multi-objective optimization problem since it requires to fulfill goals and performance indexes that are usually conflicting with each other. The decisions taken must ensure optimized usage of resources, cost reduction and better customer service. The proposed genetic algorithm produces Pareto-optimal permutations of the stored products.

In Chapter 14, it is shown that machine learning methods are of particular interest in the design of intelligent industrial systems since they can provide efficient control despite model uncertainties and imprecisions. The chapter proposes neural networks with Gauss-Hermite polynomial basis functions for the control of flexible-link manipulators. This neural model employs basis functions which are localized both in space and frequency thus allowing better approximation of the multi-frequency characteristics of vibrating structures. Gauss-Hermite basis functions have also some interesting properties: (i) they remain almost unchanged by the Fourier transform, which means that the weights of the associated neural network demonstrate the energy which is distributed to the various eigenmodes of the vibrating structure, (ii) unlike wavelet basis functions the Gauss-Hermite basis functions have a clear physical meaning since they represent the solutions of differential equations of stochastic oscillators and each neuron can be regarded as the frequency filter of the respective vibration eigenfrequency.

In Chapter 15, it is shown that machine learning methods can be of particular interest for fault diagnosis of systems that exhibit event-driven dynamics. For this type of systems fault diagnosis based on automata and finite state machine models has to be performed. In this chapter an application of fuzzy automata for fault diagnosis is given. The output of the monitored system is partitioned into linear segments which in turn are assigned to pattern classes (templates) with the use of membership functions. A sequence of templates is generated and becomes input to fuzzy automata which have transitions that correspond to the templates of the properly functioning system. If the automata reach their final states, i.e. the input sequence is accepted by the automata with a membership degree that exceeds a certain threshold, then normal operation is deduced, otherwise, a failure is diagnosed. Fault diagnosis of a DC motor is used as a case study.

In Chapter 16, applications of vision-based robotic systems are analyzed. Visual servoing over a network of synchronized cameras is an example where the significance of machine vision and distributed filtering and control for industrial robotic systems can be seen. A robotic manipulator is considered and a cameras network consisting of multiple vision nodes is assumed to provide the visual information to be used in the control loop. A derivative-free implementation of the Extended Information Filter is used to produce the aggregate state vector of the robot by processing local state estimates coming from the distributed vision nodes. The performance of the considered vision-based control scheme is evaluated through simulation experiments.

From the educational viewpoint, this book is addressed to undergraduate and post-graduate students as an upper-level course supplement. The book's content can be complementary to automatic control and robotics courses, giving emphasis to industrial systems design through the integration of control, estimation, fault diagnosis, optimization and machine intelligence methods. The book can be a useful resource for instructors since it provides teaching material for advanced topics in robotics and industrial engineering.

The book can be also a primary source of a course entitled "Modelling and Control of Intelligent Industrial Systems" which can be part of the academic programme of Electrical, Mechanical, Industrial Engineering and Computer Science Departments. It is also a suitable supplementary source for various other automatic control and robotics courses (such as Control Systems Design, Advanced Topics in Automatic Control, Dynamical Systems Identification, Stochastic Estimation and Multi-Sensor Fusion, Adaptive and Robust Control, Robotics: Dynamics, Kinematics and Basic Control Algorithms, Probabilistic Methods in Robotics, Fault Detection and Isolation of Industrial Systems, Industrial automation and Industrial Systems Optimization).

From the applied research and engineering point of view the book will be a useful companion to engineers and researchers since it analyzes a wide spectrum of problems in the area of industrial systems, such as: modelling and control of industrial robots, modelling, and control of mobile robots and autonomous vehicles, modelling and robust/adaptive control of electromechanical systems, estimation and sensor fusion based on measurements obtained from distributed sensors, fault detection/isolation, optimization for industrial production and machine intelligence for adaptive behaviour. As a textbook giving a thorough analysis of the aforementioned issues it is expected to enhance bibliography on industrial systems.

Through the aforementioned 16 chapters, the book is anticipated to provide a sufficient coverage of the topic of modeling and control for intelligent industrial systems and to motivate the continuation of research effort towards the development of adaptive algorithms for robotics and industrial engineering. By proposing an interdisciplinary approach in intelligent industrial systems design, the book can be a useful reference not only for the the robotics and control community, but also for researchers and engineers in the fields of mechatronics, signal processing, and computational intelligence.

Athens, October 2010 Gerasimos G. Rigatos

Acknowledgements

I would like to thank the people from the research and academic field who have worked with me throughout these years and who have given me the incentive to prepare this book, aiming at answering some of their questions on how to make better modeling and control for intelligent industrial systems.

Contents

Acronyms

AGV	Automated Ground Vehicle
ARE	Algebraic Riccati Equation
AVR	Automated Voltage Regulator
DC	Direct Current
DES	Discrete Event System
DPF	Distributed Particle Filter
EA	Evolutionary Algorithm
EIF	Extended Information Filter
EKF	Extended Kalman Filter
FDES	Fuzzy Discrete Event System
FDI	Fault Detection and Isolation
FFT	Fast Fourier Transform
FLR	Flexible Link Robot
GA	Genetic Algorithm
GLR	Generalized Likelihood Ratio
GPS	Global Positioning System
IM	Induction Motor
IMU	Inertial Measurement Unit
MOP	Multi-objective optimization problem
MSN	Mobile Sensors Networks
NARX	Nonlinear Autoregressive models with exogenous inputs
PF	Particle Filter
PSS	Power System Stabilizer
RNN	Recurrent Neural Network
SMC	Sliding Mode Control
SOP	Single-objective optimization problem
SPKF	Sigma Point Kalman Filters
SVD	Singular Values Decomposition
UAV	Unmanned Aerial Vehicle
UIF	Unscented Information Filter
UKF	Unscented Kalman Filter

Chapter 1
Industrial Robots in Contact-Free Operation

Abstract. A study of industrial robotic systems is provided, for the case of contact-free operation. This part of the book includes the dynamic and kinematic analysis of rigid-link robotic manipulators, and expands towards more specialized topics, such as dynamic and kinematic analysis of flexible-link robots, and control of rigid-link and flexible-link robots in contact-free operation.

1.1 Dynamic Analysis of Rigid Link Robots

In the area of industrial robots one can distinguish between two main problems: (i) robots operating in a free working space, as in the case of robotic welding, painting, or laser and plasma cutting and (ii) robots performing compliance tasks, as in the case of assembling, finishing of metal surfaces and polishing. When the robotic manipulator operates in a free environment then kinematic and dynamic analysis provide the means for designing a control law that will move appropriately the robot's end effector and will enable the completion of the scheduled tasks. The dynamic model of a multi-DOF rigid-link robotic manipulator, as the one depicted in Fig. 1.1, is obtained from the Euler-Lagrange principles. A generic rigid-link dynamic model is:

$$D(\theta)\ddot{\theta}+h(\theta,\dot{\theta})+G(\theta)=k(r_g\theta_m-\theta) \tag{1.1}$$

where $T(\theta)=k(r_g\theta_m-\theta)$ represents the control input vector (torque). In the latter relation, k is an elasticity coefficient and r_g denotes gears ratio, i.e. joints flexibility is introduced in the dynamic model of the manipulator [179],[180],[222]. The elements of the inertia matrix $D(\theta)$, the Coriolis and centrifugal forces matrix $h(\theta,\dot{\theta})$ and the gravity matrix $G(\theta)$ can be found in [107].

The physical characteristics of the manipulator and the range of values that the different variables of the system acquire in a real working environment can be defined for every type of industrial robot. The coordinates frames attached to each joint are defined using the Denavit-Hartenberg method and are depicted in Fig. 1.1. The

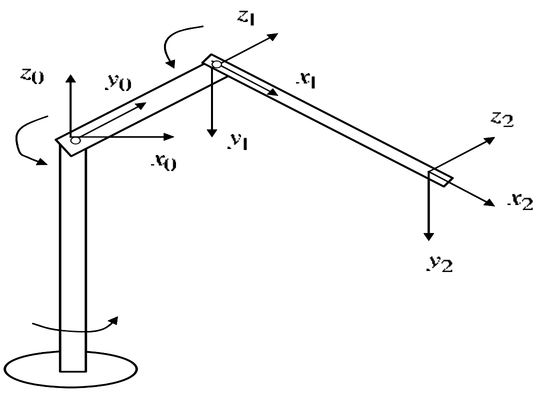


Fig. 1.1 A 3-DOF robotic manipulator with rigid links

Denavit-Hartenberg parameters for the general case of a 6-DOF robot are defined in [17] and their indicative values are given in Table 1.1:

Table 1.1 Denavit-Hartenberg parameters

i	θ_i	a_i	α_i	d_i	Joint range ^{o}
1	90	−90	0	0	−160 to 160
2	0	0	431.8mm	149.08mm	−225 to 45
3	90	90	−20.32mm	0	−45 to 225
4	0	−90	0	433.07mm	−110 to 170
5	0	90	0	0	−100 to 100
6	0	0	0	56.25mm	−266 to 266

The rigid link coordinates system and its parameters is depicted in Fig. 1.2. Considering the ith and the $(i-1)$th reference frames, the parameters of the Denavit-Hartenberg representation are defined as follows:

1. θ_i is the joint angle from the x_{i-1} axis to the x_i axis, about the z_{i-1} axis (using the right hand rule).
2. d_i is the distance from the origin of the $(i-1)$th coordinate frame to the intersection of the z_{i-1} axis, with the x_i axis along the z_{i-1} axis
3. α_i is the offset distance from the intersection of the z_{i-1} axis with the x_i axis to the origin of the ith frame along the x_i axis (or the shortest distance between the z_{i-1} and z_i axes).
4. a_i is the offset angle from the z_{i-1} axis to the z_i axis about the x_i axis (using the right hand rule)

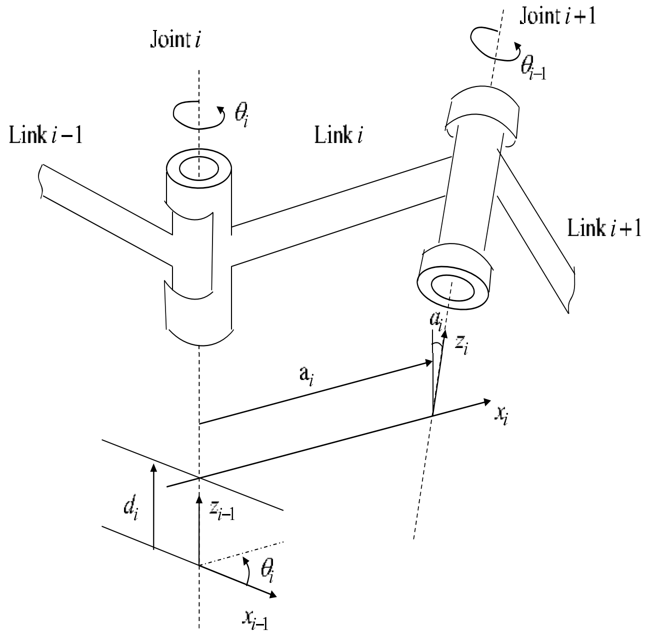


Fig. 1.2 Rigid link coordinates system and its parameters

The elements of the inertia matrix $D(\theta)$, the Coriolis and centrifugal forces matrix $h(\theta,\dot{\theta})$ and the gravity matrix $G(\theta)$ appearing in Eq. (1.1) are defined in [107] and for a 3-DOF robot are given by

$$D(\theta) = \begin{pmatrix} m_2 l_2^2 + m_3(l_2 S_2 + l_3 S_{23})^2 & 0 & 0 \\ 0 & m_2 l_2^2 + m_3(l_2^2 + 2 l_2 l_3 C_3 + l_3^2) & m_3 l_3 (l_2 C_3 + l_3) \\ 0 & m_3 l_3 (l_2 C_3 + l_3) & m_3 l_3^2 \end{pmatrix} \tag{1.2}$$

$$h(\theta,\dot{\theta}) = \begin{pmatrix} 2[m_2 l_2^2 S_2 C_2 + m_3(l_2 S_2 + l_3 S_{23})(l_2 S_2 + l_3 C_{23})]\dot{\theta}_1\dot{\theta}_2 + 2 m_3 l_3 C_{23}(l_2 S_2 + l_3 S_{23})\dot{\theta}_1\dot{\theta}_3 \\ -2 m_3 l_2 l_3 S_3 \dot{\theta}_2\dot{\theta}_3 - m_3 l_2 l_3 S_3 \dot{\theta}_3^2 - [m_2 l_2 S_2 C_2 + m_2(l_2 S_2 + l_3 S_{23})(l_2 C_2 + l_3 C_{23})]\dot{\theta}_1^2 \\ -m_3(l_2 S_2 + l_3 S_{23}) l_3 C_{23} \dot{\theta}_1^2 + m_2 l_2 l_3 S_3 \dot{\theta}_2^2 + m_3 l_3 \dot{\theta}_2 \dot{\theta}_3 \end{pmatrix} \tag{1.3}$$

$$G(\theta) = \begin{pmatrix} 0 \\ -m_2 g l_2 S_2 - m_3 g (l_2 S_2 + l_3 S_{23}) \\ m_3 g l_3 S_{23} \end{pmatrix} \tag{1.4}$$

where S_i and C_i, denote $sin(\theta_i)$ and $cos(\theta_i)$ respectively, with $i = 1,2,3$, while S_{ij} denotes $sin(\theta_i + \theta_j)$ and C_{ij} denotes $cos(\theta_i + \theta_j)$.

For the dynamic model of the 3-DOF robot shown in Fig. 1.1 and with its dynamics described in Eq. (1.1), it holds that

$$\theta = [\theta_1, \theta_2, \theta_3]^T,\ \dot{\theta} = [\dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3]^T,\ \ddot{\theta} = [\ddot{\theta}_1, \ddot{\theta}_2, \ddot{\theta}_3]^T \tag{1.5}$$

where θ is the vector of the joints angles, $\dot{\theta}$ is the vector of the angular velocities and $\ddot{\theta}$ is the vector of angular accelerations. Consequently, the robot's state vector is defined as $x{\in}R^{6\times1}$, and its derivative is given by $\dot{x}{\in}R^{6\times1}$,

$$x = [\theta_1, \theta_2, \theta_3, \dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3]^T,\ \dot{x} = [\dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3, \ddot{\theta}_1, \ddot{\theta}_2, \ddot{\theta}_3]^T \tag{1.6}$$

Then, Eq. (1.1) is written as

$$\begin{aligned} \ddot{\theta} &= D(\theta)^{-1}[-h(\theta,\dot{\theta}) - G(\theta) + k(r_g\theta_m - \theta)] \Rightarrow \\ \ddot{\theta} &= D(\theta)^{-1}[-h(\theta,\dot{\theta}) - G(\theta) + T(\theta)] \end{aligned} \tag{1.7}$$

where $T(\theta) = k(r_g\theta_m - \theta)$. The control input $u{\in}R^{3\times1}$ is defined as

$$u = D(\theta)^{-1}[-h(\theta,\dot{\theta}) - G(\theta) + T(\theta)] \tag{1.8}$$

Moreover, it holds that $\dot{x}_1 = x_4$, $\dot{x}_2 = x_5$ and $\dot{x}_3 = x_6$. Taking $0_{3\times3}$ to be 3×3 matrix with zero elements, and $I_{3\times3}$ to be the identity 3×3 matrix one obtains

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \begin{pmatrix} 0_{3\times3} & I_{3\times3} \end{pmatrix} x + \begin{pmatrix} 0_{3\times3} \end{pmatrix} u \tag{1.9}$$

Furthermore, it holds that

$$\begin{pmatrix} \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \end{pmatrix} = \begin{pmatrix} 0_{3\times3} & 0_{3\times3} \end{pmatrix} x + \begin{pmatrix} I_{3\times3} \end{pmatrix} u \tag{1.10}$$

Thus, finally the robot's dynamic model can be written in a linear state-space form given by

$$\dot{x} = Ax + Bu \tag{1.11}$$

with $A = \begin{pmatrix} 0_{3\times3} & I_{3\times3} \\ 0_{3\times3} & 0_{3\times3} \end{pmatrix}$, $B = \begin{pmatrix} 0_{3\times3} \\ I_{3\times3} \end{pmatrix}$.

The transition from the continuous time differential equations of Eq. (1.1) that describe the dynamics of the robotic manipulator, to the discrete time state-space description of Eq. (1.11) that is used in the simulation experiments can be carried out using established discretization methods and after choosing an appropriate sampling rate. Alternatively, the robot's dynamics can be simulated through numerical solution of the associated differential equations, given in Eq. (1.1).

1.2 Kinematic Analysis of Rigid Link Robots

Using the rigid-link reference system depicted in Fig. 1.2, a joint axis is established (for each joint i) at the connection of two links. This joint axis has two normals connected to it, one for each end of the links. The relative position of two such connected links (link $i-1$ and link i) is given by d_i which is the distance measured along the joint axis between the normals. The joint angle θ_i between the normals is measured in a plane that is taken to be normal to joint axis. Parameters d_i and θ_i are called the distance and angle between the adjacent links, respectively, and define the relative position of neighboring links.

A link i ($i = 1, \cdots, 6$) is connected to at most two other links, i.e link $i-1$ and link $i+1$ and two joint axes are established at the end of each connection. A fixed configuration between joints can be obtained by parameters a_i and α_i which are defined as follows: The parameter a_i is the shortest distance measured along the common normal between the joint axes, while α_i is the angle between the joint axes measured in a plane perpendicular to a_i. Equivalently, a_i and α_i are called the *length* and *twist angle* of link i.

An orthonormal cartesian coordinate system (x_i, y_i, z_i) can be established for each link at its joint axis, where $i = 1, 2, \cdots, n$ (n=number of degrees of freedom) plus the base coordinate frame. Since a rotary joint has only one degree of freedom each (x_i, y_i, z_i) coordinate frame of a robot arm corresponds to joint $i+1$ and is fixed in link i. Since the i-th coordinate system is fixed in link i it moves together with link i. Thus, the n-th coordinate frame moves the hand (link n). The base coordinates are defined as the 0-th coordinate frame (x_0, y_0, z_0) which is also the inertial coordinate frame of the robot arm. Thus for a six-axis robot arm, there are seven coordinate frames namely $(x_0, y_0, z_0), (x_1, y_1, z_1), \cdots, (x_6, y_6, z_6)$. Every coordinate frame is determined and established on the basis of three rules:

1. The z_{i-1} axis lies along the axis of motion of the i-th joint
2. The x_i axis is normal to the z_{i-1} axis and point away from it.
3. The y_i axis completes the right-handed coordinate system as required.

By these rules one is free to choose the location of coordinate frame 0 anywhere in the supporting base, as long as the z_0 axis lies along the axis of motion of the first joint. The last coordinate frame n-th frame can be placed anywhere in the robot's hand, as long as the x_n axis is normal to the z_{n-1} axis.

Once the Denavit-Hartneberg (D-H) coordinate system has been established for each link (according to the analysis given in subsection 1.1), a homogeneous transformation matrix can easily be developed relating the i-th coordinate frame to the $(i-1)$-th coordinate frame. Thus, a point r_i expressed in the i-th coordinate system may be expressed in the $(i-1)$-th coordinate system as r_{i-1} by performing the following successive transformations:

1. Rotate about the z_{i-1} axis of an angle θ_i to align the x_{i-1} axis with the x_i axis (x_{i-1} axis is parallel to x_i axis and pointing in the same direction).

2. Translate along the z_{i-1} axis a distance of d_i to bring the x_{i-1} and x_i axes into coincidence.
3. Translate along the x_i axis a distance of α_i to bring the two origins, as well as the x axis into coincidence.
4. Rotate about the x_i axis an angle of a_i to bring the two coordinate systems into coincidence.

Each of these four operations can be expressed by a basic homogeneous rotation-translation matrix and the product of these four basic homogeneous transformation matrices yields a composite homogeneous transformation matrix ${}^{i-1}A_i$, known as the D-H transformation matrix for adjacent coordinate frames i and $i-1$. Thus,

$$^{i-1}A_i = T_{z,d}T_{z,\theta}T_{x,\alpha}T_{x,a} =$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} cos(\theta_i) & -sin(\theta_i) & 0 & 0 \\ sin(\theta_i) & cos(\theta_i) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & \alpha_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(a_i) & -sin(a_i) & 0 \\ 0 & sin(a_i) & cos(a_i) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

$$\text{i.e.}\, ^{i-1}A_i = \begin{pmatrix} cos(\theta_i) & -cos(a_i)sin(\theta_i) & sin(a_i)sin(\theta_i) & \alpha_i cos(\theta_i) \\ sin(\theta_i) & cos(a_i)cos(\theta_i) & -sin(a_i)cos(\theta_i) & \alpha_i sin(\theta_i) \\ 0 & sin(a_i) & cos(a_i) & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{1.12}$$

The inverse of this transformation enables transition from the reference system i to the reference system $i-1$.

$$[^{i-1}A_i]^{-1} = ^i A_{i-1} = \begin{pmatrix} cos(\theta_i) & sin(\theta_i) & 0 & -\alpha_i \\ -cos(a_i)sin(\theta_i) & cos(a_i)cos(\theta_i) & sin(a_i) & -d_i sin(a_i) \\ sin(a_i)sin(\theta_i) & -sin(a_i)cos(\theta_i) & cos(a_i) & -d_i cos(a_i) \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{1.13}$$

where α_i, a_i, d_i are constants while θ_i is the joint variable for a revolute joint. For a prismatic joint, the joint variable is d_i, while a_i, α_i and θ_i are constants. In this case, ${}^{i-1}A_i$ becomes

$$^{i-1}A_i = T_{z,\theta}T_{z,d}T_{x,\alpha} = \begin{pmatrix} cos(\theta_i) & -cos(a_i)sin(\theta_i) & sin(a_i)sin(\theta_i) & 0 \\ sin(\theta_i) & cos(a_i)cos(\theta_i) & -sin(a_i)cos(\theta_i) & 0 \\ 0 & sin(a_i) & cos(a_i) & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{1.14}$$

and its inverse is

$$[{}^{i-1}A_i]^{-1} = {}^iA_{i-1} = \begin{pmatrix} cos(\theta_i) & sin(\theta_i) & 0 & 0 \\ -cos(a_i)sin(\theta_i) & cos(a_i)cos(\theta_i) & sin(a_i) & -d_i sin(a_i) \\ sin(a_i)sin(\theta_i) & -sin(a_i)cos(\theta_i) & cos(a_i) & -d_i cos(a_i) \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{1.15}$$

Using the $[{}^{i-1}A_i]^{-1}$ matrix, one can relate a point p_i at the rest in link i, and expressed in homogeneous coordinates with respect to the coordinate system i, to the coordinate system $i-1$ established at link $i-1$ by

$$p_{i-1} = [{}^{i-1}A_i]^{-1} p_i \tag{1.16}$$

where $p_{i-1} = (x_{i-1}, y_{i-1}, z_{i-1}, 1)^T$ and $p_i = (x_i, y_i, z_i)^T$. For the six-DOF robotic manipulator the associate coordinates transformation matrices ${}^{i-1}A_i$ are given by

$$\begin{gathered} {}^0A_1 = \begin{pmatrix} C_1 & 0 & -S_1 & 0 \\ S_1 & 0 & C_1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad {}^1A_2 = \begin{pmatrix} C_2 & -S_2 & 0 & \alpha_2 C_2 \\ S_2 & C_2 & 0 & \alpha_2 S_2 \\ 0 & 0 & 1 & d_2 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\ {}^2A_3 = \begin{pmatrix} C_3 & 0 & S_3 & \alpha_3 C_3 \\ S_3 & 0 & -C_3 & \alpha_3 S_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad {}^3A_4 = \begin{pmatrix} C_4 & 0 & -S_4 & 0 \\ S_4 & 0 & C_4 & 0 \\ 0 & -1 & 0 & d_4 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\ {}^4A_5 = \begin{pmatrix} C_5 & 0 & S_5 & 0 \\ S_5 & 0 & -C_5 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad {}^5A_6 = \begin{pmatrix} C_6 & -S_6 & 0 & 0 \\ S_5 & C_6 & 0 & 0 \\ 0 & 0 & 1 & d_6 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{gathered} \tag{1.17}$$

$$\begin{gathered} T_1 = {}^0A_1{}^1A_2{}^2A_3 = \begin{pmatrix} C_1C_{23} & -S_1 & C_1S_{23} & \alpha_2C_1C_2 + \alpha_3C_1C_{23} - d_2S_1 \\ S_1C_{23} & C_1 & S_1S_{23} & \alpha_2S_1C_2 + \alpha_3S_1C_{23} - d_2C_1 \\ -S_{23} & 0 & C_{23} & -\alpha_2S_2 - \alpha_3S_{23} \\ 0 & 0 & 0 & 1 \end{pmatrix} \\ T_2 = {}^3A_4{}^4A_5{}^5A_6 = \begin{pmatrix} C_4C_5C_6 - S_4S_6 & -C_4C_5S_6 - S_4C_6 & C_4S_5 & d_6C_4S_5 \\ S_4C_5C_6 + C_4S_6 & -S_4C_5S_6 + C_4C_6 & S_4S_5 & d_6S_4S_5 \\ -S_5C_6 & S_5S_6 & C_5 & d_6C_5 + d_4 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{gathered} \tag{1.18}$$

where $C_i = cos(\theta_i)$, $S_i = sin(\theta_i)$, $C_{ij} = cos(\theta_i + \theta_j)$, $S_{ij} = sin(\theta_i + \theta_j)$. The homogeneous matrix ${}^0T_i$ which specifies the location of the i-th coordinate frame with respect to the base coordinate system is the chain product of successive coordinate transformation matrices of ${}^{i-1}A_i$ and is expressed as

$$
\begin{aligned}
{}^{0}T_i &= {}^{0}A_1 {}^{1}A_2 \cdots {}^{i-1}A_i = \Pi_{j=1}^{i} A_j \text{ for } i = 1,2,\cdots,n \\
&= \begin{pmatrix} x_i \; y_i \; z_i \; p_i \\ 0 \; 0 \; 0 \; 1 \end{pmatrix} = \begin{pmatrix} {}^{0}R_i & {}^{0}p_i \\ 0 & 1 \end{pmatrix}
\end{aligned} \tag{1.19}
$$

1.3 Dynamic Analysis of Flexible-Link Robots

Flexible-link robots comprise an important class of systems that include lightweight arms for assembly, civil infrastructure, bridge/vehicle systems, military applications and large-scale space structures. Modelling and vibration control of flexible systems have received a great deal of attention in recent years [8],[26],[176],[327],[328],[395]. Conventional approaches to design a control system for a flexible-link robot often involve the development of a mathematical model describing the robot dynamics, and the application of analytical techniques to this model to derive an appropriate control law [11],[55],[78]. Usually, such a mathematical model consists of nonlinear partial differential equations, most of which are obtained using some approximation or simplification [176],[327].

A common approach in modelling of flexible-link robots is based on the Euler-Bernoulli model [97],[445]. This model consists of nonlinear partial differential equations, which are obtained using some approximation or simplification. In case of a single-link flexible manipulator the basic variables of this model are $w(x,t)$ which is the deformation of the flexible link, and $\theta(t)$ which is the joint's angle.

$$E{\cdot}I{\cdot}w^{''''}(x,t) + \rho\ddot{w}(x,t) + \rho x\ddot{\theta}(t) = 0 \tag{1.20}$$

$$I_t\ddot{\theta}(t) + \rho\int_0^L x\ddot{w}(x,t)dx = T(t) \tag{1.21}$$

In Eq. (1.20) and (1.21), $w^{''''}(x,t) = \frac{\partial^4 w(x,t)}{\partial x^4}, \ddot{w}(x,t) = \frac{\partial^2 w(x,t)}{\partial t^2}$, while I_t is the moment of inertia of a rigid link of length L, ρ denotes the uniform mass density and EI is the uniform flexural rigidity with units $N{\cdot}m^2$. To calculate $w(x,t)$, instead of solving analytically the above partial differential equations, modal analysis can be used which assumes that $w(x,t)$ can be approximated by a weighted sum of orthogonal basis functions

$$w(x,t) = \sum_{i=1}^{n_e} \phi_i(x)v_i(t) \tag{1.22}$$

where index $i = [1,2,\cdots,n_e]$ denotes the normal modes of vibration of the flexible link. Using modal analysis a dynamical model of finite-dimensions is derived for the flexible link robot. Without loss of generality assume a 2-link flexible robot (Fig. 1.3) and that only the first two vibration modes of each link are significant ($n_e = 2$). Σ_1 is a point on the first link with reference to which the deformation vector

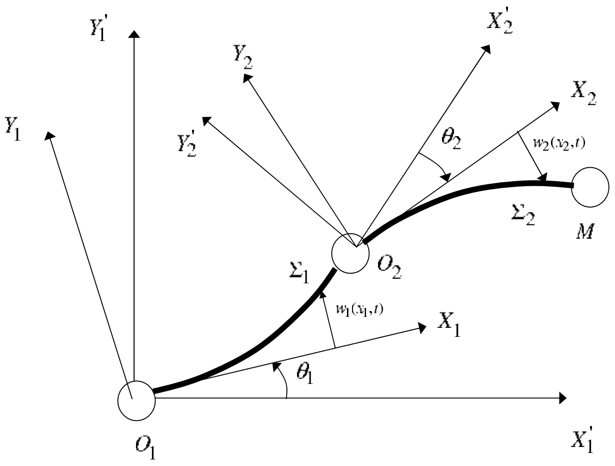


Fig. 1.3 A 2-DOF flexible-link robot

is measured. Similarly, Σ_2 is a point on the second link with reference to which the associated deformation vector is measured. In that case the dynamic model of the robot becomes [222],[445]:

$$\begin{pmatrix} M_{11}(z) & M_{12}(z) \\ M_{21}(z) & M_{22}(z) \end{pmatrix} \cdot \begin{pmatrix} \ddot{\theta} \\ \ddot{v} \end{pmatrix} + \begin{pmatrix} F_1(z,\dot{z}) \\ F_2(z,\dot{z}) \end{pmatrix} + \begin{pmatrix} 0_{2\times 2} & 0_{2\times 4} \\ 0_{4\times 2} & D(z) \end{pmatrix} \cdot \begin{pmatrix} \dot{\theta} \\ \dot{v} \end{pmatrix} + \\ + \begin{pmatrix} 0_{2\times 2} & 0_{2\times 4} \\ 0_{4\times 2} & K(z) \end{pmatrix} \cdot \begin{pmatrix} \theta \\ v \end{pmatrix} = \begin{pmatrix} T(t) \\ 0_{4\times 1} \end{pmatrix} \tag{1.23}$$

where $z = [\theta, v]^T$, with $\theta = [\theta_1, \theta_2]^T$, $v = [v_{11}, v_{12}, v_{21}, v_{22}]^T$ (vector of the vibration modes for links 1 and 2), and $[F_1(z,\dot{z}), F_2(z,\dot{z})]^T = [0,0]^T$ (centrifugal and Coriolis forces). The elements of the inertia matrix are: $M_{11} \in R^{2\times 2}$, $M_{12} \in R^{2\times 4}$, $M_{21} \in R^{4\times 2}$, $M_{22} \in R^{4\times 4}$. The damping and elasticity matrices of the aforementioned model are $D \in R^{4\times 4}$ and $K \in R^{4\times 4}$. Moreover the vector of the control torques is $T(t) = [T_1(t), T_2(t)]^T$.

1.4 Kinematic Analysis of Flexible-Link Robots

Assume the i-th link of the flexible-link robot and the associated rotating frame $O_iX_iY_i$ (Fig. 1.3). Then the vector of coordinates of the end-effector M is given by

$$p_M^i = [x_i, w_i(x_i)]^T \tag{1.24}$$

The coordinates of the end-effector in the inertial frame $O_1X_1^{'}Y_1^{'}$ is given by

$$p_M = r_i + W_i p_M^i \tag{1.25}$$

with

$$W_i = W_{i-1}E_{i-1}R_i = \hat{W}_{i-1}R_i \\ \hat{W}_0 = I \tag{1.26}$$

where R_i is the rigid rotation matrix that aligns the rotating frame of the i-th link to the inertial frame of the same link, and E_{i-1} is the flexible rotation matrix that aligns the inertial frame of link i to the rotational frame of link $i-1$:

$$R_i = \begin{pmatrix} cos(\theta_i) & -sin(\theta_i) \\ sin(\theta_i) & cos(\theta_i) \end{pmatrix}, \ E_i = \begin{pmatrix} 1 & -w_{ie}^{'} \\ w_{ie}^{'} & 1 \end{pmatrix} = \begin{pmatrix} 1 & -\frac{\partial w_i}{\partial x_i} \\ \frac{\partial w_i}{\partial x_i} & 1 \end{pmatrix} \tag{1.27}$$

$$r_i = r_{i-1} + W_i r_i^{i-1} \tag{1.28}$$

Variable r_i^{i-1} denotes the distance vector between the origin of the and i-th and the $i-1$-th frame, r_i is the distance vector between the origin of the i-th rotational frame and the inertial frame, and W_i is the rotation matrix calculated with the use of Eq.(1.26).

Using Eq. (1.25) and Eq. (1.28) in the 2-DOF flexible-link robot depicted in Fig. 1.3, one obtains

$$r_2 = r_1 + W_1 r_2^1 = \begin{pmatrix} L_1 cos(\theta_1) - w_1(L_1,t) sin(\theta_1) \\ L_1 sin(\theta_1) + w_1(L_1,t) cos(\theta_1) \end{pmatrix} \tag{1.29}$$

$$p_M = r_2 + W_2 p_M^2 \tag{1.30}$$

where

$$p_M^2 = \begin{pmatrix} L_2 \\ w_2(L_2,t) \end{pmatrix}, W_2 = R_1 E_1 R_2 = \begin{pmatrix} cos(\theta_1) & -sin(\theta_1) \\ sin(\theta_1) & cos(\theta_1) \end{pmatrix} \cdot \\ \cdot \begin{pmatrix} 1 & -w_{1e}^{'} \\ w_{1e}^{'} & 1 \end{pmatrix} \cdot \begin{pmatrix} cos(\theta_2) & -sin(\theta_2) \\ sin(\theta_2) & cos(\theta_2) \end{pmatrix} \tag{1.31}$$

The differential kinematic model of the flexible-link robot can now be calculated. The coordinates of the end-effector in the inertial frame are given by Eq. (1.25). According to modal analysis the deformation $w_i(x_i,t)$ in normal modes of vibration is given by Eq. (1.22). Using the previous 2 equations the kinematic model can be written as a function of the joint angles θ and of the normal modes of vibration v.

$$p = k(\theta, v) \tag{1.32}$$

The velocity of the end-effector is calculated through the differentiation of Eq. (1.25).

$$\dot{p}_M = \dot{r}_i + \dot{W}_i p_M^i + W_i \dot{p}_M^i \tag{1.33}$$

Moreover, it holds that $\dot{r}^i_{i+1} = \dot{p}^i_M(L_i) = [0, \dot{w}_i(x_i = L_i)]^T$ since there is no longitudinal deformation ($\dot{x}_i = 0$). It also holds that

$$\begin{aligned} \dot{W}_i &= \dot{\hat{W}}_{i-1}R_i + \hat{W}_{i-1}\dot{R}_i \\ \dot{\hat{W}}_i &= \dot{W}_i E_i + W_i \dot{E}_i \end{aligned} \tag{1.34}$$

It also holds that

$$\begin{aligned} \dot{R}_i &= SR_i\dot{\theta}_i \\ \dot{E}_i &= S\dot{w}^{'}_{ie} \end{aligned} \tag{1.35}$$

with $S = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$. Substituting Eq. (1.34) and Eq. (1.35) in Eq. (1.33) the differential kinematic model of the flexible-link robot is obtained:

$$\dot{p} = J_\theta(\theta, v)\dot{\theta} + J_v(\theta, v)\dot{v} \tag{1.36}$$

where

$J_\theta = \frac{\partial k}{\partial \theta}$: is the Jacobian with respect to θ
$J_v = \frac{\partial k}{\partial v}$: is the Jacobian with respect to v.

If the end-effector is in contact with the surface $\Omega(\theta)$ and is subject to contact-forces $F = [F_x, F_y]$ then the torques which are developed to the joints are:

$J_\theta^T F$: torques that produce the work associated with the rotation angle θ.
$J_v^T F$: torques that produce work associated with the deformation modes v.

In case of contact with a surface, the dynamic model of the flexible-link robot given initially in Eq. (1.23) is corrected into:

$$\begin{aligned} &\begin{pmatrix} M_{11}(z) & M_{12}(z) \\ M_{21}(z) & M_{22}(z) \end{pmatrix} \begin{pmatrix} \ddot{\theta} \\ \ddot{v} \end{pmatrix} + \begin{pmatrix} F_1(z,\dot{z}) \\ F_2(z,\dot{z}) \end{pmatrix} + \begin{pmatrix} 0_{2\times2} & 0_{2\times4} \\ 0_{4\times2} & D(z) \end{pmatrix} \begin{pmatrix} \dot{\theta} \\ \dot{v} \end{pmatrix} + \\ &+ \begin{pmatrix} 0_{2\times2} & 0_{2\times4} \\ 0_{4\times2} & K(z) \end{pmatrix} \begin{pmatrix} \theta \\ v \end{pmatrix} = \begin{pmatrix} T(t) - J_\theta^T(\theta, v)F \\ -J_v^T(\theta, v)F \end{pmatrix} \end{aligned} \tag{1.37}$$

For a two-link flexible robot of Fig. 1.3 one gets

$$\begin{aligned} p_M = &\begin{pmatrix} L_1 cos(\theta_1) - w_1(L_1,t)sin(\theta_1) \\ L_1 sin(\theta_1) + w_1(L_1,t)cos(\theta_1) \end{pmatrix} + \\ &+ \begin{pmatrix} cos(\theta_1+\theta_2) - w^{'}_{1e}sin(\theta_1+\theta_2) & -sin(\theta_1+\theta_2) - w^{'}_{1e}cos(\theta_1+\theta_2) \\ sin(\theta_1+\theta_2) + w^{'}_{1e}cos(\theta_1+\theta_2) & cos(\theta_1+\theta_2) - w^{'}_{1e}sin(\theta_1+\theta_2) \end{pmatrix} \begin{pmatrix} L_2 \\ w_2 \end{pmatrix} \end{aligned} \tag{1.38}$$

with

$$\begin{array}{l} w_1(L_1,t) = \phi_{11}(L_1)v_{11}(t) + \phi_{12}(L_1)v_{12}(t) \\ w_2(L_2,t) = \phi_{21}(L_2)v_{21}(t) + \phi_{22}(L_2)v_{22}(t) \\ w'_{1e} = \frac{\partial w_1(x,t)}{\partial x}|_{x=L_1} = \phi'_{11}(L_1)v_{11}(t) + \phi'_{12}(L_1)v_{12}(t) \end{array} \tag{1.39}$$

The Jacobian J_θ is

$$J_\theta = \begin{pmatrix} \frac{\partial p_M^{(1)}}{\partial \theta_1} & \frac{\partial p_M^{(1)}}{\partial \theta_2} \\ \frac{\partial p_M^{(2)}}{\partial \theta_1} & \frac{\partial p_M^{(2)}}{\partial \theta_2} \end{pmatrix} \tag{1.40}$$

$$\begin{array}{l} \frac{\partial p_M^{(1)}}{\partial \theta_1} = -L_1 sin(\theta_1) - w_1(L_1,t)cos(\theta_1) - L_2 sin(\theta_1+\theta_2) - L_2 w'_{1e} cos(\theta_1+\theta_2) - \\ -w_2(L_2,t)cos(\theta_1+\theta_2) + w_2(L_2,t)w'_{1e} sin(\theta_1+\theta_2) \\ \\ \frac{\partial p_M^{(2)}}{\partial \theta_1} = L_1 cos(\theta_1) - w_1(L_1,t)sin(\theta_1) + L_2 cos(\theta_1+\theta_2) - L_2 w'_{1e} sin(\theta_1+\theta_2) - \\ -w_2(L_2,t)sin(\theta_1+\theta_2) + w_2(L_2,t)w'_{1e} cos(\theta_1+\theta_2) \\ \\ \frac{\partial p_M^{(1)}}{\partial \theta_2} = -L_2 sin(\theta_1+\theta_2) - L_2 w'_{1e} cos(\theta_1+\theta_2) - w_2(L_2,t)cos(\theta_1+\theta_2) + \\ +w_2(L_2,t)w'_{1e} sin(\theta_1+\theta_2) \\ \\ \frac{\partial p_M^{(2)}}{\partial \theta_2} = L_2 cos(\theta_1+\theta_2) - L_2 w'_{1e} sin(\theta_1+\theta_2) - w_2(L_2,t)sin(\theta_1+\theta_2) - \\ -w_2(L_2,t)w'_{1e} cos(\theta_1+\theta_2) \end{array} \tag{1.41}$$

Similarly, the Jacobian J_v is calculated:

$$J_v = \begin{pmatrix} \frac{\partial p_M^{(1)}}{\partial v_{11}} & \frac{\partial p_M^{(1)}}{\partial v_{12}} & \frac{\partial p_M^{(1)}}{\partial v_{21}} & \frac{\partial p_M^{(1)}}{\partial v_{22}} \\ \frac{\partial p_M^{(2)}}{\partial v_{11}} & \frac{\partial p_M^{(2)}}{\partial v_{12}} & \frac{\partial p_M^{(2)}}{\partial v_{21}} & \frac{\partial p_M^{(2)}}{\partial v_{22}} \end{pmatrix} \tag{1.42}$$

$$\frac{\partial p_M^{(1)}}{\partial v_{11}} = -\phi_{11}(L_1)sin(\theta_1) - L_2\phi'_{11}(L_1)sin(\theta_1+\theta_2) - w_2(L_2,t)\phi'_{11}(L_1)cos(\theta_1+\theta_2)$$

$$\frac{\partial p_M^{(1)}}{\partial v_{12}} = -\phi_{12}(L_1)sin(\theta_1) - L_2\phi'_{12}(L_1)sin(\theta_1+\theta_2) - w_2(L_2,t)\phi'_{12}(L_1)cos(\theta_1+\theta_2)$$

$$\frac{\partial p_M^{(1)}}{\partial v_{21}} = -\phi_{21}(L_2)sin(\theta_1+\theta_2) - \phi_{21}(L_2)w'_{1e}cos(\theta_1+\theta_2)$$

$$\frac{\partial p_M^{(1)}}{\partial v_{22}} = -\phi_{22}(L_2)sin(\theta_1+\theta_2) - \phi_{22}(L_2)w'_{1e}cos(\theta_1+\theta_2)$$

$$\frac{\partial p_M^{(2)}}{\partial v_{11}} = \phi_{11}(L_1)cos(\theta_1) + L_2\phi_{11}^{'}(L_1)cos(\theta_1+\theta_2) - w_2(L_2,t)\phi_{11}^{'}(L_1)sin(\theta_1+\theta_2)$$

$$\frac{\partial p_M^{(2)}}{\partial v_{12}} = \phi_{12}(L_1)cos(\theta_1) + L_2\phi_{12}^{'}(L_1)cos(\theta_1+\theta_2) - w_2(L_2,t)\phi_{12}^{'}(L_1)sin(\theta_1+\theta_2)$$

$$\frac{\partial p_M^{(2)}}{\partial v_{21}} = \phi_{21}(L_2)cos(\theta_1+\theta_2) - \phi_{21}(L_2)w_{1_e}^{'}sin(\theta_1+\theta_2)$$

$$\frac{\partial p_M^{(2)}}{\partial v_{22}} = \phi_{22}(L_2)cos(\theta_1+\theta_2) - \phi_{22}(L_2)w_{1_e}^{'}sin(\theta_1+\theta_2)$$

1.5 Control of Rigid-Link Robots in Contact-Free Operation

The computed torque method is the basis in all control methods for robotic manipulators [107],[391]. The robotic model undergoes a linearization transformation and decoupling through state feedback and next one can design local PD-type controllers for each joint of the robot. To compensate for modelling uncertainties or external disturbances a robust control term can be included in the computed torque control signal. On the other hand when the robotic model is unknown the computed torque method can be implemented within an adaptive control scheme, where the unknown robot dynamics is learned by an adaptive algorithm.

In the computed torque method the robot manipulator is modeled as a set of rigid bodies connected in series with one end fixed to the ground and the other end free. The rigid bodies are connected via revolute or prismatic joints and a torque actuator acts at each joint. Taking into account the effect of disturbances T_d in Eq. (1.1), the dynamic equation of the manipulator is given by:

$$D(\theta)\ddot{\theta} + B(\theta,\dot{\theta}) + G(\theta) = T - T_d \tag{1.43}$$

where

T : $(n\times 1)$ is the vector of joint torques supplied by the actuators,
$D(\theta)$:$(n{\times}n)$ is the manipulator inertia matrix,
$B(\theta,\dot{\theta})$ $(n\times 1)$ is the vector representing centrifugal and Coriolis effects,
$G(\theta)$: $(n\times 1)$ is the vector representing gravity,
T_d: $(n{\times}1)$ is the vector describing unmodeled dynamics and external disturbances,
θ: $(n{\times}1)$ is the vector of joint positions,
$\dot{\theta}$: $(n\times 1)$ is the vector of joint velocities,
$\ddot{\theta}$: $(n\times 1)$ is the vector of joint accelerations.

In the computed torque method (also known as inverse model control), a completely decoupled error dynamics equation can be obtained. The overall architecture of this model is shown in Figure 1.4. If the dynamic model is exact, the dynamic

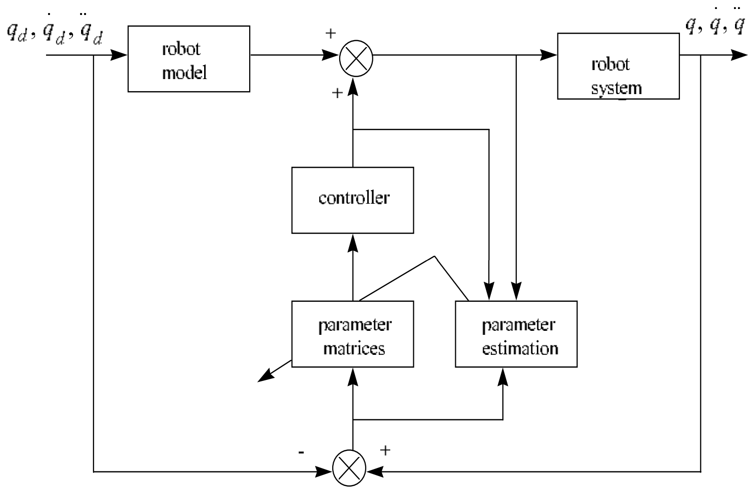


Fig. 1.4 Computed torque controller architecture for rigid-link robot control, including on-line estimation of the unknown parameters of the robot's dynamic model

perturbations are exactly canceled. The total torque that drives the manipulator is given by

$$T = D(\theta)u + \hat{B}(\theta,\dot{\theta}) + \hat{G}(\theta) + T_d, \tag{1.44}$$

where u is defined as

$$u = \ddot{\theta}_d + K_v(\dot{\theta}_d - \dot{\theta}) + K_p(\theta_d - \theta) \tag{1.45}$$

Substituting Eq. (1.44) into Eq. (1.43) and noting that $\hat{D} = D$, $\hat{B} = B$ and $\hat{G} = G$ because of the exact dynamic model assumption, one obtains

$$D(\theta)[\ddot{\theta}_d + K_v(\dot{\theta}_d - \dot{\theta}) + K_p(\theta_d - \theta)] = 0 \tag{1.46}$$

The following error vectors can be defined: $e = (\theta_d - \theta), \dot{e} = (\dot{\theta}_d - \dot{\theta})$ *and* $\ddot{e} = (\ddot{\theta}_d - \ddot{\theta})$. Under the assumption that $D(\theta)$ is positive definite, these are related by the following equation:

$$\ddot{e} + K_v\dot{e} + K_p e = 0 \tag{1.47}$$

1.6 Control of Flexible-Link Robots in Contact-Free Operation

1.6.1 Inverse Dynamics Control of Flexible-Link Robots

The inverse dynamics control approach can be also applied to flexible-link robot models. Inverse dynamics control transforms the nonlinear system of Eq. (1.23) into a linear one, so that linear control techniques can be applied. From Eq. (1.23) it holds that:

$$M_{11}\ddot{\theta}+M_{12}\ddot{v}+F_1(z,\dot{z})=T(t) \tag{1.48}$$

$$M_{21}\ddot{\theta}+M_{22}\ddot{v}+F_2(z,\dot{z})+D\dot{v}+Kv=0 \tag{1.49}$$

Eq. (1.49) is solved with respect to $\ddot{v}$

$$\ddot{v}=-M_{22}^{-1}M_{21}\ddot{\theta}-M_{22}^{-1}F_2(z,\dot{z})-M_{22}^{-1}D\dot{v}-M_{22}^{-1}Kv \tag{1.50}$$

Eq. (1.50) is substituted in Eq. (1.48) which results into:

$$\begin{aligned}&(M_{11}-M_{12}M_{22}^{-1}M_{21})\ddot{\theta}-M_{12}M_{22}^{-1}F_2(z,\dot{z})-M_{12}M_{22}^{-1}D\dot{v}-\\&-M_{12}M_{22}^{-1}Kv+F_1(z,\dot{z})=T(t)\end{aligned} \tag{1.51}$$

The following control law is now introduced [445]:

$$\begin{aligned}T(t)=&-M_{12}M_{22}^{-1}F_2(z,\dot{z})-M_{12}M_{22}^{-1}D\dot{v}-M_{12}M_{22}^{-1}Kv+\\&+F_1(z,\dot{z})+(M_{11}-M_{12}M_{22}^{-1}M_{21})u_0\end{aligned} \tag{1.52}$$

$$u_0=\ddot{\theta}_d-K_d(\dot{\theta}-\dot{\theta}_d)-K_p(\theta-\theta_d) \tag{1.53}$$

By replacing Eq. (1.52) in Eq. (1.51) one gets

$$\begin{aligned}&(M_{11}-M_{12}M_{22}^{-1}M_{21})\ddot{\theta}-M_{12}M_{22}^{-1}F_2(z,\dot{z})-M_{12}M_{22}^{-1}D\dot{v}-M_{12}M_{22}^{-1}Kv+F_1(z,\dot{z})=\\&=-M_{12}M_{22}^{-1}F_2(z,\dot{z})-M_{12}M_{22}^{-1}D\dot{v}-M_{12}M_{22}^{-1}Kv+F_1(z,\dot{z})+(M_{11}-M_{12}M_{22}^{-1}M_{21})u_0\end{aligned}$$

which finally results into

$$\ddot{\theta}=u_0 \tag{1.54}$$

Eq. (1.54) implies that linearisation and decoupling of the robotic model has been achieved. Substituting Eq. (1.53) into Eq. (1.54) gives:

$$\begin{aligned}&\ddot{\theta}-\ddot{\theta}_d+K_d(\dot{\theta}-\dot{\theta}_d)+K_p(\theta-\theta_d)=0\Rightarrow\\&\ddot{e}(t)+K_d\dot{e}(t)+K_pe(t)=0\end{aligned} \tag{1.55}$$

Gain matrices K_p and K_d are selected, so as to assure that the roots (poles) of Eq. (1.55) are in the left semiplane. This results into

$$lim_{t\to\infty}e(t)=0\Rightarrow lim_{t\to\infty}\theta(t)=\theta_d(t) \tag{1.56}$$

Consequently, for $\theta_d(t)$ =constant it holds $lim_{t\to\infty}\ddot{\theta}(t)=0$. Then Eq. (1.50) gives

$$\ddot{v}=-M_{22}^{-1}F_2-M_{22}^{-1}D\dot{v}-M_{22}^{-1}Kv \tag{1.57}$$

and for $F_2(z,\dot{z}) = 0$ results into

$$\ddot{v} + M_{22}^{-1} D\dot{v} + M_{22}^{-1} K v = 0 \tag{1.58}$$

which is the differential equation of the free damped oscillator. Suitable selection of the damping matrix D and the elasticity matrix K assures that

$$lim_{t \to \infty} v(t) = 0 \tag{1.59}$$

The objective of the above analyzed inverse-dynamics model-based control for flexible-link robots, is to make the rigid-mode variable $\theta(t)$ follow a desired trajectory or to converge to a certain set-point and at the same time to suppress the flexible modes of the links $v(t)$. However, this control approach has several weaknesses [222]:

1. The inverse dynamics model-based control for flexible link robots is based on modal analysis, i.e. on the assumption that the deformation of the flexible link can be written as a finite series expansion containing the elementary vibration modes [445]. However, this inverse-dynamics model-based control may result into unsatisfactory performance when an accurate model is unavailable, due to parameters uncertainty or truncation of high order vibration modes [222].
2. In general there are n_r flexible links, thus $\theta(t) \in R^{n_r}$. The control input available is $T(t) \in R^{n_r}$, since there is one actuator per link. Considering n_f flexible modes for each link means that $n_r \times n_f$ additional degrees of freedom are introduced. Thus appropriate control is required to suppress the vibrations. However, the number of control inputs is n_r which is less than the number of the degrees of freedom. Consequently, there is reduced control effectiveness.
3. Designing a controller for a flexible link robot without taking into account the links vibrations is an unsuccessful approach. By selecting the control input $T(t)$ to achieve practical tracking performance of the rigid variable $\theta(t)$, one actually destabilizes the flexible modes $v(t)$. This is due to the non-minimum phase nature of the zeros dynamics of the flexible-link arms.
4. Another drawback of model-based control is that the model of Eq. (1.23), is derived assuming a finite number of vibration modes. This simplification is not always applicable since higher-order modes may be excited. The proposed model-based control does not provide robustness to external disturbances.

1.6.2 Energy-Based Control of Flexible Link Robots

1.6.2.1 Energy-Based Control

To overcome the weaknesses of the inverse-dynamics model-based control for flexible link robots, model-free control methods have been proposed. These approaches are analyzed in the rest part of Chapter 1. One such approach is the energy-based control which requires only knowledge of the potential and kinetic energy of the flexible manipulator. Energy-based control of flexible-link robots assures closed-loop system stability in the case of constant set-points (point-to-point control).

The kinetic energy E_{kin} of a n-link flexible robot is given by [119],[445]

$$E_{kin} = \sum_{i=1}^{n} \frac{1}{2} \rho \int_0^{L_i} [\dot{p}_{x_i}^2 + \dot{p}_{y_i}^2] dx \tag{1.60}$$

In Eq. (1.60), p_{x_i} is the position of elementary segment of the i-th link along x-axis, while p_{y_i} is the position of elementary segment of the i-th link along y-axis. On the other hand the potential energy E_p of a planar n-link flexible robot is due to the links deformation and is given by

$$E_p = \sum_{i=1}^{n} \frac{1}{2} EI \int_0^{L_i} [\frac{\partial^2}{\partial x^2} w_i(x,t)]^2 dx \tag{1.61}$$

Thus to estimate the robot's potential energy, measurement of the flexible links strain $\frac{\partial^2 w_i(x,t)}{\partial x^2}$ is needed. The potential energy includes only the energy due to strain, while the gravitational effect as well as longitudinal and torsional deformations are neglected.

Moreover, the energy provided to the flexible-link robot by the i-th motor is given by

$$W_i = \int_0^t T_i(\tau) \dot{\theta}(\tau) d\tau \tag{1.62}$$

Consequently, the power of the i-th motor is

$$P_i(t) = T_i(t) \dot{\theta}_i(t) \tag{1.63}$$

where $T_i(t)$ is the torque of the i-th motor and $\dot{\theta}_i(t)$ is the motor's angular velocity. Thus, the aggregate motors energy is given by

$$W = \sum_{i=1}^{n} \int_0^t T_i(\tau) \dot{\theta}_i(\tau) d\tau \tag{1.64}$$

The energy that is provided to the flexible-link robot by its motors takes the form of: (i) potential energy (due to the deformation of the flexible links) and (ii) kinetic energy. This energy flow is described by

$$[E_{kin}(t) + E_p(t)] - [E_{kin}(0) + E_p(0)] = W \tag{1.65}$$

Energy-based control of flexible-link robots considers that the torque of the i-th motor (control output) is based on a PD-type controller and is given by [119],[445]:

$$\begin{array}{l} T_i(t) = -K_{p_i}[\theta_i(t) - \theta_{d_i}(t)] - K_{d_i} \dot{\theta}_i(t) - \\ -K_i w_i^{''}(x,t) \int_0^t \dot{\theta}_i(s) w_i^{''}(x,s) ds, \ i = 1,2,\cdots,n \end{array} \tag{1.66}$$

where K_{p_i} is the i-th P control gain, K_{d_i} is the i-th D control gain, θ_{d_i}, is the desirable angle of the i-th link, K_i is also a positive (constant) gain, and $w_i(x,t)$ is the deformation of the i-th link.

1.6.2.2 Stability Proof of Energy-Based Control for Flexible-Link Robots

The proposed control law of Eq. (1.66) assures the asymptotic stability of the closed-loop system in case of constant set-points (point to point control). The following Lyapunov (energy) function is considered [119],[445]:

$$V = E_{kin} + E_p + \frac{1}{2}\sum_{i=1}^{N} K_{p_i}[\theta_i(t) - \theta_{d_i}(t)]^2 + \frac{1}{2}\sum_{i=1}^{n} K_i[\int_0^t \dot{\theta}_i(s) w_i''(s,t)ds]^2 \quad (1.67)$$

where E_{kin} is given by Eq. (1.60) and denotes the kinetic energy of the robot's links, while E_p is given by Eq. (1.61) and denotes the potential energy of the robot's links due to deformation.

It holds that $V(t) > 0$ since $E_{kin} > 0$, $E_p > 0$, $\frac{1}{2}\sum_{i=1}^{n} K_{p_i}[\theta_i(t) - \theta_{d_i}(t)]^2 > 0$ and $\frac{1}{2}\sum_{i=1}^{n} K_i[\int_0^t \dot{\theta}_i(t) w_i(s,t)]^2 > 0$. Moreover, it holds that

$$\begin{gathered} \dot{V}(t) = \dot{E}_{kin} + \dot{E}_p + \sum_{i=1}^{n} K_{p_i}[\theta_i(t) - \theta_{d_i}(t)]\dot{\theta}_i(t) + \\ \frac{1}{2}\sum_{i=1}^{n} 2K_i[\int_0^t \dot{\theta}_i(s,t) w_i''(s,t)ds][\dot{\theta}_i(t) w_i''(x,t)] \end{gathered} \quad (1.68)$$

while using Eq. (1.63) and Eq. (1.65) the derivative of the robot's energy is found to be

$$\dot{E}_{kin}(t) + \dot{E}_p(t) = \sum_{i=1}^{n} T_i(t)\dot{\theta}_i(t) \quad (1.69)$$

where the torque generated by the *i*-th motor is given by Eq. (1.66). By substituting Eq. (1.69) and Eq. (1.66) in Eq. (1.68) one gets

$$\begin{gathered} \dot{V}(t) = -\sum_{i=1}^{n} K_{p_i}[\theta_i(t) - \theta_{d_i}(t)]\dot{\theta}_i(t) - \\ -\sum_{i=1}^{n} K_{d_i}\dot{\theta}_i^2(t) - \sum_{i=1}^{n}[K_i w_i''(x,t)\int_0^t \dot{\theta}_i(s) w_i''(s,t)ds]\dot{\theta}_i(t) \\ +\sum_{i=1}^{n} K_{p_i}[\theta_i(t) - \theta_{d_i}(t)]\dot{\theta}_i(t) + \sum_{i=1}^{n}[K_i w_i''(x,t)\int_0^t \dot{\theta}_i(s) w_i''(s,t)ds]\dot{\theta}_i(t) \end{gathered} \quad (1.70)$$

which finally results into,

$$\dot{V}(t) = -\sum_{i=1}^{n} K_{d_i}\dot{\theta}_i^2 \quad (1.71)$$

Obviously, from Eq. (1.71) it holds that $\dot{V}(t) \leq 0$, which implies stability of the closed-loop system, but not asymptotic stability. Asymptotic stability can be proved as follows [445]: If the *i*-th link did not converge to the desirable angle, i.e. $lim_{t\to\infty}\theta_i(t) = a_i \neq \theta_{d_i}(t)$ then the torque of the *i*-th motor would become equal to a small positive constant. This is easy to prove from Eq.(1.66) where the terms $K_{d_i}\dot{\theta}_i(t) = 0$, $K_i w_i(x,t)\int_0^t \dot{\theta}_i(s) w_i''(s,t)ds = 0$, while the term $K_{p_i}[\theta_i(t) - \theta_{d_i}(t)] = K_{p_i} a_i$ becomes equal to a positive constant.

However, if $T_i(t)=\text{constant}\neq 0$ then the i-th link should continue to rotate. This means that $\theta_i(t)\neq a_i$, which contradicts the initial assumption $lim_{t\to\infty}\theta_i(t)=a_i$. Therefore, it must hold $lim_{t\to\infty}T_i(t)=0$ and $lim_{t\to\infty}\theta_i(t)=\theta_{d_i}(t)$. Consequently, $lim_{t\to\infty}V(t)=0$.

The proposed energy-based controller is a decentralized controller since the control signals $T_i(t)$ of the i-th motor are calculated using only the angle $\theta_i(t)$ and the deformation $w_i(x,t)$ of the i-th link.

The performance of the previously analyzed model-free control methods (energy-based control and neural adaptive control) are compared to the performance of model-based techniques (inverse-dynamics control), in a simulation case study for planar 2-DOF manipulators.

1.6.3 Adaptive Neural Control of Flexible Manipulators

Adaptive neural network control of robotic manipulators has been extensively studied [118],[222]. Following [411] a method of neural adaptive control for flexible-link robots will be proposed.

Eq. (1.51) represents the dynamics of the flexible-link manipulator. It actually refers to a nonlinear transformation (mapping) from inputs (torques $T(t)$ generated by the motors) to outputs (motion of the joints). This nonlinear model can be written in the general form:

$$\ddot{\theta}=G(\theta,\dot{\theta},v,\dot{v},T(t)) \tag{1.72}$$

Consequently, the inverse dynamics of the flexible-link manipulator, is a relation that provides the torque that should be generated by the motors of the joints so as the joints angle, angular velocity and acceleration to take certain values. The inverse model of Eq. (1.72) is given by

$$T(t)=G^{-1}(\theta,\dot{\theta},\ddot{\theta},v,\dot{v}) \tag{1.73}$$

The dynamic model and its inverse are time dependent. If the inverse dynamic model of Eq. (1.73) can be explicitly calculated then a suitable control law for the flexible-link robot is available.

However, this model is not usually available and the system dynamics has to be adaptively identified. In this paper, a neural network model will be used to effectively approximate the inverse dynamical model of Eq. (1.73). Variables $\theta,\dot{\theta},\ddot{\theta}$ can be measured while variables $v,\dot{v}$ are non-measurable. Thus, the inverse dynamics of the manipulator can be decomposed into n sub-models given in the following form:

$$T(t)=G^{-1}(\theta,\dot{\theta},\ddot{\theta})=\begin{pmatrix} g_1^{-1}(\theta,\dot{\theta},\ddot{\theta}) \\ g_2^{-1}(\theta,\dot{\theta},\ddot{\theta}) \\ \cdots \\ g_n^{-1}(\theta,\dot{\theta},\ddot{\theta}) \end{pmatrix} \tag{1.74}$$

where each g_i^{-1}, $(i=1,2,\cdots,n)$ defines the inverse dynamics of the corresponding joint, while n is the number of joints of the manipulator.

A neural network can be employed to approximate each sub-model g_i^{-1} of the flexible robot's inverse dynamics. Therefore, the inverse dynamics of the overall system can be represented by a neural network $N(\theta,\dot{\theta},\ddot{\theta},w)$ [411]

$$T(t)\simeq N(\theta,\dot{\theta},\ddot{\theta},w)=\begin{pmatrix}N_1(\theta,\dot{\theta},\ddot{\theta},w_1)\\ N_2(\theta,\dot{\theta},\ddot{\theta},w_2)\\ \cdots \\ N_n(\theta,\dot{\theta},\ddot{\theta},w_n)\end{pmatrix} \tag{1.75}$$

where $N_i(\theta,\dot{\theta},\ddot{\theta},w_i)$ $i=1,2,\cdots,n$ is the i-th neural network that approximates the i-th sub-model of robot's inverse dynamics and w_i is the associated weights vector.

Using a NN to model the dynamics of the flexible-link robot provides a mapping of the joint angles vector to the motor torques vector. The torque appearing at the output of the neural network can be combined with a PD-feedback controller to generate the overall control signal that will finally drive the motors [422]. Therefore, the control scheme can be given as

$$T(t)=N(\theta,\dot{\theta},\ddot{\theta},w)+K_pe+K_d\dot{e} \tag{1.76}$$

where $N(\theta,\dot{\theta},\ddot{\theta},w)$ is the neural network approximation of the actual inverse dynamics of the manipulators, while $K_p \in R^{n\times m}$ and $K_d \in R^{n\times m}$ are the diagonal gain matrices with entries K_p and K_d, respectively, denoting a servo feedback that is introduced to stabilize the system. In Eq. (1.76) $e=\theta_d-\theta$, and $\dot{e}=\dot{\theta}_d-\dot{\theta}$ denote the position and velocity error of the robot's joints, respectively, and $\theta_d \in R^2$ is the vector of the position and velocity set-points.

The architecture of NN-based control of flexible-link robots is depicted in Fig. 1.5. Making use of Eq. (1.74) and Eq. (1.75), the neural network controller that is described by Eq. (1.76) becomes

$$N(\theta,\dot{\theta},\ddot{\theta},w)+K_pe+K_d\dot{e}=G^{-1}(\theta,\dot{\theta},\ddot{\theta}) \tag{1.77}$$

or equivalently,

$$K_pe+K_d\dot{e}=G^{-1}(\theta,\dot{\theta},\ddot{\theta})-N(\theta,\dot{\theta},\ddot{\theta},w)=\bar{N}(\theta,\dot{\theta},\ddot{\theta},w) \tag{1.78}$$

Eq. (1.78) represents a decoupled linear system, driven by the nonlinear vector function $\bar{N}(\theta,\dot{\theta},\ddot{\theta},w) \in R^n$. This function represents the error between the actual inverse dynamics $G^{-1}(\theta,\dot{\theta},\ddot{\theta})$ and its estimated model $N(\theta,\dot{\theta},\ddot{\theta},w)$ and can be written as

$$\bar{N}(\theta,\dot{\theta},\ddot{\theta})=\begin{pmatrix}\bar{N}_1(\theta,\dot{\theta},\ddot{\theta},w)\\ \cdots \\ \bar{N}_n(\theta,\dot{\theta},\ddot{\theta},w)\end{pmatrix}=\begin{pmatrix}g_1^{-1}(\theta,\dot{\theta},\ddot{\theta})-N_1(\theta,\dot{\theta},\ddot{\theta},w)\\ \cdots \\ g_n^{-1}(\theta,\dot{\theta},\ddot{\theta})-N_n(\theta,\dot{\theta},\ddot{\theta},w)\end{pmatrix} \tag{1.79}$$

Instead of using one neural network to approximate the inverse dynamics of the flexible-link robot one could use a separate network for each joint of the

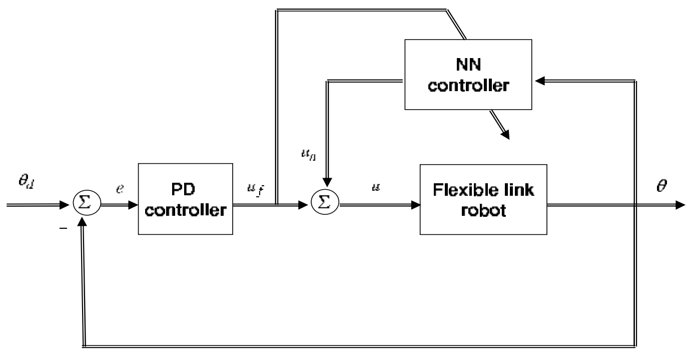


Fig. 1.5 Adaptive neural control of the flexible-link robot

manipulator. In that case and using Eq. (1.79), the error equation for the i-th joint of the manipulator becomes

$$K_p^i e_i + K_d^i \dot{e}_i = \bar{N}_i(\theta, \dot{\theta}, \ddot{\theta}, w), \ i = 1, 2, \cdots, n \tag{1.80}$$

where e_i and $\dot{e}_i$ denote the position and velocity errors of the i-th joint respectively, K_p^i and K_d^i are the proportional and derivative gains of the PD controller of the i-th joint, and $\bar{N}_i(\theta, \dot{\theta}, \ddot{\theta}, w)$ is the approximation error of the NN assigned to the i-th joint. The objective is to eliminate the approximation error, i.e. to succeed $lim_{t\to\infty}\bar{N}_i(\theta, \dot{\theta}, \ddot{\theta}, w) = 0$. In that case, a suitable selection of the control gains K_p^i and K_d^i results into

$$\begin{gathered} K_p^i e_i + K_d^i \dot{e}_i = 0 \Rightarrow \\ lim_{t\to\infty} e_i(t) = 0, \ lim_{t\to\infty} \dot{e}_i(t) = 0 \end{gathered} \tag{1.81}$$

Eq. (1.81) denotes that, the convergence condition for the closed-loop system is to make the error surface $\varepsilon_i = K_p^i e_i + K_d^i \dot{e}_i$ to approach zero. Then the suitable selection of the control gains will result in the asymptotic convergence of $e(t)$ to 0

$$K_p^i e_i(t) + K_d^i \dot{e}_i(t) = 0 \Rightarrow \dot{e}_i(t) = -\frac{K_p^i}{K_d^i} e_i(t) \Rightarrow e_i(t) = e_i(0) e^{-\frac{K_p^i}{K_d^i} t} \tag{1.82}$$

Thus a measure of the output error can be considered to be

$$\varepsilon_i = \bar{N}_i(\theta, \dot{\theta}, \ddot{\theta}, w) \tag{1.83}$$

which reflects the discrepancy between the actual inverse dynamics of the manipulator and its neural network approximation. The update of the weights of the neural network has to be carried out in such a way that the stability of the closed-loop system is maintained. To this end, the following cost function is defined for each joint

$$E_i(t) = \frac{1}{2}\varepsilon_i^2 \tag{1.84}$$

This cost function gives the squared distance of the error function $\varepsilon_i = K_p^i e_i(t) + K_d^i \dot{e}_i(t)$ from 0. The weights update algorithm is derived from the minimization of the cost function $E_i(t)$ over the weight space of the corresponding NN model.

1.6.4 Approximation of the Flexible-Links Dynamics

A diagonal recurrent neural network (RNN) is employed first to approximate the inverse dynamics of each link of the flexible manipulator. The structure of the i-th diagonal recurrent neural network is depicted in Fig. 1.6. The input vector of the RNN is $X = [\theta_i, \dot{\theta}_i, \ddot{\theta}_i]^T$, i.e. it consists of the position, angular velocity and acceleration of the robot's joints. The self-feedback connections at the hidden nodes of the RNN enable the neural network to approximate dynamical systems. On the other hand feed-forward neural networks without tapped delays, provide only a static input-output mapping [203]. Thus, RNN are deemed as more appropriate for modelling of the inverse dynamics of the flexible-link robot.

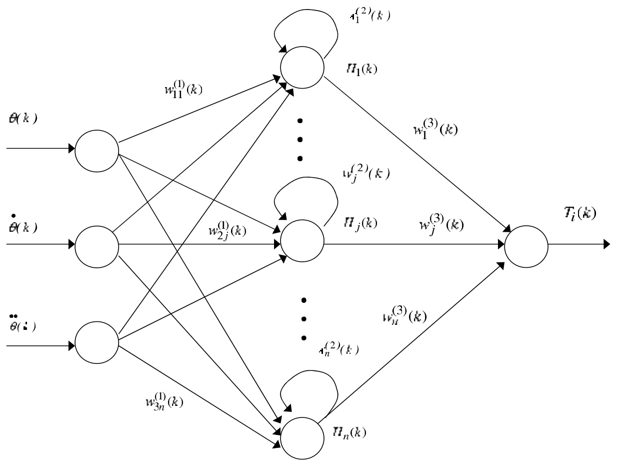


Fig. 1.6 Structure of the diagonal recurrent neural network

The weights update of the RNN consists of the forward and backwards computation. The forward computation is described as follows:

1. *Input layer*:

$$I_m(k) = x_m(k),\ m = 1,2,3 \tag{1.85}$$

where $[x_1, x_2, x_3] = [\theta_i, \dot{\theta}_i, \ddot{\theta}_i]$.

2. *Hidden layer*:

$$\begin{gathered} H_j(k) = F_j(u_j(k)) \\ u_j(k) = w_j^{(2)} H_j(k-1) + \textstyle\sum_m w_{mj}^{(1)} x_m(k) \end{gathered} \tag{1.86}$$

3. *Output layer*:

$$O(k) = N_i(\theta, \dot{\theta}, \ddot{\theta}, w) = \sum_j w_j^{(3)} H_j(k) \tag{1.87}$$

where $u_j(k)$ and $H_j(k)$ are the input and output of the hidden layer at the j-th unit, respectively, $O(k)$ is the output of the neural network (torque for the i-th joint), $F_j()$ is a sigmoid function, written as $F_j(u_j) = \frac{1}{1+e^{-u_j}}$, $w_{mj}^{(1)}$ is the weight that connects the m-th input layer neuron with the j-th hidden layer neuron, $w_j^{(2)}$ is the self-feedback of the j-th neuron of the hidden layer and $w_j^{(3)}$ is the weight that connects the j-th node of the hidden layer with the output layer node.

The update of the network weights is based on gradient descent [411]:

$$\frac{\partial E_i}{\partial w_j^{(3)}} = -\varepsilon_i(k) \frac{\partial O(k)}{\partial w_j^{(3)}} \tag{1.88}$$

$$\frac{\partial E_i}{\partial w_j^{(2)}} = -\varepsilon_i(k) \frac{\partial O(k)}{\partial H_j(k)} \frac{\partial H_j(k)}{\partial w_j^{(2)}} = -\varepsilon_i(k) w_j^{(3)} \delta_j^{(i)}(k) \tag{1.89}$$

$$\frac{\partial E_i}{\partial w_{mj}^{(1)}} = -\sum_j [\varepsilon_i(k) \frac{\partial O(k)}{\partial H_j(k)}] \frac{\partial H_j(k)}{\partial w_{mj}^{(1)}} = -\sum_j \varepsilon_i(k) w_j^{(3)} \beta_{mj}^{(i)}(k) \tag{1.90}$$

where index i denotes the neural network that models the inverse dynamics of the i-th flexible link

$$\delta_j^{(i)}(k) = \frac{\partial H_j(k)}{\partial w_j^{(2)}} = F'(u_j(k))[H_j(k-1) + w_j^{(2)} \delta_j^{(i)}(k-1)] \tag{1.91}$$

$$\beta_{mj}^{(i)}(k) = \frac{\partial H_j(k)}{\partial w_{mj}^{(1)}} = F'(u_j(k))[x_m(k) + w_j^{(2)} \beta_{mj}^{(i)}(k-1)] \tag{1.92}$$

with $F_j'(u_j) = F_j(u_j)(1 - F_j(u_j))$. The formula for the weights update is given by

$$w(k+1) = w(k) - \alpha \nabla E_i(w) \tag{1.93}$$

and the gradient $\nabla E_i(w)$ is given by

$$\nabla E_i(w) = [\frac{\partial E_i}{\partial w_{mj}^{(1)}}, \frac{\partial E_i}{\partial w_j^{(2)}}, \frac{\partial E_i}{\partial w_j^{(3)}}]^T \tag{1.94}$$

where $w = [w_{mj}^{(1)}, w_j^{(2)}, w_j^{(3)}]^T$ and α is the learning rate. From Eq. (1.83), (1.84) and (1.93) it can be seen that the parameters involved in the adaptation of the network's weights are: the output of the robot system $\theta, \dot{\theta}, \ddot{\theta}$, the current output of the PD controller and the learning rate α.

It is noted that, the neural-adaptive control scheme depicted in Fig. 1.5 is generic since in place of the diagonal recurrent neural network, a different neural network (such as a RBF neural network, which does not include self-feedback in the hidden layer neurons) can be considered and can perform well in the suppression of the flexible-links vibrations.

1.6.4.1 Local Stability Properties of Adaptive Neural Control

Local stability properties of the closed-loop system that consists of the flexible-link robot, the PD controller and the neural adaptive controller, will be investigated. The weights update law given in Eq. (1.93) can be suitably tuned so as to assure the stability of the closed-loop system. To this end, the following discrete-time Lyapunov function is defined

$$V_L(k) = \frac{1}{2}\varepsilon_i^2(k) \tag{1.95}$$

Thus, the change of the Lyapunov function due to the weights update is given by

$$\Delta V_L(k) = V_L(k+1) - V_L(k) = \frac{1}{2}[\varepsilon_i^2(k+1) - \varepsilon_i^2(k)] \tag{1.96}$$

The evolution of the modelling error ε_i in the discrete time is given by

$$\varepsilon_i(k+1) = \varepsilon_i(k) + \Delta\varepsilon_i(k) = \varepsilon_i(k) + [\frac{\partial E_i}{\partial w}]^T \Delta w \tag{1.97}$$

where Δw represents a change in an arbitrary weight vector. From the laws for weights update, given in Eq. (1.88) to Eq. (1.90) one obtains for the weights of the output layer

$$\Delta w = -\alpha \nabla E_i(w) = \alpha \varepsilon_i(k) \frac{\partial O(k)}{\partial w} \tag{1.98}$$

Based on the above, the following general convergence lemma has been stated [203]:

Lemma 1: Let α be the learning rate for the weights of the RNN and z_{max} be defined as $z_{max} = max||z(k)||^2$, where $z(k) = \frac{\partial O(k)}{\partial w}$ and $||\cdot||$ is the Euclidean norm in R^n. The convergence is guaranteed if α is chosen as

$$0 < \alpha < \frac{2}{z_{max}^2} \tag{1.99}$$

Using the previous Lemma the learning rate α that assures the convergence of the RNN weights update, can be found. The acceptable ranges of variation of the learning rate α are given in the following theorem [411]:

Theorem 1: Let α_3 be the learning rate for the RNN weights $w^{(3)}$. The dynamic back propagation algorithm converges if $0 < |w_j^{(3)}| < 1,\ (j = 1, 2, \cdots, h)$ and the learning rate α_3 is selected as

$$0 < \alpha_3 < \frac{2}{h} \tag{1.100}$$

where h is the number of recurrent neurons in the hidden layer.

Proof: From Eq. (1.88) it holds that

$$z(k) = \frac{\partial O(k)}{\partial w^{(3)}} = H(k) \tag{1.101}$$

where $H(k) = [h_1(k), h_2(k), ..., h_h(k)]^T$ and $h_j(k)$ is the output value of the j-th neuron in the hidden layer. Since $0 < h_j(k) < 1$ (due to the sigmoid function $F(x) = 1/(1+e^{-x})$) and using the definition of the Euclidean norm in R^h, it holds that $||z(k)|| < \sqrt{h}$ and $z_{max}(k) = h$. Then from Lemma 1, Eq. (1.100) can be obtained.

Similarly, the learning rates α_2 and α_1 for the weights $w^{(2)}$ and $w^{(1)}$ can be obtained

$$0 < \alpha_2 < \frac{1}{h}[\frac{1}{w_{max}^{(2)}}]^2 \tag{1.102}$$

$$0 < \alpha_1 < \frac{1}{h+3}[\frac{1}{w_{max}^{(1)} x_{max}}]^2 \tag{1.103}$$

where $w_{max}^{(2)} = max||w^{(2)}(k)||$.

1.7 Simulation of Flexible-Link Robot Control

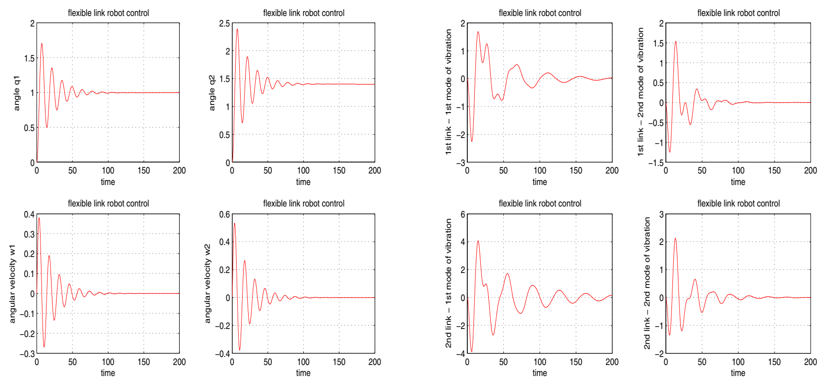
The performance of the previously analyzed model-free control methods (energy-based control and neural adaptive control) are compared to the performance of model-based techniques (inverse-dynamics control), in a simulation case study for planar 2-DOF manipulators, in contact-free operation.

1.7.1 Model-Based Control of Flexible-Link Robots

The 2-DOF flexible link robot of Fig. 1.3 is considered. The robot is planar and consists of two flexible links of length $L_1 = 0.45m$ and $L_2 = 0.45m$, respectively. The dynamic model of the robot is given by Eq. (1.23). The elements of the inertia matrix M are:

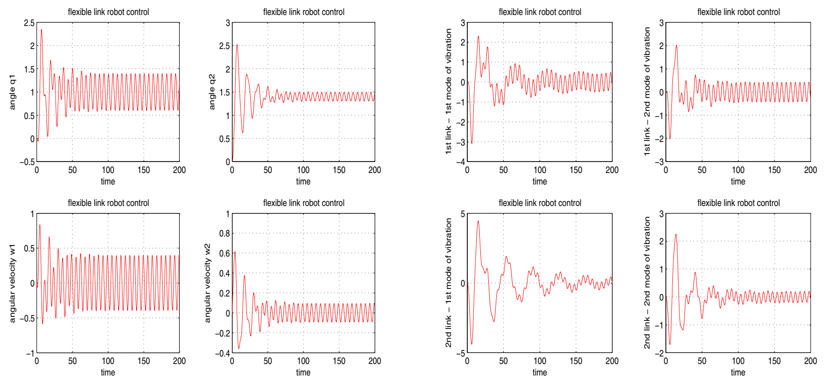
$$M_{11} = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}, \ M_{12} = M_{21}^T = \begin{pmatrix} 1 & 1 & 0.2 & 0.3 \\ 0.5 & 0.1 & 2 & 0.7 \end{pmatrix}, \ M_{22} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

The damping matrix was taken to be $D = diag\{0.04, 0.08, 0.03, 0.06\}$ while the stiffness matrices was selected as $K = diag\{0.02, 0.04, 0.03, 0.06\}$. The inverse dynamics control law given in Eq. (1.52) and Eq. (1.53) is employed. The selection of the gain matrices K_p and K_d determines the transient response of the closed loop system. The following controller gains have been considered: $K_p = diag\{0.2, 0.2\}$ and $K_d = diag\{0.1, 0.1\}$. The desirable joints positions are $\theta_{d_1} = 1$ rad and $\theta_{d_2} = 1.4$ rad. The performance of the model-based controller is given in Fig. 1.23.



(a) **(b)**

Fig. 1.7 Model-based control of a 2-link flexible robot (a) joints' angles and joints' angular velocity, (b) the first two vibration modes for each link



(a) **(b)**

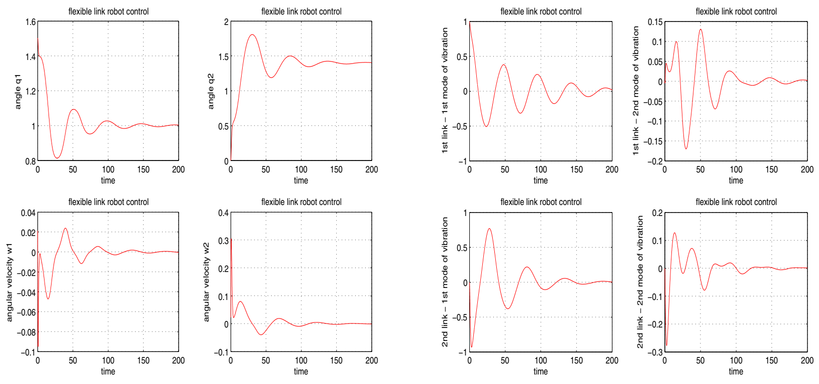
Fig. 1.8 Model-based control of a 2-link flexible robot in the presence of additive motor-torques disturbances (a) joints' angles and joints' angular velocity, (b) the first two vibration modes for each link

Moreover, it is considered that an additive disturbance torque appears on each joint. The disturbance is given by $d_i(t) = 0.3cos(t)$. The performance of the model-based controller of the flexible-link robot in the presence of disturbance is depicted in Fig. 1.8. It can be seen that vibrations around the desirable joint positions cannot be eliminated.

1.7.2 Energy-Based Control

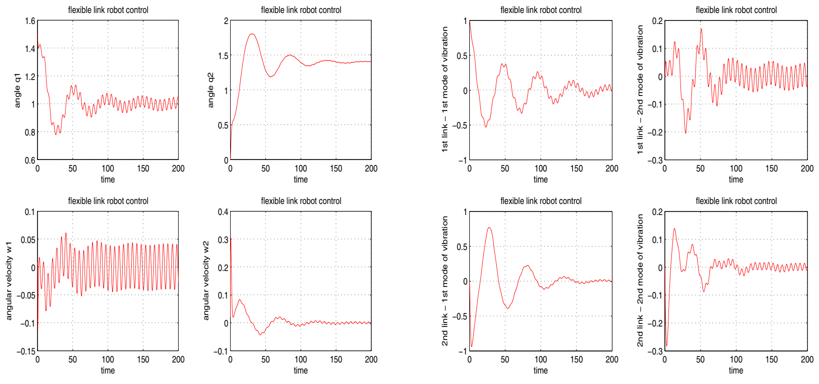
The same robotic model as in Subsection 1.7.1 is used to simulate the variation of the manipulator's joints with respect to time. Energy-based control of flexible-link robots is based on Eq. (1.66). The following controller gains have been used: $K_p = diag\{1.9, 5.6\}, K_d = diag\{7.2, 23.3\}$ and $K_i = diag\{0.1, 0.1\}$. The desirable joint positions are again $\theta_{d_1} = 1.0\ rad$ and $\theta_{d_2} = 1.4\ rad$. To derive the control signal of Eq. (1.66) the strains at the base of each link were used, i.e. $w_i''(0,t)$. The performance of the energy-based controller in the case of the 2-DOF flexible link robot is shown in Fig. 1.9.

Moreover, the performance of the energy-based controller in presence of the external disturbances of Subsection 1.7.1 is given in Fig. 1.10. Suppression of the



(a) **(b)**

Fig. 1.9 Energy-based control of a 2-link flexible robot (a) joints' angles and joints' angular velocity, (b) the first two vibration modes for each link



(a) **(b)**

Fig. 1.10 Energy-based control of a 2-link flexible robot in the presence of additive motor-torques disturbances (a) joints' angles and joints' angular velocity, (b) the first two vibration modes for each link

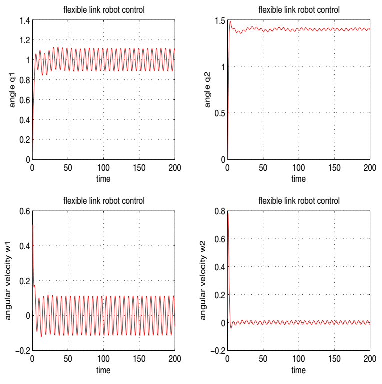
vibrations can be achieved if the elements of the gain matrix K_d are given higher values.

1.7.3 Adaptive Neural Control

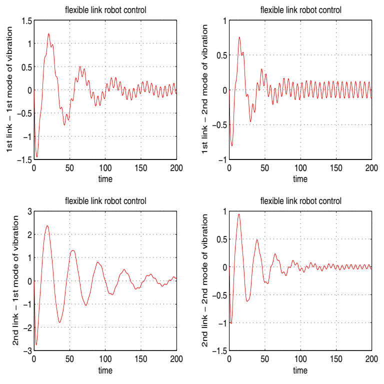
The control scheme shown in Fig. 1.5 was implemented. A diagonal recurrent neural network with 5 nodes at the hidden layer was first used. The following proportional-derivative (PD) controller gains have been chosen: $K_p = diag\{1.4, 5.6\}$ and $K_d = diag\{2.7, 6.8\}$. The additive motor-torques disturbance of Subsection 1.7.1 were considered. The performance of a PD controller without additional torque from the neural adaptive controller is depicted in Fig. 1.11.

To compensate for the external disturbances the neural adaptive controller of Fig. 1.5 was used. The simulation results are depicted in Fig. 1.13. It can be observed that the neural adaptive controller achieves disturbance rejection and results in suppression of the vibrations of the flexible links.

The flexible-link robot control was also implemented with the use of a Radial Basis Function (RBF) neural network. The hidden layer of the NN contained 5 nodes with Gaussian Basis functions, and unlike the diagonal recurrent neural network no self-feedback in the hidden nodes was considered. The additive motor-torques disturbance of Subsection 1.7.1 were considered. To compensate for the external disturbances the neural adaptive controller of Fig. 1.5 was used, but this time using a RBF neural network. The simulation results are depicted in Fig. 1.13. It can be observed that the RBF-based neural adaptive controller succeeds suppression of the vibrations of the flexible links and achieves also convergence of the robot's joints to the desirable set-points.

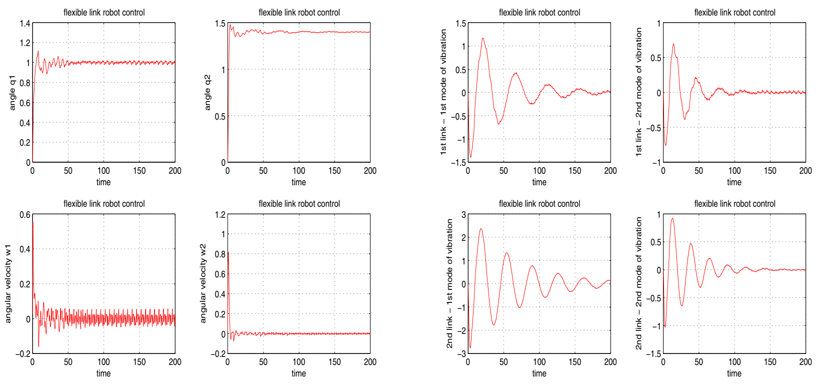


(a)



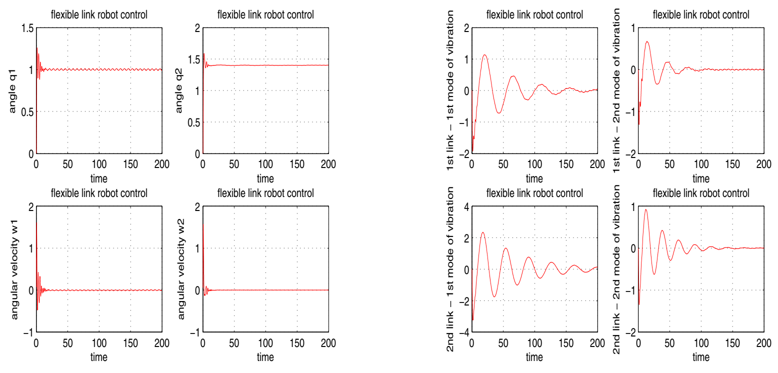
(b)

Fig. 1.11 PD control of a 2-link flexible robot without additional torque from the neural adaptive controller (a) joints' angles and joints' angular velocity, (b) the first two vibration modes for each link



(a) **(b)**

Fig. 1.12 DRNN-based neural network control of a 2-link flexible robot in the presence of additive motor-torques disturbances (a) joints' angles and joints' angular velocity, (b) the first two vibration modes for each link



(a) **(b)**

Fig. 1.13 RBF-based neural network control of a 2-link flexible robot in the presence of additive motor-torques disturbances (a) joints' angles and joints' angular velocity, (b) the first two vibration modes for each link

The ISE (integral of the square error) criterion $\int_0^T e^2(t)dt$ of the examined control loops is also calculated assuming for all controllers the same proportional-derivative controller gains $K_p = \mathrm{diag}\{1.2, 1.5\}$ and $K_d = \mathrm{diag}\{2.4, 1.8\}$. The results are summarized in Table 1.2. It can be observed that the examined model-free controllers of the flexible-link robot (i.e. the energy-based controller, the DRNN and RBF neural-adaptive controllers) are equally effective to (or in certain cases outperform)

Table 1.2 ISE tracking performance of the control loops

parameter	θ_1	θ_2	$\dot{\theta}_1$	$\dot{\theta}_2$
inverse-dynamics control	130.31	173.79	31.89	82.41
energy-based control	104.36	180.64	24.83	91.62
DRNN adaptive neural control	88.68	163.76	33.82	85.11
RBF adaptive neural control	108.96	152.97	30.92	100.11

the inverse-dynamics model-based controller. Moreover, it should be taken into account that the model-free controllers use less a-priori known information about the flexible-link robot's dynamics than the inverse-dynamics controller.

Chapter 2
Industrial Robots in Compliance Tasks

Abstract. An analysis of industrial robot control is given, for the case of compliance tasks. First, rigid-link robotic models are considered and the impedance control and hybrid position-force control methods are analyzed. Next, force control methods are generalized in the case of flexible-link robots performing compliance tasks.

2.1 Impedance Control

Up to now the study of control methods for flexible-link robots followed the assumption that the robot operates in the free space. However, when in contact with a surface, forces are exerted to the robot's end-effector and a significant issue that has to be taken into account in the design of the robotic controller is force control [186],[187]. To solve the force control problem, the kinematic model of the flexible-link robot described in Chapter 1 will be used. In the case of compliance tasks, the objective is not only to control the end effector's position but also to regulate the force developed due to contact to the processed surface [189],[190]. There are established approaches for simultaneous position and force control of robotic manipulators which were initially designed for rigid-link robots and which were subsequently extended to flexible-link robots.

Tasks like grinding, milling, polishing or assembling need the control of the force that the end-effector exerts on the workpieces, as well as the control of its position. The impedance control method is used for simultaneous control of both the position and the force of the robotic manipulator. The implementation of impedance control can be understood through an example concerning the deburring process (see Fig. 2.1). The aim of the deburring process is to remove the burrs from a rough surface. The deburring quality depends on

1. the rotary speed of the grinding tool ω_r
2. the grinding tools penetration depth x
3. the grinding tools velocity parallel to the metals surface $\dot{y}$.

The above variables can be represented by the command signal C_d

$$C_d = w_\omega \omega_r + w_x x + w_y \dot{y} \tag{2.1}$$

which is equivalent to the force that has to be applied on the metal's surface (contact force) in order to succeed removal of all burrs and perform perfect smoothing of the surface. Thus, the deburring problem initially aiming at finding the values of ω_r, x, and $\dot{y}$ which assure perfect smoothing of the surface, can now be formulated as aiming at finding the contact force C_d^* which assures perfect smoothing of the surface.

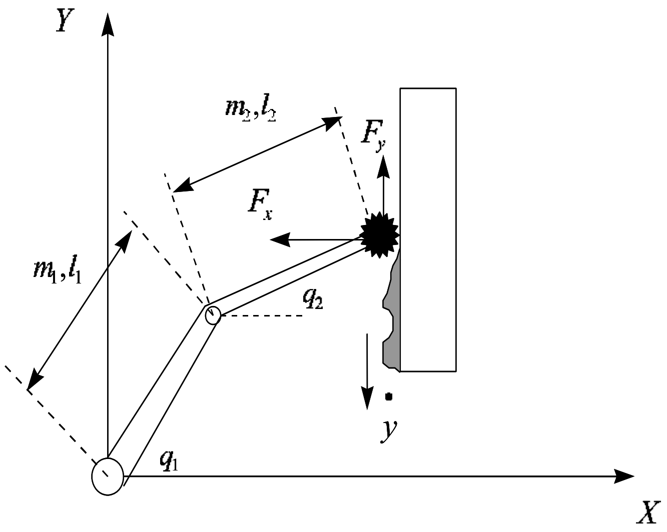


Fig. 2.1 Contact forces to a robot's end effector while performing a deburring task

The reaction force which is applied by the surface on the robot's end effector is described by the relation (see [204],[210])

$$F_e = F_n + F_j \tag{2.2}$$

F_f is the tangential reaction force (i.e., the friction) and F_n is the perpendicular to the metal surface reaction force. The analytic expressions of F_f and F_n are

$$F_n = K_e \frac{ax + by + c}{\sqrt{a^2 + b^2}} \quad F_f = \rho F_n \tag{2.3}$$

ρ friction coefficient and $|F_e| = \sqrt{F_n^2 + F_f^2}$, K_e spring stiffness coefficient, a,b,c parameters related to surface curvature. Every force setpoint C_d can be mapped to a setpoint (x_d, y_d) such that the condition $|C_d| = |F_n|$ is satisfied, i.e.,

$$C_d = K_e \frac{a}{\sqrt{a^2+b^2}} x_d + K_e \frac{b}{\sqrt{a^2+b^2}} y_d + K_e \frac{c}{\sqrt{a^2+b^2}}$$
$$\text{i.e. } C_d = K_x x_d + K_y y_d + K_c \tag{2.4}$$

To calculate the setpoint (x_d, y_d) which corresponds to the command force C_d the following steps are performed: 1) parameters K_e, x_d, and y_d are initialized arbitrarily and 2) a continuous update of the above parameters with a gradient algorithm follows.

The Euler-Lagrange joint-space equation of a n-DOF robot is

$$D(\theta)\ddot{\theta} + B(\theta,\dot{\theta}) + G(\theta) = \tau + J^T F_e + \tau_d \tag{2.5}$$

$q \in R^{n\times 1}$ is the joint angles vector, $D(q) \in R^{n\times n}$ is the inertia matrix, $B(q,\dot{q}) \in R^{n\times 1}$ is the vector of the Coriolis and centrifugal forces, $G(q) \in R^{n\times 1}$ is the gravity force vector, τ is the torque vector, J is the Jacobian matrix of the robot, F_e is the vector of the reaction contact forces applied to end-effector, and $\tau_d \in R^{n\times 1}$ is the vector of the external disturbances.

Unifying $B(\theta,\dot{\theta}), G(\theta)$ in $h(\theta,\dot{\theta}) = B(\theta,\dot{\theta}) + G(\theta)$, one gets

$$D(\theta)\ddot{\theta} + h(\theta,\dot{\theta}) = \tau + J^T F + \tau_d. \tag{2.6}$$

It is also possible to express the robot dynamic equation in a Cartesian coordinates system [204],[210]. In this approach, X denotes the position and orientation of the

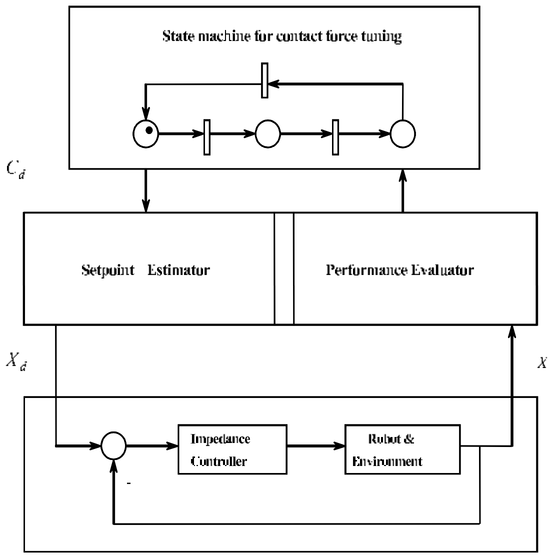


Fig. 2.2 Impedance controller for robotic deburring

end-effector, $\dot{X}$ denotes its linear velocity, and $\ddot{X}$ denotes its linear acceleration. Using the relations $\dot{X} = J(\theta)\dot{\theta}$, $\ddot{X} = \dot{J}\dot{\theta} + J\ddot{\theta}$, and $\tau = J^T F$ the robots dynamic equation in Cartesian coordinates is found to be

$$\tilde{D}\ddot{X} + \tilde{h} = F + F_e + F_d \tag{2.7}$$

where $\tilde{D} = J^{-T}DJ^{-1}$,$\tilde{h} = J^{-T}h - \tilde{D}\dot{J}\dot{q}$,$F_e$ is the reaction force applied to the end-effector, and F_d is the disturbance force vector.

When the robots end-effector comes in contact with the surface, the reaction force F_e applied to the end-effector is expressed by a generalized spring model [204],[210]

$$F_e = A\delta\ddot{X} + B\delta\ddot{X} + K\delta X \tag{2.8}$$

where $\delta X = X_d - X$ displacement from the desired position, K stiffness matrix (diagonal to achieve static decoupling). Here, X_d is intentionally designed so that X will not be able to reach X_d in the steady state $(X \neq X_d)$ and is known as virtual commanded position. The matrices A, B, and K are positive-definite and influence the stability and the transient behavior of position tracking. The concept of impedance control is to find a control law that will transform equation 2.7 into equation 2.8. Assuming no external disturbance, an appropriate impedance control law is [204],[210]

$$F = \tilde{D}A^{-1}[B\delta\dot{X} + K\delta X - F_e] + \tilde{D}\ddot{X}_d + \tilde{h} + (A\delta\ddot{X} + B\delta\dot{X} + K\delta X) \tag{2.9}$$

Using the impedance control law equation of Eq. (2.9), one succeeds in transforming the Cartesian dynamic model of Eq. (2.7) of the robot to the equivalent form of Eq. (2.8). If the matrices A,B, and K in Eq. (2.8) are appropriately selected such that the polynomial $Z(s) = As^2 + Bs + K$ to be Hurwitz stable (to have poles in the left complex semiplane), then one can succeed tracking of the desirable force profile.

2.2 Hybrid Position/Force Control

Apart from impedance control, hybrid position/force control, is an approach for handling compliance tasks. The milling process will be used as an example of the hybrid position/force control applications to industrial systems. The hybrid controller is based on the idea of weighting the joint torque's components which are responsible for the control of the robot's degrees of freedom (position, force), via the selection matrix S. Hybrid position/force control is an expansion of the computed torque control method [242],[314],[457],[466]. The computed torque method, in combination with the conversion of joint control signals to the end-effector coordinates and the selection matrix S, gives the following equation for the hybrid position/force controller:

$$\begin{aligned}\tau = {} & D(\theta)J^{-1}[(I-S)u_x - \dot{J}\dot{\theta}] + h(\theta,\dot{\theta}) \\ & + h(\theta,\dot{\theta}) + g(\theta) + F_{jc} + J^T f + J^T S u_f\end{aligned} \tag{2.10}$$

where $D(\theta)$ is the inertia matrix, J is the robot Jacobian, θ is the vector of joint angles, $h(\theta,\dot{\theta})$ is the vector of Coriolis and centrifugal forces, g is the vector of gravity forces, F_{jc} is the Coulomb friction of the robot joints, f is the force applied to the environment and u_x, u_f are the position and force control signals respectively. The selection matrix S consists of unit and zero elements on the diagonal depending on the control way (position or force) of the respective degree of freedom. The control signals u_x and u_f are provided by controllers which may be conventional or some kind of adaptive controllers.

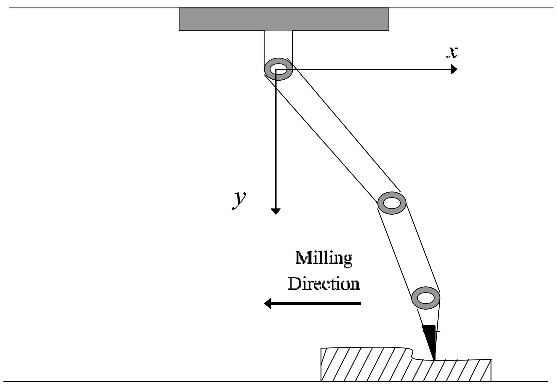


Fig. 2.3 Robotic milling

This architecture contains two independent control loops, one of which performs the position control while the other performs the force control. The methodology does not introduce any new control law but solves the problem of cooperation of the two independent control loops in several compliance tasks [108].

2.2.1 Stiffness Identification in Compliance Tasks

In the milling process the knowledge of the materials' stiffness is necessary. The evaluation of the stiffness, except from its real value, is a function of the sharpness of the tool which applies the force to the object. So, if a force is exerted to an object with a tool of low sharpness the depth obtained will be smaller than that achieved if the tool is more pointed. So the computed stiffness will be different when the manipulator uses another milling tool. For this reason stiffness identification algorithms are needed, and can be performed by the manipulator to useless experimental piece of the material, before the actual milling process starts [108],[183].

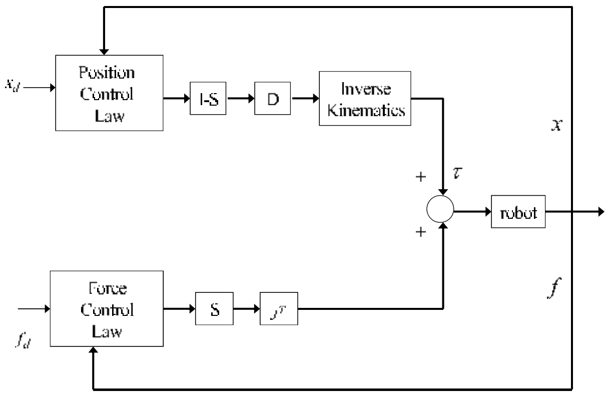


Fig. 2.4 Hybrid positiobforce controller

To this end, a stiffness identification method which is based on a gradient algorithm and the hybrid position/force control method will be presented. The goal of this method is to achieve a desirable depth by changing the set-point of the force controller, while the manipulator is not moved to the other degrees of freedom. Let k_n be an arbitrary, initial value for the stiffness of the material to be milled, smaller than the real stiffness and $d_{\max}$ a desirable depth. Initially a force set-point is calculated using the equation:

$$f_d = k_n d_{\max} \tag{2.11}$$

When the force controller succeeds convergence to this set-point, then the tool is in real depth d_r. This real depth satisfies the equation:

$$f_d = k_r d_r \tag{2.12}$$

where k_r is the real stiffness of the object. From Eq. (2.12) and Eq. (2.11) the relation of d_r and $d_{\max}$ is found to be:

$$d_r = (\frac{k_n}{k_r})d_{\max} \tag{2.13}$$

The error e_d between the desirable and real depth is:

$$e_d = d_{\max} - d_r \tag{2.14}$$

Now the error cost function $J = \frac{1}{2}e_d^2$ is defined. The aim of the gradient algorithm is to minimize J. Differentiating J with respect to $k_n$and applying the chain rule gives:

$$\begin{aligned} \partial J/\partial k_n = (\partial J/\partial e_d)(\partial e_d/\partial k_n) = \\ e_d \partial(d_{\max} - \tfrac{k_n}{k_r} d_{\max})/\partial k_n = \tfrac{-e_d d_{\max}}{k_r} \end{aligned} \tag{2.15}$$

The stiffness identification algorithm can not use k_r from Eq. (2.15) because it is unknown. Replacing the ratio $d_{\max}/k_r$ in Eq. (2.15) by its value obtained from Eq. 2.13 yields :

$$\partial J/\partial k_n = -\frac{e_d d_r}{k_n} \tag{2.16}$$

Therefore the updating equation of the object's stiffness is:

$$k_n(\mathrm{k}+1) = k_n(\mathrm{k}) + \eta \; e_d \; d_r/k_n \tag{2.17}$$

where η is the renewal rate. The implementation of the stiffness identification algorithm consists of the following five steps:

Step 1: Initialize the unknown stiffness to an arbitrary value and define the desired depth.

Step 2: Calculate the force set-point based on the present stiffness value and the desired depth.

Step 3: Wait until the force controller succeeds convergence to the desired force set-point.

Step 4: If the desired depth has been achieved go to step 5, otherwise compute the new stiffness value using Eq. (2.17) and go to step 2.

Step 5 : The stiffness has been identified and is equal to the present stiffness value obtained from Eq. (2.17).

2.2.2 Application of Robot Hybrid Position/Force Control

The milling process is widely met in metal industry and is usually carried out by appropriate CNC machines. However, when the surface of the metallic object is of varying curvature (e.g. ship propellers) it might be difficult to tune and use a conventional milling machine. Therefore when dextrous manipulations are required, robot-based milling can be more suitable.

In this section the implementation of the robot hybrid position/force control on the milling process by a desired depth curve will be presented. As desired depth curve the $\sin^2$ function is selected. The milling process will be performed at 10cm width interval on the object and will have four peaks in this interval. The shape of this curve is shown in Fig. 2.5.

The milling process will be performed at an horizontal velocity of 2 cm/sec. It will be assumed that in the horizontal motion there is friction between the tool and the object which is opposite to the motion. This friction will be assumed proportional to the real depth by a coefficient value 1500 N/m giving a 3 Nt maximum friction value at the maximum depth of 2 mm.

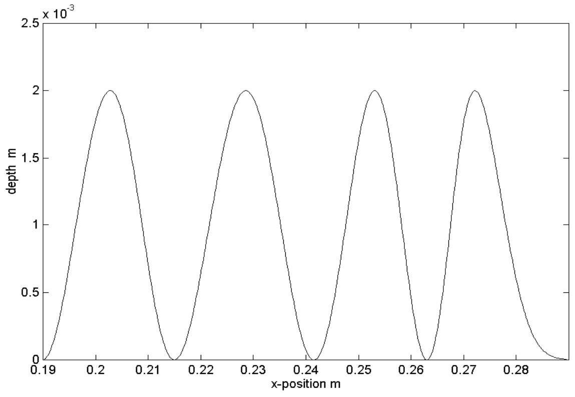


Fig. 2.5 Desired depth curve for the milling process

The conventional resolved acceleration hybrid control method and the neuro-fuzzy hybrid control were applied using a 3-DOF robot manipulator. The robot has the following parameters: $m_1 = 5kg$, $m_2 = 5kg$, $m_3 = 2.5kg$ and $l_1 = 30cm$, $l_2 = 30cm$ and $l3 = 15cm$ respectively. The dynamic model of this robot is:

$$\tau = D(\theta)\ddot{\theta} + h(\theta,\dot{\theta}) + g(\theta) + F_{jc} + J^T F \tag{2.18}$$

The Jacobian of the manipulator is:

$$J = \begin{pmatrix} -l_3S_{123} - l_2S_{12} - l_1S_1 & -l_3S_{123} - l_2S_{12} & -l_3S_{123} \\ l_3C_{123} + l_2C_{12} + l_1C_1 & l_3C_{123} + l_2C_{12} & l_3C_{123} \\ 1 & 1 & 1 \end{pmatrix} \tag{2.19}$$

where $S_i = \sin(\theta_i)$,$S_{ji} = \sin(\theta_i + \theta_j)$, $C_i = \cos(\theta_i)$, $C_{ji} = \cos(\theta_i + \theta_j)$ and θ_i is the angle between the links i and $i+1$.

The parameters (gains) of the controller are selected as: a) Angle Controller: $k_{p\varphi} = 700$, $k_{v\varphi} = 1500$, b) Position controller: $k_{px} = 15$, $k_{vx} = 3.5$, c) Force Controlle: $k_{pf} = 15$, $k_{vf} = 5000$.

From the requirement the tool to be vertical to the object during the milling process the desired angle set–point is found to be 90^o. The aim of the horizontal position controller is to move the tool with a desired velocity, and the force controller has to apply the desired force to the object. The trials were performed at desired horizontal velocities 4 cm/sec and 2 cm/sec and with several kinds of material and forces. It was also assumed that there is friction between the tool and the object which is proportional to the real depth by a coefficient with value 1500. The motion started from the point $x = 0.29027$m and directed opposite to the x axis, while the tool was vertical to the object from the beginning. The results are shown in Fig. 2.6.

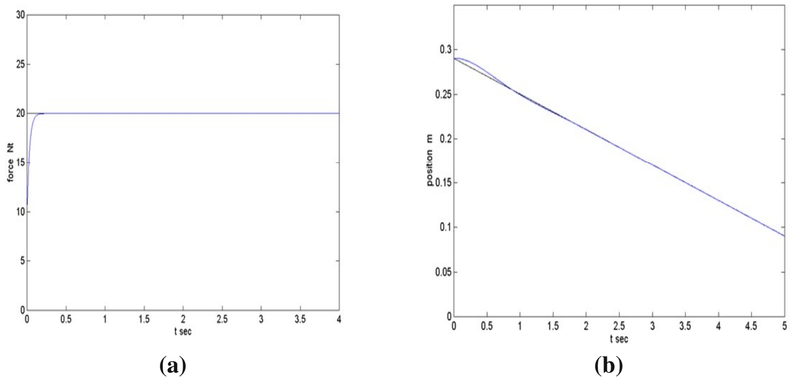


Fig. 2.6 Stiffness 10000 Nt/m, desired force 20Nt and horizontal movement by 4cm/sec: (a) force control, (b) position control

2.3 Force Control of Flexible-Link Robots

2.3.1 Interaction with the Compliant Surface

The flexible-link robot model of Eq. (1.37) is now considered. A simple model of elastic force due to contact of the end-effector with a surface is given by:

$$F = K_e \eta \eta^T (p - p_e) \tag{2.20}$$

where $p = k(\theta, v)$ are the coordinates of the end-effector which are calculated from the kinematic model, and η is a vector normal to the surface p_e. From the second line of the dynamic model of Eq. (1.37) one obtains:

$$M_{21}\ddot{\theta} + M_{22}\ddot{v} + D\dot{v} + Kv = -J_v^T(\theta, v)F \tag{2.21}$$

In the steady-state one obtains

$$\begin{gathered} v = -K^{-1}J_v^T(\theta,v)F - K^{-1}J_v^T(\theta,v)\eta K_e(p_n - p_{en}) \Rightarrow \\ v = -K^{-1}J_{vn}K_e(p_n - p_{en}) \end{gathered} \tag{2.22}$$

where $p_n = \eta^T p$, $p_{e_n} = \eta^T p_e$, and $J_{v_n} = J_v^T \eta$. The derivative of Eq. (2.22) with respect to time t is calculated.

$$\begin{gathered} \dot{v} = \frac{\partial v}{\partial \theta}\frac{\partial \theta}{\partial t} = \frac{\partial}{\partial \theta}\{-K^{-1}J_{v_n}K_e(p_n - p_{en})\}\dot{\theta} \Rightarrow \\ \dot{v} = -K^{-1}\frac{\partial J_{v_n}}{\partial \theta}K_e(p_n - p_{en}) + K^{-1}J_{v_n}K_e\frac{\partial p_{en}}{\partial \theta}\dot{\theta} \end{gathered} \tag{2.23}$$

which finally results into

$$\dot{v} = -K^{-1}K_e J_f(\theta)\dot{\theta} \tag{2.24}$$

with $J_f(\theta) = \frac{\partial J_{vn}}{\partial \theta} K_e(p_n - p_{en}) + K^{-1} J_{v_n} K_e \frac{\partial p_{en}}{\partial \theta}$. Substituting Eq. (2.24) into Eq. (1.36) gives:

$$\begin{aligned} \dot{p} &= J_\theta(\theta, v)\dot{\theta} + J_v(\theta, v)\{-K^{-1} K_e J_f(\theta)\} \Rightarrow \\ \dot{p} &= \{J_\theta(\theta, v) - K^{-1} K_e J_v(\theta, v) J_f(\theta)\} \end{aligned} \tag{2.25}$$

The overall Jacobian matrix J_p is defined as:

$$J_p = J_\theta(\theta, v) - K^{-1} K_e J_v(\theta, v) J_f(\theta) \tag{2.26}$$

which relates the velocity of the end-effector with the angular velocity of the joints

$$\dot{p} = J_p(\theta, v)\dot{\theta} \tag{2.27}$$

2.3.2 Force Control for Flexible-Link Robots

The desirable contact force along the normal vector of surface p_e is denoted as F_d and corresponds to the desirable position p_d. The relation between F_d and p_d is given by

$$p_{d_n} = \eta^T p_d = K_e^{-1} F_d + p_{e_n} \tag{2.28}$$

or equivalently $p_{d_n} - p_{e_n} = K_e^{-1} F_d \Rightarrow \eta^T p_d - \eta^T p_e = K_e^{-1} F_d$. Thus to succeed contact force equal to F_d the end-effector should reach the depth $\eta^T p_d - \eta^T p_e$. The design of the force controller comprises the following steps [380]:

1. For a certain force set-point F_d the corresponding position of the end-effector is calculated using Eq. (2.28).
2. Knowing p_d an inverse kinematics algorithm is used to calculate the associated joint angles θ_d and the vibration modes v_d.
3. The values of θ_d and v_d are used as set-points of a simple proportional-derivative joint controller, as the ones described in the previous sections.

The inverse kinematics problem can be solved with the use of an inverse kinematics algorithm which enables the calculation of θ_d and v_d through the convergence of the closed-loop system:

$$\dot{\theta} = J_P^T(\theta, v) K_p (p_d - p) \tag{2.29}$$

where J_p is the Jacobian of Eq. (2.26), p is the current position of the end-effector, p_d is the desirable position of the end-effector, and K_p is the diagonal feedback matrix of Eq. (2.29). The convergence conditions of the inverse kinematics algorithm have been studied [380]. The calculated values θ_d and v_d which are associated with the desirable position p_d are introduced as set-points in the PD controller of each link. This is given in:

$$T(t) = K_1(\theta_d - \theta) + K_2\dot{\theta} + J_\theta^T(\theta_d, v_d) F_d n \tag{2.30}$$

where $F_dn = K_e[\eta^T p_d - \eta^T p_e]\eta$ and J_θ is the Jacobian of Eq. (1.40). The term $J_\theta^T F_dn$ is added to the control signal to compensate for the torques which are induced to the joints due to the contact forces.

The discrete-time solution of the inverse kinematics gives

$$\theta_d(k+1) = \theta_d(k) + T_s J_P^T(\theta_d(k), v_d(k))K_p[p_d(k) - p(k)] \tag{2.31}$$

where the Jacobian J_p is given by Eq.(2.26), and T_s is the sampling period. From Eq. (2.24), one obtains for the normal vibration modes,

$$v_d(k+1) = -K^{-1}K_e J_{v_n}(\theta_d(k))[p_n(k) - p_{e_n}(k)] \tag{2.32}$$

with $p_n = \eta^T p$, $p_{e_n} = \eta^T p_e$ and $J_{v_n} = J_v^T \eta$.

2.4 Simulation of Force Control for Flexible-Link Robots

In the previous example it was assumed that the flexible-link robot operates in free space. In the sequel it will be assumed that the manipulator's end effector contacts a metallic surface of stiffness K_e and is thus subject to compliance forces.

The 2-DOF flexible manipulator of Fig. 2.7 is considered. The first link of the robot is rigid and its length is $L_1 = 0.45m$ while the second link is flexible and its length is $L_2 = 0.38m$. The angle setpoints for the two joints are $\theta_{d_1} = 1\ rad$ and $\theta_{d_2} = 0.4\ rad$. The elements of the inertia matrix M are:

$$M_{11} = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix},\ M_{12} = M_{21}^T = \begin{pmatrix} 1.2 & 1.3 \\ 2.0 & 1.7 \end{pmatrix},\ M_{22} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Moreover, the damping and stiffness matrices are $D = diag\{0.03, 0.06\}$ and $K = diag\{0.03, 0.06\}$, respectively. The constraint surface is described as $\Omega(X_1^{'}, Y_1^{'}) = 0$. The position of the end-effector (X_M, Y_M) can be expressed in terms of the joints angles θ_1 and θ_2

$$\begin{aligned} X_M &= L_1 cos(\theta_1) + L_2 cos(\theta_1 + \theta_2) - w(L_2, t) sin(\theta_1 + \theta_2) \\ Y_M &= L_1 sin(\theta_1) + L_2 sin(\theta_1 + \theta_2) + w(L_2, t) cos(\theta_1 + \theta_2) \end{aligned} \tag{2.33}$$

where $w(L_2, t)$ is the deformation of the flexible link at $x = L_2$. The dynamic model of the manipulator, initially given in Eq. (1.23) can now be expressed as

$$\begin{pmatrix} M_{11}(z) & M_{12}(z) \\ M_{21}(z) & M_{22}(z) \end{pmatrix} \begin{pmatrix} \ddot{\theta} \\ \ddot{v} \end{pmatrix} + \begin{pmatrix} F_1(z,\dot{z}) \\ F_2(z,\dot{z}) \end{pmatrix} + \begin{pmatrix} 0_{2\times2} & 0_{2\times2} \\ 0_{2\times2} & D(z) \end{pmatrix} \begin{pmatrix} \dot{\theta} \\ \dot{v} \end{pmatrix} + \begin{pmatrix} 0_{2\times2} & 0_{2\times2} \\ 0_{2\times2} & K(z) \end{pmatrix} \begin{pmatrix} \theta \\ v \end{pmatrix} = \begin{pmatrix} T(t) \\ 0_{2\times1} \end{pmatrix} + \begin{pmatrix} J^T F \\ 0_{2\times1} \end{pmatrix} \tag{2.34}$$

In Eq. (2.34) $T(t)$ is the vector of the motor torques, J^T is the Jacobian of the manipulator with respect to the joint angles $\theta_i,\ i = 1,2$ and $F = [F_x, F_y]^T$ is the

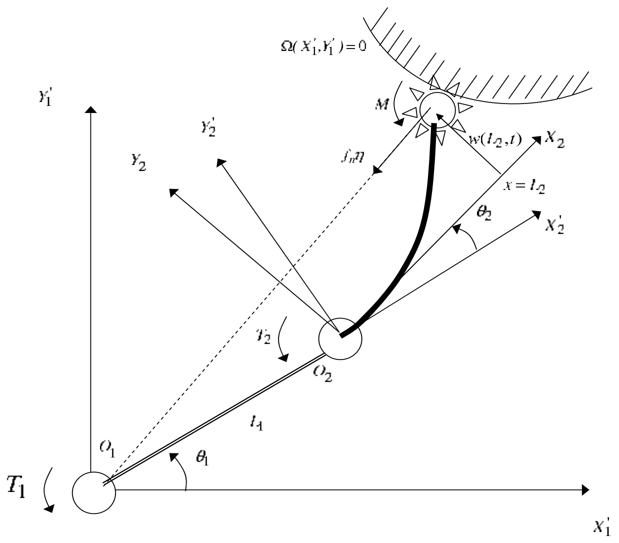


Fig. 2.7 A flexible-link robot which operates in the presence of compliance forces

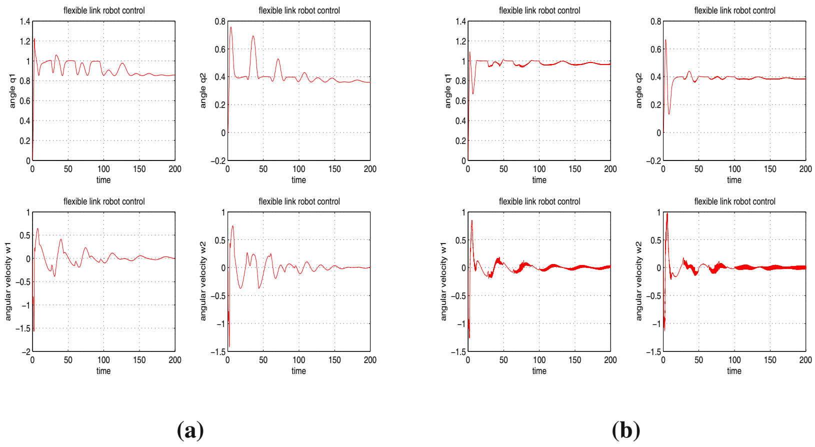


Fig. 2.8 Control of a 2-link flexible robot in the presence of disturbances induced by compliance forces (a) joints' angles and joints' angular velocity when only a PD controller was used, (b) joints angle and joints angular velocity when an adaptive neural controller was included in the control-loop

vector of the compliance forces which are applied to the robot's end-effector due to contact with the constraint surface $\Omega(X_1', Y_1') : X + Y - 1 = 0$. In the above dynamic model it has been assumed that the Jacobian with respect to the vibration modes is 0 and thus the impact of elastic reaction forces can be neglected.

From Eq. (2.33) the calculation of the Jacobian J with respect to the joint angles $\theta_i,\ i = 1,2$ is straightforward. Moreover, the compliance forces F_x and F_y are proportional to the stiffness of the metallic surface K_e [421]. If the position (X_M, Y_M) of the end effector can be measured then its projection (X_Ω, Y_Ω) on the surface $\Omega(X_1^{'}, Y_1^{'})$ can be calculated, and the compliance forces will be: $F_x = K_e(X_M - X_\Omega)$ and $F_y = K_e(Y_M - Y_\Omega)$.

The control architecture of Fig. 1.5 was employed. The following PD controller gains were used: $K_p = diag\{11.1, 26.5\}$ and $K_d = diag\{17.9, 45.9\}$. In the simulation experiments the value of the stiffness was set to $\bar{K}_e = 200\ Nt/m$ while its nominal value was $K_e = 100\ Nt/m$. Thus the effect of the compliance forces could not be compensated effectively by subtracting the term $J^T F$ from the torques vector [380]. Position error of the robot's joints should be expected as shown in Fig. 2.8(a). However, the neural adaptive controller was able to compensate for the disturbance induced to the robot's joints by the aforementioned compliance forces and to eliminate the position error. The associated simulation results are depicted in Fig. 2.8(b).

Chapter 3
Mobile Robots and Autonomous Vehicles

Abstract. An analysis of the kinematic model of automatic ground vehicles is given and nonlinear control for this type of vehicles is presented. Moreover, the kinematic and dynamic model of unmanned surface vessels is studied and nonlinear control for the dynamic ship positioning problem is in turn formulated.

3.1 Kinematic Analysis of Mobile Robots

3.1.0.1 Nonholonomic Constraints of Autonomous Vehicles

The problem of control of autonomous ground vehicles (AGVs) is first considered. The position of such a vehicle is described by the coordinates (x,y) of the center of its rear axis and its orientation is given by the angle θ between the x-axis and the axis of the direction of the vehicle. The steering angle ϕ and the speed u are considered to be the inputs of the system. The kinematic model of autonomous vehicles can be expressed in the general form [211]

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} cos(\theta) & 0 \\ sin(\theta) & 0 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} v(t) \\ v(t)\rho(t) \end{pmatrix} \tag{3.1}$$

where (x,y) are the coordinates of the center of the vehicle's rear wheels axis, $v(t)$ is the velocity of the vehicle, and θ is the angle between the transversal axis of the vehicle and axis OX. The autonomous vehicle is a nonholonomic system. Nonholonomic systems are characterized by nonintegrable differential expressions, such as

$$\sum_{i=i}^{n} f_{ij}(q_1,q_2,\cdots,q_n,t)\dot{q}_i = 0,\ j = 1,2,\cdots,m \tag{3.2}$$

where $\dot{q}_i$ represents the n-th generalized coordinate (state variable), m is the number of equations defining the nonholonomic constraints, $\dot{q}_i$ represents the generalized speed and f_{ij} are nonlinear functions of q_i at time t. For the kinematic model of Eq. (3.1) the following nonholonomic constraint exists:

$$\dot{x}sin(\theta) - \dot{y}cos(\theta) = 0 \tag{3.3}$$

The curvature radius for any path can be written as

$$R(t) = \frac{1}{\rho(t)} = \frac{L}{tan(\phi)} \tag{3.4}$$

where L is the distance between the front and the back wheels, and ϕ (namely the steering angle) is the angle defined by the main axis of the vehicle and the velocity vector of the front wheel (for cart like vehicles as shown in Fig. 3.1), or the central front point (for car-like vehicles as shown in Fig. 3.1). The value of $R(t)$ is usually bounded by R_{min}, the minimum curvature radius.

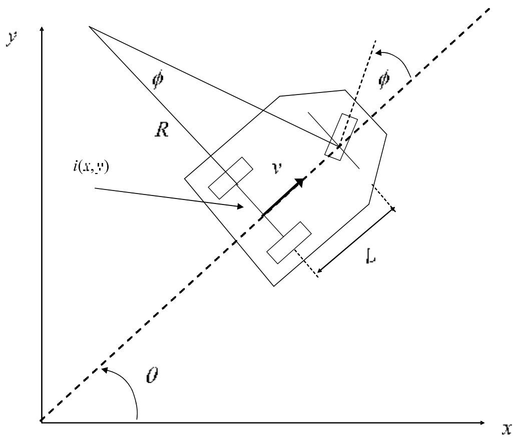


Fig. 3.1 The model of the unicycle autonomous vehicle (cart-like vehicle)

3.2 Control of Autonomous Ground Vehicles

The kinematic model of an autonomous vehicle (robotic unicycle) is considered. This comes from Eq.(3.1) and is given by

$$\dot{x} = vcos(\theta),\ \dot{y} = vsin(\theta),\ \dot{\theta} = \omega = \tfrac{v}{L}tan(\phi) \tag{3.5}$$

where $v(t)$ is the velocity of the vehicle, L is the distance between the front and the rear wheel axis of the vehicle, θ is the angle between the transversal axis of the vehicle and axis OX, and ϕ is the angle of the steering wheel with respect to the transversal axis of the vehicle (the sign of v and ϕ depends on whether the vehicle moves forward or backwards).

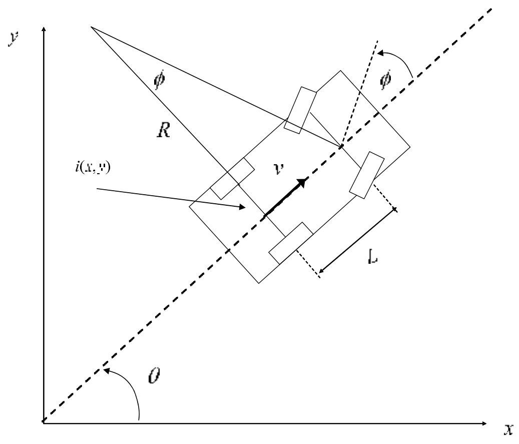


Fig. 3.2 The model of the 4-wheel autonomous vehicle (car-like vehicle)

Flatness-based control can be used for steering the vehicle along a desirable trajectory or for making the robot perform specific tasks such as the parallel parking task. Flatness-based control is based on the property of differential flatness [99],[209],[221][257],[264]. A system is said to be differentially flat if all its state variables and the control inputs can be written as function of a suitably selected variable and its derivatives, where this particular variable is known as *flat output* [355],[356]. For linear systems, differential flatness is equivalent to controllability. For nonlinear systems, flatness-based control enables linearization and calculation of an appropriate control law for the linearized model [44],[273],[342],[333],[438].

3.2.1 Differential Flatness for Finite Dimensional Systems

Flatness-based control is proposed for steering an autonomous ground vehicle (AGV) along a desirable trajectory [98],[295],[438]. The main principles of flatness-based control are as follows [361]: A finite dimensional system is considered. This can be written in the general form of an ODE, i.e.

$$S_i(w,\dot{w},\ddot{w},\cdots,w^i),\ \ i=1,2,\cdots,q \tag{3.6}$$

The quantity w denotes the system variable while w^i, $i=1,2,\cdots,q$ are its derivatives (these can be for instance the elements of the system's state vector). The system of Eq. (1) is said to be differentially flat if there exists a collection of m functions $y=(y_1,\cdots,y_m)$ of the system variables w_i, $i=1,\cdots,s$ and of their time-derivatives, i.e.

$$y_i=\phi(w,\dot{w},\ddot{w},\cdots,w^{\alpha_i}),\ i=1,\cdots,m \tag{3.7}$$

such that the following two conditions are satisfied [98], [333]:

1. There does not exist any differential relation of the form

$$R(y, \dot{y}, \cdots, y^{\beta}) = 0 \tag{3.8}$$

which implies that the derivatives of the flat output are not coupled in the sense of an ODE, or equivalently it can be said that the flat output is differentially independent.

2. All system variables, i.e. the components of w (elements of the system's state vectors) can be expressed using only the flat output y and its time derivatives

$$w_i = \psi_i(y, \dot{y}, \cdots, y^{\gamma_i}),\ i = 1, \cdots, s \tag{3.9}$$

An equivalent definition of differentially flat systems is as follows:

Definition: The system $\dot{x} = f(x,u)$, $x \in R^n$, $u \in R^m$ is differentially flat if there exist relations $h:\ R^n \times R^m \rightarrow R^m$, $\phi:\ (R^m)^r \rightarrow R^n$ and $\psi:\ (R^m)^{r+1} \rightarrow R^m$, such that $y = h(x, u, \dot{u}, \cdots, u^{(r)})$, $x = \phi(y, \dot{y}, \cdots, y^{(r-1)}, y^{(r)})$ and $u = \psi(y, \dot{y}, \cdots, y^{(r-1)}, y^{(r)})$. This means that all system dynamics can be expressed as a function of the flat output and its derivatives, therefore the state vector and the control input can be written as $x(t) = \phi(y(t), \dot{y}(t), \cdots, y^{(r)}(t))$ and $u(t) = \psi(y(t), \dot{y}(t), \cdots, y^{(r)}(t))$.

3.2.2 Flatness-Based Control of the Autonomous Vehicle

In the case of the autonomous vehicle of Eq. (3.5) the flat output is the cartesian position of the center of the wheel axis, denoted as $\eta = (x,y)$, while the other model parameters can be written as:

$$v = \pm||\dot{\eta}|| \begin{pmatrix} cos(\theta) \\ sin(\theta) \end{pmatrix} = \frac{\dot{\eta}}{v}\ tan(\phi) = l det(\dot{\eta}\ddot{\eta})/v^3 \tag{3.10}$$

These formulas show simply that θ is the tangent angle of the curve and $tan(\phi)$ is the associated curvature. With reference to a generic driftless nonlinear system

$$\dot{q},\ q \in R^n, w \in R^m \tag{3.11}$$

dynamic feedback linearization consists in finding a feedback compensator of the form

$$\begin{aligned} \dot{\xi} &= \alpha(q,\xi) + b(q,\xi)u \\ w &= c(q,\xi) + d(q,\xi)u \end{aligned} \tag{3.12}$$

with state $\xi \in R^v$ and input $u \in R^m$, such that the closed-loop system of Eq. (3.11) and Eq. (3.12) is equivalent under a state transformation $z = T(q,\xi)$ to a linear system. The starting point is the selection of a m-dimensional output $\eta = h(q)$ to which a desired behavior can be assigned (this is the previously defined *flat output*). One then proceeds by successively differentiating the output until the input appears in a non-singular way. If the sum of the output differentiation orders equals the dimension $n+v$ of the extended state space, full input-state-output linearization is obtained. The closed-loop system is then equivalent to a set of decoupled input-output chains of integrators from u_i to η_i. The exact linearization procedure is illustrated for the unicycle model of Eq. (21). As flat output $\eta = (x,y)$ the coordinates of the center of the wheel axis is considered . Differentiation with respect to time then yields [295],[333]

$$\dot{\eta} = \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} cos(\theta) & 0 \\ sin(\theta) & 0 \end{pmatrix} \cdot \begin{pmatrix} v \\ \omega \end{pmatrix} \tag{3.13}$$

showing that only v affects $\dot{\eta}$, while the angular velocity ω cannot be recovered from this first-order differential information. To proceed, one needs to add an integrator (whose state is denoted by ξ) on the linear velocity input

$$v = \xi,\ \dot{\xi} = \alpha \Rightarrow \dot{\eta} = \xi \begin{pmatrix} cos(\theta) \\ sin(\theta) \end{pmatrix} \tag{3.14}$$

where α denotes the linear acceleration of the vehicle. Differentiating further one obtains

$$\ddot{\eta} = \dot{\xi} \begin{pmatrix} cos(\theta) \\ sin(\theta) \end{pmatrix} + \xi \dot{\theta} \begin{pmatrix} sin(\theta) \\ cos(\theta) \end{pmatrix} = \begin{pmatrix} cos(\theta) & -\xi sin(\theta) \\ sin(\theta) & \xi cos(\theta) \end{pmatrix} \begin{pmatrix} \alpha \\ \omega \end{pmatrix} \tag{3.15}$$

and the matrix multiplying the modified input (α, ω) is nonsingular if $\xi \neq 0$. Under this assumption one defines

$$\begin{pmatrix} \alpha \\ \omega \end{pmatrix} = \begin{pmatrix} cos(\theta) & -\xi sin(\theta) \\ sin(\theta) & \xi cos(\theta) \end{pmatrix} \cdot \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \tag{3.16}$$

and $\ddot{\eta}$ is denoted as

$$\ddot{\eta} = \begin{pmatrix} \ddot{\eta}_1 \\ \ddot{\eta}_2 \end{pmatrix} = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = u \tag{3.17}$$

which means that the desirable linear acceleration and the desirable angular velocity can be expressed using the transformed control inputs u_1 and u_2. Then, the resulting dynamic compensator is (return to the initial control inputs v and ω)

$$\begin{gathered} \dot{\xi} = u_1 cos(\theta) + u_2 sin(\theta) \\ v = \xi \\ \omega = \frac{u_2 cos(\theta) - u_1 sin(\theta)}{\xi} \end{gathered} \tag{3.18}$$

Being $\xi \in R$, it is $n+v = 3+1 = 4$, equal to the output differentiation order in Eq. (3.17). In the new coordinates

$$\begin{aligned} z_1 &= x \\ z_2 &= y \\ z_3 &= \dot{x} = \xi cos(\theta) \\ z_4 &= \dot{y} = \xi sin(\theta) \end{aligned} \tag{3.19}$$

The extended system is thus fully linearized and described by the chains of integrators, in Eq. (3.17), and can be rewritten as

$$\begin{aligned} \ddot{z}_1 &= u_1 \\ \ddot{z}_2 &= u_2 \end{aligned} \tag{3.20}$$

The dynamic compensator of Eq. (3.18) has a potential singularity at $\xi = v = 0$, i.e. when the vehicle is not moving, which is a case not met while executing the parking manoeuvres. It is noted however, that the occurrence of such a singularity is structural for non-holonomic systems. In general, this difficulty must be obviously taken into account when designing control laws on the equivalent linear model.

A nonlinear controller for output trajectory tracking, based on dynamic feedback linearization, is easily derived. Assume that the autonomous vehicle must follow a smooth trajectory $(x_d(t), y_d(t))$ which is persistent, i.e. for which the nominal velocity $v_d = (\dot{x}_d^2 + \dot{y}_d^2)^{\frac{1}{2}}$ along the trajectory never goes to zeros (and thus singularities are avoided). On the equivalent and decoupled system of Eq. (3.20), one can easily design an exponentially stabilizing feedback for the desired trajectory, which has the form

$$\begin{aligned} u_1 &= \ddot{x}_d + k_{p_1}(x_d - x) + k_{d_1}(\dot{x}_d - \dot{x}) \\ u_2 &= \ddot{y}_d + k_{p_1}(y_d - y) + k_{d_1}(\dot{y}_d - \dot{y}) \end{aligned} \tag{3.21}$$

and which results in the following error dynamics for the closed-loop system

$$\begin{aligned} \ddot{e}_x + k_{d_1}\dot{e}_x + k_{p_1}e_x &= 0 \\ \ddot{e}_y + k_{d_2}\dot{e}_y + k_{p_2}e_y &= 0 \end{aligned} \tag{3.22}$$

where $e_x = x - x_d$ and $e_y = y - y_d$. The proportional-derivative (PD) gains are chosen as $k_{p_i} > 0$ and $k_{d_i} > 0$ for $i = 1,2$. Knowing the control inputs u_1, u_2, for the linearized system one can calculate the control inputs v and ω applied to the vehicle, using Eq. (3.12). The above result is valid, provided that the dynamic feedback compensator does not meet the singularity. In the general case of design of flatness-based controllers, the following theorem assures the avoidance of singularities in the proposed control law [295]:

Theorem: Let λ_{11}, λ_{12} and λ_{21}, λ_{22}, be respectively the eigenvalues of two equations of the error dynamics, given in Eq. (3.12). Assume that, for $i = 1,2$ it is $\lambda_{i1} < \lambda_{i2} < 0$ (negative real eigenvalues), and that $|\lambda_{i2}|$ is sufficiently small. If

$$\min_{t \geq 0} || \begin{pmatrix} \dot{x}_d(t) \\ \dot{y}_d(t) \end{pmatrix} || \geq \begin{pmatrix} \dot{\varepsilon}_x^0 \\ \dot{\varepsilon}_y^0 \end{pmatrix} \tag{3.23}$$

with $\dot{\varepsilon}_x^0 = \dot{\varepsilon}_x(0) \neq 0$ and $\dot{\varepsilon}_y^0 = \dot{\varepsilon}_y(0) \neq 0$, then the singularity $\xi = 0$ is never met.

3.3 Kinematic and Dynamic Models of Surface Vessels

3.3.1 A Generic Kinematic and Dynamic Ship Model

The motion of a ship is described by two reference frames: (i) a local geographical earth-fixed frame denoted as OXY and, (ii) a body-fixed frame denoted as $O^{'}X^{'}Y^{'}$ which is attached to the vessel (see Fig. 3.3). The components of the position vector of the vessel are $[x,y,\psi]^T$ where (x,y) are the coordinates of the ship's center of symmetry in a local geographical frame and ψ is the orientation angle with reference to the OX axis of the local coordinates frame.

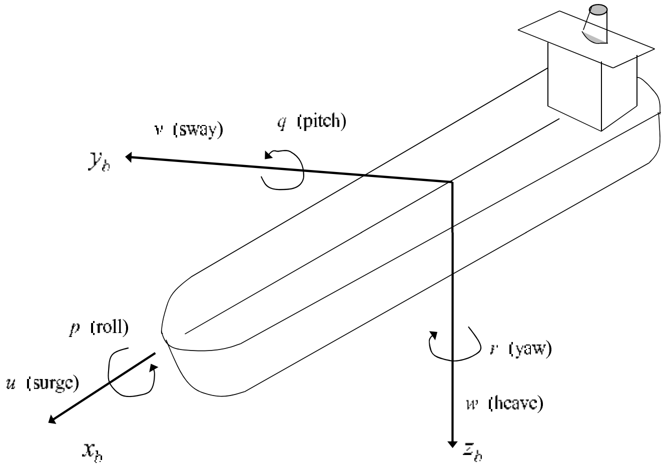


Fig. 3.3 (i) Components of the linear velocity vector of the vessel in a body-fixed frame denoted as surge, sway and heave, (ii) Components of the angular velocity of the vessel in a body-fixed frame denoted as roll, pitch and yaw Euler angles.

The components of the ship's velocity vector, denoted as $u_s = [u,v,r]^T$, are the surge and sway velocities (u,v) and the yaw rate r. The body-fixed velocities u and v are the time derivatives of the position of the origin of the body-fixed frame relative to the origin of the local geographical frame expressed in the body-fixed frame. The yaw rate r stands for the angular velocity of the body-fixed frame with respect to the local geographical reference frame expressed in the body-fixed frame. Although in the most general case the ship's angular velocity has three components, for ship positioning and heading control, the translational motion is assumed to be confined to the horizontal plane and the angular velocity to have only one component, which is the rotation rate r (yaw) about the axis perpendicular to the horizontal plane. A model for vessel kinematics, relating the ship's position vector to the ship velocity vector, can be expressed as

$$\dot{\eta} = R(\psi)v \tag{3.24}$$

The kinematic transformation of Eq. (3.24) relates the body-fixed velocities to the position derivatives in the local geographical frame. The transformation is described by the rotation matrix

$$R(\psi) = \begin{pmatrix} cos\psi & -sin\psi & 0 \\ sin\psi & cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad R^{-1}(\psi) = R^T(\psi) \tag{3.25}$$

The equation of the ship dynamics describes the relation between the ship's velocity and the generalized forces vector (forces and torques) which is applied to the vessel.

$$M\dot{v} + C_{RB}(v)v + d(V_{rc}, \gamma_c) = \tau_{control} + \tau_{wind} + \tau_{waves} \tag{3.26}$$

The inertia matrix M is the sum of two matrices M_A and M_{RB}. When a vessel moves in the water, the changes in the pressure of the hull are proportional to the velocities and accelerations of the vessel relative to the fluid. The coefficients used to express the pressure induced forces proportional to the accelerations are called added-mass coefficients. These forces show the change in momentum in the fluid due to the vessel accelerations. The positive-definite hydrodynamic added-mass matrix M_A is represented by

$$M_A = \begin{pmatrix} -X_{\dot{u}} & 0 & 0 \\ 0 & -Y_{\dot{v}} & -Y_{\dot{r}} \\ 0 & -Y_{\dot{r}} & -N_{\dot{r}} \end{pmatrix} \tag{3.27}$$

where the added-mass coefficients $X_{\dot{u}}$, $Y_{\dot{v}}$, and $N_{\dot{r}}$ depend on the hull shape. On the other hand the positive definite rigid-body mass matrix M_{RB} and the skew-symmetric Coriolis-centripetal matrix $C_{RB}(v)$ are given by

$$M_{RB} = \begin{pmatrix} m & 0 & 0 \\ 0 & m & mx_g \\ 0 & mx_g & I_z \end{pmatrix} \tag{3.28}$$

where x_g denotes the longitudinal position of the center of gravity of the vessel relative to the body-fixed frame.

A frequently used form of the ship's inertia matrix M is

$$M = \begin{pmatrix} m_{11} & 0 & 0 \\ 0 & m_{22} & m_{23} \\ 0 & m_{23} & m_{33} \end{pmatrix} \tag{3.29}$$

The Coriolis-centripetal terms matrix C_{RB} is given by

$$C_{RB} = \begin{pmatrix} 0 & 0 & -m(x_g r + v) \\ 0 & 0 & mu \\ m(x_g r + v) & -mu & 0 \end{pmatrix} \tag{3.30}$$

The Coriolis-centripetal terms appear as a consequence of expressing the equations of motion in body-fixed coordinates. It is noted that when a vessel operates under positioning control the velocities are small and thus the Coriolis-centripetal terms $C_{RB}(v)v$ in Eq. (3.26) can be omitted from the ship's dynamic model.

3.3.2 Models of Current, Wind and Wave Forces

Apart from forces due to control, the right hand side of Eq.(3.26) contains terms which represent wind and wave forces respectively. Moreover, the term $d(V_{rc}, \gamma_c)$ on the left hand side of Eq. (3.26) represents the current and damping forces. The speed of the current is denoted as V_{rc} while the angle of the current is denoted as γ_{rc} and is defined relative to the bow of the vessel.

It is common practice to write the current forces in surge, sway and yaw as functions of nondimensional current coefficients $C_{X_c}(\gamma_{r_c})$, $C_{Y_c}(\gamma_{rc})$, $C_{N_c}(\gamma_{r_c})$ which is

$$d(V_{rc}, \gamma_{rc}) = \frac{1}{2}\rho V_{rc}^2 \begin{pmatrix} A_{F_c} C_{X_c}(\gamma_{rc}) \\ A_{L_c} C_{Y_c}(\gamma_{rc}) \\ A_{L_c} L_{0\alpha} C_{N_c}(\gamma_{rc}) \end{pmatrix} \tag{3.31}$$

where ρ is the water density, A_{F_c} and A_{L_c} are frontal and lateral projected areas of the submerged part of the vessel and $L_{0\alpha}$ is the length of the ship. However, the current coefficients $C_{X_c}(\gamma_{r_c})$, $C_{Y_c}(\gamma_{r_c})$, $C_{N_c}(\gamma_{r_c})$ are difficult to estimate with accuracy. In such cases, it is common practice to simplify the model of Eq. (3.31), in terms of a linear damping term and a bias term which finally takes the form

$$d(V_{rc}, \gamma_{rc}) \simeq D(v)v - R^T(\psi)d \tag{3.32}$$

where

$$D = D^T = \begin{pmatrix} D_{11} & 0 & 0 \\ 0 & D_{22} & D_{23} \\ 0 & D_{32} & D_{33} \end{pmatrix}, \quad d = \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix} \tag{3.33}$$

The linear damping term also models the transfer of energy from the vessel to the fluid due to the waves that are generated as a consequence of the vessel's motion.

The wind forces and moments can be represented in a similar way to the current forces and moments, i.e.

$$\tau_{wind} = \frac{1}{2}\rho_\alpha V_{rw}^2 \begin{pmatrix} A_{Fw} C_{Xw}(\gamma_{rw}) \\ A_{Lw} C_{Yw}(\gamma_{rw}) \\ A_{Lw} L_{0\alpha} C_{Nw}(\gamma_{rw}) \end{pmatrix} \tag{3.34}$$

where ρ_0 is the air density, A_{Fw} and A_{Lw} are the frontal and lateral projected wind areas and $L_{0\alpha}$ is the vessel's overall length. The wind speed is V_{rw} and its direction is γ_{rw} in earth-fixed coordinates. The wind model coefficients can be obtained by model tests while with reference to the control problem obtaining measurements of the wind's speed and direction enables to compensate τ_{wind} using a feed-forward term $\hat{\tau}_{wind}$. The difference (modelling error) between τ_{wind} and $\hat{\tau}_{wind}$ can be described by a bias term $R^T(\psi)d$, as in the case of the current bias term that was given in Eq. (3.32).

Wave forces are usually modeled as the sum of a linear and a nonlinear component, i.e.

$$\tau_{waves} = \tau_{waves}^{lin} + \tau_{waves}^{nlin} \tag{3.35}$$

The low-frequency nonlinear wave forces can be modeled again by a bias term, and are modeled as an input disturbance. On the other hand the linear wave forces are considered to be output disturbances. Therefore, the observation (measurement) equation of the ship is given by

$$y = \eta + n_w + v_1 \tag{3.36}$$

where n is the vessel's position calculated using the ship's dynamic model of Eq. (3.44), v_1 is sensor measurement noise and n_w is the ship's displacement due to the linear wave forces. It has been proposed to approximate the displacement due to linear wave forces by a linear model driven by Gaussian noise w

$$\begin{aligned} \dot{\xi} &= A_w\xi + E_w w \\ n_w &= C_w\xi \end{aligned} \tag{3.37}$$

3.3.3 Ship Model for the Dynamic Positioning Problem

Using Eq. (3.32) and the above assumptions about the wind and waves forces, the vessel's kinematic and dynamic model described in Eq.(3.24) and Eq. (3.26) respecively, is given by

$$\dot{\eta} = R(\psi)v \tag{3.38}$$

$$\dot{v} + M^{-1}Dv = M^{-1}R^T(\psi)d + M^{-1}\tau_{\text{control}} + w \tag{3.39}$$

$$\dot{d} = w \tag{3.40}$$

$$y = \eta + n_w + v_1 \text{ or } y = n + v \tag{3.41}$$

The bias is an additive disturbance in the ship's dynamic model which can be estimated with the use of a state observer. Once the bias is accurately estimated it can be compensated by including a suitable control term in the right hand side of Eq. (3.39). This additional control term provides the required robustness to compensate for the bias effects.

3.3.4 Ship Actuator Model

Without loss of generality the model of a vessel with two propellers and one bow thruster is considered (see Fig. 3.4). The vector of the ship's control forces and torques $\tau \in R^3$ are related to propeller pitch ratios vector u_p (or propeller revolutions for fixed blade propellers) as follows [126]

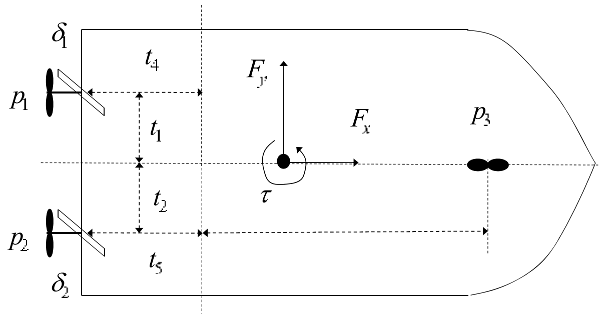


Fig. 3.4 Model of a vessel with two propellers and a bow thruster

$$\tau = T{\cdot}K(U){\cdot}u_p \tag{3.42}$$

where U is the magnitude of the ship's velocity in the xy-plane i.e. $U = \sqrt{u^2+v^2}$ while u denotes the surge velocity and v denotes the sway velocity. Vector u is defined as $u_p = [f_1(p_1), f_2(p_2), f_3(p_3), f_4(\delta_1), f_5(\delta_2)]^T$. For the (fully actuated) ship model of Fig. 3.4 with two propellers p_1 and p_2, one thruster p_3 and two rudders δ_1 and δ_2, matrix $T{\in}R^{3\times 6}$ depends on the position of the actuators p_1, p_2 and p_3, while matrix $K(U){\in}R^{6\times 6}$ depends on the ship's velocity and the type of the actuators. The coefficients of matrices T and K are defined as follows: p_i, $(i=1,2,3)$ are the propeller pitch ratios (or for fixed-blade propellers are the propeller revolutions), δ_i, $(i=1,2)$ are the rudder angles, t_i, $(i=1,\cdots,5)$ are distances of the actuators from the ship's symmetry axes, and k_i, $(i=1,\cdots,5)$ are the force coefficients.

3.4 Feedback Linearization for Ship Dynamic Positioning

3.4.1 Ship Control Using Dynamic Feedback Linearization

As mentioned above the kinematic and dynamic model of the ship is given by

$$\dot{\eta} = R{\cdot}v \tag{3.43}$$

$$M\dot{v} + D(v)v - R^T d = \tau \tag{3.44}$$

From Eq. (3.44) one obtains $v = R^{-1}\dot{\eta}$ or since $R^T = R^{-1}$ it can be written as $v = R^T\dot{\eta}$. Similarly one obtains $\dot{v} = \dot{R}^T\dot{\eta} + R^T\ddot{\eta}$. Consequently, this gives

$$\begin{aligned} M(\dot{R}^T\dot{\eta} + R^T\ddot{\eta}) + DR^T\dot{\eta} - R^T d = \tau \Rightarrow \\ RMR^T\ddot{\eta} + RM\dot{R}^T\dot{\eta} + RD(v)R^T\dot{\eta} - RR^T d = R\tau \end{aligned} \tag{3.45}$$

Defining the matrices $J(\eta)=RMR^T \in R^{3\times 3}$, $C(\eta,\dot{\eta})=RM\dot{R}^T \in R^{3\times 3}$, $F(\eta)=RDR^T \in R^{3\times 3}$ and $\tau^*=R\tau$ the dynamic model of the vessel comes to the form of the dynamic model of a robotic manipulator [94].

$$J(\eta)\ddot{\eta}+C(\eta,\dot{\eta})\dot{\eta}+F(\eta)\dot{\eta}-d=\tau^* \tag{3.46}$$

Using the form of the ship inertia matrix M given in Eq. (3.29) and the form of the damping matrix D given in Eq. (3.33) one obtains the following description for matrices $J(\eta)\ddot{\eta}$, $C(\eta,\dot{\eta})$ and $F(\eta)$

$$J(\eta)\ddot{\eta}=\begin{pmatrix} m_{11}cos^2(\psi)+m_{22}sin^2(\psi) & (m_{11}-m_{22})sin(\psi)cos(\psi) & -m_{23}sin(\psi) \\ (m_{11}-m_{12})sin(\psi)cos(\psi) & m_{11}sin^2(\psi)+m_{22}cos^2(\psi) & m_{23}cos(\psi) \\ -m_{23}sin(\psi) & m_{23}cos(\psi) & m_33 \end{pmatrix} \tag{3.47}$$

$$C(\eta,\dot{\eta})=\begin{pmatrix} \dot{\psi}(m_{22}-m_{11})sin(\psi)cos(\psi) & \dot{\psi}(m_{11}cos^2(\psi)+m_{22}sin^2(\psi)) & 0 \\ -\dot{\psi}(m_{11}sin^2(\psi))+m_{22}cos^2(\psi))) & \dot{\psi}(m_{22}-m_{11})sin(\psi)cos(\psi) & 0 \\ -\dot{\psi}(m_{23}cos(\psi) & -\dot{\psi}(m_{23}sin(\psi) & 0 \end{pmatrix} \tag{3.48}$$

$$F(\eta)=\begin{pmatrix} d_{11}cos^2(\psi)+d_{22}sin^2(\psi) & (d_{11}-d_{12})sin(\psi)cos(\psi) & -d_{23}sin(\psi) \\ (d_{11}-d_{12})sin(\psi)cos(\psi) & d_{11}sin^2(\psi)+d_{22}cos^2(\psi) & d_{23}cos(\psi) \\ -d_{32}sin(\psi) & d_{32}cos(\psi) & d_{23} \end{pmatrix} \tag{3.49}$$

If all parameters of the model are known, then a controller can be designed based on dynamic feedback linearization.

$$\ddot{\eta}+J(\eta)^{-1}C(\eta,\dot{\eta})\dot{\eta}+J(\eta)^{-1}F(\eta)\dot{\eta}-J^{-1}(\eta)d=J^{-1}(\eta)\tau^* \tag{3.50}$$

Then choosing the control signal to be

$$\tau^*=J(\eta)[\ddot{\eta}_d+J(\eta)^{-1}C(\eta,\dot{\eta})\dot{\eta}+J(\eta)^{-1}F(\eta)\dot{\eta}-J(\eta)^{-1}d-K_D\dot{\tilde{\eta}}-K_P\tilde{\eta}] \tag{3.51}$$

where $\tilde{\eta}=\eta-\eta_d$ is the tracking error and $K_D=diag[k_{d_1},k_{d_2},k_{d_3}]$ and $K_P=diag[k_{p_1},k_{p_2},k_{p_3}]$ are feedback gain matrices. This finally results into the tracking error dynamics

$$\begin{gathered}\ddot{\eta}-\ddot{\eta}_d+K_D\dot{\tilde{\eta}}+K_P\tilde{\eta}=0 \\ \text{or } \ddot{\tilde{\eta}}+K_D\dot{\tilde{\eta}}+K_P\tilde{\eta}=0\end{gathered} \tag{3.52}$$

3.4.2 Estimation of the Unknown Additive Disturbances

Estimation of unknown model parameters and external disturbances affecting a dynamical system can be performed with the use of a state observer [469]. This approach is also applicable to the problem of dynamic ship positioning. It has been shown that the nonlinear ship model is of the form

$$\ddot{\eta}+J(\eta)^{-1}C(\eta,\dot{\eta})\dot{\eta}+J(\eta)^{-1}F(\eta)\dot{\eta}=J^{-1}(\eta)\tau+J^{-1}(\eta)d \tag{3.53}$$

Defining the generalized state vector $x=[\eta,d,\dot{\eta},\dot{d}]^T$ and considering invariance of the disturbance d for specific time periods, one obtains the generalized ship state-space model

$$\begin{gathered}\ddot{\eta}+J(\eta)^{-1}C(\eta,\dot{\eta})\dot{\eta}+J(\eta)^{-1}F(\eta)\dot{\eta}-J^{-1}(\eta)d=J^{-1}(\eta)\tau\\ \ddot{\eta}=0\end{gathered} \tag{3.54}$$

Setting $x_1=\eta$, $x_2=d$, $x_3=\dot{\eta}$, $x_4=\dot{d}$ one obtains

$$\begin{pmatrix}\dot{x}_1\\ \dot{x}_2\\ \dot{x}_3\\ \dot{x}_4\end{pmatrix}=\begin{pmatrix}0_{3\times 3} & 0_{3\times 3} & I_{3\times 3} & 0_{3\times 3}\\ 0_{3\times 3} & 0_{3\times 3} & 0_{3\times 3} & I_{3\times 3}\\ 0_{3\times 3} & J^{-1}(x) & -J^{-1}(x)[C(x,\dot{x})+F(x)] & 0_{3\times 3}\\ 0_{3\times 3} & 0_{3\times 3} & 0_{3\times 3} & 0_{3\times 3}\end{pmatrix}\cdot\begin{pmatrix}x_1\\ x_2\\ x_3\\ x_4\end{pmatrix}+\begin{pmatrix}0_{3\times 3}\\ 0_{3\times 3}\\ J^{-1}(x)\\ 0_{3\times 3}\end{pmatrix}\tau \tag{3.55}$$

where $x_i\in R^{3\times 1}$, $i=1,2,3,4$ and $\tau\in R^{3\times 1}$. The measurement vector of the ship's model is given by

$$y=\left(x\ y\ \psi\ d^1\right) \tag{3.56}$$

where x,y are measurements of the ship's cartesian coordinates, ψ is a measurement of the ship's orientation and d^1 is a measurement of the ship's distance from the coast, measured e.g. with the use of a coast radar. Taking into account the existence of process and measurement noise the ship's model is written as

$$\begin{gathered}\dot{x}=Ax+Bu+w\\ y=\gamma(x)+v\end{gathered} \tag{3.57}$$

there w and v are the vectors of process and measurement noise, respectively. State vector x can be estimated by processing a sequence of output measurements y with the use of Kalman or Particle Filtering.

3.5 Backstepping Control for the Ship Steering Problem

3.5.1 The Ship Steering Problem

In the recent years, sophisticated controllers have been proposed for the ship steering problem, based on advanced control engineering concepts. These include model reference adaptive control, self-tuning control, optimal control and neural control [104],[103],[387],[431],[475].

The complete dynamic model of the ship stems from Euler-Lagrange dynamic analysis [240]. The mathematical model relating the rudder angle δ of the ship to the heading ψ (Fig. 3.5) was proposed by Norbin (1963) [103], [212]:

$$\begin{aligned} T\ddot{\psi} + K \cdot H_N(\dot{\psi}) &= K \cdot \delta \\ H_N(\dot{\psi}) &= n_3\dot{\psi}^3 + n_2\dot{\psi}^2 + n_1\dot{\psi} + n_0 \end{aligned} \tag{3.58}$$

where $H_N(\dot{\psi})$ is the nonlinear maneuvering characteristic. For a course unstable ship $n_1 < 0$, whereas a course-stable ship satisfies $n_1 > 0$. For single-screwed ships (one propeller) $n_0 \neq 0$. Similarly, symmetry in the hull implies that $n_2 = 0$. Usually, the bias term n_0 cannot be identified accurately due to the influence of the environmental disturbances (wind, waves and currents). The function $H_N(\dot{\psi})$, can be found from the relationship between δ and ψ in steady state, such that $\ddot{\psi} = \dot{\psi} = \dot{\delta} = 0$. An experiment known as the *spiral test* has shown that $H_N(\dot{\phi})$ can be approximated by [105], [459]

$$H_N(\dot{\psi}) = n_3\dot{\psi}^3 + n_1\dot{\psi} \tag{3.59}$$

In the above equations K is a gain (sec^{-1}), and T is a time constant (sec). These parameters are function of ship's forward velocity and its length. The state-space model of the ship can be written as

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -\tfrac{K}{T}H(x_2) + \tfrac{K}{T}u \end{aligned} \tag{3.60}$$

where $y = x_1$ and $u = \delta$. Setting $x_1 = \psi$ and $x_2 = \dot{\psi}$ the state equation of the ship is obtained:

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} (f(x,t) + g(x,t)u + \tilde{d}) \tag{3.61}$$

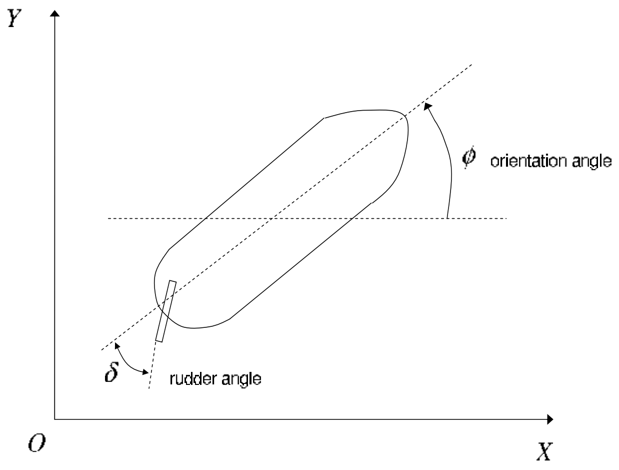


Fig. 3.5 Kinematic model of the ship

$$y = \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \tag{3.62}$$

where $\tilde{d}$ stands for the external disturbances while the nonlinear functions $f(x,t)$ and $g(x,t)$ are given by

$$\begin{aligned} f(x,t) &= -\tfrac{K}{T} H_N(\dot{\psi}) = -\tfrac{K}{T}\{n_3\dot{\psi}^3 + n_1\dot{\psi}\} \\ g(x,t) &= \tfrac{K}{T} \end{aligned} \tag{3.63}$$

3.5.2 Nonlinear Backstepping

Backstepping control can provide solution to the ship steering problem. Nonlinear backstepping is related to feedback linearization. Backstepping control is based on a change of coordinates (diffeomorfism). The transformed system is decomposed in two cascaded subsystems and a stabilizing function is introduced to the first one [104]. The stability proof is recursive, i.e. starting from the second subsystem and moving backwards. Unlike adaptive fuzzy H_∞ control, backstepping control assumes knowledge of the dynamic model of the system. For the nonlinear system

$$\begin{aligned} \dot{x} &= f(x) + g(x)u \\ y &= h(x) \end{aligned} \tag{3.64}$$

where $x = [x_1, \cdots, x_n]^T$, it holds that $\dot{y} = \frac{\partial h(x)}{\partial x}\dot{x} = \frac{\partial h(x)}{\partial x}[f(x) + g(x)u] = L_f h(x) + L_g h(x)u$, where the Lie derivatives are defined as

$$L_f h(x) = \frac{\partial h(x)}{\partial x} f(x), \;\; L_g h(x) = \frac{\partial h(x)}{\partial x} g(x)$$

To apply SISO nonlinear backstepping control, the system of Eq. (3.64) has to be written in a *SISO strict feedback form*, which means the following lower triangular form

$$\begin{aligned} \dot{x}_1 &= f_1(x_1) + g_1(x_1)x_2 \\ \dot{x}_2 &= f_2(x_1,x_2) + g_2(x_1,x_2)x_3 \\ \dot{x}_3 &= f_3(x_1,x_2,x_3) + g_3(x_1,x_2,x_3)x_4 \\ &\cdots \\ &\cdots \\ \dot{x}_{n-1} &= f_{n-1}(x_1,x_2,\cdots,x_{n-1}) + g_{n-1}(x_1,x_2,\cdots,x_{n-1})x_n \\ \dot{x}_n &= f_n(x_1,x_2,\cdots,x_n) + g_n(x_1,x_2,\cdots,x_n)u \\ y &= h(x_1) \end{aligned} \tag{3.65}$$

It is considered that $y = h(x_1) = x_1$ and that the error is $e = y - y_d = x_1 - y_d$. The objective is to succeed $lim_{t\to\infty}e(t) \to 0$. The following theorem holds [201]

Theorem: N-th order backstepping control law

The $n-th$ order SISO backstepping controller is given by

$$\begin{gathered}
\alpha_1 = \frac{1}{L_{g_1}h(x_1)}[\dot{y}_d - L_{f_1}h(x_1) - k_1 z_1 - n_1(z_1)z_1] \\
\alpha_2 = \frac{1}{g_2(x_1,x_2)}[\dot{\alpha}_1 - f_2(x_1,x_2) - L_{g_1}h(x_1)z_1 - k_2 z_2 - n_2(z_2)z_2] \\
\vdots \\
\alpha_i = \frac{1}{g_i(x_1,x_2,\cdots,x_i)}[\dot{\alpha}_i - f_i(x_1,\cdots,x_i) - g_{i-1}(x_1,x_2,\cdots,x_{i-1})z_{i-1} - k_i z_i - n_i(z_i)z_i] \\
\cdots \\
\alpha_n = \frac{1}{g_n(x_1,\cdots,x_n)}[\dot{\alpha}_{n-1} - f_n(x_1,\cdots,x_n) - g_{n-1}(x_1,\cdots,x_{n-2})z_{n-1} - k_n z_n - n_n(z_n)z_n] \\
u = \alpha_n
\end{gathered} \tag{3.66}$$

where $z_1 = h(x_1) - y_d$, $z_i = x_i - \alpha_{i-1}$, $(i = 2,\cdots,n)$ and the stabilizing functions α_i are selected so as to compensate for all nonlinearities. This control law leads asymptotically the system of Eq. (3.64) to the equilibrium $z = 0$. The controller gains are $k_i > 0$ and $n_i(z_i) \geq 0$ for $i = 1,\cdots,n$. ◇

Denoting $z = [z_1,\cdots,z_n]^T$, the resulting error dynamics is given by

$$\dot{z} = -K(z)z + S(x)z, \tag{3.67}$$

with $K(z) = diag\{k_1 + n_1(z_1), k_2 + n_2(z_2), \cdots, k_n + n_n(z_n)\}$

$$S(x) = \begin{pmatrix}
0 & L_{g_1}(x_1) & 0 & \cdots & 0 & 0 & 0 \\
-L_{g_1}(x_1) & 0 & g_2(x_1,x_2) & \cdots & 0 & 0 & 0 \\
0 & -g_2(x_1,x_2) & \cdots & 0 & 0 & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & \cdots & 0 & g_{n-1}(x_1,\cdots,x_{n-1}) & 0 \\
0 & 0 & 0 & \cdots & -g_{n-1}(x_1,\cdots,x_{n-1}) & 0 & 0 \\
0 & 0 & 0 & \cdots & 0 & -g_n(x_1,\cdots,x_n) & 0
\end{pmatrix} \tag{3.68}$$

It holds that $z^T S(x)z = 0\ \forall x,z$ and $K(z) > 0$ thus for $V_n(z) = \frac{1}{2}z^T z$ one gets $\dot{V}_n(z) = -z^T K(z)z < 0$.

3.5.3 Automated Ship Steering Using Backstepping Control

A nonlinear ship model connecting the ship's heading angle ψ to the rudder's angle δ is considered. This model given by Eq. (3.58), i.e.

$$T\ddot{\psi} + H_N(\dot{\psi}) = K\delta$$

$$H_N(\psi) = n_3\dot{\psi}^3 + n_2\dot{\psi}^2 + n_1\dot{\psi} + n_0$$

The SISO backstepping controller is designed in two steps [104]:

Step 1: The error is defined as $e = \psi - \psi_d$. Thus

$$\begin{array}{c} z_1 = e \\ \dot{z}_1 = \dot{\psi} - \dot{\psi}_d \end{array} \tag{3.69}$$

where $y = a_1 + z_2$, thus

$$\dot{z}_1 = \alpha_1 + z_2 - \dot{\psi}_d \tag{3.70}$$

The stabilizing function α_1 is selected so as to assure that $z_1 \to 0$. Therefore,

$$\alpha_1 = \dot{\theta}_d - k_1 z_1 - n_1(z_1) z_1 \tag{3.71}$$

Then, using Eq. (3.70) and (3.71) one gets:

$$\dot{z}_1 = \dot{\psi}_d - k_1 z_1 - n_1(z_1) z_1 + z_2 - \dot{\psi}_d \Rightarrow \dot{z}_1 = -[k_1 + n_1(z_1)] z_1 + z_2$$

where k_1 and $n_1(z_1) \geq 0$ are design parameters. A candidate Lyapunov function for z_1 is:

$$V_1 = \tfrac{1}{2} z_1^2 \Rightarrow \dot{V}_1 = z_1 \dot{z}_1 \Rightarrow \dot{V}_1 = -[k_1 + n_1(z_1)] z_1^2 + z_1 z_2$$

Step 2: For the stabilization of the dynamics of z_2 one has

$$\dot{z}_2 = \ddot{\psi} - \dot{\alpha}_1 = \frac{K}{T}\delta - \frac{1}{T} H_N(\dot{\psi}) - \dot{\alpha}_1 \tag{3.72}$$

The second candidate Lyapunov function is

$$\begin{array}{c} V_2 = V_1 + \tfrac{1}{2} z_2^2 \Rightarrow \dot{V}_2 = \dot{V}_1 + z_2 \dot{z}_2 \Rightarrow \dot{V}_2 = -[k_1 + n_1(z_1)] z_1^2 + z_1 z_2 + z_2 \dot{z}_2 \Rightarrow \\ \dot{V}_2 = -[k_1 + n_1(z_1)] z_1^2 + z_2 [z_1 + \tfrac{K}{T}\delta - \tfrac{1}{T} H_N(\dot{\psi}) - \dot{\alpha}_1] \end{array}$$

The objective is to find the rudder's angle δ for which V_2 will become negative definite.

The SISO back-stepping control law is given by

$$\begin{array}{c} \alpha_1 = \dot{\psi}_d - k_1 z_1 - n_1(z_1) z_1 \\ u = \tfrac{T}{K} [\alpha_1 - z_1 + \tfrac{1}{T} H_N(x_2) - k_2 z_2 - n_2(z_2) z_2] \end{array} \tag{3.73}$$

where $x_1 = \psi$ and $x_2 = \dot{\psi}$, and $k_2 > 0$, $n_2(z_2) \geq 0$ are design parameters. Thus, one gets

$$\begin{array}{c} \dot{V}_2 = -[k_1 + n_1(z_1)] z_1^2 + z_2 [z_1 + \tfrac{K}{T}\tfrac{T}{K}\{\dot{\alpha}_1 - [k_2 + n_2(z_2)] z_2 - z1\}] \\ + \tfrac{K}{T}\tfrac{1}{K} H_N(\dot{\psi}) - \tfrac{1}{T} H_N(\dot{(\psi)}) - \dot{\alpha}_1] \Rightarrow \dot{V}_2 = -[k_1 + n_1(z_1)] z_1^2 - [k_2 + n_2(z_2)] z_2^2 < 0 \\ \forall\, z_1 \neq 0, z_2 \neq 0 \end{array}$$

3.5.4 Calculation of the SISO Backstepping Nonlinear Controller

The system of Eq. (3.58) is written in a SISO strict feedback form, according to Eq. (3.65)

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = -\tfrac{1}{T}[n_3 {x_2}^3 + n_1 x_2] + \tfrac{K}{T}u$$

where $f_2(x_1,x_2) = -\frac{1}{T}[n_3x_2^3 + n_2x_2^2 + n_1x_2 + n_0]$ and $g_2(x_1,x_2) = \frac{K}{T}$. Therefore, $n = 2$ and the SISO backstepping controller is given by the following relations

$$\begin{array}{c} a_1 = \frac{1}{L_{g_1}h(x_1)}[\dot{y}_d - L_{f_1}h(x_1) - k_1 z_1 - \eta_1(z_1)z_1] \\ a_2 = \frac{1}{g_2(x_1,x_2)}[\dot{\alpha}_1 - f_2(x_1,x_2) - L_{g_1}h(x_1)z_1 - k_2 z_2 - \eta_2(z_2)z_2] \\ u = \alpha_2 \end{array} \tag{3.74}$$

It holds that

$$\begin{array}{c} L_{g_1}h(x_1) = \frac{\partial h(x_1)}{\partial x_1}g_1(x_1) = \frac{\partial x_1}{\partial x_1}1 = 1 \\ L_{f_1}h(x_1) = \frac{\partial h(x_1)}{\partial x_1}f_1(x_1) = \frac{\partial x_1}{\partial x_1}0 = 0 \end{array} \tag{3.75}$$

thus,

$$\begin{array}{c} \alpha_1 = [\dot{y}_d - k_1 z_1 - \eta_1(z_1)z_1] \\ \alpha_2 = \frac{T}{K}[\dot{\alpha}_1 + \frac{1}{T}H_N(\dot{\theta}) - 1z_1 - k_2 z_2 - \eta_2(z_2)z_2] \\ u = \alpha_2 \end{array} \tag{3.76}$$

i.e.

$$\begin{array}{c} \alpha_1 = [\dot{y}_d - k_1 z_1 - \eta_1(z_1)z_1] \\ \alpha_2 = \frac{T}{K}[\dot{\alpha}_1 - z_1 + \frac{1}{T}H_N(\dot{x_2}) - k_2 z_2 - \eta_2(z_2)z_2] \\ u = \alpha_2 \end{array} \tag{3.77}$$

The following gains were used $k_1 = 2.0$, $k_2 = 2.0$, $n_1(z_1) = 1.5$ and $n_2(z_1) = 1.2$.

The performance of backstepping nonlinear control has been tested in the case of a see-saw set-point and a sinusoidal setpoint. The results are depicted in Fig. 3.6 and Fig. 3.7.

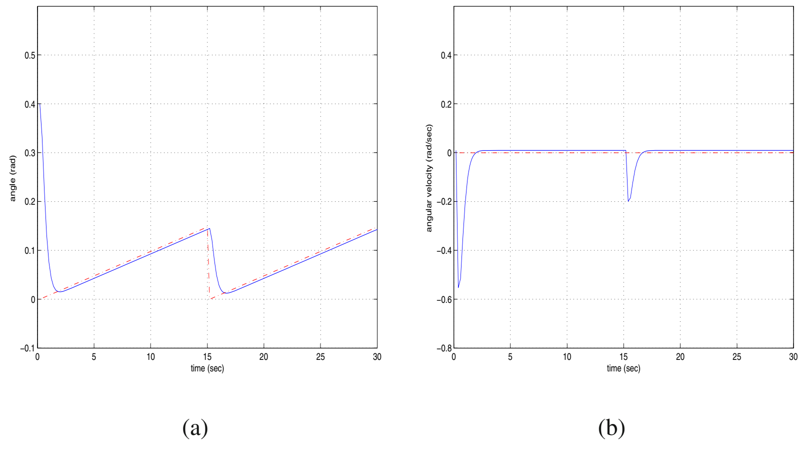


Fig. 3.6 (a) Heading ψ (continuous line) and desirable heading ψ_d (dashed line) of the ship control and (b) Angular velocity $\dot{\psi}$ (continuous line) and desirable desirable angular velocity $\dot{\psi}_d$ (dashed line) of the ship when using backstepping control

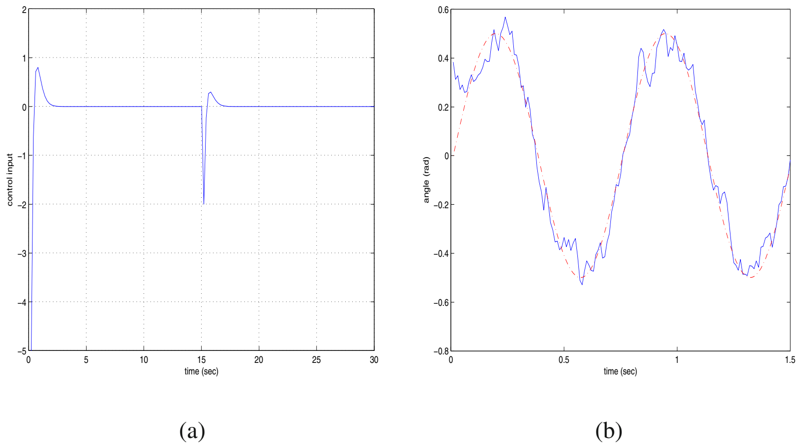


Fig. 3.7 (a) Control input (rudder angle δ) of the ship using backstepping control when tracking a see-saw set-point, (b) Tracking of a sinusoidal set-point using backstepping control in the presence of disturbance and measurement noise.

Chapter 4
Adaptive Control Methods for Industrial Systems

Abstract. A method for the design of stable adaptive control schemes for a class of industrial systems is first studied. The considered adaptive controllers can be based either on feedback of the complete state vector or on feedback of the system's output. In the latter case the objective is to succeed simultaneous estimation of the system's state vector and identification of the unknown system dynamics. Lyapunov analysis provides necessary and sufficient conditions in the controller's design that assure the stability of the control loop. Examples of adaptive control applications to industrial systems are presented.

4.1 Adaptive Control of Industrial Systems with Full State Feedback

4.1.1 Problem Statement

Controller design for systems having complex nonlinear dynamics is an important research field [238],[239],[460]. Many results in this area have been obtained owing to advances in feedback linearization techniques [118],[330],[345],[446] . Neuro-fuzzy networks have been made particularly attractive for modeling and control of nonlinear systems, due to their approximation capabilities, learning and adaptation and parallel distributed features. The feasibility of applying neuro-fuzzy networks to model unknown dynamic systems has been demonstrated in several studies [61],[227],[339],[392],[396],[478]. Both state feedback and output feedback linearization methods have been presented. Under certain assumptions, output feedback controllers can guarantee the global stability of the closed-loop system, based on state observers [120],[121],[122],[213],[215],[412].

In this chapter a method for the control of uncertain dynamical systems is analyzed. In certain cases the state vector of the system can be completely measurable while in other cases the full state vector is unavailable and has to be reconstructed with the use of state observers. The latter case forms a complex problem because it requires to succeed simultaneous convergence of the system's output to the desirable set-point and convergence of the observer's output to zero observation error.

The chapter proposes neuro-fuzzy estimators to approximate the unknown dynamics of the system and an H_∞ control term to compensate for external disturbances. First it is assumed that the complete state vector of the system is measurable. In this case the overall control signal consists of i) the equivalent control term which is based on neurofuzzy approximators, ii) the H_∞ control term which compensates for modelling inaccuracies and external disturbances.

Next it is considered that the system's state vector is not directly measurable and thus it has to be estimated from output measurements with the use of a state observer. In the latter case the overall control signal consists of: i) the equivalent control term which is based on the neurofuzzy approximators, ii) the H_∞ control term which compensates for modelling inaccuracies and external disturbances, iii) a control term which compensates for the observation error [260]. The convergence of the closed-loop that consists of the controller, the neuro-fuzzy approximators and the state observer depends on the simultaneous existence of solution for two Riccati equations. Parameters that also affect the closed-loop robustness are: i) the feedback gain vector K, ii) the observer's gain vector K_o, iii) the positive definite matrices P_1 and P_2 which stem from the solution of the aforementioned algebraic Riccati equations and which weigh the state observer and supervisory control terms. The proposed control architecture guarantees that, the output of the closed-loop system will asymptotically track the desired trajectory and that H_∞ performance will be achieved .

Comparing to model-based approaches the advantages of the proposed adaptive fuzzy H_∞ control are summarized as follows: (i) removal of any dependence upon identification of the mathematical model expressing the dynamics of the system under control, (ii) removal of any dependence upon complete knowledge of the system's state vector. Control is succeeded only through output feedback, (iii) since training of the neuro-fuzzy approximators is repeatedly undertaken in every control cycle, any changes to the system's dynamics can be identified online, and hence the control approach is useful for time-varying models, (iv) regarding operation under external disturbances and measurement noise the adaptive fuzzy H_∞ controller offers improved robustness.

The following non-linear SISO system is considered:

$$x^{(n)} = f(x,t) + g(x,t)u + \tilde{d} \tag{4.1}$$

where $f(x,t)$, $g(x,t)$ are unknown nonlinear functions and $\tilde{d}$ is an unknown additive disturbance. The objective is to force the system's output $y = x$ to follow a given bounded reference signal x_d. In the presence of non-gaussian disturbances w, successful tracking of the reference signal is denoted by the H_∞ criterion [61],[83].

$$\int_0^T e^T Qedt \leq \rho^2 \int_0^T w^T wdt \tag{4.2}$$

where ρ is the attenuation level and corresponds to the maximum singular value of the transfer function $G(s)$ of the linearized equivalent of Eq. (4.1).

Remark: From the H_∞ control theory, the H_∞ norm of a linear system with transfer function $G(s)$, is denoted by $||G||_\infty$ and is defined as $||G||_\infty = sup_\omega \sigma_{max}[G(j\omega)]$ [83],[208],[241]. In this definition sup denotes the supremum or least upper bound of the function $\sigma_{max}[G(j(\omega)]$, and thus the H_∞ norm of $G(s)$ is the maximum value of $\sigma_{max}[G(j(\omega)]$ over all frequencies ω. H_∞ norm has a physically meaningful interpretation when considering the system $y(s) = G(s)u(s)$. When this system is driven with a unit sinusoidal input at a specific frequency, $\sigma_{max}|G(j\omega)|$ is the largest possible output for the corresponding sinusoidal input. Thus, the H_∞ norm is the largest possible amplification over all frequencies of a sinusoidal input.

4.1.2 Transformation to a Regulation Problem

For measurable state vector x and uncertain functions $f(x,t)$ and $g(x,t)$ an appropriate control law for Eq. (4.1) would be

$$u = \frac{1}{\hat{g}(x,t)}[x_d^{(n)} - \hat{f}(x,t) + K^T e + u_c] \tag{4.3}$$

with $e^T = [e, \dot{e}, \ddot{e}, \cdots, e^{(n-1)}]^T$, $K^T = [k_n, k_{n-1}, \cdots, k_1]$, such that the polynomial $e^{(n)} + k_1 e^{(n-1)} + k_2 e^{(n-2)} + \cdots + k_n e$ is Hurwitz. The control law of Eq. (4.3) results into

$$e^{(n)} = -K^T e + u_c + [f(x,t) - \hat{f}(x,t)] + [g(x,t) - \hat{g}(x,t)]u + \tilde{d}, \tag{4.4}$$

where the supervisory control term u_c aims at the compensation of the approximation error

$$w = [f(x,t) - \hat{f}(x,t)] + [g(x,t) - \hat{g}(x,t)]u, \tag{4.5}$$

as well as of the additive disturbance $\tilde{d}$. The above relation can be written in a state-equation form. The state vector is taken to be $e^T = [e, \dot{e}, \cdots, e^{(n-1)}]$, which after some operations yields

$$\dot{e} = (A - BK^T)e + Bu_c + B\{[f(x,t) - \hat{f}(x,t)] + g(x,t) - \hat{g}(x,t)]u + \tilde{d}\} \tag{4.6}$$

$$e_1 = C^T e \tag{4.7}$$

where

$$A = \begin{pmatrix} 0 & 1 & 0 & \cdots & \cdots & 0 \\ 0 & 0 & 1 & \cdots & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & \cdots & 1 \\ 0 & 0 & 0 & \cdots & \cdots & 0 \end{pmatrix}, \quad B = \begin{pmatrix} 0 \\ 0 \\ \cdots \\ \cdots \\ 0 \\ 1 \end{pmatrix}, \quad K = \begin{pmatrix} k_0 \\ k_1 \\ \cdots \\ \cdots \\ k_{n-2} \\ k_{n-1} \end{pmatrix}, \quad C^T = \begin{pmatrix} 1 \\ 0 \\ \cdots \\ \cdots \\ 0 \\ 0 \end{pmatrix} \tag{4.8}$$

and e_1 denotes the output error $e_1 = x - x_d$. Eq. (4.6) and (4.7) describe a regulation problem. The control signal u_c is the H_∞ control term, used for the compensation of $\tilde{d}$ and w

$$u_c = -\tfrac{1}{r} B^T P e \tag{4.9}$$

4.1.3 Approximators of Unknown System Dynamics

The approximation of functions $f(x,t)$ and $g(x,t)$ of Eq. (4.1) can be carried out with Takagi-Sugeno neuro-fuzzy networks of zero or first order (Fig. 4.1). The estimation of $f(x,t)$ and $g(x,t)$ can be written as

$$\hat{f}(x|\theta_f) = \theta_f^T \phi(x), \ \hat{g}(x|\theta_g) = \theta_g^T \phi(x) \tag{4.10}$$

where $\phi(x)$ are kernel functions with elements

$$\phi^l(x) = \frac{\Pi_{i=1}^n \mu_{A_i}^l(x_i)}{\Sigma_{l=1}^L \Pi_{i=1}^n \mu_{A_i}^l(x_i)} \quad l = 1, 2, \cdots, L$$

It is assumed that the weights θ_f and θ_g vary in the bounded areas M_{θ_f} and M_{θ_g} which are defined as

$$\begin{aligned} M_{\theta_f} &= \{\theta_f \in R^h : ||\theta_f|| \leq m_{\theta_f}\}, \\ M_{\theta_g} &= \{\theta_g \in R^h : ||\theta_g|| \leq m_{\theta_g}\} \end{aligned} \tag{4.11}$$

with m_{θ_f} and m_{θ_g} positive constants. The values of θ_f and θ_g that give optimal approximation are:

$$\theta_f^* = \arg\ \min_{\theta_f \in M_{\theta_f}} [sup_{x \in U_x} |f(x) - \hat{f}(x|\theta_f)|]$$

$$\theta_g^* = \arg\ \min_{\theta_g \in M_{\theta_g}} [sup_{x \in U_x} |g(x) - \hat{g}(x|\theta_g)|]$$

The approximation error of $f(x,t)$ and $g(x,t)$ is given by

$$w = [\hat{f}(x|\theta_f^*) - f(x,t)] + [\hat{g}(x|\theta_g^*) - g(x,t)]u \Rightarrow$$

$$\begin{aligned} w = \{&[\hat{f}(x|\theta_f^*) - f(x|\theta_f)] + [f(x|\theta_f) - f(x,t)]\} + \\ +\{&[\hat{g}(\hat{x}|\theta_g^*) - g(x|\theta_g)] + [g(x|\theta_g) - g(x,t)]u\} \end{aligned}$$

where: i) $\hat{f}(x|\theta_f^*)$ is the approximation of f for the best estimation θ_f^* of the weights' vector θ_f, ii) $\hat{g}(x|\theta_g^*)$ is the approximation of g for the best estimation θ_g^* of the weights' vector θ_g. The approximation error w can be decomposed into w_a and w_b, where

$$w_a = [\hat{f}(x|\theta_f) - \hat{f}(x|\theta_f^*)] + [\hat{g}(x|\theta_g) - \hat{g}(x|\theta_g^*)]u$$

$$w_b = [\hat{f}(x|\theta_f^*) - f(x,t)] + [\hat{g}(x|\theta_g^*) - g(x,t)]u$$

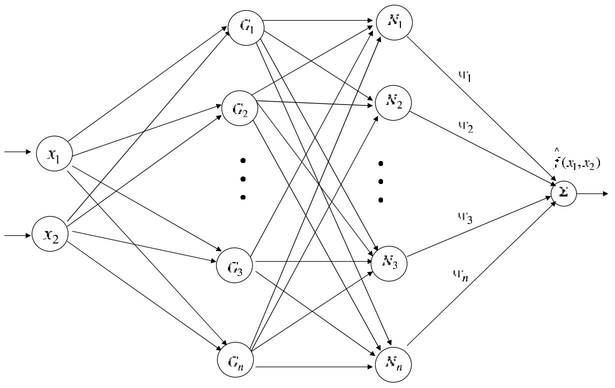


Fig. 4.1 Neuro-fuzzy approximator: G_i Gaussian basis function, N_i: normalization unit

Finally, the following two parameters are defined: $\tilde{\theta}_f = \theta_f - \theta_f^*$, $\tilde{\theta}_g = \theta_g - \theta_g^*$.

4.1.4 Lyapunov Stability Analysis in the Case of Full State Feedback

The adaptation law of the weights θ_f and θ_g as well as of the supervisory control term u_c are derived by the requirement for negative definiteness of the Lyapunov function

$$V = \frac{1}{2}e^T Pe + \frac{1}{2\gamma_1}\tilde{\theta}_f^T\tilde{\theta}_f + \frac{1}{2\gamma_2}\tilde{\theta}_g^T\tilde{\theta}_g \tag{4.12}$$

Substituting Eq. (4.6) into Eq. (4.12) and differentiating gives

$$\dot{V} = \tfrac{1}{2}\dot{e}^T Pe + \tfrac{1}{2}e^T P\dot{e} + \tfrac{1}{\gamma_1}\tilde{\theta}_f^T\dot{\tilde{\theta}}_f + \tfrac{1}{\gamma_2}\tilde{\theta}_g^T\dot{\tilde{\theta}}_g \Rightarrow$$

$$\dot{V} = \tfrac{1}{2}e^T\{(A-BK^T)^T P + P(A-BK^T)\}e + B^T Pe(u_c + w + \tilde{d}) + \tfrac{1}{\gamma_1}\tilde{\theta}_f^T\dot{\tilde{\theta}}_f + \tfrac{1}{\gamma_2}\tilde{\theta}_g^T\dot{\tilde{\theta}}_g$$

Assumption 1: For given positive definite matrix Q there exists a positive definite matrix P, which is the solution of the following matrix equation

$$(A-BK^T)^T P + P(A-BK^T) - PB(\frac{2}{r} - \frac{1}{\rho^2})B^T P + Q = 0 \tag{4.13}$$

Substituting Eq. (4.13) into $\dot{V}$ yields after some operations

$$\dot{V} = -\tfrac{1}{2}e^T Qe + \tfrac{1}{2}e^T PB(\tfrac{2}{r} - \tfrac{1}{\rho^2})B^T Pe + B^T Pe(-\tfrac{1}{r}e^T PB) + B^T Pe(w+d) +$$
$$+\tfrac{1}{\gamma_1}\tilde{\theta}_f^T\dot{\tilde{\theta}}_f + \tfrac{1}{\gamma_2}\tilde{\theta}_g^T\dot{\tilde{\theta}}_g.$$

It holds that $\dot{\tilde{\theta}}_f = \dot{\theta}_f - \dot{\theta}_f^* = \dot{\theta}_f$ and $\dot{\tilde{\theta}}_g = \dot{\theta}_g - \dot{\theta}_g^* = \dot{\theta}_g$. The following weight adaptation laws are considered

$$\dot{\theta}_f = \{ \begin{matrix} -\gamma_1 e^T PB\phi(x) \ \ if \ ||\theta_f|| < m_{\theta_f} \\ 0 \ \ ||\theta_f|| \geq m_{\theta_f} \end{matrix} \tag{4.14}$$

$$\dot{\theta}_g = \{ \begin{matrix} -\gamma_2 e^T PB\phi(x)u_c \ \ if \ ||\theta_g|| < m_{\theta_g} \\ 0 \ \ ||\theta_g|| \geq m_{\theta_g} \end{matrix} \tag{4.15}$$

$\dot{\theta}_f$ and $\dot{\theta}_g$ are set to 0, when $||\theta_f|| \geq m_{\theta_f}$, and $||\theta_g|| \geq m_{\theta_g}$ [446]. The update of θ_f stems from a LMS algorithm on the cost function $\frac{1}{2}(f-\hat{f})^2$. The update of θ_g is also of the LMS type, while u_c implicitly tunes the adaptation gain γ_2. Substituting Eq. (4.14) and (4.15) in $\dot{V}$ finally gives

$$\dot{V} = -\frac{1}{2}e^TQe - \frac{1}{2\rho^2}e^TPBB^TPe + e^TPB(w+d) -$$
$$-e^TPB(\theta_f - \theta_f^*)^T\phi(x) - e^TPB(\theta_g - \theta_g^*)^T\phi(x)u_c$$

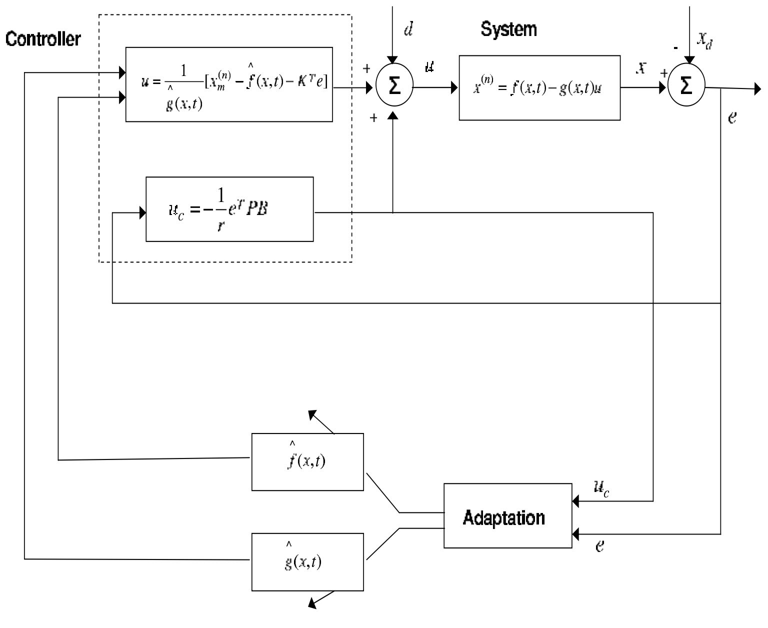
$$\Rightarrow \dot{V} = -\frac{1}{2}e^TQe - \frac{1}{2\rho^2}e^TPBB^TPe + e^TPB(w+d) + e^TPBw_\alpha$$

The control scheme is depicted in Fig. 4.2

Denoting $w_1 = w + d + w_\alpha$ one gets

$$\dot{V} = -\frac{1}{2}e^TQe - \frac{1}{2\rho^2}e^TPBB^TPe + e^TPBw_1 \text{ or equivalently,}$$

$$\dot{V} = -\frac{1}{2}e^TQe - \frac{1}{2\rho^2}e^TPBB^TPe + \frac{1}{2}e^TPBw_1 + \frac{1}{2}w_1^TB^TPe.$$



Controller
$u = \frac{1}{\hat{g}(x,t)}[x_m^{(n)} - \hat{f}(x,t) - K^Te]$
$u_c = -\frac{1}{r}e^TPB$
d
System
$x^{(n)} = f(x,t) - g(x,t)u$
u
x
x_d
+
-
e
$\hat{f}(x,t)$
$\hat{g}(x,t)$
Adaptation
u_c
e

Fig. 4.2 The proposed H_∞ control scheme in the case of full state feedback

Lemma: The following inequality holds

$$\frac{1}{2}e^T PBw_1 + \frac{1}{2}w_1^T B^T Pe - \frac{1}{2}\rho^2 e^T PBB^T Pe \leq \frac{1}{2}\rho^2 w_1^T w_1 \tag{4.16}$$

Proof: The binomial $(\rho a - \frac{1}{\rho}b)^2 \geq 0$ is considered. Expanding the left part of the above inequality one gets

$$\rho^2 a^2 + \tfrac{1}{\rho^2}b^2 - 2ab \geq 0 \Rightarrow \tfrac{1}{2}\rho^2 a^2 + \tfrac{1}{2\rho^2}b^2 - ab \geq 0 \Rightarrow$$

$$ab - \tfrac{1}{2\rho^2}b^2 \leq \tfrac{1}{2}\rho^2 a^2 \Rightarrow \tfrac{1}{2}ab + \tfrac{1}{2}ab - \tfrac{1}{2\rho^2}b^2 \leq \tfrac{1}{2}\rho^2 a^2.$$

The following substitutions are carried out $a = w_1$ and $b = e^T PB$ and the previous relation becomes

$$\tfrac{1}{2}w_1^T B^T Pe + \tfrac{1}{2}e^T PBw_1 - \tfrac{1}{2\rho^2}e^T PBB^T P_2 e \leq \tfrac{1}{2}\rho^2 w_1^T w_1$$

The previous inequality is used in $\dot{V}$, and the right part of the associated inequality is enforced

$$\dot{V} \leq -\frac{1}{2}e^T Qe + \frac{1}{2}\rho^2 w_1^T w_1 \tag{4.17}$$

Eq. (4.17) can be used to show that the H_∞ performance criterion is satisfied . The integration of $\dot{V}$ from 0 to T gives

$$\int_0^T \dot{V}(t)dt \leq -\tfrac{1}{2}\int_0^T ||e||^2 dt + \tfrac{1}{2}\rho^2 \int_0^T ||w_1||^2 dt \Rightarrow$$

$$2V(T) + \int_0^T ||e||_Q^2 dt \leq 2V(0) + \rho^2 \int_0^T ||w_1||^2 dt.$$

If there exists a positive constant $M_w > 0$ such that $\int_0^\infty ||w_1||^2 dt \leq M_w$. Therefore one gets

$$\int_0^\infty ||e||_Q^2 dt \leq 2V(0) + \rho^2 M_w \tag{4.18}$$

Thus, the integral $\int_0^\infty ||e||_Q^2 dt$ is bounded and according to Barbalat's Lemma one obtains $lim_{t\to\infty} e(t) = 0$.

4.2 Adaptive Control of Industrial Systems with Output Feedback

4.2.1 Transformation to a Regulation Problem

For measurable state vector x of the system and uncertain functions $f(x,t)$ and $g(x,t)$ an appropriate control law for Eq. (4.49) is given by Eq. (4.3). When an observer is used to reconstruct the state vector x of Eq. (4.3), the control law of Eq. (4.3) is written as

$$u = \frac{1}{\hat{g}(\hat{x},t)}[x_m^{(n)} - \hat{f}(\hat{x},t) + K^T e + u_c] \tag{4.19}$$

The following definitions are used: i) error of the state vector $e = x - x_m$, ii) error of the estimated state vector $\hat{e} = \hat{x} - x_m$, iii) observation error $\tilde{e} = e - \hat{e} = (x - x_m) - (\hat{x} - x_m)$. Applying Eq. (4.19) to Eq. (4.49), after some algebraic operations, results into

$$x^{(n)} = x_m^{(n)} - K^T \hat{e} + u_c + [f(x,t) - \hat{f}(\hat{x},t)] + [g(x,t) - \hat{g}(\hat{x},t)]u + \tilde{d}$$

It holds $e = x - x_m \Rightarrow x^{(n)} = e^{(n)} + x_m^{(n)}$. Substituting $x^{(n)}$ in the above equation gives

$$\begin{aligned} \dot{e} &= Ae - BK^T \hat{e} + Bu_c + B\{[f(x,t) - \hat{f}(\hat{x},t)] + [g(x,t) - \hat{g}(\hat{x},t)]u + \tilde{d}\} \\ e_1 &= C^T e \end{aligned} \tag{4.20}$$

$\hat{e} = \left(\hat{e}, \dot{\hat{e}}, \ddot{\hat{e}}, \cdots, \hat{e}^{(n-1)}\right)^T$ and A, C, K are given by Eq. (4.8). According to Eq. (4.20) the observer is:

$$\begin{aligned} \dot{\hat{e}} &= A\hat{e} - BK^T \hat{e} + K_o[e_1 - C^T \hat{e}] \\ \hat{e}_1 &= C^T \hat{e} \end{aligned} \tag{4.21}$$

The observation gain $K_o = [k_{o_0}, k_{o_1}, \cdots, k_{o_{n-2}}, k_{o_{n-1}}]$ is selected so as to assure the convergence of the observer. Subtracting Eq. (4.21) from Eq. (4.20) one gets

$$\begin{aligned} \dot{\tilde{e}} &= (A - K_o C^T)\tilde{e} + Bu_c + B\{[f(x,t) - \hat{f}(\hat{x},t)] + [g(x,t) - \hat{g}(\hat{x},t)]u + \tilde{d}\} \\ \tilde{e}_1 &= C\tilde{e} \end{aligned} \tag{4.22}$$

The additional term u_c which appeared in Eq. (4.3) is also introduced in the observer-based control scheme to compensate for: i) The external disturbances $\tilde{d}$, ii) The state vector estimation error $\tilde{e} = e - \hat{e} = x - \hat{x}$, iii) The approximation error of the nonlinear functions $f(x,t)$ and $g(x,t)$, denoted as $w = [f(x,t) - \hat{f}(\hat{x},t)] + [g(x,t) - \hat{g}(\hat{x},t)]u$. The control u_c consists of: i) the H_∞ control term u_a, for the compensation of d and w, ii) the control term u_b, for the compensation of the observation error $\tilde{e}$. The control scheme is depicted in Fig. 4.3.

$$\begin{aligned} u_a &= -\tfrac{1}{r} B^T P_2 \tilde{e} \\ u_b &= -K_o^T P_1 \hat{e} \end{aligned} \tag{4.23}$$

4.2.2 Approximation of Unknown System Dynamics

The approximation of functions $f(\hat{x},t)$ and $g(\hat{x},t)$ of Eq. (4.19) can be carried out again with Takagi-Sugeno neuro-fuzzy networks of zero or first order (see again Fig. 4.1). These consist of rules of the form:

$$\begin{array}{l} R^l: \ \text{IF} \hat{x} \text{ is } A_1^l \text{AND } \dot{\hat{x}} \text{ is } A_2^l \text{ AND } \cdots \text{ AND } \hat{x}^{(n-1)} \text{ is } A_n^l \text{ THEN} \\ \bar{y}^l = \sum_{i=1}^n w_i^l \hat{x}_i + b^l, \ \ l = 1, 2, \cdots, L \end{array} \tag{4.24}$$

The output of the Takagi-Sugeno model is calculated by taking the average of the consequent part of the rules

$$\hat{y} = \frac{\sum_{l=1}^L \bar{y}^l \prod_{i=1}^n \mu_{A_i}^l(\hat{x}_i)}{\sum_{l=1}^L \prod_{i=1}^n \mu_{A_i}^l(\hat{x}_i)}$$

where $\mu_{A_i^l}$ is the membership function of x_i in the fuzzy set A_i^l. The training of the neuro-fuzzy networks is carried out with 1^{st} order gradient algorithms, in pattern mode, i.e. by processing only one data pair (x_i, y_i) at every time step i. The estimation of $f(x,t)$ and $g(x,t)$ can be written as

$$\hat{f}(\hat{x}|\theta_f) = \theta_f^T \phi(\hat{x}) \ \hat{g}(\hat{x}|\theta_g) = \theta_g^T \phi(\hat{x}) \tag{4.25}$$

where $\phi(\hat{x})$ are kernel functions with elements

$$\phi^l(\hat{x}) = \frac{\prod_{i=1}^n \mu_{A_i}^l(\hat{x}_i)}{\sum_{l=1}^L \prod_{i=1}^n \mu_{A_i}^l(\hat{x}_i)} \ \ l = 1, 2, \cdots, L$$

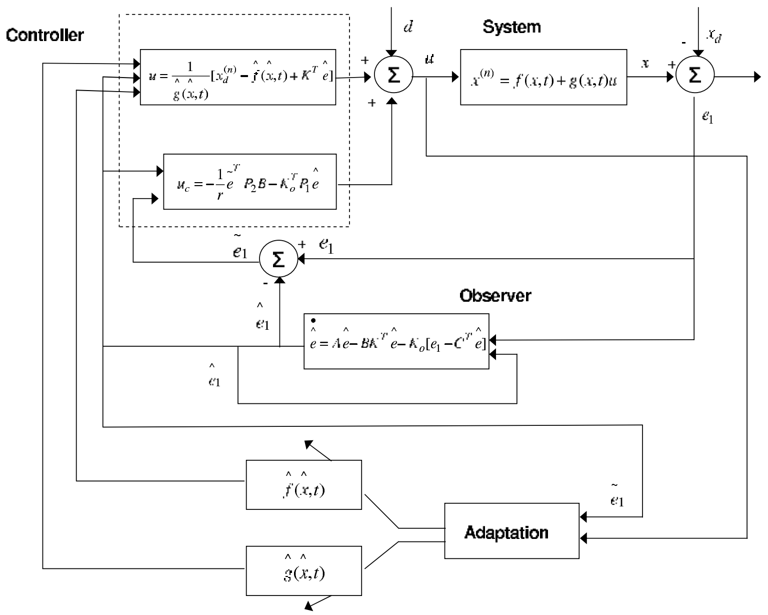

Controller
$u = \frac{1}{\hat{g}(\hat{x},t)}[x_d^{(n)} - \hat{f}(\hat{x},t) + K^T \hat{e}]$
$u_c = -\frac{1}{r}\tilde{e}^T P_2 B - K_o^T P_1 \hat{e}$
d
u
System
$x^{(n)} = f(x,t) + g(x,t)u$
x
x_d
e_1
$\tilde{e}_1$
$\hat{e}_1$
Observer
$\dot{\hat{e}} = A\hat{e} - BK^T\hat{e} - K_o[e_1 - C^T\hat{e}]$
$\hat{f}(\hat{x},t)$
$\hat{g}(\hat{x},t)$
Adaptation

Fig. 4.3 The proposed H_∞ control scheme in the case of output feedback

It is assumed that that the weights θ_f and θ_g vary in the bounded areas M_{θ_f} and M_{θ_g}, while x and $\hat{x}$ remain in the bounded areas U_x and $U_{\hat{x}}$ respectively. The values of θ_f and θ_g for optimal approximation are:

$$\theta_f^* = arg \ \ min_{\theta_f \in M_{\theta_f}} [sup_{x \in U_x, \hat{x} \in U_{\hat{x}}} |f(x) - \hat{f}(\hat{x}|\theta_f)|],$$

$$\theta_g^* = arg \ \ min_{\theta_g \in M_{\theta_g}} [sup_{x \in U_x, \hat{x} \in U_{\hat{x}}} |g(x) - \hat{g}(\hat{x}|\theta_g)|].$$

The approximation error of $f(x,t)$ and $g(x,t)$ is given by

$$w = [\hat{f}(\hat{x}|\theta_f^*) - f(x,t)] + [\hat{g}(\hat{x}|\theta_g^*) - g(x,t)]u \Rightarrow$$

$$\begin{aligned} w = &\{[\hat{f}(\hat{x}|\theta_f^*) - f(x|\theta_f^*)] + [f(x|\theta_f^*) - f(x,t)]\} + \\ &+\{[\hat{g}(\hat{x}|\theta_g^*) - g(\hat{x}|\theta_g^*)] + [g(\hat{x}|\theta_g^*) g(x,t)]u\} \end{aligned}$$

where, i) $\hat{f}(\hat{x}|\theta_f^*)$ is the approximation of f for the best estimation θ_f^* of the weights' vector θ_f, ii) $\hat{g}(\hat{x}|\theta_g^*)$ is the approximation of g for the best estimation θ_g^* of the weights' vector θ_g. The approximation error w can be decomposed into w_a and w_b, where

$$w_a = [\hat{f}(\hat{x}|\theta_f) - \hat{f}(\hat{x}|\theta_f^*)] + [\hat{g}(\hat{x}|\theta_g) - \hat{g}(\hat{x}|\theta_g^*)]u,$$

$$w_b = [\hat{f}(\hat{x}|\theta_f^*) - f(x,t)] + [\hat{g}(\hat{x}|\theta_g^*) - g(x,t)]u.$$

Finally, the following two parameters are defined: $\tilde{\theta}_f = \theta_f - \theta_f^*$ and $\tilde{\theta}_g = \theta_g - \theta_g^*$.

4.2.3 Lyapunov Stability Analysis in the Case of Output Feedback

The adaptation law of the neuro-fuzzy approximators weights θ_f and θ_g as well as of the supervisory control term u_c are derived from the requirement for negative definiteness of the Lyapunov function

$$V = \tfrac{1}{2}\hat{e}^T P_1 \hat{e} + \tfrac{1}{2}\tilde{e}^T P_2 \tilde{e} + \tfrac{1}{2\gamma_1}\tilde{\theta}_f^T \tilde{\theta}_f + \tfrac{1}{2\gamma_2}\tilde{\theta}_g^T \tilde{\theta}_g \tag{4.26}$$

The selection of the Lyapunov function is based on the following two principles of indirect adaptive control: i) $\hat{e}: \lim_{t\to\infty}\hat{x}(t) = x_d(t)$, ii) $\tilde{e}: \lim_{t\to\infty}\hat{x}(t) = x(t)$ which yields $lim_{t\to\infty}x(t) = x_d(t)$. Substituting Eq. (4.20), and Eq. (4.22), into Eq. (4.26) and differentiating results into

$$\begin{aligned} \dot{V} = &\tfrac{1}{2}\hat{e}^T(A - BK^T)^T P_1 \hat{e} + \tfrac{1}{2}\tilde{e}^T C K_o^T P_1 \hat{e} + \tfrac{1}{2}\hat{e}^T P_1 (A - BK^T)\hat{e} + \\ &+\tfrac{1}{2}\hat{e}^T P_1 K_o C^T \tilde{e} + \tfrac{1}{2}\tilde{e}^T (A - K_o C^T)^T P_2 \tilde{e} + \tfrac{1}{2}B^T P_2 \tilde{e}(u_c + w + d) + \\ &+\tfrac{1}{2}\tilde{e}^T P_2 (A - K_o C^T)\tilde{e} + \tfrac{1}{2}\tilde{e}^T P_2 B(u_c + w + d) + \tfrac{1}{\gamma_1}\tilde{\theta}_f^T \dot{\tilde{\theta}}_f + \tfrac{1}{\gamma_2}\tilde{\theta}_g^T \dot{\tilde{\theta}}_g \end{aligned}$$

Assumption 1: For given positive definite matrices Q_1 and Q_2 there exist positive definite matrices P_1 and P_2, which are the solution of the following Riccati equations

$$(A-BK^T)^T P_1 + P_1(A-BK^T) + Q_1 = 0$$

$$(A-K_oC^T)^T P_2 + P_2(A-K_oC^T) - P_2B(\tfrac{2}{r}-\tfrac{1}{\rho^2})B^TP_2 + Q_2 = 0 \tag{4.27}$$

$$P_2B = C$$

The conditions given in Eq. (4.27) are related to the requirement that the systems described by Eq. (4.21) and Eq. (4.22), are strictly positive real. Substituting Eq. (4.27) into $\dot{V}$ yields

$$\begin{aligned}\dot{V} = &-\tfrac{1}{2}\hat{e}^TQ_1\hat{e} + \tilde{e}^TCK_o^TP_1\hat{e} - \tfrac{1}{2}\tilde{e}^T\{Q_2 - P_2B(\tfrac{2}{r}-\tfrac{1}{\rho^2})B^TP_2\}\tilde{e}+ \\ &+B^TP_2\tilde{e}(u_c+w+d) + \tfrac{1}{\gamma_1}\tilde{\theta}_f^T\dot{\tilde{\theta}}_f + \tfrac{1}{\gamma_2}\tilde{\theta}_g^T\dot{\tilde{\theta}}_g\end{aligned} \tag{4.28}$$

Substituting u_a and u_b in $\dot{V}$ and assuming that Eq. (4.27) holds, after some operations one gets

$$\begin{aligned}\dot{V} = &-\tfrac{1}{2}\hat{e}^TQ_1\hat{e} - \tfrac{1}{2}\tilde{e}^TQ_2\tilde{e} - \tfrac{1}{2\rho^2}\tilde{e}^TP_2BB^TP_2\tilde{e} + B^TP_2\tilde{e}(w+d)+ \\ &+\tfrac{1}{\gamma_1}\tilde{\theta}_f^T\dot{\tilde{\theta}}_f + \tfrac{1}{\gamma_2}\tilde{\theta}_g^T\dot{\tilde{\theta}}_g\end{aligned} \tag{4.29}$$

It holds that

$$\dot{\tilde{\theta}}_f = \dot{\theta}_f - \dot{\theta}_f^* = \dot{\theta}_f$$
$$\dot{\tilde{\theta}}_g = \dot{\theta}_g - \dot{\theta}_g^* = \dot{\theta}_g$$

The following weight adaptation laws are considered [446]:

$$\dot{\theta}_f = \{ \begin{matrix} -\gamma_1\tilde{e}^TP_2B\phi(\hat{x}) \ \ if \ ||\theta_f|| \in M_{\theta_f} \\ 0 \ \ ||\theta_f|| \notin M_{\theta_f} \end{matrix} \tag{4.30}$$

$$\dot{\theta}_g = \{ \begin{matrix} -\gamma_2\tilde{e}^TP_2B\phi(\hat{x})u_c \ \ if \ ||\theta_g|| \in M_{\theta_g} \\ 0 \ \ ||\theta_g|| \notin M_{\theta_g} \end{matrix} \tag{4.31}$$

Substituting Eq. (4.30) and using Eq. (4.25) and (4.30) results into

$$\begin{aligned}\dot{V} = &-\tfrac{1}{2}\hat{e}^TQ_1\hat{e} - \tfrac{1}{2}\tilde{e}^TQ_2\tilde{e} - \tfrac{1}{2\rho^2}\tilde{e}^TP_2BB^TP_2\tilde{e} + \tilde{e}^TP_2B(w+d) - \tilde{e}^TP_2B\{[\hat{f}(\hat{x}|\theta_f)+ \\ &+\hat{g}(\hat{x}|\theta_f)u] - [\hat{f}(\hat{x}|\theta_f^*) + \hat{g}(\hat{x}|\theta_g^*)u]\}\end{aligned} \tag{4.32}$$

where

$$[\hat{f}(\hat{x}|\theta_f) + \hat{g}(\hat{x}|\theta_f)u] - [\hat{f}(\hat{x}|\theta_f^*) + \hat{g}(\hat{x}|\theta_g^*)u] = w_a.$$

Thus setting $w_1 = w + w_a + d$ one finally gets

$$\dot{V} = -\tfrac{1}{2}\hat{e}^T Q_1 \hat{e} - \tfrac{1}{2}\tilde{e}^T Q_2 \tilde{e} - \tfrac{1}{2\rho^2}\tilde{e}^T P_2 B B^T P_2 \tilde{e} + \tfrac{1}{2} w_1^T B^T P_2 \tilde{e} + \tfrac{1}{2}\tilde{e}^T P_2 B w_1$$

Lemma: The following inequality holds

$$\tfrac{1}{2}\tilde{e}^T P_2 B w_1 + \tfrac{1}{2} w_1^T B^T P_2 \tilde{e} - \tfrac{1}{2\rho^2}\tilde{e}^T P_2 B B^T P_2 \tilde{e} \leq \tfrac{1}{2}\rho^2 w_1^T w_1 \tag{4.33}$$

Proof: The binomial $(\rho a - \frac{1}{\rho} b)^2 \geq 0$ is considered. Expanding the left part of the above inequality results in:

$$ab - \tfrac{1}{2\rho^2} b^2 \leq \tfrac{1}{2}\rho^2 a^2 \Rightarrow \tfrac{1}{2}ab + \tfrac{1}{2}ab - \tfrac{1}{2\rho^2} b^2 \leq \tfrac{1}{2}\rho^2 a^2 \tag{4.34}$$

Substituting $a = w_1$ and $b = \tilde{e}^T P_2 B$ and the previous relation one gets Eq. (4.33) $\diamond$.

Eq. (4.33) is used in $\dot{V}$, and the right part of the associated inequality is enforced

$$\dot{V} \leq -\frac{1}{2}\hat{e}^T Q_1 \hat{e} - \frac{1}{2}\tilde{e}^T Q_2 \tilde{e} + \frac{1}{2}\rho^2 w_1^T w_1 \tag{4.35}$$

Eq. (4.35) is used to show that, the H_∞ performance criterion of Eq.(4.2) is derived. For ρ sufficiently small Eq. (4.35) will be true and the H_∞ tracking criterion will hold. In that case, the integration of $\dot{V}$ from 0 to T gives

$$2V(T) + \textstyle\int_0^T ||E||_Q^2 dt \leq 2V(0) + \rho^2 \int_0^T ||w_1||^2 dt \tag{4.36}$$

where $E = [\hat{e}, \tilde{e}]^T$ and $Q = diag[Q_1, Q_2]^T$. If there exists a positive constant $M_w > 0$ such that $\int_0^\infty ||w_1||^2 dt \leq M_w$, then for the integral $\int_0^\infty ||E||_Q^2 dt$ one gets

$$\textstyle\int_0^\infty ||E||_Q^2 dt \leq 2V(0) + \rho^2 M_w \tag{4.37}$$

Thus, the integral $\int_0^\infty ||E||_Q^2 dt$ is bounded and according to Barbalat's Lemma

$$lim_{t\to\infty} E(t) = 0 \Rightarrow \ lim_{t\to\infty}\hat{e}(t) = 0, lim_{t\to\infty}\tilde{e}(t) = 0 \tag{4.38}$$

Therefore $lim_{t\to\infty} e(t) = 0$.

4.2.4 Riccati Equation Coefficients and H_∞ Control Robustness

The linear system of Eq. (4.20) is considered again, i.e.

$$\dot{\tilde{e}} = (A - K_o C^T)\tilde{e} + B u_c + B\{[f(x,t) - \hat{f}(\hat{x},t)] + [g(x,t) - \hat{g}(\hat{x},t)]u + \tilde{d}\}$$
$$e_1 = C\tilde{e}$$

The aim of H_∞ control is to eliminate the impact of the modelling errors $w = [f(x,t) - \hat{f}(\hat{x},t)] + [g(x,t) - \hat{g}(\hat{x},t)]u$ and the external disturbances $\tilde{d}$ which are not white noise signals. This implies the minimization of the quadratic cost function [208],[241]:

$$J(t) = \frac{1}{2}\int_0^T \tilde{e}^T(t)\tilde{e}(t) + ru_c^T(t)u_c(t) - \rho^2(w+\tilde{d})^T(w+\tilde{d})dt, \;\; r,\rho > 0 \qquad (4.39)$$

The weight r determines how much the control signal should be penalized and the weight ρ determines how much the disturbances influence should be rewarded in the sense of a mini-max differential game. The H_∞ control law is $u(t) = -\frac{1}{r}B^T P_2 \tilde{e}(t)$ where P_2 is the positive definite symmetric matrix derived from the algebraic Riccati equation Eq. (4.27).

The parameter ρ in Eq. (4.39), is an indication of the closed-loop system robustness. If the values of $\rho > 0$ are excessively decreased with respect to r, then the solution of the Riccati equation is no longer a positive definite matrix. Consequently there is a lower bound ρ_{min} of ρ for which the H_∞ control problem has a solution. The acceptable values of ρ lie in the interval $[\rho_{min}, \infty)$. If ρ_{min} is found and used in the design of the H_∞ controller, then the closed-loop system will have increased robustness. Unlike this, if a value $\rho > \rho_{min}$ is used, then an admissible stabilizing H_∞ controller will be derived but it will be a suboptimal one.

The Hamiltonian matrix

$$H = \begin{pmatrix} A & -(\frac{2}{r} - \frac{1}{\rho^2})BB^T \\ -Q & -A^T \end{pmatrix} \qquad (4.40)$$

provides a criterion for the existence of a solution of the Riccati equation Eq. (4.27). A necessary condition for the solution of the algebraic Riccati equation to be a positive semi-definite symmetric matrix is that H has no imaginary eigenvalues [83].

4.3 Application to the Control of Electric Motors

4.3.1 The DC Motor Model

DC motors are widely used in industrial systems, such as robotic manipulators, because their control is relatively simple and they are reliable for a wide range of operating conditions. DC motors are usually modelled as linear systems and then linear control approaches are implemented. However, most linear controllers have unsatisfactory performance due to changes of the motor/load dynamics and due to nonlinearities introduced by the armature reaction. Neglecting the impact of external disturbances and of nonlinearities may risk the stability of the closed-loop system. For the aforementioned reasons DC motor control based on conventional PID or model-based feedback controllers can be inadequate and more effective control approaches are needed. If the nonlinearities of the motor are known functions, then adaptive tracking control methods with the technique of input-output linearization can be used [155],[192]. However, when these nonlinearities or disturbances are unknown, neural or fuzzy control is more suitable for succeeding satisfactory performance of the closed-loop system [146],[313],[358],[359].

Results on the successful application of neural identification and control to DC motor drives have been given in [90],[450],[451], were neural-network controllers for a DC motor were introduced. The unknown nonlinear dynamics of the motor and the external load torque were approximated by a multi-layer neural network. The obtained model was used to generate the control input to the DC motor, following the principles of indirect adaptive control. Several other examples on fuzzy/neural modelling and control of DC motors can be noted. In [205] an identification approach based on Takagi-Sugeno fuzzy models is applied to the DC motor model. In [72] fuzzy logic is applied to the modelling of the dynamics of a DC motor drive, and the obtained model is used to design a controller that compensates for nonlinear disturbances. In [166] a fuzzy logic controller with self-tuning properties was proposed to remove dead-zone effects from DC motor drives. In [199] a self-learning fuzzy logic controller was applied to the position control of a chopper-fed DC servo system. In [268],[358],[360] nonlinear neurocontrollers with online learning capabilities were developed for controlling the speed of brushless DC motors.

The design of nonlinear controllers for high performance servo systems is an ongoing research topic, which can further be advanced using results in the wider area of neuro-fuzzy control [118],[330],[345],[446]. The feasibility of applying neuro-fuzzy networks to model unknown dynamic systems has been demonstrated in several studies. Both state feedback and output feedback linearization methods have been presented [61],[227],[348],[392],[396],[478]. Moreover it has been shown that, neural control based on output feedback controllers and state observers can guarantee the global stability of the closed-loop system [120],[121],[122],[215],[412].

This paper proposes a method for the control of DC motors, which can be applied to linear or nonlinear models, and which is also robust to uncertainties or external disturbances. The paper extends the results of [341],[348]. Two cases can be distinguished: (i) control with feedback of the full state vector, (ii) control using only output feedback. In the first case the closed-loop system consists of the DC motor and an adaptive fuzzy controller based on H_∞ theory [83],[208],[241]. Neuro-fuzzy networks are used to approximate the unknown motor dynamics and subsequently this information is used for the generation of the control signal. In the second case the closed-loop system consists of the DC motor, a state observer that estimates the parameters of the state vector from output measurements, and an adaptive fuzzy H_∞ controller that uses the estimated state vector. Neuro-fuzzy estimators are employed as in the first case to approximate the unknown dynamics of the system, but this time they receive as input the estimated state vector [341]. Moreover, it is shown that the proposed adaptive fuzzy control method can be applied to field-oriented induction motors, following the results of [287],[442],[443],[444].

Comparing to model-based approaches, the advantages of the proposed adaptive fuzzy control are summarized in the following: (i) there is no dependence upon identification of the mathematical model (linear or nonlinear) expressing the dynamics of the DC motor, (ii) since training of the neuro-fuzzy approximators is repeatedly undertaken in every control cycle, any changes to the motor dynamics can be identified online, and hence the control approach is useful for time-varying models, (iii) regarding operation under external disturbances and measurement noise the proposed adaptive fuzzy controller offers improved robustness. Finally, in case that the control is based only on output feedback there is no need to use specific sensors (for instance accelerometers) to measure all elements of the motor's state vector.

4.3.2 State Feedback Controller of the DC Motor Model

A direct current (DC) motor model converts electrical energy into mechanical energy. The torque developed by the motor shaft is proportional to the magnetic flux in the stator field and to the current in the motor armature (iron cored rotor wound with wire coils) [357]. There are two main ways in controlling a DC motor: the first one named *armature control* consists of maintaining the stator magnetic flux constant, and varying (use as control input) the armature current. Its main advantage is a good torque at high speeds and its disadvantage is high energy losses. The second way is called *field control*, and has a constant voltage to set up the armature current, while a variable voltage applied to the stator induces a variable magnetic flux. Its advantages are energy efficiency, inexpensive controllers and its disadvantages are a torque that decreases at high speeds. A linear model that approximates the dynamics of the DC motor is derived as follows: the torque developed by the motor is proportional to the stator's flux and to the armature's current thus one has

$$\Gamma = k_f \Psi K_\alpha I \tag{4.41}$$

where Γ is the shaft torque, Ψ is the magnetic flux in the stator field, I is the current in the motor armature. Since the flux is maintained constant the torque of Eq. (4.41) can be written as

$$\Gamma = k_T I, \text{ where } k_T = k_f \Psi K_\alpha \tag{4.42}$$

Apart from this, when a current carrying conductor passes through a magnetic field, a voltage V_b appears corresponding to what is called electromagnetic force (EMF)

$$V_b = k_e \omega \tag{4.43}$$

where ω is the rotation speed of the motor shaft. The constants k_T and k_e have the same value. Kirchhoff's law yields the equation of the motor (Fig. 4.4):

$$V - V_{\text{res}} - V_{\text{coil}} - V_b = 0 \tag{4.44}$$

where V is the input voltage, $V_{\text{res}} = RI$ is the armature resistor voltage (R denotes the armature resistor), $V_{\text{coil}} = L\dot{I}$ is the armature inductance voltage. The motor's electric equation is then

$$L\dot{I} = -k_e \omega - RI + V \tag{4.45}$$

From the mechanics of rotation it holds that

$$J\dot{\omega} = \Gamma - \Gamma_{\text{damp}} - \Gamma_d \tag{4.46}$$

The DC motor model is finally

$$\begin{aligned} L\dot{I} &= -k_e\omega - RI + V \\ J\dot{\omega} &= k_e I - k_d\omega - \Gamma_d \end{aligned} \tag{4.47}$$

with the following notations

Notation	Significance
L	armature inductance
I	armature current
k_e	motor electrical constant
R	armature resistance
V	input voltage, taken as control input
J	motor inertia
ω	rotor rotation speed
k_d	mechanical dumping constant
Γ_d	disturbance torque

where the armature designates the iron cored rotor wound with wired coils. Assuming $\Gamma_d = 0$ and denoting the state vector as $[x_1, x_2, x_3]^T = [\theta, \dot{\theta}, \ddot{\theta}]^T$, a linear model of the DC motor is obtained:

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & \frac{-k_e^2 - k_d R}{JL} & \frac{-JR - k_d L}{JL} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \frac{k_e}{JL} \end{pmatrix} V \tag{4.48}$$

Usually the DC-motor model is considered to be linear by neglecting the effect of armature reaction or by assuming that the compensating windings remove this effect. Introducing the armature reaction leads to a nonlinear system and in that case a nonlinear model may be appropriate. In that case the dynamic model of the DC-motor model can be written as [146]:

$$\dot{x} = f(x) + g(x)u \tag{4.49}$$

with $\dot{x}$ denoting the derivative of the motor's state vector, $x = [x_1, x_2, x_3]^T = [\theta, \dot{\theta}, i_\alpha]$ (θ is the position of the motor, $\dot{\theta}$ is the angular velocity of the motor and i_α is the armature current). Functions $f(x)$ and $g(x)$ are vector field functions defined as:

$$f(x) = \begin{pmatrix} x_2 \\ k_1 x_2 + k_2 x_3 + k_3 x_3^2 + k_4 T_1 \\ k_5 x_2 + k_6 x_2 x_3 + k_7 x_3 \end{pmatrix}, g(x) = \begin{pmatrix} 0 \\ 0 \\ k_8 \end{pmatrix} \tag{4.50}$$

where $k_1 = -F/J$, $k_2 = A/J$, $k_3 = B/J$, $k_4 = -1/J$, $k_5 = -A/L$, $k_6 = -B/L$, $k_7 = -R/L$, $k_8 = -1/L$, R and L are the armature resistance and induction respectively, and J is the rotor's inertia, while F is the friction.

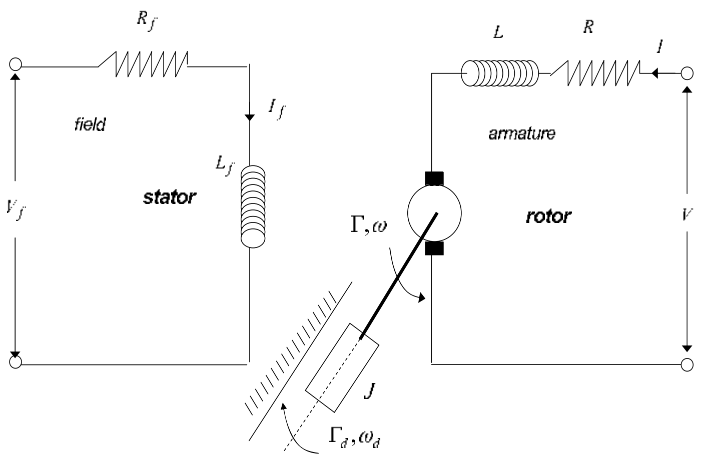


Fig. 4.4 Parameters of the DC motor model

Now choosing the motor's angle to be the system output, the state space equation of the DC motor can be rewritten as

$$\begin{aligned}\dot{x}_1 &= x_2\\ \dot{x}_2 &= k_1x_2+k_2x_3+k_3x_3^2+k_4T_1\\ \dot{x}_3 &= k_5x_2+k_6x_2x_3+k_7x_3+k_8u\end{aligned} \tag{4.51}$$

$$y = x_1$$

where T_1 the load torque and u is the terminal voltage. From the second row of Eq. (4.51) one obtains,

$$\begin{aligned}\ddot{x}_2 &= k_1\dot{x}_2+k_2\dot{x}_3+2k_3x_3\dot{x}_3 \Rightarrow\\ \ddot{x}_2 &= k_1\dot{x}_2+k_2\dot{x}_3+2k_3k_5x_2x_3+2k_3k_6x_2x_3^2+2k_3k_7x_3^2+2k_3k_8x_3u\end{aligned} \tag{4.52}$$

Thus the input-output relation can be written as

$$\ddot{x}_2 = \bar{f}(x)+\bar{g}(x)u \tag{4.53}$$

where, $\bar{f}(x) = k_1\dot{x}_2+k_2\dot{x}_3+2k_3k_5x_2x_3+2k_3k_6x_2x_3^2+2k_3k_7x_3^2$, and $\bar{g}(x) = 2k_3k_8x_3$. The control approach that will be developed in this paper is model-free and generic and can be applied to both linear and nonlinear models.

4.3.3 State Feedback Controller for the DC Motor

The performance of the previously analyzed adaptive fuzzy controller with full-state feedback, when applied to the DC motor model, is tested in the tracking of several reference trajectories. The time step of the simulation experiments was taken to be $T_s = 0.01$ sec [338].

For $r = 1.0$ and $\rho = 1.0$ the Riccati equation given in Eq. (4.13) was solved. The basis functions used in the estimation of $f(x,t)$ and $g(x,t)$ were $\mu_{A_j}(\hat{x}) = e^{(\frac{\hat{x}-c_j}{\sigma})^2}$, $j = 1,\cdots,3$. In the associated fuzzy rule base there are three inputs $x_1 = \theta$, $\dot{x}_1 = \dot{\theta}$ and $\ddot{x}_1 = \ddot{\theta}$. The universe of discourse of each input variable consisted of 3 fuzzy sets. Consequently 27 fuzzy rules were derived which had the following form:

$$R^l : IF \ x_1 \ is \ A_1^l \ AND \ \dot{x}_1 \ is \ A_2^l \ AND \ \ddot{x}_1 \ is \ A_3^l \ THEN \ \hat{f}^l \ is \ b^l \tag{4.54}$$

and the approximation of function $f(x,t)$ in the motor's model of Eq. (4.49) was given by

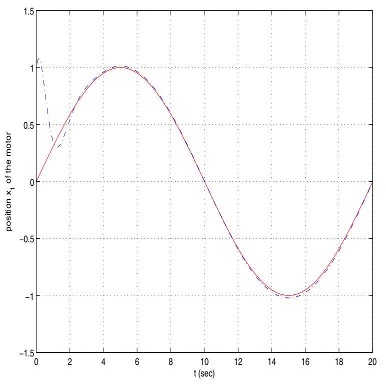
$$\hat{f}(x,t) = \frac{\sum_{l=1}^{27} \hat{f}^l \prod_{i=1}^{3} \mu_{A_i}^l(x_i)}{\sum_{l=1}^{27} \prod_{i=1}^{3} \mu_{A_i}^l(x_i)} \tag{4.55}$$

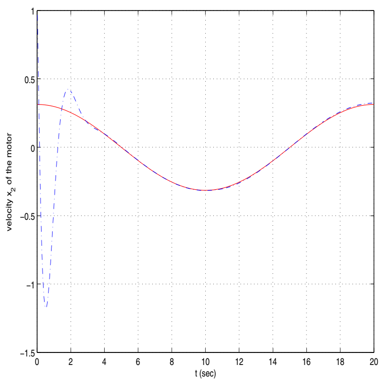
The centers $c_i^{(l)}$, $i = 1,\cdots,3$ can take values in the set $\{-1.0, 0.0, 1.0\}$ while the variances $v_i^{(l)}$, $i = 1,\cdots,3$ were given the value $v_i^{(l)} = 2.2$. Thus taking the possible combinations the following R^l, $l = 1,\cdots,27$ are derived where the associated centers and variances are defined as:

Rule	$c_1^{(l)}$	$c_2^{(l)}$	$c_3^{(l)}$	$v^{(l)}$
$R^{(1)}$	-1.0	-1.0	-1.0	2.2
$R^{(2)}$	-1.0	-1.0	0.0	2.2
$R^{(3)}$	-1.0	-1.0	1.0	2.2
$R^{(4)}$	-1.0	0.0	-1.0	2.2
$R^{(5)}$	-1.0	0.0	0.0	2.2
$R^{(6)}$	-1.0	0.0	1.0	2.2
...	...	...	...	...
...	...	...	...	...
$R^{(27)}$	1.0	1.0	1	1

Similar was the fuzzy rule base that was used in the approximation of function $g(x,t)$ of Eq. (4.49). The learning rates γ_1 and γ_2 of the neurofuzzy networks were suitably tuned. The controller's gain $K = [k_0, k_1, k_2]^T$ was suitably selected so as to result in a Hurwitz stable polynomial and to assure the asymptotic convergence of the tracking error to zero. In the first half of the simulation time the training of the neuro-fuzzy approximators was carried out. In the second half, the estimated functions $\hat{f}(x,t)$ and $\hat{g}(x,t)$ were used to derive the control signal. First the performance of the proposed state feedback controller was tested in the tracking of a sinusoidal set-point.

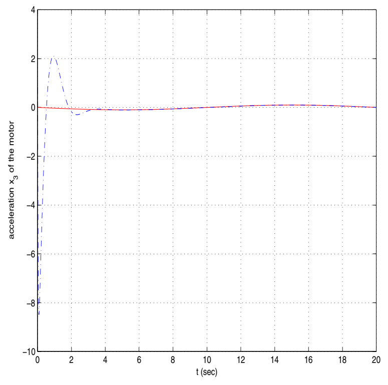
- The position and velocity variations for a sinuoidal set-point are depicted in Fig. 4.5(a) and Fig. 4.5(b), respectively.

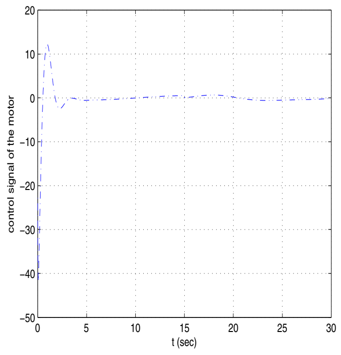




(a) **(b)**

Fig. 4.5 Full state feedback control of the DC motor: (a) state x_1 (dashed line) tracks a sinusoidal set-point (continuous line) (b) x_2 (dashed line) tracks a sinusoidal set-point (continuous line)





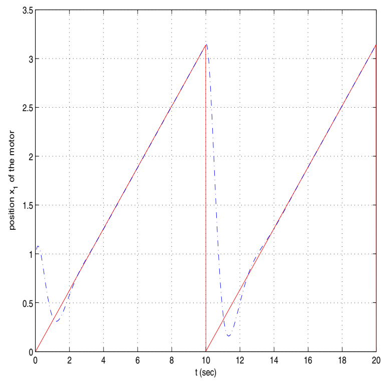
(a) **(b)**

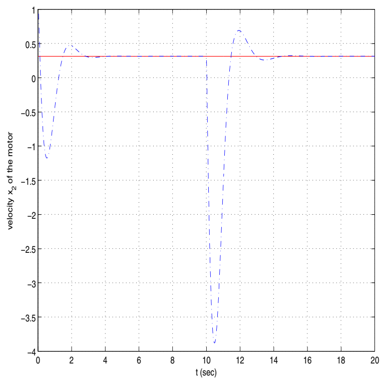
Fig. 4.6 Full state feedback control of the DC motor: (a) state x_3 (dashed line) tracks a sinusoidal set-point (continuous line) (b) control signal (dashed line) for the dc-motor

- The acceleration tracking succeeded for the sinusoidal set-point is shown in Fig. 4.6(a), while the associated control input is shown in Fig. 4.6(b)

The simulation tests were also extended to the tracking of a see-saw set-point.

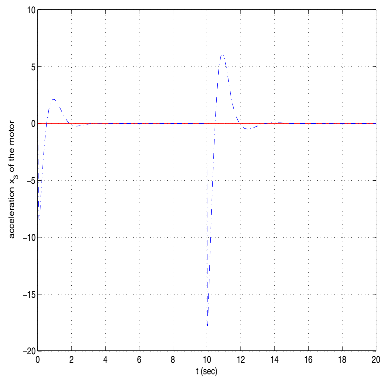
- The position and velocity variations for a see-saw set-point are depicted in Fig. 4.7(a) and Fig. 4.7(b), respectively.
- The acceleration tracking succeeded for the see-saw set-point is shown in Fig. 4.8(a), while the associated control input is shown in Fig. 4.8(b)

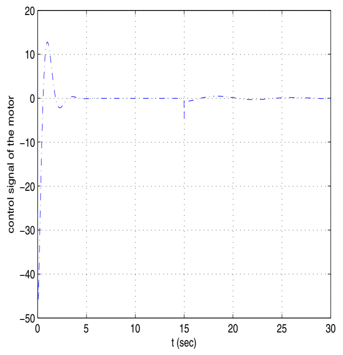




(a) **(b)**

Fig. 4.7 Full state feedback control of the DC motor: (a) state x_1 (dashed line) tracks a see-saw set-point (continuous line) (b) state x_2 (dashed line) tracks the associated set-point (continuous line)





(a) **(b)**

Fig. 4.8 Full state feedback control of the DC motor for see-saw reference output: (a) state x_3 (dashed line) tracks the associated set-point (continuous line) (b) control signal (dashed line) for the dc-motor

From the simulation tests the following remarks can be made: (i) adaptive fuzzy H_∞ control based on state feedback succeeds excellent tracking of the reference motor's angle θ_d. Overshoot depends on the selection of the feedback gain K, (ii) Excellent tracking of the reference angular velocity $\dot{\theta}_d$ is also achieved, (iii) The variation of the control input (field voltage) is smooth. This was due to the proper selection of the feedback gain K, (iv) The neuro-fuzzy networks can succeed good approximations of the unknown functions $f(x,t)$ and $g(x,t)$. The accuracy in the estimation of $g(x,t)$ is important for the convergence of the control algorithm.

4.3.4 Output Feedback Controller for the DC Motor

The performance of the output feedback controller was also tested in the tracking of several set-points. The time step was again taken to be $T_s = 0.01$ sec.

The controller's feedback gain $K = [k_0, k_1, k_2]^T$ and the observer's gain $K_o = [k_{o_0}, k_{o_1}, k_{o_2}]^T$ were suitably selected so as to assure the asymptotic elimination of the tracking and observation errors respectively. The basis functions used in the estimation of $f(x,t)$ and $g(x,t)$ were $\mu_{A_j}(\hat{x}) = e^{(\frac{\hat{x}-c_j}{\sigma})^2}, j = 1, \cdots, 3$. Since there were three inputs $\hat{x}_1$, $\dot{\hat{x}}_1$ and $\ddot{\hat{x}}_1$ and the associated universes of discourse consisted of 3 fuzzy sets there were again 27 fuzzy rules of the form:

$$R^l : IF \ \hat{x}_1 \ is \ A_1^l \ AND \ \dot{\hat{x}}_1 \ is \ A_2^l \ AND \ \ddot{\hat{x}}_1 \ is \ A_3^l \ THEN \ \hat{f}^l \ is \ b^l \tag{4.56}$$

and

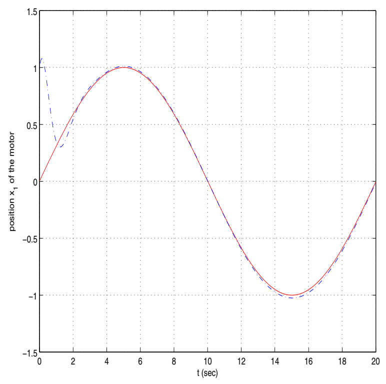
$$\hat{f}(\hat{x},t) = \frac{\sum_{l=1}^{27} \hat{f}^l \prod_{i=1}^{2} \mu_{A_i}^l(\hat{x}_i)}{\sum_{l=1}^{27} \prod_{i=1}^{2} \mu_{A_i}^l(\hat{x}_i)}. \tag{4.57}$$

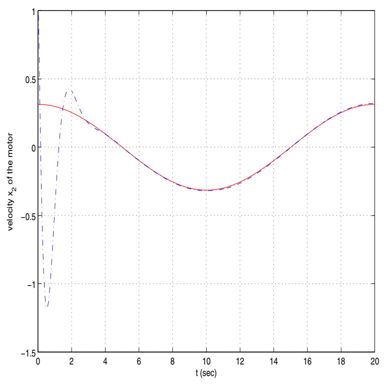
The centers c_i^l, $i = 1, \cdots, 3$ take values from the set $\{-1.0, 0.0, 1.0\}$ while the variance of the fuzzy sets v_i^l, $i = 1, \cdots, 3$ is given again the value 1. Thus, the centers $c_i^{(l)}$, $i = 1, 2, 3$ and the variances $v^{(l)}$ of each rule are as follows

Rule	$c_1^{(l)}$	$c_2^{(l)}$	$c_3^{(l)}$	$v^{(l)}$
$R^{(1)}$	-1.0	-1.0	-1.0	2.2
$R^{(2)}$	-1.0	-1.0	0.0	2.2
$R^{(3)}$	-1.0	-1.0	1.0	2.2
$R^{(4)}$	-1.0	-0.0	-1.0	2.2
$R^{(5)}$	-1.0	-0.0	0.0	2.2
$R^{(6)}$	-1.0	-0.0	1.0	2.2
...	...	...	...	...
...	...	...	...	...
$R^{(27)}$	1.0	1.0	1.0	2.2

Similar was the fuzzy rule base that provided the approximation of function $g(x,t)$ of Eq. (4.49). The first half of the simulation time was used for training the neuro-fuzzy approximators and a measurable state vector was used. Matrices P_1 and P_2 were obtained from the solution of the Riccati equation given in Eq. (4.27). First, the proposed controller was used for tracking a sinusoidal set-point:

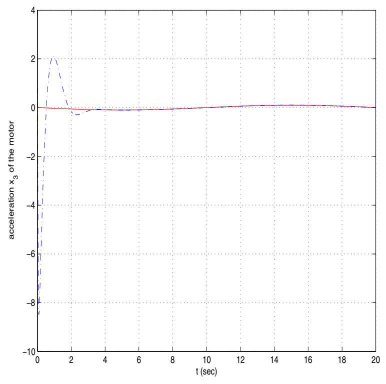
- The position and velocity tracking succeeded in the case of the sinusoidal set-point are depicted in Fig. 4.9(a) and Fig. 4.9(b), respectively.
- The acceleration tracking succeeded for the sinusoidal set-point is shown in Fig. 4.10(a), while the associated control input is shown in Fig. 4.10(b).

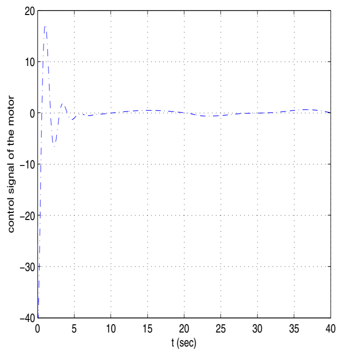




(a) **(b)**

Fig. 4.9 Control of the DC motor using output feedback (a) state x_1 (dashed line) tracks a sinusoidal set-point (continuous line) (b) state x_2 (dashed line) tracks a sinusoidal set-point (continuous line)



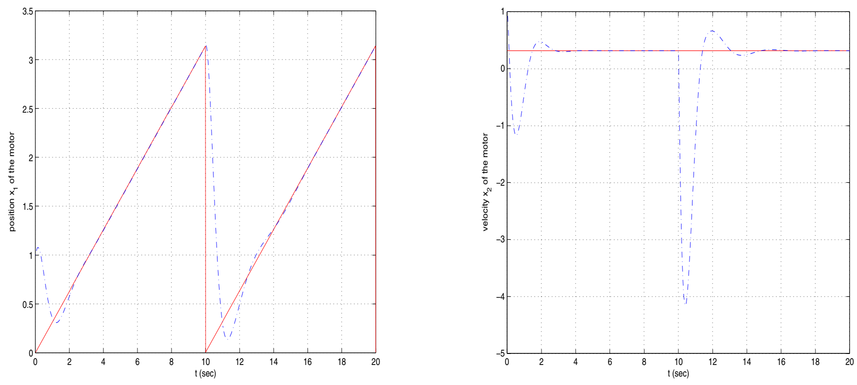


(a) **(b)**

Fig. 4.10 Control of the DC motor using output feedback (a) state x_3 (dashed line) tracks a sinusoidal set-point (continuous line) (b) control signal (dashed line) for the dc-motor

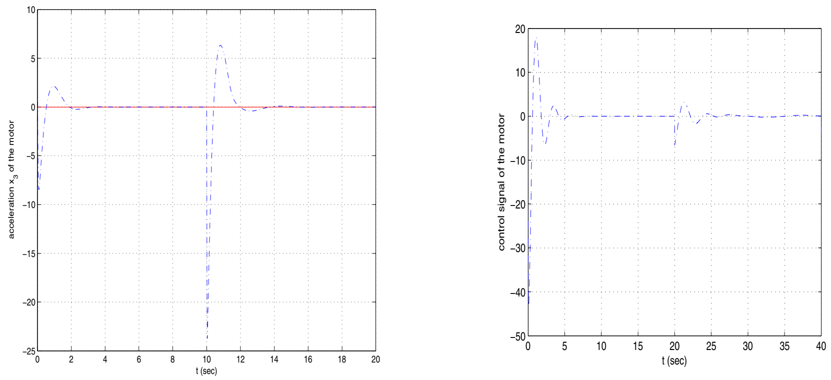
The simulation experiments for the adaptive fuzzy control based on output feedback were also extended to the tracking of a see-saw set-point

- The position and velocity tracking succeeded in the case of the see-saw set-point are depicted in Fig. 4.11(a) and Fig. 4.11(b), respectively.
- The acceleration tracking succeeded for the see-saw set-point is shown in Fig. 4.12(a) associated control input is shown in Fig. 4.12(b)



(a) **(b)**

Fig. 4.11 Control of the DC motor using output feedback (a) state x_1 (dashed line) tracks a see-saw set-point (continuous line) (b)state x_2 (dashed line) tracks the associated set-point (continuous line)



(a) **(b)**

Fig. 4.12 Control of the DC motor using output feedback for see-saw reference output (a) state x_3 (dashed line) tracks the associated set-point (continuous line) (b) control signal (dashed line) for the dc-motor

The estimations succeeded for the state $x_1 = \theta$ of the motor in the case of the sinusoidal and the see-saw set-point are given in Fig. 4.13(a) and (b) respectively. Finally, the approximation of the DC motor dynamics (function $\hat{g}(x,t)$) by the neural-fuzzy approximators, in the case of output feedback, is depicted in Fig. 4.14(a) and (b).

Adaptive fuzzy H_∞ control with output feedback has the same advantages as adaptive fuzzy control with state feedback. These are summarized in the following: (i) removal of any dependence upon identification of the mathematical model

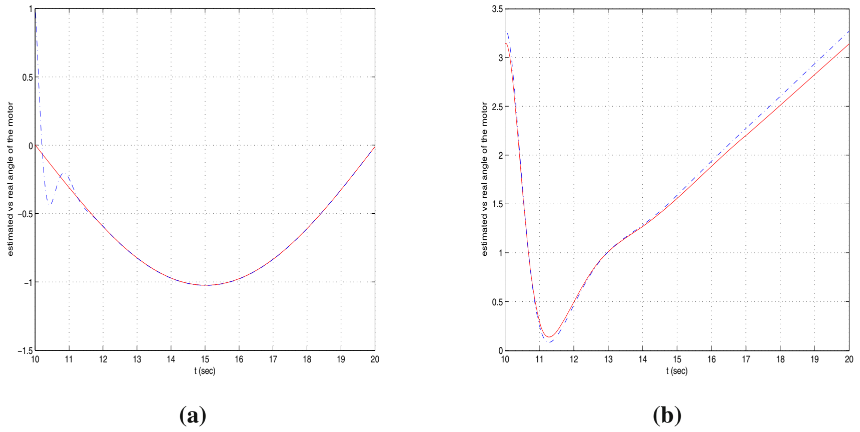


Fig. 4.13 Control of the DC motor with output feedback: estimated (dashed line) vs real value (continuous line) of the angle θ in the case of (a) sinusoidal set-point (b) see-saw set-point

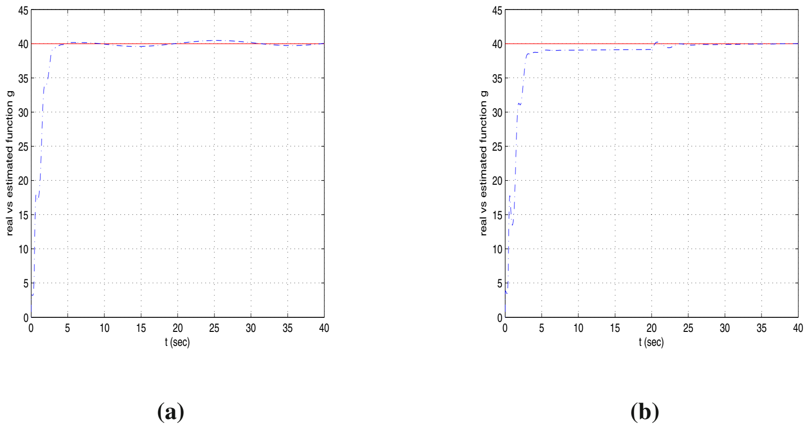


Fig. 4.14 Control of the DC motor with output feedback: approximation of the DC motor dynamics (function $\hat{g}(x,t)$) when tracking (a) a sinusoidal set-point, (b) a see-saw set-point.

expressing the dynamics of the motor, (ii) since training of the neurofuzzy approximators contained in the adaptive fuzzy H_∞ controller is repeatedly undertaken in every control cycle, any changes to the motor dynamics can be identified on-line, and hence the control strategy is useful for time varying motor models, (iii) regarding operation under external disturbances and measurement noise, robustness of the closed loop is succeeded. Moreover, it should be noted that in the case of adaptive fuzzy control with output feedback there is no need to use additional sensors to measure the velocity and the acceleration of the motor, since the state vector is reconstructed with the use of an observer.

4.3.5 Application to the Field-Oriented Induction Motor

4.3.5.1 The Model of the Induction Motor

With the field-oriented method, the dynamic behavior of the induction motor is rather similar to that of a separately excited DC motor [287],[442],[443]. A decoupled relationship is obtained by means of a proper selection of state coordinates and thus, the rotor speed is asympotically decoupled from the rotor flux, while the speed can be controlled only by the torque current. However, the control performance of the induction motor is still influenced by the uncertainties of the motor's dynamic model, such as mechanical parameters uncertainty, and external load disturbance [40],[229],[254],[443]. To compensate for these uncertainties and to control effectively the field-oriented induction motor the adaptive fuzzy controller analyzed in the previous sections is proposed.

To derive the dynamic model of an induction motor the three-phase variables are first transformed to two-phase ones. This two-phase system can be described in the stator-coordinates frame $\alpha - b$, and the associated voltages are denoted as v_{s_α} and v_{s_b}, while the currents of the stator are i_{s_α} and i_{s_b}, and the components of the rotor's magnetic flux are ψ_{r_α} and ψ_{r_b}. Then, the rotation angle of the rotor with respect to the stator is denoted by δ. Next, the rotating reference frame $d - q$ on rotor, is defined (Fig. 4.15).

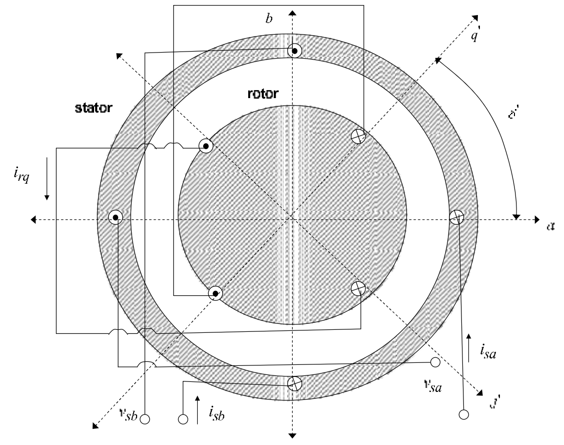


Fig. 4.15 AC motor circuit, with the $a - b$ stator reference frame and the $d - q$ rotor reference frame

The state vector of the motor is defined as $x = [\theta, \omega, \psi_{r_\alpha}, \psi_{r_b}, i_{s_\alpha}, i_{s_b}]$ and the dynamic model of the induction motor is written as [5]:

$$\dot{x} = f(x) + g_\alpha(x) v_{s_\alpha} + g_b(x) v_{s_b} \tag{4.58}$$

with

$$f(x) = \begin{pmatrix} x_2 \\ \mu_1(x_3x_6 - x_4x_5) - \frac{T_L}{J} \\ \alpha_1 x_3 - n_p x_2 x_4 + \alpha_1 M x_5 \\ n_p x_2 x_3 - \alpha_1 x_4 + \alpha_1 M x_6 \\ \alpha_1 \beta_1 x_3 + n_p \beta_1 x_2 x_4 - \gamma x_5 \\ -n_p \beta_1 x_2 x_3 + \alpha_1 \beta_1 x_4 - \gamma_1 x_6 \end{pmatrix} \tag{4.59}$$

$$g_\alpha = [0,0,0,0,\tfrac{1}{\sigma L_s},0]^T \ g_b = [0,0,0,0,0,\tfrac{1}{\sigma L_s}]^T \tag{4.60}$$

J is the rotor's inertia, and T_L is the external load torque. The rest of the model parameters are $\alpha_1 = \frac{R_r}{L_r}$, $\beta_1 = \frac{M}{\sigma L_s L_r}$, $\gamma_1 = (\frac{M^2 R_r}{\sigma L_s L_r^2} + \frac{R_s}{\sigma L_s})$, $\mu_1 = \frac{n_p M}{J L_r}$, where L_s, L_r are the stator and rotor autoinductances, M is the mutual inductance and n_p is the number of poles.

4.3.5.2 Field-Oriented Control of the Induction Motor

The classical method for induction motors control is based on a transformation of the stator's currents (i_{s_α} and i_{s_b}) and of the magnetic fluxes of the rotor (ψ_{r_α} and ψ_{r_b}) to the reference frame d_q which rotates together with the rotor (Fig. 4.15). In the $d-q$ frame there will be only one non-zero component of the magnetic flux ψ_{r_d}, while the component of the flux along the q axis equals 0 [340]. The new inputs of the system are considered to be v_{s_d}, v_{s_q}, which are connected to v_{s_α}, v_{s_b} of Eq. (4.58) according to the relation

$$\begin{pmatrix} v_{s\alpha} \\ v_{sb} \end{pmatrix} = ||\psi|| \cdot \begin{pmatrix} \psi_{s\alpha} & \psi_{sb} \\ \psi_{sb} & \psi_{s\alpha} \end{pmatrix}^{-1} \begin{pmatrix} v_{sd} \\ v_{sq} \end{pmatrix} \tag{4.61}$$

where $\psi = \psi_{r_d}$ and $||\psi|| = \sqrt{\psi_{s_\alpha}^2 + \psi_{s_b}^2}$. Next, the following nonlinear feedback control law is defined

$$\begin{pmatrix} v_{sd} \\ v_{sq} \end{pmatrix} = \sigma L_s \begin{pmatrix} -n_p \omega i_{sq} - \frac{\alpha M {i_{sq}}^2}{\psi_{rd}} - \alpha b \psi_{rd} + v_d \\ n_p \omega i_{sd} + b n_p \omega \psi_{rd} + \frac{\alpha M i_{sq} i_{sd}}{\psi_{rd}} + v_q \end{pmatrix} \tag{4.62}$$

The control signal in the coordinates system $\alpha - b$ is

$$\begin{pmatrix} v_{s\alpha} \\ v_{sb} \end{pmatrix} = ||\psi|| \sigma L_s \begin{pmatrix} \psi_{s\alpha} & \psi_{sb} \\ -\psi_{sb} & \psi_{s\alpha} \end{pmatrix} \cdot^{-1} \begin{pmatrix} -n_p \omega i_{sq} - \frac{\alpha M {i_{sq}}^2}{\psi_{rd}} - \alpha \beta \psi_{rd} + v_d \\ n_p \omega i_{sd} + \beta n_p \omega \psi_{rd} + \frac{\alpha M i_{sq} i_{sd}}{\psi_{rd}} + v_q \end{pmatrix} \tag{4.63}$$

Substituting Eq. (4.63) into Eq. (4.58) one obtains

$$\frac{d}{dt} = \mu \psi_{rd} i_{sq} - \frac{T_L}{J} \tag{4.64}$$

$$\frac{d}{dt}i_{sq} = -\gamma i_{sq} + v_q \tag{4.65}$$

$$\frac{d}{dt}\psi_{rd} = -\alpha\psi_{rd} + \alpha M i_{sd} \tag{4.66}$$

$$\frac{d}{dt}i_{sd} = -\gamma i_{sd} + v_d \tag{4.67}$$

$$\frac{d}{dt}\rho = n_p\omega + \alpha M\frac{i_{sq}}{\psi_{rd}} \tag{4.68}$$

The system of Eq. (4.64) to Eq. (4.68) consists of two linear subsystems, where the first one has as output the magnetic flux ψ_{rd} and the second has as output the rotation speed ω, i.e.

$$\frac{d}{dt}\psi_{rd} = -\alpha\psi_{rd} + \alpha M i_{sd} \tag{4.69}$$

$$\frac{d}{dt}i_{sd} = -\gamma i_{sd} + v_d \tag{4.70}$$

$$\frac{d}{dt}\omega = \mu\psi_{rd}i_{sq} - \frac{T_L}{J} \tag{4.71}$$

$$\frac{d}{dt}i_{sq} = -\gamma i_{sq} + v_q \tag{4.72}$$

If $\psi_{rd}\rightarrow\psi_{rd}^{\text{ref}}$, i.e. the transient phenomena for ψ_{rd} have been eliminated and therefore ψ_{rd} has converged to a steady state value then Eq. (4.71) is decoupled ψ_{rd}, then the two subsystems described by Eq. (4.69)-(4.70) and Eq. (4.71)-(4.72) are decoupled.

The subsystem that is described by Eq. (4.69) and Eq. (4.70) is linear and has as control input v_{sd}, and can be controlled using methods of linear control, such as optimal control, or PID control. For instance the following PI controller has been proposed for the control of the magnetic flux

$$v_d(t) = -k_{d1}(\psi_{rd} - \psi_{rd}^{\text{ref}}) - k_{d2}\int_0^t(\psi_{rd}(\tau) - \psi_{rd}{}^{\text{ref}}(\tau))d\tau \tag{4.73}$$

If Eq. (4.73) is applied to the subsystem that is described by Eq. (4.69) and Eq. (4.70), then one can succeed $\psi_{rd}(t)\rightarrow\psi_{rd}^{\text{ref}}(t)$. If $\psi_{rd}(t)$ is not sufficiently measurable using Hall sensors then it can be reconstructed using some kind of observer.

Now the subsystem that consists of Eq. (4.71) and Eq. (4.72) is examined. The term $T = \mu\psi_{rd}{}^{\text{ref}}i_{sq}$ denotes the torque developed by the motor. The above mentioned subsystem is a model equivalent to that of a DC motor and thus after succeeding $\psi_{rd}\rightarrow\psi_{rd}^{\text{ref}}$, one can also control the motor's speed ω, using control algorithms already applied to the control of DC motors.

4.3.5.3 Adaptive Fuzzy Control for the Decoupled Induction Motor

It is assumed that the motor rotates a rigid link of length l, while a lumped mass m is attached to the end of the link. Then the model of Eq. (4.71) and Eq. (4.72) can be written as [340]

$$J^{'}\frac{d}{dt}\omega = \mu^{'}\psi_{rd}^{ref}i_{s_q} - b\dot{\theta} - mglsin(\theta) \tag{4.74}$$

$$\frac{d}{dt}i_{s_q} = -\gamma i_{s_q} + v_d \tag{4.75}$$

where $J^{'} = J + ml^2$ is the total inertia, $b^{'}$ is the friction coefficient, and $\mu^{'} = \frac{n_p}{J^{'}}\frac{M}{L_r}$. The above nonlinear model can be written as $J^{'}\ddot{\theta} = H_N(\theta) + Ki_{s_q}$ which is of the form

$$\ddot{x} = f(x) + g(x)u \tag{4.76}$$

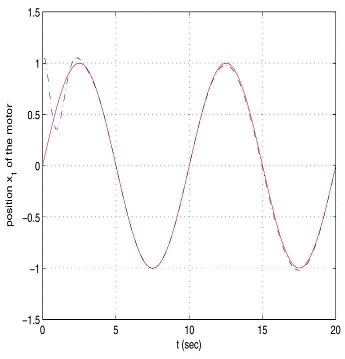
with $f(x) = \frac{H_N(\theta)}{J^{'}}$ and $g(x) = \frac{K}{J^{'}}$. The objective is to force the system's output (angle of the motor) $y = x$ to follow a given bounded reference signal x_d. The decoupled model of the induction motor can be controlled using the adaptive fuzzy H_∞ controller that was analyzed in the previous sections.

Again, in the presence of non-gaussian disturbances w, successful tracking of the reference signal is denoted by the H_∞ criterion of Eq. (4.2). As in the case of the DC motor, for measurable state vector and uncertain functions $f(x,t)$ and $g(x,t)$ an appropriate control law for Eq. (4.76) is again given by Eq. (4.3), i.e. $u = \frac{1}{\hat{g}(x,t)}[x_d^{(n)} - \hat{f}(x,t) + K^T e + u_c]$. The error vector is given by $e^T = [e, \dot{e}, \ddot{e}, \cdots, e^{(n-1)}]^T$, and the feedback gain $K^T = [k_n, k_{n-1}, \cdots, k_1]$, is selected so as the polynomial $e^{(n)} + k_1 e^{(n-1)} + k_2 e^{(n-2)} + \cdots + k_n e$ to be Hurwitz. The control law of Eq. (4.3) results into the error dynamics described in Eq. (4.4) to Eq. (4.8), which describe a regulation problem. The control signal u_c is the H_∞ control term, used for the compensation of $\tilde{d}$ and w, as defined in Eq. (4.9), i.e $u_c = -\frac{1}{r}B^T Pe$.

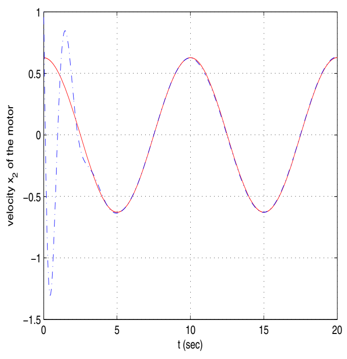
The adaptive fuzzy controller described above was used in the control of the field-oriented induction motor, while matrix P in the control law of Eq. (4.9) was found from Eq. (4.13) . The following controller parameters were used: the feedback gain was $K = [10, 2.65]^T$ and for $r = 0.1$ and $\rho = 0.235$ the solution of Eq. (4.13) was

$$P = \begin{pmatrix} 12 & 2 \\ 2 & 0.8 \end{pmatrix}, \quad Q = \begin{pmatrix} 47.57 & 4.32 \\ 4.32 & 1.45 \end{pmatrix} \tag{4.77}$$

Finally, the learning rates of the neurofuzzy approximators were set to $\gamma_1 = 0.025$, and $\gamma_2 = 0.235$. The results of set-point tracking are given in Fig. 4.16 to 4.17. It can be observed that although prior knowledge of the induction motor's dynamic model was not available, the adaptive fuzzy H_∞ controller resulted in satisfactory tracking of the reference trajectory.

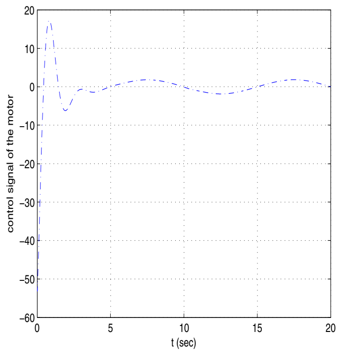


(a)

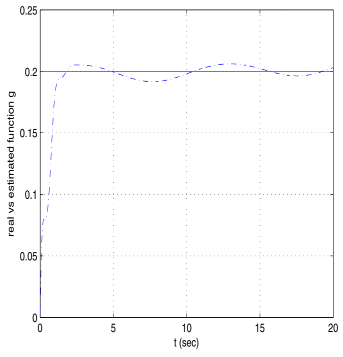


(b)

Fig. 4.16 Adaptive fuzzy control: (a) state variable $x_1 = \omega_r$ of the motor (dashed line), (b) state variable $x_2 = i_{s_q}$ of the motor (dashed line)



(a)



(b)

Fig. 4.17 Adaptive fuzzy control: (a) control input (dashed line) to the field-oriented induction motor, (b) approximation of function $g(x,t)$ by the fuzzy neural network

4.4 Application to the Ship Steering Control Problem

Using the previously analyzed methodology an adaptive fuzzy controller is designed for the ship steering problem described in Chapter 3. The performance of adaptive fuzzy H_∞ control is depicted in Fig. 4.18 [339]. The control input to the ship model generated by adaptive fuzzy H_∞ control and nonlinear backstepping control is depicted in Fig. 4.22(b).

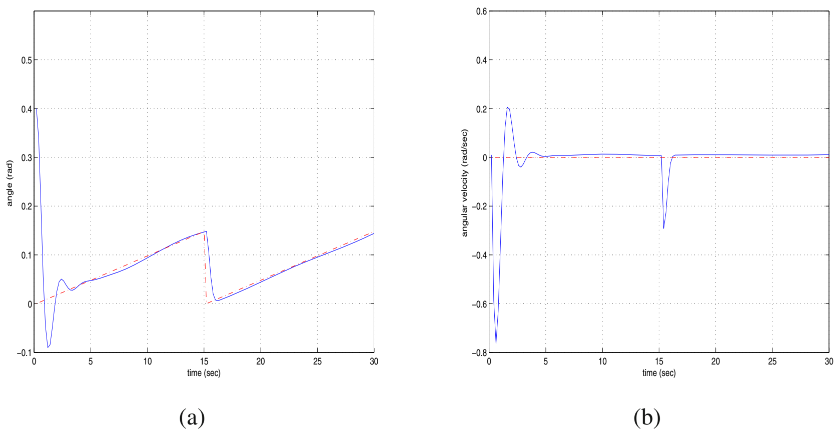


Fig. 4.18 (a) Heading ψ (continuous line) and desirable heading ψ_d (dashed line) of the ship when tracking a see-saw setpoint using adaptive fuzzy H_∞ control (b) Angular velocity $\dot{\psi}$ (continuous line) and desirable desirable angular velocity $\dot{\psi}_d$ (dashed line) of the ship when tracking a see-saw set-point using adaptive fuzzy H_∞ control.

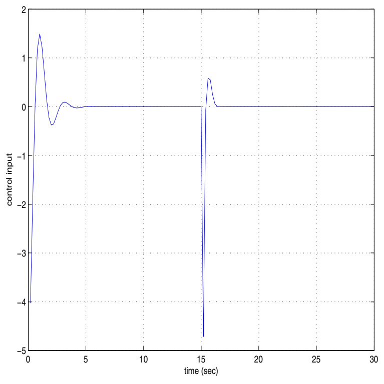


Fig. 4.19 Control input (rudder angle δ) of the ship when tracking a see-saw set-point using adaptive fuzzy H_∞ control

The approximation of the unknown nonlinear functions $\hat{f}(x,t)$ and $\hat{g}(x,t)$ that were obtained by the neuro-fuzzy approximators in the case of a see-saw setpoint using adaptive fuzzy H_∞ control are shown in Fig. 4.20.

The performance of adaptive fuzzy H_∞ control was also examined in the case of a sinuoidal set-point. The simulation results are given in Fig. 4.21. To further increase the simulation difficulty, measurement noise was considered. Specifically, a random sequence was added to the measurement of the ship's heading angle ψ and to the ship's velocity $\dot{\psi}$. The noise in the measurement of ψ varied in the interval $[-1.0^o, +1.0^o]$, while the noise in the measurement of the change of the ship's

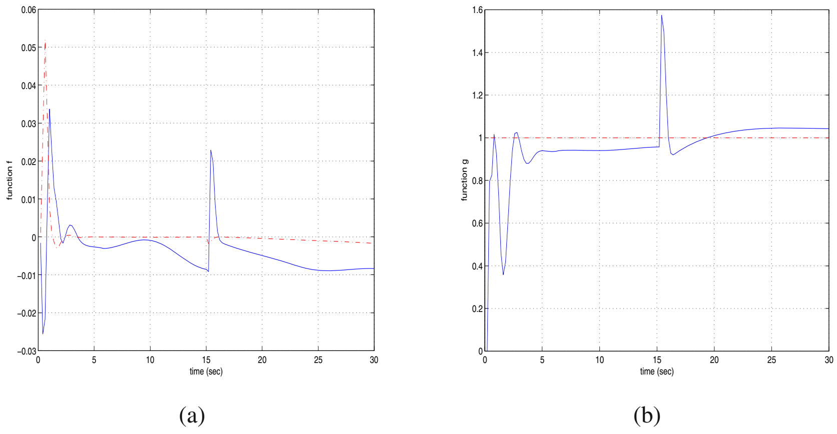


Fig. 4.20 Tracking of a see-saw setpoint with adaptive fuzzy H_∞ control (a) approximation $\hat{f}(x,t)$ (continuous line) of function $f(x,t)$ (dashed line) (b) approximation $\hat{g}(x,t)$ (continuous line) of function $g(x,t)$ (dashed line).

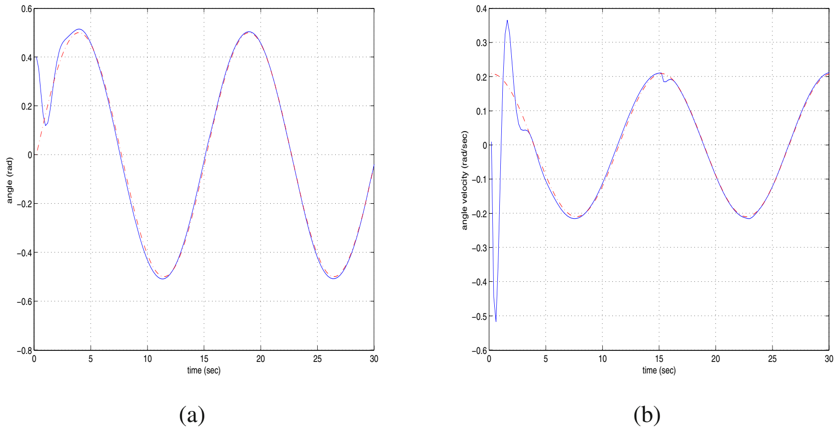


Fig. 4.21 Tracking of a sinusoidal setpoint (dashed line) with adaptive fuzzy H_∞ control (a) angle ψ (b) angular velocity $\dot{\psi}$.

heading $\dot{\psi}$ varied in the interval $[-0.3^o/sec, 0.3^o/sec]$. The tracking of a sinusoidal setpoint in the presence of measurement noise is depicted in Fig. 4.22(a).

From the simulation tests given in the sequel, as well as from the simulation results on ship steering with the use of backstepping control which have been presented in subsection 3.5.4 the following remarks can be made: (i) In Fig. 4.18(a) it can be observed that adaptive fuzzy H_∞ control succeeds tracking of the reference angle ϕ_d that is comparable to that of nonlinear back-stepping control. Overshooting depends on the selection of the feedback gain K. (ii) From Fig. 4.18(b) it is clear that

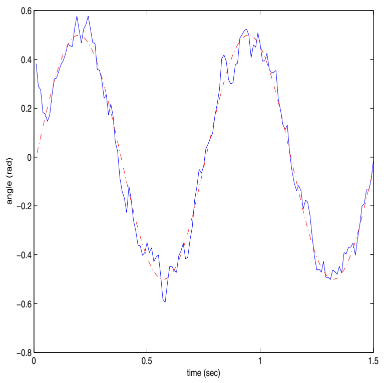


Fig. 4.22 Tracking of a sinusoidal set-point in the presence of disturbance and measurement noise, using adaptive fuzzy H_∞ control

both control methods result in excellent tracking of the reference angular velocity $\dot{\phi}_d$. (iii) In Fig. 4.19 the variation of the control input appeared to be more abrupt in the case of adaptive fuzzy H_∞ control. This was due to the selection of the feedback gain K, (iv) From Fig. 4.20 it can be seen that the neuro-fuzzy networks succeed good approximations of the unknown functions $f(x,t)$ and $g(x,t)$. The accuracy in the estimation of $g(x,t)$ is critical for the convergence of the control algorithm (v) In Fig. 4.21 the excellent performance of adaptive fuzzy H_∞ control is verified in the case of sinusoidal set-points. (vi) Finally, in Fig. 4.22 the performance of adaptive fuzzy H_∞ control and nonlinear backstepping control is evaluated in case that measurement noise is added to the ship's state vector. It can be observed that both controllers maintain the output of the closed-loop system within acceptable levels.

Comparing to nonlinear backstepping control, the advantages of adaptive fuzzy H_∞ control are summarized in the following: (i) removal of any dependence upon identification of the mathematical model expressing the dynamics of the ship, (ii) since training of the neurofuzzy approximators contained in the adaptive fuzzy H_∞ controller is repeatedly undertaken in every control cycle, any changes to the ship dynamics can be identified on-line, and hence the control strategy is useful for time varying ship models. Moreover regarding operation under external disturbances and measurement noise, the adaptive fuzzy H_∞ controller performs equally well to the nonlinear back-stepping controller.

4.5 Application to the Stabilization of Electromechanical Systems

The performance of the proposed adaptive fuzzy H_∞ controller was also tested in the benchmark problem of cart-pole balancing (Fig. 4.23).

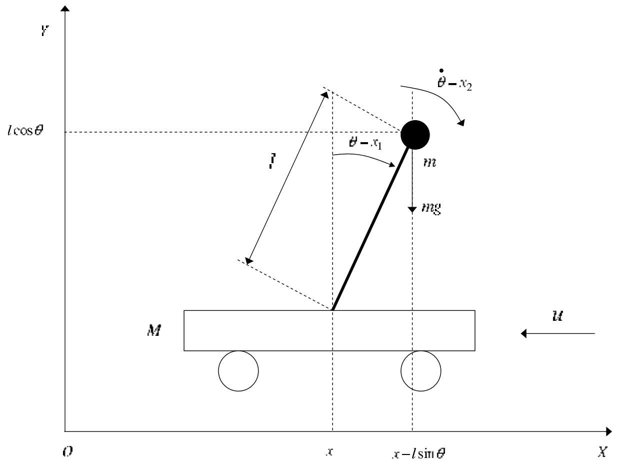


Fig. 4.23 The cart-pole balancing problem

The derivation of the state equations of the cart-pole system stems from the solution of an Euler-Lagrange equation. The model presented here, follows [446],[447], and considers only the dynamics of the pole (inverted pendulum). Denoting by θ the angle of the pole, m the mass of the pole, M the mass of the cart and l the length of the pole, one gets

$$\ddot{\theta} = \frac{mlsin(\theta)cos(\theta)\dot{\theta}^2-(m+M)gsin(\theta)}{mlcos^2(\theta)-2(m+M)l} + \frac{cos(\theta)}{mlcos^2(\theta)-2(m+M)l}u \tag{4.78}$$

Setting $x_1 = \theta$ and $x_2 = \dot{\theta}$ the state equation of the cart-pole system is obtained:

$$\begin{pmatrix}\dot{x}_1\\ \dot{x}_2\end{pmatrix} = \begin{pmatrix}0 & 1\\ 0 & 0\end{pmatrix}\begin{pmatrix}x_1\\ x_2\end{pmatrix} + \begin{pmatrix}0\\ 1\end{pmatrix}(f(x,t)+g(x,t)u+\tilde{d}) \tag{4.79}$$

$$y = \begin{pmatrix}1 & 0\end{pmatrix}\begin{pmatrix}x_1\\ x_2\end{pmatrix} \tag{4.80}$$

where the nonlinear functions $f(x,t)$ and $g(x,t)$ are given by

$$f(x,t) = \frac{mlx_2^2sin(x_1)cos(x_1)-(M+m)gsin(x_1)}{mlcos^2(x_1)-2l(M+m)} \tag{4.81}$$

$$g(x,t) = \frac{cos(x_1)}{mlcos^2(x_1)-2l(M+m)} \tag{4.82}$$

The following parameter values were chosen: $m = 0.1kg$, $M = 1.0kgr$, $l = 0.5m$. The learning rates of the neuro-fuzzy approximators were taken to be $\gamma_1 = 0.015$, $\gamma_2 = 0.035$. The controller's gain was $K = [0.3, 0.9]^T$ while the observer's gain was $K_o = [0.8, 1.0]^T$. The basis functions used in the estimation of $f(x,t)$ and $g(x,t)$

were $\mu_{A_j}(\hat{x}) = e^{(\frac{\hat{x}-c_j}{\sigma})^2}, j = 1,\cdots,5$. Since there are two inputs x_1 and $\dot{x}_1$ and the associated universes of discourse consist of 5 fuzzy sets there will be 25 fuzzy rules of the form:

$$R^l : IF \ \hat{x}_1 \ is \ A_1^l \ AND \ \dot{\hat{x}}_1 \ is \ A_2^l \ THEN \ \hat{f}^l \ is \ b^l \tag{4.83}$$

and $\hat{f}(\hat{x},t) = \frac{\Sigma_{l=1}^{25} \hat{f}^l \Pi_{i=1}^{2} \mu_{A_i}^l(\hat{x}_i)}{\Sigma_{l=1}^{25} \Pi_{i=1}^{2} \mu_{A_i}^l(\hat{x}_i)}$. The centers $c_i^{(l)}$, $i = 1,2$ and the variances $v^{(l)}$ of each rule are as follows

Rule	$c_1^{(l)}$	$c_2^{(l)}$	$v^{(l)}$
$R^{(1)}$	-0.3	-0.03	3
$R^{(2)}$	-0.3	-0.01	3
$R^{(3)}$	-0.3	0.00	3
$R^{(4)}$	-0.3	0.01	3
$R^{(5)}$	-0.3	0.03	3
$R^{(6)}$	-0.1	-0.03	3
$\cdots$	$\cdots$	$\cdots$	$\cdots$
$\cdots$	$\cdots$	$\cdots$	$\cdots$
$R^{(25)}$	0.3	0.03	3

The estimation $\hat{g}(\hat{x},t)$ was derived in a similar way. The overall simulation time was $t_s = 30sec$. In the first half of the simulation time a measurable state vector was assumed and used for the training of the neuro-fuzzy approximators. The positive definite matrix P_1 stems from the solution of the algebraic Riccati equation (4.27), which for $Q = I_2$ resulted into

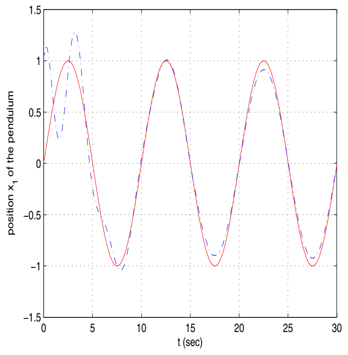
$$P_1 = \begin{pmatrix} 0.372 & 0.167 \\ 0.167 & 0.741 \end{pmatrix} \tag{4.84}$$

The positive definite matrix P_2 stems from the solution of the algebraic Riccati equation (4.27), which for $\rho = 0.773$ and $r = 0.003$, and resulted into

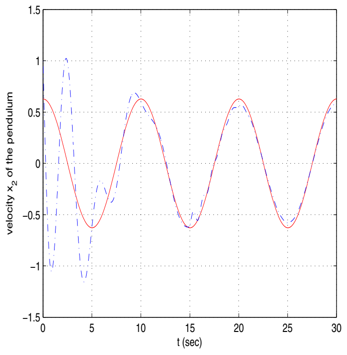
$$P_2 = \begin{pmatrix} 133.2 & 1.0 \\ 1.0 & 0.2 \end{pmatrix} \ Q_2 = \begin{pmatrix} 880.12 & 0.8 \\ 0.8 & 24.6 \end{pmatrix} \tag{4.85}$$

with $P_2 B \approx C$. For the selected values of P_2, Q_2, ρ and r it can be verified that Eq. (4.40) has no imaginary eigenvalues.

The time step was taken to be 0.01 sec. Two different set-points were studied: i) a sinusoidal signal of amplitude 0.1 and period $T = 15sec$, ii) a see-saw set-point of amplitude 0.15 and period $T = 15sec$. At the beginning of the second half of the simulation time an additive sinusoidal disturbance of amplitude $A = 1.0$ and period $T = 7.5$sec was applied to the system. The approximations $\hat{f}$ and $\hat{g}$ were used in the derivation of the control law, given by Eq. (4.19).



(a)



(b)

Fig. 4.24 (a) Tracking of a sinusoidal position set-point (dashed line) by the angle of the pendulum (continuous line), using adaptive fuzzy control (b) Tracking of a sinusoidal velocity set-point (dashed line) by the angular velocity of the pendulum (continuous line) using adaptive fuzzy control

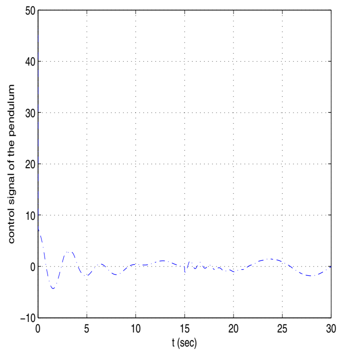


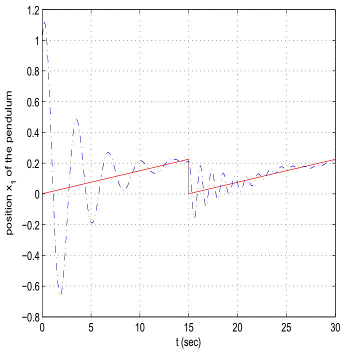
Fig. 4.25 Control input u of the cart-pole system when tracking a sinusoidal set-point under the use of adaptive fuzzy control

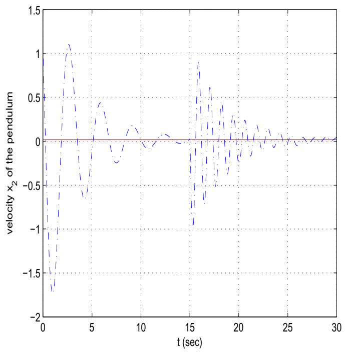
Simulation experiments are presented for the case in which the state vector of the cart-pole model in reconstructed with the use of an observer. A sinusoidal and a see-saw set-point are considered.

The position and velocity variations for the sinusoidal set-point are depicted in Fig. 4.24(a) and Fig. 4.24(b), respectively.

The associated control input is shown in Fig. 4.25.

The performance of the proposed adaptive fuzzy H_∞ control is also tested in the tracking of a see-saw set-point. The position and velocity variation are demonstrated in Fig. 4.26(a) and Fig. 4.26(b), respectively. The variation of the control input for the see-saw set-point is depicted in Fig. 4.27.





(a) (b)

Fig. 4.26 (a) Tracking of a see-saw position set-point (dashed line) by the angle of the pendulum (continuous line), using adaptive fuzzy control (b) Tracking of a see-saw velocity set-point (dashed line) by the angular velocity of the pendulum (continuous line) using adaptive fuzzy control

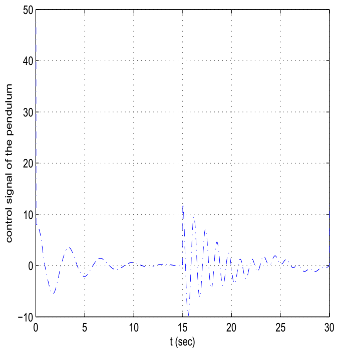


Fig. 4.27 Control input *u* of the cart-pole system when tracking a see-saw set-point under the use of adaptive fuzzy control

Chapter 5
Robust Control Methods for Industrial Systems

Abstract. Robust control approaches for industrial systems are studied. Such methods are based on sliding-mode control theory where the controller's design is performed in the time domain and Kharitonov's stability theory where the controller's design is performed in the frequency domain. Applications of robust control to industrial systems are given.

5.1 Robust Control with Sliding-Mode Control Theory

5.1.1 Sliding-Mode Control

The following nonlinear and non-autonomous open-loop system is considered:

$$x^{(n)}(t) = f(x,t) + b(x,t)u + \tilde{d}, x^n = \frac{d^n x}{dt^n}, \tag{5.1}$$

where $x(t) = (x, \dot{x}, \dots, x^{(n-1)})^T$ is the state vector, $\tilde{d}(x,t)$ is a time-dependent disturbance with known upper bound,and $f(x,t)$ and $b(x,t)$ are nonlinear functions.

The tracking problem for Eq. (5.1) is to find a control law for a desirable trajectory $x^d(t)$ such that the tracking error $x(t) - x^d(t)$ tends to zero independently of the system's uncertainties. The tracking error of the state vector is

$$\begin{gathered} e(t) = x(t) - x^d(t) = (e, \dot{e}, \dots, e^{(n-1)})^T \\ \text{with } e(t) = x(t) - x^d(t) \end{gathered} \tag{5.2}$$

The sliding surface $s(x,t) = 0$ is defined,where

$$s(x,t) = (\frac{d}{dt} + \lambda)^{n-1} e = \sum_{k=0}^{n-1} \binom{n-1}{k} \lambda^k e^{(n-1-k)} \tag{5.3}$$

with initial condition $e(0) = 0$. This means that,by setting $s(x,t) = 0$, one has an homogenous differential equation which has a unique solution $e = 0$. Consequently, an appropriate control rule u has to be found,to keep the state vector e on the sliding surface $s(x,t)=0$. To this end,a Lyapunov function is defined

$$V=\frac{1}{2}s^2 \tag{5.4}$$

with $V(0)=0$ and $V(s)>0$ for $s>0$. An efficient condition for the stability of the system is (see [233],[300],[388],[454]):

$$\dot{V}=\tfrac{1}{2}\tfrac{d}{dt}s^2 \leq -\eta|s| \tag{5.5}$$

which leads to the convergence condition:

$$s\dot{s}\leq-\eta|s| \Rightarrow s\dot{s}\leq-\eta sign(s)s \Rightarrow \dot{s}sign(s)\leq-\eta. \tag{5.6}$$

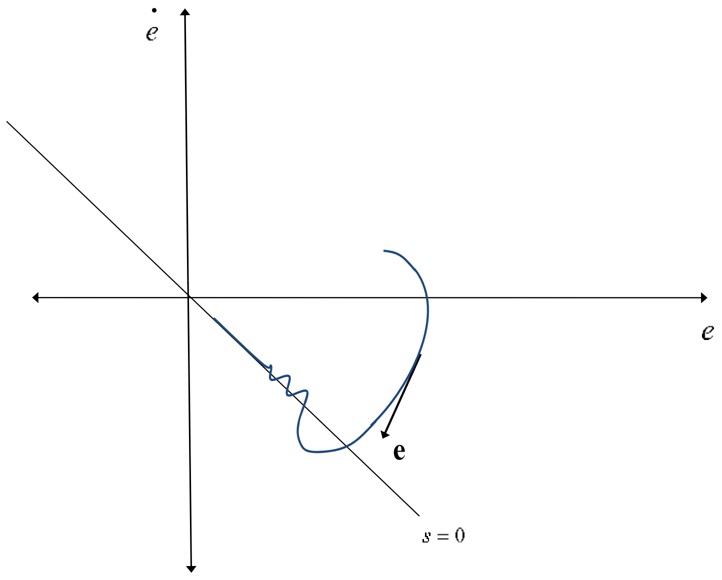


Fig. 5.1 Error state vector in the sliding mode

If $\eta>0$, then system is driven to the sliding mode. This means that, if the state trajectory $[e,\dot{e}]^T$ has reached the sliding surface $s=0$,then it remains on it while at the same time it slides to the origin $s=0$ independently of the system's parametric uncertainties and disturbances.

For a second-order system, convergence to the sliding mode is illustrated in the $(e,\dot{e})$- plane (Figure 5.1). The first step in the design of a sliding-mode controller (SMC) is the selection of the parameter λ.The linear differential Equation (5.3) can be considered as a chain of $(n-1)$-order low-pass filters,where the scalar s plays the role of the input, λ is the break frequency (bandwidth) and e is the output (see Figure 2).

The parameter λ must be selected such that the unmodelled frequencies of the system to be rejected. From the elementary first-order filter $H(p)=1/(\lambda+p)$, where $p=d/dt$, it can be observed that a sufficient condition for frequency rejection is $\lambda \ll p$.Thus in order to reject all unmodelled frequencies one should select $\lambda \ll v_{umin}$, where v_{umin} is the lower bound of the system's unmodelled frequencies v_u.

The next step is to find the control law that will keep the system in sliding mode. Equation (3) gives a sufficient condition for the asymptotic stability of the closed-loop system.The first derivative $\dot{s}$ is calculated as follows:

$$\begin{aligned} s(x,t) &= (\tfrac{d}{dt}+\lambda)^{n-1}e = \sum_{k=0}^{n-1}\binom{n-1}{k}\lambda^k e^{n-1-k} \\ &= e^{(n-1)} + \binom{n-1}{1}\lambda e^{(n-2)} + \binom{n-1}{2}\lambda^2 e^{(n-3)} + \ldots + \lambda^{(n-1)}e \end{aligned} \tag{5.7}$$

or

$$\dot{s}(x,t) = e^{(n)} + \binom{n-1}{1}\lambda e^{(n-1)} + \binom{n-1}{2}\lambda^2 e^{(n-2)} + \ldots + \lambda^{(n-1)}\dot{e} \tag{5.8}$$

or

$$\dot{s}(x,t) = x^{(n)} - x_d^{(n)} + \sum_{k=1}^{n-1}\binom{n-1}{k}\lambda^k e^{(n-k)}. \tag{5.9}$$

Using Eq. (5.9) and Eq. (5.1),one gets

$$([f(x,t)+b(x,t)u+d] - x_d^{(n)} + \sum_{k=1}^{n-1}\binom{n-1}{k}\lambda^k e^{(n-k)})sign(s) \leq -\eta. \tag{5.10}$$

The sliding control law is now defined via the following equations:

$$\begin{aligned} u &= \hat{b}^{-1}(\tilde{u}-\hat{f}) \\ \tilde{u} &= G(\hat{u} - K(x,t)sign(s)) \\ \hat{u} &= x_d^{(n)} - \sum_{k=1}^{n-1}\binom{n-1}{k}\lambda^k e^{(n-k)} \end{aligned} \tag{5.11}$$

where $K(x,t)>0$, and $\hat{f}$ and $\hat{b}$ are estimates of the functions f and b,respectively. To choose the multiplicative coefficient (gain) G, the following bounds are defined:

$$0 \leq \beta^{min} \leq b\hat{b}^{-1} \leq \beta^{max} \tag{5.12}$$

Then gain G is defined as

$$G = (\beta^{min}\beta^{max})^{-1/2} \tag{5.13}$$

and the gain margin β is defined as

$$\beta = (\tfrac{\beta^{max}}{\beta^{min}})^{1/2} \tag{5.14}$$

It now remains to find $K(x,t)$ so as to satisfy Equation 5.6: $\dot{s}sign(s) \leq -\eta$. Introducing Eq. (5.11) into Eq. (5.10) yields

$$sign(s)(f + b\hat{b}^{-1}(\tilde{u} - f) + \tilde{d} - x_d^{(n)} + \sum_{k=1}^{n-1}\binom{n-1}{k}\lambda^k e^{(n-k)}) \leq -\eta \tag{5.15}$$

i.e.,

$$\begin{aligned} &sign(s)(f - b\hat{b}^{-1}f + b\hat{b}^{-1}G\hat{u} - b\hat{b}^{-1}GK(x,t)sign(s) + \\ &+\tilde{d} - x_d^{(n)} + \sum_{k=1}^{n-1}\binom{n-1}{k}\lambda^k e^{(n-k)}) \leq -\eta \end{aligned} \tag{5.16}$$

whence

$$(\Delta f + (b\hat{b}^{-1}G - 1)\hat{u} + \tilde{d})sign(s) - b\hat{b}^{-1}GK(x,t) \leq -\eta \tag{5.17}$$

where $\Delta f = f - b\hat{b}^{-1}f$.The above inequality is satisfied if

$$b\hat{b}^{-1}GK(x,t) \geq |\Delta f + (b\hat{b}^{-1}G - 1)\hat{u} + \tilde{d}| + \eta \tag{5.18}$$

or

$$b\hat{b}^{-1}GK(x,t) \geq |\Delta f| + |(b\hat{b}^{-1}G - 1)||\hat{u}| + |\tilde{d}| + \eta \tag{5.19}$$

Now, if $b\hat{b}^{-1}$ is replaced by its lower bound β^{min}, and after using the relation $\beta^{min}G = (\frac{\beta^{min}}{\beta^{max}})^1/2 = \beta^{-1}$,one gets

$$\beta^{-1}K(x,t) \geq |\Delta f| + |1 - \beta^{-1}||\hat{u}| + |\tilde{d}| + \eta \tag{5.20}$$

whence

$$K(x,t) \geq \beta(|\Delta f| + (1 - \beta^{-1}))|\hat{u}| + |\tilde{d}| + \eta \tag{5.21}$$

The upper bounds $\tilde{F}$, D and U

$$|\Delta f| < \tilde{F}, \ \ |\tilde{d}| < D, \ \ |\hat{u}| < U \tag{5.22}$$

are supposed to be known from the system's analysis. Thus, a sufficient condition are the control law to make the sliding surface $s = 0$ a domain of attraction,is

$$K(x,t) \geq \beta(\tilde{F} + (1 - \beta^{-1}))U + D + \eta. \tag{5.23}$$

5.1.2 An Application Example of Sliding-Mode Control

A second order system is considered

$$\ddot{x} = f(x) + u \tag{5.24}$$

where the dynamics of $f = a(t)\dot{x}^2cos(3x)$ is not exactly known but estimated as $\hat{f}$. The estimation error on f is assumed to be bounded by some known function $F(x,\dot{x})$ i.e.

$$|\hat{f}-f|\leq F \tag{5.25}$$

For example it is assumed that $1\leq a(t)\leq 2$ and $\hat{f}=-1.5\dot{x}^2cos(3x)$ where the bound F is given by $F=0.5\dot{x}^2|cos(3x)|$. The associated sliding surface is defined as

$$\dot{s}=\ddot{x}-\ddot{x}_d+\lambda\dot{\tilde{x}}=f+u-\ddot{x}_d+\lambda\tilde{x} \tag{5.26}$$

The equivalent control law results from $\dot{s}=0$ and gives

$$\hat{u}=\hat{f}+\ddot{x}_d-\lambda\dot{\tilde{x}} \tag{5.27}$$

To satisfy the sliding condition, despite the uncertainty of the dynamics f, a discontinuous control term is added to the equivalent control term $\hat{u}$, across the surface $s=0$. Thus, the overall control input becomes

$$u=\hat{u}-ksgn(s) \tag{5.28}$$

where

$$sgn(s)=\begin{cases}+1 \text{ if } s>0\\ -1 \text{ if } s<0\end{cases} \tag{5.29}$$

The selection of the value of gain K results from the requirement to satisfy the sliding condition given in Eq. (5.6). Indeed from Eq. (5.27) and Eq. (5.28) one has

$$\frac{1}{2}\frac{d}{dt}s^2=s{\cdot}\dot{s}=[f\hat{f}-ksgn(s)]s=(f-\hat{f})s=-k|s| \tag{5.30}$$

thus, setting

$$k=F+\eta \tag{5.31}$$

one gets from Eq. (5.25)

$$\frac{1}{2}\frac{d}{dt}s^2\leq-\eta|s| \tag{5.32}$$

It is noted that the control discontinuity along the sliding surface $s=0$ increases with the extent of parametric uncertainty, while $\hat{f}$ and F need not depend only on x and $\dot{x}$. They may depend on any measured variables external to the system of Eq. (5.24).

5.1.3 Sliding-Mode Control with Boundary Layer

An essential drawback of SMC is that, owing to the sign term $K(x,t)sign(s)$, it causes abrupt changes (chattering) to the control signal u. However, this can be avoided by introducing a Boundary Layer (BL) from both sides of the sliding surface $s=0$. If the term $K(x,t)sign(s)$ exceeds the width of the BL, then it becomes saturated, and is assigned the maximum (minimum) permissible value. The width of BL is selected to be 2Φ.

Assume that $|s|$ is the distance between the state vector e and the sliding surface $s=0$. Then,the state e is inside the boundary layer if $|s|<\Phi$, and is outside the BL if $|s|>\Phi$.If the BL is imported in the control law of Eq. (5.11) one gets:

$$\begin{aligned}
&u=\hat{b}^{-1}(\tilde{u}-\hat{f})\\
&\tilde{u}=G(\hat{u}-K(x,t)sat(s/\Phi))\\
&\hat{u}=x_d^{(n)}-\sum_{k=1}^{n-1}\binom{n-1}{k}\lambda^k e^{(n-k)}
\end{aligned} \tag{5.33}$$

where the saturation function sat$(\cdot)$ is defined as

$$sat(z)=\begin{cases} z, \text{if } |z|<1\\ sign(z) \text{ if } |z|\geq 1\end{cases} \tag{5.34}$$

If $K(x,t)$ is chosen according to Eq. (5.23),then the BL becomes a domain of attraction and the asymptotic stability of the closed loop system is guaranteed. Obviously, this is a weaker requirement than making the sliding surface $s(x,t)=0$ a domain of attraction.The result is that the BL reduces the chattering phenomenon,at the price of increased tracking error. The next step in the design of SMC with BL is the selection of Φ. Equations (5.33) inside the boundary layer (BL) take the form:

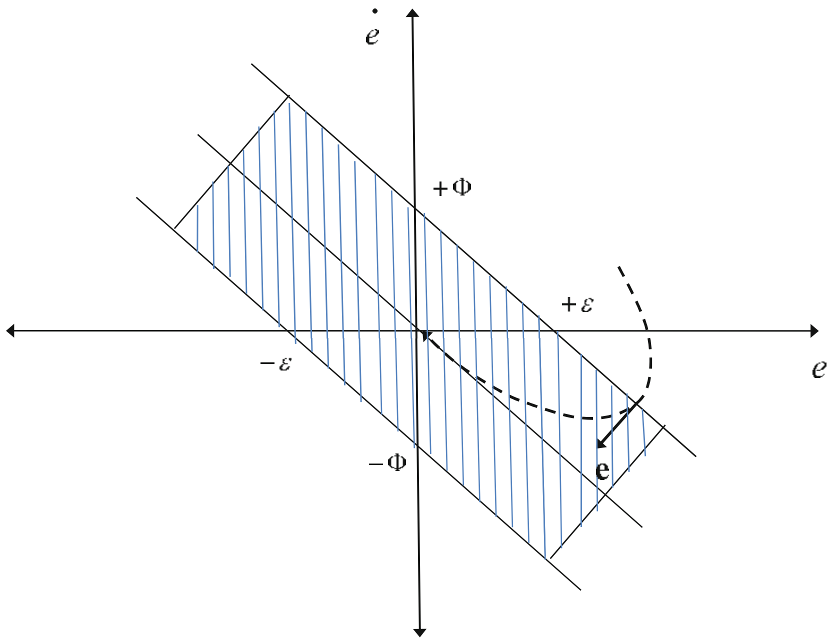


Fig. 5.2 Sliding mode control with boundary layer

$$\begin{aligned} u &= \hat{b}^{-1}(\tilde{u} - \hat{f}) \\ \tilde{u} &= G(\hat{u} - K(x,t)\tfrac{s}{\Phi}) \\ \hat{u} &= x_d^{(n)} - \sum_{k=1}^{n-1} \binom{n-1}{k} \lambda^k e^{(n-k)} \end{aligned} \tag{5.35}$$

Introducing Eq. (5.1) and $\hat{u}$ from Eq. (5.11) into Eq. (5.9)

$$\begin{aligned} \dot{s}(x,t) &= f(x,t) + b(x,t)u + \tilde{d} - \hat{u} \\ &= f(x,t) + b(x,t)(\hat{b}^{-1}(\hat{u} - \hat{f})) + \tilde{d} - \hat{u} \end{aligned} \tag{5.36}$$

or

$$\dot{s}(x,t) = b\hat{b}^{-1}\tilde{u} + (f - b\hat{b}^{-1}\hat{f}) + \tilde{d} - \hat{u} \tag{5.37}$$

or

$$\dot{s}(x,t) = b\hat{b}^{-1}\tilde{u} + \Delta f + \tilde{d} - \hat{u} \tag{5.38}$$

Now, using $\tilde{u}$ from Eq. (5.35) yields

$$\dot{s}(x,t) = b\hat{b}^{-1}G(\hat{u} - K\tfrac{s}{\Phi}) + \Delta f + \tilde{d} - \hat{u} \tag{5.39}$$

whence

$$\dot{s}(x,t) + \tfrac{b\hat{b}^{-1}GK}{\Phi}s = \hat{u}(b\hat{b}^{-1}G - 1) + \Delta f + \tilde{d} \tag{5.40}$$

Equation (5.40) represents a low-pass filter with input $\hat{u}(b\hat{b}^{-1}G - 1) + \Delta f + \tilde{d}$, output s, and break frequency $(b\hat{b}^{-1}GK)/\Phi$. So far, it has been shown how to compute the numerator of the break frequency expression. It only remains to determine the width Φ of BL.There are two choices: The first is to select Φ in proportion to the desirable tracking accuracy ε. Variable ε is selected such that

$$\varepsilon = \frac{\Phi}{\lambda^{n-1}}. \tag{5.41}$$

The second choice is to select the bandwidth $b\hat{b}^{-1}GK/\Phi$ equal to λ. This choice is known as "balanced condition"

$$\frac{b\hat{b}^{-1}GK}{\Phi} = \lambda. \tag{5.42}$$

From the above discussion one can see that the design of SMC with BL is identical to the design of a simple SMC.The only additional step required is the selection of the width Φ which can be done either Eq. (5.41) or Eq. (5.42).

5.2 Robust Control with Interval Polynomials Theory

5.2.1 Basics of Kharitonov's Theory

The stability theory developed by Kharitonov deals with the robust Hurwitz stability of interval polynomials and can be used in the design of robust power

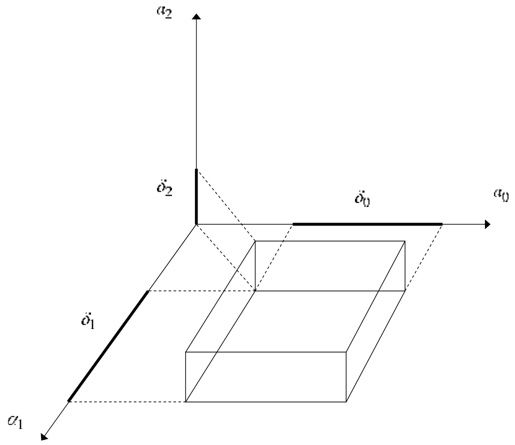


Fig. 5.3 The hyperrectangle defined by the parameters of the interval characteristic polynomial

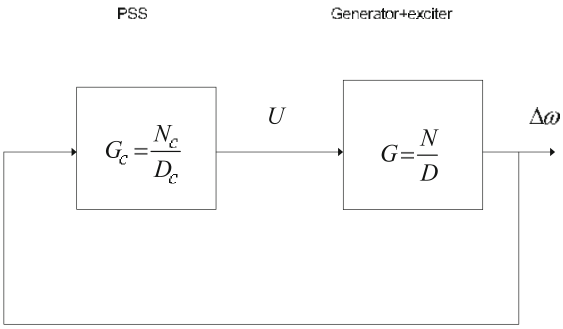


Fig. 5.4 Closed-loop for power system stabilization

system stabilizers. The main points of Kharitonov's theory have been studied in [35],[148],[149],[389],[403],[414], and are summarized as follows:

Let $I(s)$ be the set of the n-th order real polynomials of the form $\delta x = \delta_0 + \delta_1 s + \delta_2 s^2 + \cdots + \delta_n s^n$ where the coefficients $\delta_i,\ i = 1,\cdots,n$ take values in the intervals $\delta_0 \in [x_0, y_0]$, $\delta_1 \in [x_1, y_1], \cdots$, $\delta_n \in [x_n, y_n]$. The coefficient vector δ is defined as $\delta = [\delta_0, \delta_1, \cdots, \delta_n]$. Therefore, δ lies in an hyper-rectangle box of coefficients, as shown in Fig. 5.3 $\Pi = \delta \ :\ \delta \in R^{n+1},\ x_i \leq p_i \leq y_i$.

It is assumed that 0 is not in $[x_n, y_n]$. This kind of polynomials are called 'interval polynomials' and $I(s)$ is the family of 'interval polynomials'. Kharitonov's theorem provides a simple necessary and sufficient condition for the family of interval polynomials $I(s)$. The following theorem holds:

Theorem 1: Each polynomial of the family $I(s)$ is Hurwitz if and only if the following four extreme polynomials are Hurwitz

$$\begin{aligned}
K^1(s) &= x_0+x_1s+y_2s^2+y_3s^3+x_4s^4+x_5s^5+\cdots \\
K^2(s) &= x_0+y_1s+y_2s^2+x_3s^3+x_4s^4+y_5s^5+\cdots \\
K^3(s) &= y_0+x_1s+x_2s^2+y_3s^3+y_4s^4+x_5s^5+\cdots \\
K^4(s) &= y_0+y_1s+x_2s^2+x_3s^3+y_4s^4+y_5s^5+\cdots
\end{aligned} \tag{5.43}$$

An assumption made in Kharitonov's theorem is that the coefficients perturb independently. It is noted however that, in some problems this assumption does not hold, since the characteristic polynomial coefficients perturb inter-dependently through other primary parameters. However, even in these cases Kharitonov's theorem can give computationally simple answers by overbounding the actual perturbations by an axis parallel box Π in the coefficient space.

5.2.2 Extremal Properties of Kharitonov Polynomials

The case of a closed-loop control system with unit feedback is considered, where the forward transfer function $G(s) = n(s)/d(s)$ contains parameter uncertainty. The polynomials $n(s)$ and $d(s)$ vary in independent polynomial families $N(s)$ and $D(s)$ respectively. Moreover, the family of transfer functions of Eq. (5.44) is considered:

$$G(s) = \frac{n(s)}{d(s)} : \; n(s) \in N(s), \; d(s) \in D(s) \tag{5.44}$$

where each interval polynomial of degree n can be written $\delta(s) = \delta_0 + \delta_1 s + \delta_2 s^2 + \cdots + \delta_n s^n$ with the coefficients lying in the ranges $\delta_0 \in [x_0, y_0]$, $\delta_1 \in [x_1, y_1]$, $\cdots$, $\delta_n \in [x_n, y_n]$. For the closed-loop system with unity feedback containing $G(s)$ to be robustly stable, the associated characteristic polynomial given by $P(s) = d(s+n(s))$ should be Hurwitz for all $n(s),\; d(s) \in N(s)\times D(s)$. Let $K_N^i(s),\; i = 1,2,3,4$, and $K_D^j(s),\; j = 1,2,3,4$ denote the Kharitonov polynomials associated with $N(s)$ and $D(s)$ respectively. The positive set of Kharitonov systems $G_K^+(s)$ which are associated with the interval family $G(s)$ is now defined as:

$$G_K^+(s) = \frac{K_N^i(s)}{K_D^j(s)}, \; i,j = 1,2,3,4 \tag{5.45}$$

The following two theorems hold:

Theorem 2: The closed-loop system with unity feedback containing $G(s)$ is robustly stable if and only if each of the positive Kharitonov systems in $G_K^+(s)$ is robustly stable.

Theorem 3: (Extremal Gain Margin Theorem) The worst case gain margin of a system at an operating point P over the interval family $G(s)$ is the minimum gain margin corresponding to the positive Kharitonov systems $G_K^+(s)$.

Finally, the Kharitonov plants (linear system descriptions) are defined as follows: the interval plant family P is considered consisting of all plants of the form:

$$P(s,q,r) = \frac{N(s,q)}{D(s,r)} = \frac{q_m s^m + q_{m-1}s^{m-1} + \cdots + q_1 s + q_0}{s^n + r_{m-1}s^{m-1} + \cdots + r_1 s + r_0} \tag{5.46}$$

where $N(s,q)$ and $D(s,r)$ are interval polynomials, with coefficients which vary in the rectangles Q and R as defined by the cartesian products of $[q_i^-, q_i^+]$, $i = 0, 1, \cdots, m$ and $[r_i^-, r_i^+]$, $i = 0, 1, \cdots, n-1$. It is assumed that $N(s,q)$ and $D(s,r)$ do not have common roots and that $D(s,r)$ does not have roots on the imaginary axis for all $r \in R$. Given $c \in C$, the polynomials family $N(s,q) - cD(s,r)$ is said to be robustly stable if all its roots are in the left complex semi-plane, $\forall q \in Q$ and $r \in R$. The interval polynomial $N(s,q)$ is Hurwitz $\forall q \in Q$ if and only if the 4 Kharitonov polynomials $N_i(s)$, $i = 1,2,3,4$, which are defined in Eq. (5.47), are Hurwitz. The interval polynomial $D(s,r)$ is Hurwitz if and only if the 4 Kharitonov polynomials $D_k(s)$, $k = 1,2,3,4$, which are defined in Eq. (5.47) are Hurwitz.

$$\begin{aligned}
N_1(s) &= q_0^+ + q_1^+ s + q_2^- s^2 + q_3^- s^3 + q_4^- s^4 + q_5^+ s^5 + \cdots \\
N_2(s) &= q_0^+ + q_1^- s + q_2^+ s^2 + q_3^+ s^3 + q_4^- s^4 + q_5^- s^5 + \cdots \\
N_3(s) &= q_0^- + q_1^+ s + q_2^+ s^2 + q_3^- s^3 + q_4^- s^4 + q_5^+ s^5 + \cdots \\
N_4(s) &= q_0^+ + q_1^- s + q_2^- s^2 + q_3^+ s^3 + q_4^+ s^4 + q_5^- s^5 + \cdots \\
\\
D_1(s) &= r_0^+ + r_1^+ s + r_2^- s^2 + r_3^- s^3 + r_4^- s^4 + r_5^+ s^5 + \cdots \\
D_2(s) &= r_0^+ + r_1^- s + r_2^+ s^2 + r_3^+ s^3 + r_4^- s^4 + r_5^- s^5 + \cdots \\
D_3(s) &= r_0^- + r_1^+ s + r_2^+ s^2 + r_3^- s^3 + r_4^- s^4 + r_5^+ s^5 + \cdots \\
D_4(s) &= r_0^+ + r_1^- s + r_2^- s^2 + r_3^+ s^3 + r_4^+ s^4 + r_5^- s^5 + \cdots
\end{aligned} \tag{5.47}$$

5.3 Application to the Stabilization of Electric Power Systems

Modern large-scale power systems have commonly experienced adverse impacts on their operation and come against the risk for destabilization due to under-damped oscillations [206]. Several research papers have tried to resolve the problem of suppression of the oscillatory behavior of electric power transmission networks using either frequency response control methods or state-feedback control algorithms [224],[248],[341]. The objective of the subsequent analysis is to develop robust power system stabilizers that will be able to suppress the oscillatory behavior of the power systems over a wide range of operating conditions, thus assuring their secure operation. Since power generation systems are actually nonlinear, conventional fixed parameter PSS cannot cope with great changes in operating conditions.

Power system instability will mostly happen when a remote station sends a large amount of power to the rest of the grid through a relatively weak transmission line. This situation can be well modeled by a single-machine infinite-bus system. In this representation the dynamic interaction between the various machines in the system is not considered, however the resulting model is still adequate for many types of studies, especially when the machines are identical and operate at nearly the same load levels. In this chapter the model of a single-machine infinite-bus power system is considered and a stabilizer is designed using Kharitonov's extremal gain margin theorem. According to Kharitonov's theory the nonlinear power system can be modeled by a set of transfer functions, where each transfer function is described by the ratio of two interval polynomials. Such polynomials have the form $p(s) = \alpha_n s^n + \alpha_{n-1} s^{n-1} + \cdots + \alpha_0$, where each coefficient α_i is independent of the others and varies within an interval of known lower and upper bounds, i.e. $\alpha_i \in [\alpha_i^-, \alpha_i^+]$. Thus, the problem of stabilization of the nonlinear power system is transformed into the problem of stabilization of a set of linear systems and can be succeeded with the use of simple phase-lead compensators, and frequency response techniques, such as the root-locus diagram [353],[389],[426].

5.3.1 The Problem of Power System Stabilization

In a power generation system, the electromechanical coupling between the rotor and the rest of the system components results in a behavior similar to that of a damped oscillator. Thus, disturbances such as sudden changes in loads, changes in transmission line parameters, or fluctuations in the output of turbine and faults, cause low frequency oscillations around the equilibrium state. Furthermore, the use of fast acting high gain Automated Voltage Regulators (AVR) and the development of interconnected power systems with transfer of power across transmission lines can result in further degradation of the power system's oscillatory behavior. These oscillations limit the power transmission capability of the network and, sometimes, even cause a loss of generators' synchronism and a subsequent breakdown of the entire system [346]. The application of Power System Stabilizers (PSS) can help in damping out these oscillations and in improving the system's stability.

Usually, there are two approaches to succeed efficient operation of power systems in variable conditions: (i) adaptive control, and (ii) robust control [89],[224]. Adaptive control is based on the idea of continuously adapting the controller's parameters according to recent measurements. However, the performance of adaptive controllers maybe unsatisfactory during the learning phase, particularly when they are improperly initialized. Successful operation of adaptive controllers requires the measurements to satisfy persistence of excitation conditions, otherwise the adjustment of the controller's parameters fails. Moreover, the design of nonlinear adaptive controllers can be complicated, and the selection of the adaptation gains which assures closed stability may require extensive calculations [224]. Robust control can

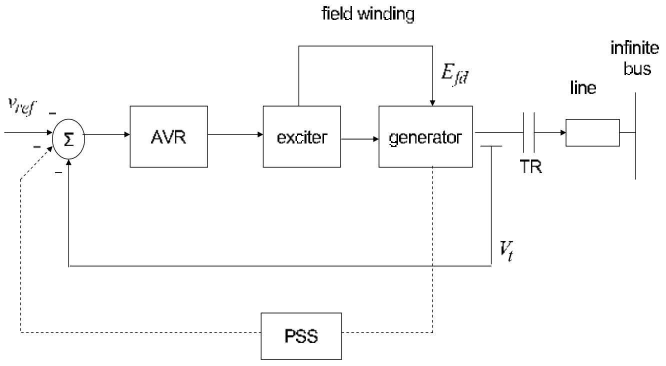


Fig. 5.5 Basic components of the single-machine infinite bus power system

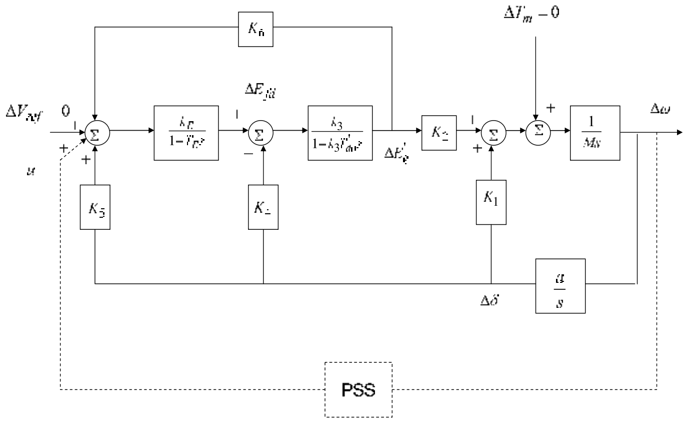


Fig. 5.6 Transfer function elements of the linearized single-machine infinite bus power system

be an effective approach to dealing with uncertainties introduced by variations of the power system operating conditions. Among many techniques available in robust control theory H_∞ control has received significant attention in applications of power system stabilization [4],[35],[60],[89],[136]. The H_∞ controller is selected so that the system's closed-loop response lies between pre-specified bounds. The design of such H_∞ controllers requires (i) full state feedback or observer-based estimation of the high-order parameters of the machine's state vector (ii) the solution of Riccati equations, which can be complicated if prior reduction of the linear model's order is not performed.

Taking into account the flaws and limitations of the aforementioned PSS design approach, this chapter proposes a simple, yet robust controller for power system stabilization, using Kharitonov's stability theory. The diagram of the single-machine infinite-bus systems is given in Fig. 5.5 and 5.6. The PSS receives as input the machine's speed deviation signal $\Delta\omega$. To select the parameters of the PSS, usually the power system dynamics is linearized at an operating point, and once a transfer function is obtained linear control techniques can be applied. However, in practice the parameters of such a linear model are load dependent and thus the PSS has to be adjusted at different loads. This may include calculations over a large number of operating points. In the method for robust power system stabilization proposed by this paper, the PSS parameters will be chosen so as to assure stability of the power system model and suppression of the oscillatory behavior for a wide range of operating conditions. The steps of design of the proposed PSS according to Kharitonov's theory will be explained in the sequel.

5.3.2 Transfer Function of the Single-Machine Infinite-Bus Model

The fourth-order transfer function for the single-machine infinite-bus system is given by [89],[389]

$$\frac{\Delta\omega}{U} = \frac{b_s}{\alpha_4 s^4 + \alpha_3 s^3 + \alpha_2 s^2 + \alpha_1 s + \alpha_0} \tag{5.48}$$

(1) The transfer function coefficients are $\alpha_4 = M{\cdot}T{\cdot}T_E$, $\alpha_3 = M(T+T_E)$, $\alpha_2 = M + 314k_1T{\cdot}T_E + k_Ek_3k_6M$, $\alpha_1 = 314k_1(T+T_E) - 314k_2k_3k_4T_E$, where $T = k_3T_{do}{}'$. The parameters of the single machine infinite-bus system are as follows: $k_1,\cdots,k_6$ are the parameters of the power system block diagram as depicted in Fig. 5.6, T is the mechanical torque, T_{do}' is the open-circuit d-axis transient time constant, M is the inertia coefficient, k_E and T_E are the exciter gain and time constant respectively, $\Delta\omega$ is the machine's speed deviation and U is the stabilizing signal (PSS output).

5.3.3 Kharitonov's Theory for Power System Stabilization

The transfer function that describes the single-machine infinite-bus model has been given in Eq. (5.48). Using [389] indicative ranges of the parameters of the model are $b{\in}[21,96.1]$, $\alpha_4 = 8.34$, $\alpha_3 = 167.3$, $\alpha_2 \in [427.1, 744.4]$, $\alpha_1 \in [2941, 7500]$, $\alpha_0 \in [1093, 1258]$. First, the control-loop of the nominal model is considered to contain a lead-lag compensator of the form

$$G_c(s) = K_c \frac{s+\mu_1}{s+\pi_1}\frac{s+\mu_2}{s+\pi_2} \tag{5.49}$$

where the μ_1 and π_1 are the zeros/poles of the phase-lead part while μ_2, π_2 are the poles/zeros of the phase-lag part. Using the stability conditions of the closed loop

$$||G_c(s)G(s)|| = 1 \ \angle G_c(s)G(s) = -180^o \tag{5.50}$$

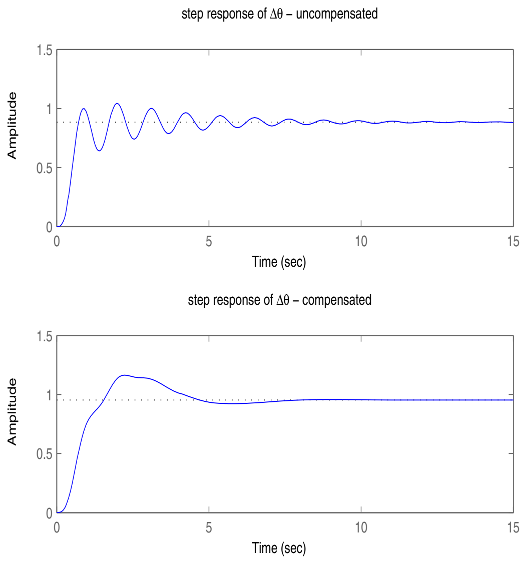


Fig. 5.7 The inclusion of the compensator $G_c(s)$ in the control-loop results in the desirable transient and steady-state performance of the nominal power system model

and the requirement that the steady-state error of the control loop should be minimized one obtains the compensator

$$G_c(s) = 65.38\frac{s+2}{s+2.33}\frac{s+1.76}{s+0.05} \tag{5.51}$$

Assuming step input to the control loop, the machine's angle difference $\Delta\delta$ is depicted in Fig. 5.7. It can be observed that by introducing the compensator in the control-loop the desirable transient and steady-state performance are achieved (according to selected dominant poles for the closed-loop). It can be seen that the steady-state becomes practically 0, oscillations are suppressed and settling time is reduced.

Next, the design of a robust phase-lead compensator was performed, assuming that the model's parameters where defined within the uncertainty ranges described above. The phase lead compensator was $G_c(s) = K_c\frac{s+\mu_1}{s+\pi_1}$. According to Theorem 3, to find the stabilizing gain K_c one has to calculate the gain K_c^i which stabilizes each one of the 16 Kharitonov plant's G_K^+ defined in Eq. (5.45). However, since coefficient b has two extreme values b_{min} and b_{max} the 16 Kharitonov plants of Eq. (5.45) are reduced to only 8 models. Thus one obtains the following extreme Kharitonov plants

$$G_k^1(s)=\frac{21s}{8.34s^4+167.3s^3+744.4s^2+2941s+1093}\quad G_k^2(s)=\frac{96.1s}{8.34s^4+167.3s^3+744.4s^2+2941s+1093}$$
$$G_k^3(s)=\frac{21s}{8.34s^4+167.3s^3+427.1s^2+7500s+1258}\quad G_k^4(s)=\frac{96.1s}{8.34s^4+167.3s^3+744.4s^2+2941s+1258}$$
$$G_k^5(s)=\frac{21s}{8.34s^4+167.3s^3+427.1s^2+2941s+1258}\quad G_k^6(s)=\frac{96.1s}{8.34s^4+167.3s^3+427.1s^2+2941s+1258}$$
$$G_k^7(s)=\frac{21s}{8.34s^4+167.3s^3+744.4s^2+7500s+1093}\quad G_k^8(s)=\frac{96.1s}{8.34s^4+167.3s^3+744.4s^2+7500s+1093} \tag{5.52}$$

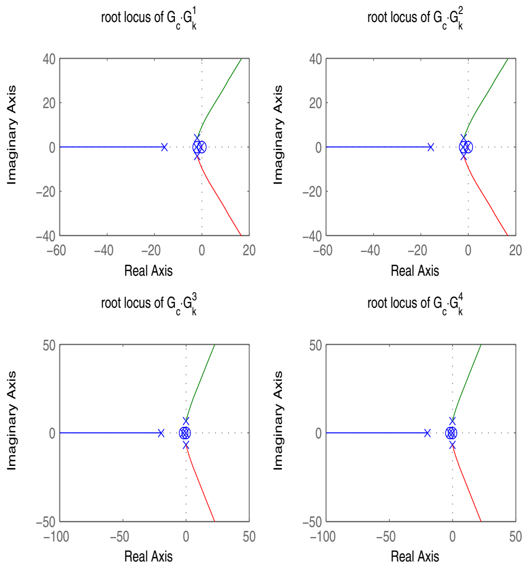


Fig. 5.8 Root-locus of the Kharitonov extremal plants $G_c(s)G_k^1(s)$ to $G_c(s)G_k^4(s)$

The root-locus diagrams which correspond to the models $G_c(s)G_k^i(s)$, are depicted in Fig. 5.8 and Fig. 5.9. From these diagrams one obtains the variation range $\Delta K_c^i=[K_{cmin}^i,K_{cmax}^i]$, $i=1,\cdots,8$ of the gain K_c^i which assures stability to each one of the associated loops, while the overall stabilizing gain belongs to the intersection of ΔK_c^i . A suitable stabilizing gain is thus taken to be $K_c=2$.

The impulse response of the 8 unit-feedback closed loops which contain the stabilizing gain K_c, the phase lead compensator $G_c(s)=\frac{s+\mu_1}{s+\pi_1}$ and each one of the extreme Kharitonov polynomials G_k^i is shown in Fig. 5.10 and Fig. 5.11, as well as in Fig. 5.12 and Fig. 5.13. Comparing to the response of the unit-feedback closed-loop without phase lead compensator and with gain which has been selected according to the stability ranges of the nominal plant, as depicted in Fig. 5.11 and Fig. 5.13, one can observe the improvements in the closed-loop performance which is due to the use of the power system stabilizer. It can be seen that the proposed robust power

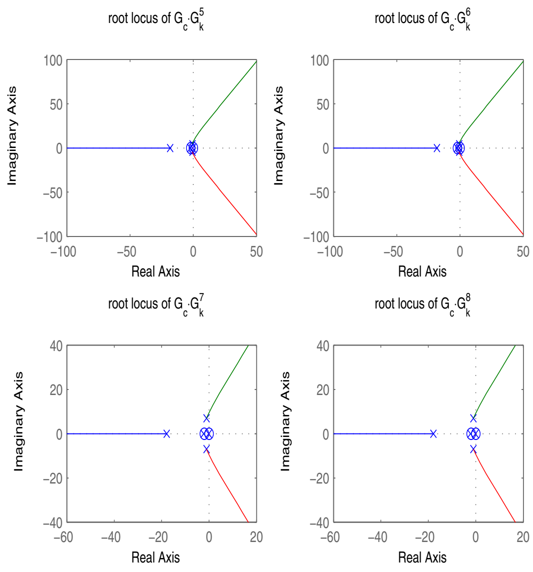


Fig. 5.9 Root-locus of the Kharitonov extremal plants $G_c(s)G_k^5(s)$ to $G_c(s)G_k^8(s)$

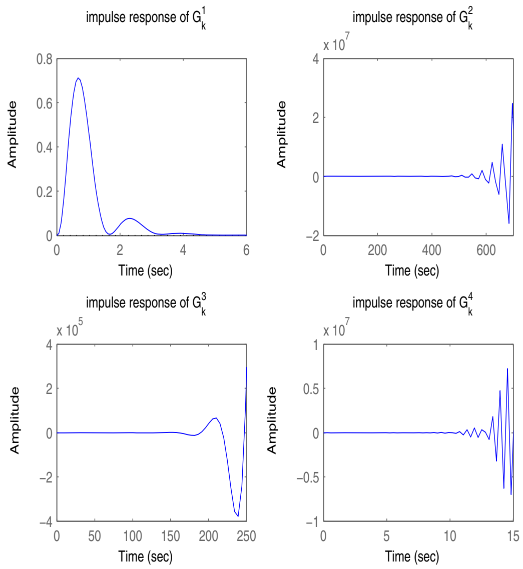


Fig. 5.10 Impulse response of the Kharitonov extremal plants G_k^1 to G_k^4, in a unit-feedback closed-loop which includes a phase-lead compensator

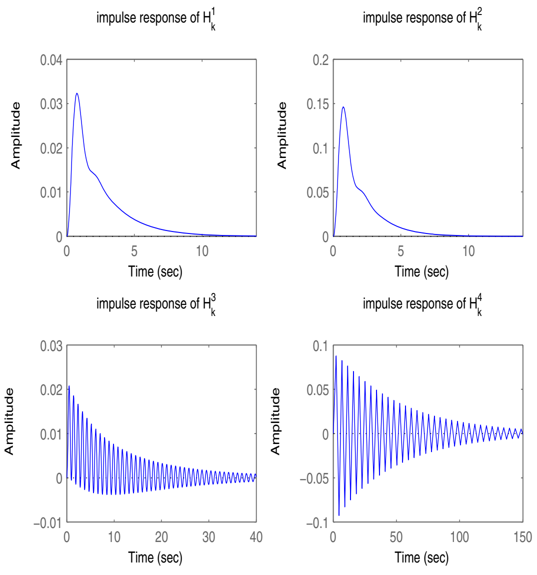


Fig. 5.11 Impulse response of the Kharitonov extremal plants G_k^1 to G_k^4, in a unit-feedback closed-loop with proportional gain K_c chosen according to the nominal plant's stability ranges

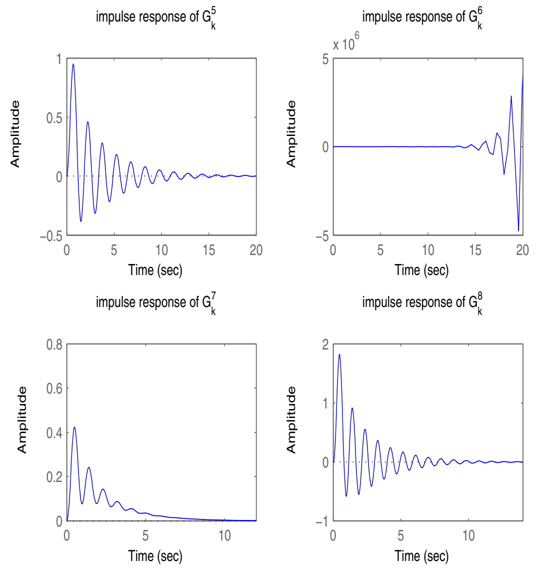


Fig. 5.12 Impulse response of the Kharitonov extremal plants G_k^5 to G_k^8, in a unit-feedback closed-loop which includes a phase-lead compensator

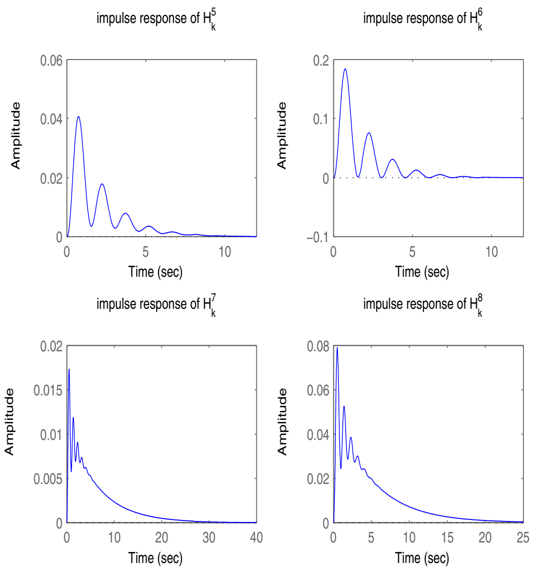


Fig. 5.13 Impulse response of the Kharitonov extremal plants G_k^5 to G_k^8, in a unit-feedback closed-loop with proportional gain K_c chosen according to the nominal plant's stability ranges

system stabilizer results in fast decay of oscillations and suppression of the oscillations' amplitude. On the other hand the design of a control-loop which does not take into account the parametric uncertainty in the single-machine infinite-bus model results in undesirable oscillations and may even lead the power system to instability as shown in Fig. 5.10 and in Fig. 5.12.

Chapter 6
Filtering and Estimation Methods for Industrial Systems

Abstract. Filtering and stochastic estimation methods are proposed for the control of linear and nonlinear dynamical systems. Starting from the theory of linear state observers the chapter proceeds to the standard Kalman filter and its generalization to the nonlinear case which is the Extended Kalman Filter. Additionally, Sigma-Point Kalman Filters are proposed as an improved nonlinear state estimation approach. Finally, to circumvent the restrictive assumption of Gaussian noise used in Kalman filter and its variants, the Particle Filter is proposed. Applications of filtering and estimation methods to industrial systems control with a reduced number of sensors are presented.

6.1 Linear State Observers

First, the linear dynamical system of Eq. (1) is considered

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t) \\ y(t) = Cx(t) \end{cases} \tag{6.1}$$

where $x \in R^{m\times 1}$ is the system's state vector $u \in R^{1\times 1}$ is the control input, and $y \in R^{p\times 1}$ is the system's output. It is assumed that the elements of the state vector x are not completely measurable, either due to the system's structure or due to high cost of measurement sensors. In that case the system's state vector can be reconstructed using the sequence of output measurements $y(t)$ and the associated sequence of control inputs $u(t)$. The basic requirement to perform the state vector's reconstruction is the linear system to be observable as defined by the pair of matrices (A,C). An important application of state observers is the design of state estimation-based control schemes. For linear dynamical systems the principle of separation holds: (i) the state feedback controller is designed assuming that the complete state vector of the system is available, (ii) the state observer is designed for those state variables which cannot be measured directly. The concept of state observers is due to Luenberger and includes the Kalman Filter as a special case [175]. Actually, the Kalman Filter is an optimal state observer in the sense that it can compensate in optimal way for the effect that process and measurement noises have on the estimation of the system's state vector.

For the continuous time dynamical system of Eq. (6.1) the state observer is

$$\dot{\hat{x}} = A\hat{x} + Bu + K(y - C\hat{x}) \tag{6.2}$$

From Eq. (6.2) it can be seen that the linear state observer uses the state-space equation of the dynamical system augmented by the additional term $K(y - \hat{y})$ where the signal $y - C\hat{x}$ is called residual (or innovation) and is the difference between the real measurement $y(t)$ and the estimated output $\hat{y}(t)$. Defining the state vector estimation error as $e = x - \hat{x}$, the dynamics of the observer becomes

$$\dot{e}(t) = (A - KC)e(t) \Rightarrow \dot{e}(t) = \hat{A}e(t) \tag{6.3}$$

where $\hat{A} = A - KC$. The main problem in the design of the state observer is to select gain K such that matrix $\hat{A} = A - KC$, which defines the estimation error dynamics, to have eigenvalues strictly in the left complex semi-plane. Two typical methods for the selection of gain K are:

1. All eigenvalues of matrix $\hat{A}$ can be moved to desirable positions at the left complex semi-plane using pole-placement techniques.
2. Gain K can be selected by solving an optimization problem (which finally provides the Kalman Filter). This is expressed as the minimization of a quadratic cost functional (as in the case of Linear Quadratic Regulator - LQR optimal control) and is performed through the solution of a Riccati equation. In that case the observer's gain K is calculated by $K = PC^TR^{-1}$ considering an optimal control problem for the dual system (A^T, C^T), where the covariance matrix of the estimation error P is found by the solution of the continuous-time Riccati equation of the form

$$\dot{P} = AP + PA^T + Q - PC^TR^{-1}CP \tag{6.4}$$

where matrices Q and R stand for the process and measurement noise covariance matrices, respectively.

6.2 The Continuous-Time Kalman Filter for Linear Models

Next, the continuous-time dynamical system of Eq. (5) is assumed [174],[175]:

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t) + w(t), \ t \geq t_0 \\ y(t) = Cx(t) + v(t), \ t \geq t_0 \end{cases} \tag{6.5}$$

where again $x \in R^{m \times 1}$ is the system's state vector, and $y \in R^{p \times 1}$ is the system's output. Matrices A,B and C can be time-varying and $w(t)$,$v(t)$ are uncorrelated white Gaussian noises. The covariance matrix of the process noise $w(t)$ is $Q(t)$, while the covariance matrix of the measurement noise is $R(t)$. Then the Kalman Filter is again a linear state observer which is given by

$$\begin{cases} \dot{\hat{x}} = A\hat{x} + Bu + K[y - C\hat{x}],\ \hat{x}(t_0) = 0 \\ K(t) = PC^T R^{-1} \\ \dot{P} = AP + PA^T + Q - PC^T R^{-1} CP \end{cases} \tag{6.6}$$

where $\hat{x}(t)$ is the optimal estimation of the state vector $x(t)$ and $P(t)$ is the covariance matrix of the state vector estimation error with $P(t_0) = P_0$. It can be seen that as in the case of the Luenberger observer, the Kalman Filter consists of the system's state equation plus a corrective term $K[y - C\hat{x}]$. The associated Riccati equation for calculating the covariance matrix $P(t)$ has the same form as Eq. (6.4).

6.3 The Discrete-Time Kalman Filter for Linear Systems

In the discrete-time case the dynamical system is assumed to be expressed in the form of a discrete-time state model:

$$\begin{cases} x(k+1) = \Phi(k)x(k) + L(k)u(k) + w(k) \\ z(k) = Cx(k) + v(k) \end{cases} \tag{6.7}$$

where the state $x(k)$ is a m-vector, $w(k)$ is a m-element process noise vector and Φ is a $m \times m$ real matrix. Moreover the output measurement $z(k)$ is a p-vector, C is an $p \times m$-matrix of real numbers, and $v(k)$ is the measurement noise. It is assumed that the process noise $w(k)$ and the measurement noise $v(k)$ are uncorrelated.

Now the problem of interest is to estimate the state $x(k)$ based on the measurements $z(1), z(2), \cdots, z(k)$. The initial value of the state vector $x(0)$, the initial value of the error covariance matrix $P(0)$ is unknown and an estimation of it is considered, i.e. $\hat{x}(0) =$ a guess of $E[x(0)]$ and $\hat{P}(0)$=a guess of $Cov[x(0)]$.

For the initialization of matrix P one can set $\hat{P}(0) = \lambda I$, with $\lambda > 0$. The state vector $x(k)$ has to be estimated taking into account $\hat{x}(0)$, $\hat{P}(0)$ and the output measurements $Z = [z(1), z(2), \cdots, z(k)]^T$, i.e. there is a function relationship:

$$\hat{x}(k) = \alpha_n(\hat{x}(0), \hat{P}(0), Z(k)) \tag{6.8}$$

Actually, this is a linear minimum mean squares estimation problem (LMMSE) which is solved recursively, through the function relationship

$$\hat{x}(k+1) = a_{n+1}(\hat{x}(k), z(k+1)) \tag{6.9}$$

The process and output noise are white and their covariance matrices are given by: $E[w(i)w^T(j)] = Q\delta(i-j)$ and $E[v(i)v^T(j)] = R\delta(i-j)$.

Using the above, the discrete-time Kalman Filter can be decomposed into two parts: i) time update, and ii) measurement update. The first part employs an estimate of the state vector $x(k)$ made before the output measurement $z(k)$ is available (a priori estimate). The second part estimates $x(k)$ after $z(k)$ has become available (a posteriori estimate).

- When the set of measurements $Z^{-}=\{z(1),\cdots,z(k-1)\}$ is available, from Z^{-} an a priori estimation of $x(k)$ is obtained which is denoted by $\hat{x}^{-}(k)=$ the estimate of $x(k)$ given Z^{-}.
- When $z(k)$ becomes available, the set of the output measurements becomes $Z=\{z(1),\cdots,z(k)\}$, where $\hat{x}(k)=$ the estimate of $x(k)$ given Z.

The associated estimation errors are defined by

$$\begin{array}{c} e^{-}(k)=x(k)-\hat{x}^{-}(k)=\textit{the a priori error} \\ e(k)=x(k)-\hat{x}(k)=\textit{the a posteriori error} \end{array} \tag{6.10}$$

The estimation error covariance matrices associated with $\hat{x}(k)$ and $\hat{x}^{-}(k)$ are defined as [139],[175]

$$\begin{array}{c} P^{-}(k)=Cov[e^{-}(k)]=E[e^{-}(k)e^{-}(k)^{T}] \\ P(k)=Cov[e(k)]=E[e(k)e^{T}(k)] \end{array}$$

From the definition of the trace of a matrix, the mean square error of the estimates can be written as

$$\begin{array}{c} MSE(\hat{x}^{-}(k))=E[e^{-}(k)e^{-}(k)^{T}]=tr(P^{-}(k)) \\ MSE(x(k))=E[e(k)e^{T}(k)]=tr(P(k)) \end{array}$$

Finally, the linear Kalman filter equations in cartesian coordinates are

measurement update: obtain measurement $z(k)$ and compute

$$\begin{array}{l} K(k)=P^{-}(k)C^{T}[C{\cdot}P^{-}(k)C^{T}+R]^{-1} \\ \hat{x}(k)=\hat{x}^{-}(k)+K(k)[z(k)-C\hat{x}^{-}(k)] \\ P(k)=P^{-}(k)-K(k)CP^{-}(k) \end{array} \tag{6.11}$$

time update: compute

$$\begin{array}{l} P^{-}(k+1)=\Phi(k)P(k)\Phi^{T}(k)+Q(k) \\ \hat{x}^{-}(k+1)=\Phi(k)\hat{x}(k)+L(k)u(k) \end{array} \tag{6.12}$$

6.4 The Extended Kalman Filter for Nonlinear Systems

The following nonlinear time-invariant state model is now considered [331]:

$$\begin{array}{c} x(k+1)=\phi(x(k))+L(k)u(k)+w(k) \\ z(k)=\gamma(x(k))+v(k) \end{array} \tag{6.13}$$

where $x{\in}R^{m\times 1}$ is the system's state vector and $z{\in}R^{p\times 1}$ is the system's output, while $w(k)$ and $v(k)$ are uncorrelated, zero-mean, Gaussian noise processes with covariance matrices $Q(k)$ and $R(k)$ respectively. The operators $\phi(x)$ and $\gamma(x)$ are $\phi(x)=[\phi_1(x),\phi_2(x),\cdots,\phi_m(x)]^{T}$, and $\gamma(x)=[\gamma_1(x),\gamma_2(x),\cdots,\gamma_p(x)]^{T}$, respectively. It is assumed that ϕ and γ are sufficiently smooth in x so that each one has a valid series Taylor expansion. Following a linearization procedure, ϕ is expanded into Taylor series about $\hat{x}$:

$$\phi(x(k)) = \phi(\hat{x}(k)) + J_{\phi}(\hat{x}(k))[x(k) - \hat{x}(k)] + \cdots \tag{6.14}$$

where $J_{\phi}(x)$ is the Jacobian of ϕ calculated at $\hat{x}(k)$:

$$J_{\phi}(x) = \frac{\partial \phi}{\partial x}|_{x=\hat{x}(k)} = \begin{pmatrix} \frac{\partial \phi_1}{\partial x_1} & \frac{\partial \phi_1}{\partial x_2} & \cdots & \frac{\partial \phi_1}{\partial x_m} \\ \frac{\partial \phi_2}{\partial x_1} & \frac{\partial \phi_2}{\partial x_2} & \cdots & \frac{\partial \phi_2}{\partial x_m} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial \phi_m}{\partial x_1} & \frac{\partial \phi_m}{\partial x_2} & \cdots & \frac{\partial \phi_m}{\partial x_m} \end{pmatrix} \tag{6.15}$$

Likewise, γ is expanded about $\hat{x}^{-}(k)$

$$\gamma(x(k)) = \gamma(\hat{x}^{-}(k)) + J_{\gamma}[x(k) - \hat{x}^{-}(k)] + \cdots \tag{6.16}$$

where $\hat{x}^{-}(k)$ is the estimation of the state vector $x(k)$ before measurement at the k-th instant to be received and $\hat{x}(k)$ is the updated estimation of the state vector after measurement at the k-th instant has been received. The Jacobian $J_{\gamma}(x)$ is

$$J_{\gamma}(x) = \frac{\partial \gamma}{\partial x}|_{x=\hat{x}^{-}(k)} = \begin{pmatrix} \frac{\partial \gamma_1}{\partial x_1} & \frac{\partial \gamma_1}{\partial x_2} & \cdots & \frac{\partial \gamma_1}{\partial x_m} \\ \frac{\partial \gamma_2}{\partial x_1} & \frac{\partial \gamma_2}{\partial x_2} & \cdots & \frac{\partial \gamma_2}{\partial x_m} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial \gamma_p}{\partial x_1} & \frac{\partial \gamma_p}{\partial x_2} & \cdots & \frac{\partial \gamma_p}{\partial x_m} \end{pmatrix} \tag{6.17}$$

The resulting expressions create first order approximations of ϕ and γ. Thus the linearized version of the plant is obtained:

$$\begin{aligned} x(k+1) &= \phi(\hat{x}(k)) + J_{\phi}(\hat{x}(k))[x(k) - \hat{x}(k)] + w(k) \\ z(k) &= \gamma(\hat{x}^{-}(k)) + J_{\gamma}(\hat{x}^{-}(k))[x(k) - \hat{x}^{-}(k)] + v(k) \end{aligned} \tag{6.18}$$

Now, the EKF recursion is as follows: First the time update is considered: by $\hat{x}(k)$ the estimation of the state vector at instant k is denoted. Given initial conditions $\hat{x}^{-}(0)$ and $P^{-}(0)$ the recursion proceeds as [331]:

- *Measurement update*. Acquire $z(k)$ and compute:

$$\begin{aligned} K(k) &= P^{-}(k)J_{\gamma}^{T}(\hat{x}^{-}(k))\cdot[J_{\gamma}(\hat{x}^{-}(k))P^{-}(k)J_{\gamma}^{T}(\hat{x}^{-}(k)) + R(k)]^{-1} \\ \hat{x}(k) &= \hat{x}^{-}(k) + K(k)[z(k) - \gamma(\hat{x}^{-}(k))] \\ P(k) &= P^{-}(k) - K(k)J_{\gamma}(\hat{x}^{-}(k))P^{-}(k) \end{aligned} \tag{6.19}$$

- *Time update*. Compute:

$$\begin{aligned} P^{-}(k+1) &= J_{\phi}(\hat{x}(k))P(k)J_{\phi}^{T}(\hat{x}(k)) + Q(k) \\ \hat{x}^{-}(k+1) &= \phi(\hat{x}(k)) + L(k)u(k) \end{aligned} \tag{6.20}$$

The schematic diagram of the EKF loop is given in Fig. 6.1.

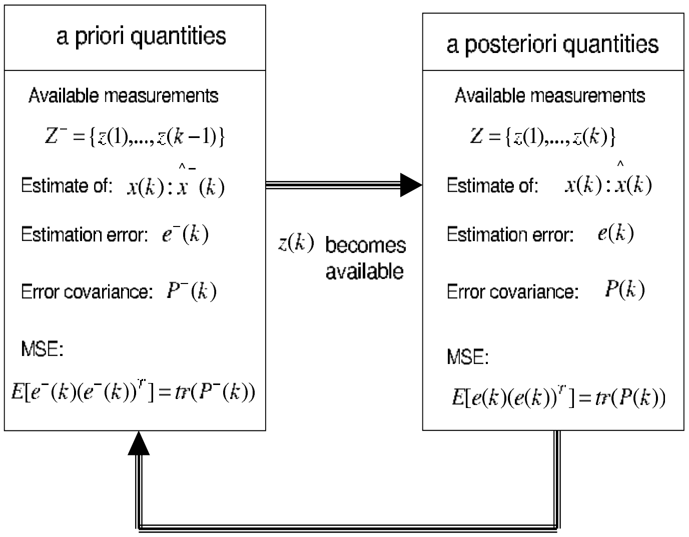

a priori quantities
Available measurements
$Z^- = \{z(1),...,z(k-1)\}$
Estimate of: $x(k) : \hat{x}^-(k)$
Estimation error: $e^-(k)$
Error covariance: $P^-(k)$
MSE:
$E[e^-(k)(e^-(k))^T] = tr(P^-(k))$
$z(k)$ becomes available
a posteriori quantities
Available measurements
$Z = \{z(1),...,z(k)\}$
Estimate of: $x(k) : \hat{x}(k)$
Estimation error: $e(k)$
Error covariance: $P(k)$
MSE:
$E[e(k)(e(k))^T] = tr(P(k))$

Fig. 6.1 Schematic diagram of the EKF loop

6.5 Sigma-Point Kalman Filters

The Sigma-Point Kalman Filter overcomes the flaws of Extended Kalman Filtering [263]. Unlike EKF no analytical Jacobians of the system equations need to be calculated as in the case for the EKF. This makes the sigma-point approach suitable for application in ”black-box” models where analytical expressions of the system dynamics are either not available or not in a form which allows for easy linearization. This is achieved through a different approach for calculating the posterior 1st and 2nd order statistics of a random variable that undergoes a nonlinear transformation. The state distribution is represented again by a Gaussian Random Variable but is now specified using a minimal set of deterministically chosen weighted sample points. The basic sigma-point approach can be described as follows:

1. A set of weighted samples (sigma-points) are deterministically calculated using the mean and square-root decomposition of the covariance matrix of the system's state vector. As a minimal requirement the sigma-point set must completely capture the first and second order moments of the prior random variable. Higher order moments can be captured at the cost of using more sigma-points.
2. The sigma-points are propagated through the true nonlinear function using functional evaluations alone, i.e. no analytical derivatives are used, in order to generate a posterior sigma-point set.

3. The posterior statistics are calculated (approximated) using tractable functions of the propagated sigma-points and weights. Typically, these take on the form of a simple weighted sample mean and covariance calculations of the posterior sigma points.

It is noted that the sigma-point approach differs substantially from general stochastic sampling techniques, such as Monte-Carlo integration (e.g Particle Filtering methods) which require significantly more sample points in an attempt to propagate an accurate (possibly non-Gaussian) distribution of the state. The sigma-point approach results in posterior approximations that are accurate to the third order for Gaussian inputs for all nonlinearities. For non-Gaussian inputs, approximations are accurate to at least the second-order, with the accuracy of third and higher-order moments determined by the specific choice of weights and scaling factors.

The Unscented Kalman Filter (UKF) is a special case of Sigma-Point Kalman Filters. The UKF is a discrete time filtering algorithm which uses the unscented transform for computing approximate solutions to the filtering problem of the form

$$\begin{aligned} x(k+1) &= \phi(x(k)) + L(k)U(k) + w(k) \\ y(k) &= \gamma(x(k)) + v(k) \end{aligned} \tag{6.21}$$

where $x(k)\in R^n$ is the system's state vector, $y(k)\in R^m$ is the measurement, $w(k)\in R^n$ is a Gaussian process noise $w(k)\sim N(0,Q(k))$, and $v(k)\in R^m$ is a Gaussian measurement noise $v(k)\sim N(0,R(k))$. The mean and covariance of the initial state $x(0)$ are $m(0)$ and $P(0)$, respectively.

Some basic operations performed in the UKF algorithm (*Unscented Transform*) are summarized as follows:

1) Denoting the current state mean as $\hat{x}$, a set of $2n+1$ sigma points is taken from the columns of the $n\times n$ matrix $\sqrt{(n+\lambda)P_{xx}}$ as follows:

$$\begin{aligned} x^0 &= \hat{x} \\ x^i &= \hat{x} + [\sqrt{(n+\lambda)}P_{xx}]_i,\ i = 1,\cdots,n \\ x^i &= \hat{x} - [\sqrt{(n+\lambda)}P_{xx}]_i,\ i = n+1,\cdots,2n \end{aligned} \tag{6.22}$$

and the associated weights are computed:

$$\begin{aligned} W_0^{(m)} &= \tfrac{\lambda}{(n+\lambda)} & W_0^{(c)} &= \tfrac{\lambda}{(n+\lambda)+(1-\alpha^2+b)} \\ W_i^{(m)} &= \tfrac{1}{2(n+\lambda)},\ i = 1,\cdots,2n & W_i^{(c)} &= \tfrac{1}{2(n+\lambda)} \end{aligned} \tag{6.23}$$

where $i = 1,2,\cdots,2n$ and $\lambda = \alpha^2(n+\kappa) - n$ is a scaling parameter, while α, β and κ are constant parameters. Matrix P_{xx} is the covariance matrix of the state x.

2) Transform each of the sigma points as

$$z^i = h(x^i)\ \ i = 0,\cdots,2n \tag{6.24}$$

3) Mean and covariance estimates for z can be computed as

$$\begin{gathered}\hat{z} \simeq \sum_{i=0}^{2n} W_i^{(m)} z^i \\ P_{zz} = \sum_{i=0}^{2n} W_i^{(c)} (z^i - \hat{z})(z^i - \hat{z})^T\end{gathered} \tag{6.25}$$

4) The cross-covariance of x and z is estimated as

$$P_{xz} = \simeq \sum_{i=0}^{2n} W_i^{(c)} (x^i - \hat{x})(z^i - \hat{z})^T \tag{6.26}$$

The matrix square root of positive definite matrix P_{xx} means a matrix $A = \sqrt{P_{xx}}$ such that $P_{xx} = AA^T$ and a possible way for calculation is Singular Values Decomposition (SVD).

Next the basic stages of the *Unscented Kalman Filter* are given:

As in the case of the Extended Kalman Filter and the Particle Filter, the Unscented Kalman Filter also consists of prediction stage (time update) and correction stage (measurement update) [171], [368].

Time update: Compute the predicted state mean $\hat{x}^-(k)$ and the predicted covariance $P_{xx}{}^-(k)$ as

$$\begin{gathered}[\hat{x}^-(k), P_{xx}^-(k)] = UT(f_d, \hat{x}(k-1), P_{xx}(k-1)) \\ P_{xx}^-(k) = P_{xx}(k-1) + Q(k-1)\end{gathered} \tag{6.27}$$

Measurement update: Obtain the new output measurement z_k and compute the predicted mean $\hat{z}(k)$ and covariance of the measurement $P_{zz}(k)$, and the cross covariance of the state and measurement $P_{xz}(k)$

$$\begin{gathered}[\hat{z}(k), P_{zz}(k), P_{xz}(k)] = UT(h_d, \hat{x}^-(k), P_{xx}^-(k)) \\ P_{zz}(k) = P_{zz}(k) + R(k)\end{gathered} \tag{6.28}$$

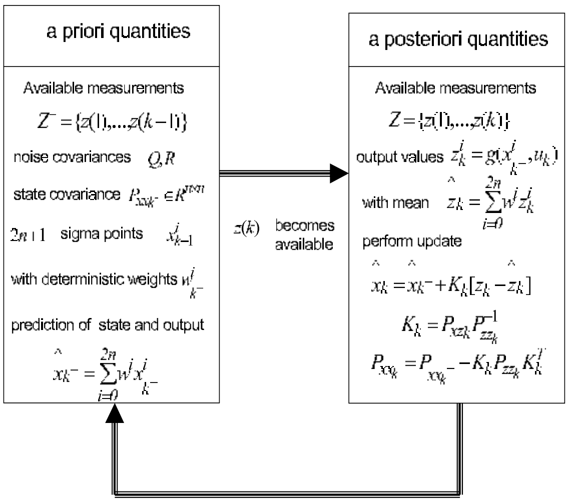

a priori quantities
Available measurements
$Z^- = \{z(1), \dots, z(k-1)\}$
noise covariances Q, R
state covariance $P_{xx_k^-} \in R^{nxn}$
$2n+1$ sigma points x_{k-1}^i
with deterministic weights $w_{k^-}^i$
prediction of state and output
$\hat{x}_k^- = \sum_{i=0}^{2n} w^i x_{k^-}^i$
$z(k)$ becomes available
a posteriori quantities
Available measurements
$Z = \{z(1), \dots, z(k)\}$
output values $z_k^i = g(x_{k^-}^i, u_k)$
with mean $\hat{z}_k = \sum_{i=0}^{2n} w^i z_k^i$
perform update
$\hat{x}_k = \hat{x}_k^- + K_k[z_k - \hat{z}_k]$
$K_k = P_{xz_k} P_{zz_k}^{-1}$
$P_{xx_k} = P_{xx_k^-} - K_k P_{zz_k} K_k^T$

Fig. 6.2 Schematic diagram of the Unscented Kalman Filter loop

Then compute the filter gain $K(k)$, the state mean $\hat{x}(k)$ and the covariance $P_{xx}(k)$, conditional to the measurement $y(k)$

$$\begin{aligned} K(k) &= P_{xz}(k)P_{zz}^{-1}(k) \\ \hat{x}(k) &= \hat{x}^{-}(k) + K(k)[z(k) - \hat{z}(k)] \\ P_{xx}(k) &= P_{xx}^{-}(k) - K(k)P_{zz}(k)K(k)^T \end{aligned} \tag{6.29}$$

The filter starts from the initial mean $m(0)$ and covariance $P_{xx}(0)$. The stages of state vector estimation with the use of the unscented filtering algorithm are depicted in Fig. 6.2.

6.6 Particle Filters

6.6.1 The Particle Approximation of Probability Distributions

The functioning of the Particle Filters will be explained. Particle Filtering is a method for state estimation that is not dependent on the probability density function of the measurements [128],[133]. In the general case, the equations of the optimal filter used for the calculation of the state-vector of a dynamical system do not have an explicit solution. This happens for instance when the process noise and the noise of the output measurement do not follow a Gaussian distribution. In that case approximation through Monte-Carlo methods can be used [12],[45],[82],[409]. A sampling of size N is assumed, i.e. N i.i.d. (independent identically distributed) variables $\xi^1, \xi^2, \cdots, \xi^N$. This sampling follows the p.d.f. $p(x)$ i.e. $\xi^{1:N} \sim p(x)$. Instead of $p(x)$ the function $p(x) \simeq p^N(x) = \frac{1}{N}\sum_{i=1}^{N} \delta_{\xi^i}(x)$ can be used. It is assumed that all points ξ^i have an equal weighted contribution to the approximation of $p(x)$. A more general approach would be if weight factors were assigned to the points ξ^i, which will also satisfy the normality condition $\sum_{i=1}^{N} w^i = 1$. In the latter case

$$p(x) \simeq p^N(x) = \sum_{i=1}^{N} w^i \delta_{\xi^i}(x) \tag{6.30}$$

If $p(\xi^i)$ is known then the probability $P(x)$ can be approximated using the discrete values of the p.d.f. $p(\xi^i) = w^i$. If sampling over the p.d.f. $p(x)$ is unavailable, then one can use a p.d.f. $\bar{p}(x)$ with similar support set, i.e. $p(x) = 0 \Rightarrow \bar{p}(x) = 0$. Then it holds $E(\phi(x)) = \int \phi(x)p(x)dx = \int \phi(x)\bar{p}(x)\frac{p(x)}{\bar{p}(x)}dx$. If the N samples of $\bar{p}(x)$ are available at the points $\tilde{\xi}^1 \cdots \tilde{\xi}^N$, i.e. $\bar{p}(\tilde{\xi})^i = \delta_{\tilde{\xi}^i}(x)$ and the weight coefficients w^i are defined as $w^i = \frac{p(\tilde{\xi}^i)}{\bar{p}(\tilde{\xi}^i)}$, then it is easily shown that

$$E(\phi(x)) \simeq \sum_{i=1}^{N} w^i \phi(\tilde{\xi}^i), \text{ where} \left\{ \begin{array}{c} \tilde{\xi}^{1:N} \sim \bar{p}(x) \\ w^i = p(\tilde{x}^i)/\bar{p}(\tilde{x}^i) \end{array} \right. \tag{6.31}$$

The meaning of Eq. (6.31) is as follows: assume that the p.d.f. $p(x)$ is unknown (target distribution), however the p.d.f. $\bar{p}(x)$ (importance law) is available. Then, it is sufficient to sample on $\bar{p}(x)$ and find the associated weight coefficients w^i so as to calculate $E(\phi(x))$.

6.6.2 The Prediction Stage

As in the case of the Kalman Filter or the Extended Kalman Filter the particles filter consists of the measurement update (correction stage) and the time update (prediction stage) [49],[350],[352],[409]. The prediction stage calculates $p(x(k)|Z^-)$ where $Z^- = \{z(1), z(2), \cdots, z(n-1)\}$ according to Eq. (6.30). It holds that:

$$p(x(k-1)|Z^-) = \sum_{i=1}^{N} w_{k-1}^i \delta_{\xi_{k-1}^i}(x(k-1)) \tag{6.32}$$

while from Bayes formula it holds $p(x(k)|Z^-) = \int p(x(k)|x(k-1))p(x(k-1)|Z^-)dx$. Using also Eq. (6.32) one finally obtains

$$\begin{array}{l} p(x(k)|Z^-) = \sum_{i=1}^{N} w_{k-1}^i \delta_{\xi_{k^-}^i}(x(k)) \\ \text{with } \xi_{k^-}^i \sim p(x(k)|x(k-1) = \xi_{k-1}^i) \end{array} \tag{6.33}$$

The meaning of Eq. (6.33) is as follows: the state equation of the system is executed N times, starting from the N previous values of the state vectors $x(k-1) = \xi_{k-1}^i$

$$\begin{array}{c} \hat{x}(k+1) = \phi(\hat{x}(k)) + L(k)u(k) + w(k) \\ z(k) = \gamma(\hat{x}(k)) + v(k) \end{array} \tag{6.34}$$

Thus estimations of the current value of the state vector $\hat{x}(k)$ are obtained, and consequently the mean value of the state vector will be given from Eq. (6.33). This means that the value of the state vector which is calculated in the prediction stage is the result of the weighted averaging of the state vectors which were calculated after running the state equation, starting from the N previous values of the state vectors ξ_{k-1}^i.

6.6.3 The Correction Stage

The a-posteriori probability density is found using Eq. (6.33). Now, a new position measurement $z(k)$ is obtained and the objective is to calculate the corrected probability density $p(x(k)|Z)$, where $Z = \{z(1), z(2), \cdots, z(k)\}$. From Bayes law it holds that $p(x(k)|Z) = \frac{p(Z|x(k))p(x(k))}{p(Z)}$ which can be also written as

$$p(x(k)|Z) = \frac{p(z(k)|x(k))p(x(k)|Z^-)}{\int p(z(k)|x(k), Z^-)p(x(k)|Z^-)dx} \tag{6.35}$$

Substituting Eq. (6.33) into Eq. (6.35) and after intermediate calculations one finally obtains

$$\begin{gathered} p(x(k)|Z) = \sum_{i=1}^{N} w_k^i \delta_{\xi_{k^-}^i}(x(k)) \\ \text{where } w_k^i = \frac{w_{k^-}^i p(z(k)|x(k)=\xi_{k^-}^i)}{\sum_{j=1}^{N} w_{k^-}^j p(z(k)|x(k)=\xi_{k^-}^j)} \end{gathered} \tag{6.36}$$

Eq. (6.36) denotes the corrected value for the state vector. The recursion of the Particle Filter proceeds in a way similar to the update of the Kalman Filter or the Extended Kalman Filter, i.e. [333],[334],[343]:

- *Measurement update*: Acquire $z(k)$ and compute

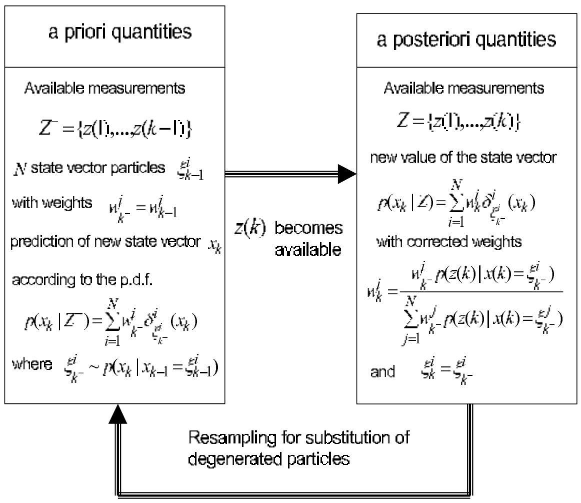
$$\begin{gathered} \text{new value of the state vector} \\ p(x(k)|Z) = \sum_{i=1}^{N} w_k^i \delta_{\xi_{k^-}^i}(x(k)) \\ \text{with corrected weights} \\ w_k^i = \frac{w_{k^-}^i p(z(k)|x(k)=\xi_{k^-}^i)}{\sum_{j=1}^{N} w_{k^-}^i p(z(k)|x(k)=\xi_{k^-})^i} \text{and } \xi_k^i = \xi_{k^-}^i \end{gathered} \tag{6.37}$$

  Resampling for substitution of the degenerated particles

- *Time update*: compute state vector $x(k+1)$ according to the pdf

$$\begin{gathered} p(x(k+1)|Z) = \sum_{i=1}^{N} w_k^i \delta_{\xi_k^i}(x(k)) \\ \text{where } \xi_k^i \sim p(x(k+1)|x(k) = \xi_k^i) \end{gathered} \tag{6.38}$$

The stages of state vector estimation with the use of the Particle Filtering algorithm are depicted in Fig. 6.3.



a priori quantities
Available measurements
$Z^- = \{z(1),\ldots,z(k-1)\}$
N state vector particles ξ_{k-1}^i
with weights $w_{k^-}^i = w_{k-1}^i$
prediction of new state vector x_k
according to the p.d.f.
$p(x_k|Z^-) = \sum_{i=1}^{N} w_{k^-}^i \delta_{\xi_{k^-}^i}(x_k)$
where $\xi_{k^-}^i \sim p(x_k|x_{k-1} = \xi_{k-1}^i)$
$z(k)$ becomes available
a posteriori quantities
Available measurements
$Z = \{z(1),\ldots,z(k)\}$
new value of the state vector
$p(x_k|Z) = \sum_{i=1}^{N} w_k^i \delta_{\xi_{k^-}^i}(x_k)$
with corrected weights
$w_k^i = \frac{w_{k^-}^i p(z(k)|x(k)=\xi_{k^-}^i)}{\sum_{j=1}^{N} w_{k^-}^j p(z(k)|x(k)=\xi_{k^-}^j)}$
and $\xi_k^i = \xi_{k^-}^i$
Resampling for substitution of degenerated particles

Fig. 6.3 Schematic diagram of the Particle Filter loop

6.6.4 The Resampling Stage

The algorithm of particle filtering which is described through Eq. (6.33) and Eq. (6.36) has a significant drawback: after a certain number of iterations k, almost all the weights w_k^i become 0. In the ideal case, all the weights should converge to the value $\frac{1}{N}$, i.e. the particles should have the same significance. The criterion used to define a sufficient number of particles is $N_k^{\text{eff}} = \frac{1}{\sum_{i=1}^{N} {w_k^i}^2} \in [1,N]$. When N_k^{eff} is close to value N then all particles have almost the same significance. However using the algorithm of Eq. (6.33) and Eq. (6.36) results in $N_k^{\text{eff}} \rightarrow 1$, which means that the particles are degenerated, i.e. they lose their effectiveness. Therefore, it is necessary to modify the algorithm so as to assure that degeneration of the particles will not take place [335],[381],[409],[472].

When N_k^{eff} is small, then most of the particles have weights close to 0 and consequently they have a negligible contribution to the estimation of the state vector. To overcome this drawback, the PF algorithm weakens such particles in favor of particles that have a non-negligible contribution. Therefore, the particles of low weight factors are removed and their place is occupied by duplicates of the particles with high weight factors. The total number of particles remains unchanged (equal to N) and therefore this procedure can be viewed as a ”resampling” or ”redistribution” of the particles set.

The particles resampling presented above maybe slow if not appropriately tuned. There are improved versions of it which substitute the particles of low importance with those of higher importance. A first choice would be to perform a multinomial resampling. N particles are chosen between $\{\xi_k^1, \cdots, \xi_k^N\}$ and the corresponding weights are $w_k^1, \cdots, w_k^N$. The number of times each particle is selected is given by $[j_1, \cdots, j_n]$. Thus a set of N particles is again created, the elements of which are chosen after sampling with the discrete distribution $\sum_{i=1}^{N} w_k^i \delta_{\xi_k^i}(x)$. The particles $\{\xi_k^1, \cdots, \xi_k^N\}$ are chosen according to the probabilities $\{w_k^1, \cdots, w_k^N\}$. The selected particles are assigned with equal weights $\frac{1}{N}$.

6.6.5 Approaches to the Implementation of Resampling

Although sorting of the particles’ weights is not necessary for the convergence of the particle filter algorithm, there are variants of the resampling procedure of $(\xi_k^i, w_k^i \; i = 1, \cdots, N)$ which are based on previous sorting in decreasing order of the particles’ weights [50]. Sorting of particles’ weights gives $w^{s[1]} > w^{s[2]} > \cdots > w^{s[N]}$. A random numbers generator is evoked and the resulting numbers $u^{i:N} \sim U[0,1]$ fall in the partitions of the interval $[0,1]$. The width of these partitions is w^i and thus a redistribution of the particles is generated. For instance, in a wide partition of width w^j will be assigned more particles than to a narrow partition of witdh w^m (see Fig. 6.4).

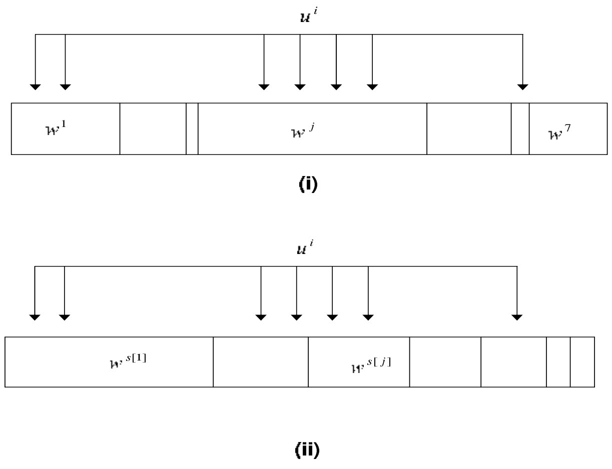
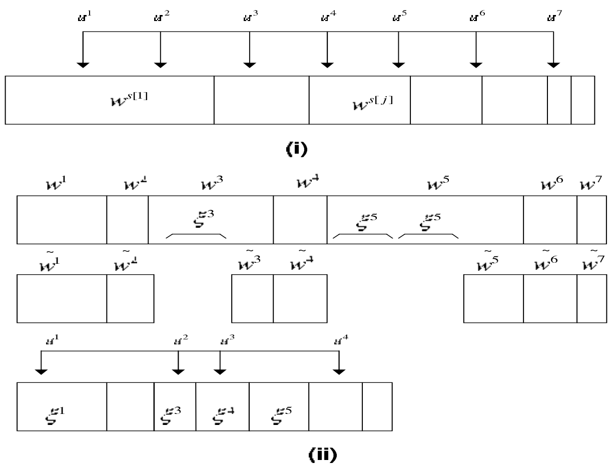


Fig. 6.4 Multinomial resampling: (i) conventional resampling, (ii) resampling with sorted weights

Two other methods that have been proposed for the implementation of resampling in Particle Filtering are explained in the sequel. These are Kitagawa's approach and the residuals resampling approach [50]. In *Kitagawa's resampling* the speed of the resampling procedure is increased by using less the random numbers generator. The weights are sorted again in decreasing order $w^{s[j]}$ so as to cover the region that corresponds to the interval $[0,1]$. Then the random numbers generator is used to produce the variable $u_1 \sim U[0,\frac{1}{N}]$, according to $u_1 \sim U[0,\frac{1}{N}]$ and $u^i = u^1 + \frac{i}{N}$, $i = 2,\cdots,N$. The rest of the variables u^i are produced in a deterministic way (see Fig. 6.5(i)) [193].

In the *residuals resampling* approach, the redistribution of the residuals is performed as follows: at a first stage particle ξ^i is chosen in a deterministic way $[w^i/N]$ times (with rounding). The residual weights are $\tilde{w}^i = w^i - N[w^i/N]$ and are normalized. Thus, a probability distribution is generated. The rest of the particles are selected according to the multinomial resampling described above (see Fig. 6.5(ii)). The method can be applied if the number $\tilde{N}$ which remains at the second stage is small, i.e. when $N^{\text{eff}} = 1/\sum_{i=1}^{N} w_i^2$ is small.

Finally, it is mentioned that tuning of the resampling procedure is of importance for succeeding improved performance of the Particle Filter algorithm and convergence to an accurate estimation of the state vector. To tune resampling the following issues are taken into account [333],[409]: (i) control of the diversity of the particles population, (ii) avoidance of particles' impoverishment, i.e. avoidance of absence of particles in the vicinity of the correct state, (iii) selection of the optimal number of particles, (iv) selection of the particles' subsets to be substituted



u^1 u^2 u^3 u^4 u^5 u^6 u^7
$w^{s[1]}$
$w^{s[j]}$
(i)
w^1 w^2 w^3 w^4 w^5 w^6 w^7
ξ^3 ξ^5 ξ^5
$\tilde{w}^1$ $\tilde{w}^2$ $\tilde{w}^3$ $\tilde{w}^4$ $\tilde{w}^5$ $\tilde{w}^6$ $\tilde{w}^7$
u^1 u^2 u^3 u^4
ξ^1 ξ^3 ξ^4 ξ^5
(ii)

Fig. 6.5 Multinomial resampling: (i) Kitagawa's approach, (ii) residuals approach

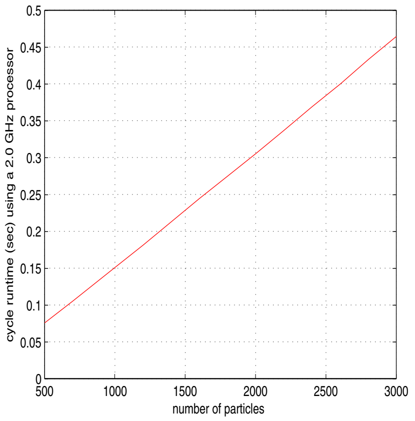


Fig. 6.6 Runtime of the particle filtering algorithm with respect to the number of particles

(stratified resampling), and (v) parallel implementation of resampling for improving the speed of the PF algorithm. It is noted that when resampling does not include particles' sorting the computation time of PF scales up linearly with the number of particles [370], as shown in Fig. 6.6. On the other hand when particles' sorting is performed during resampling the computational complexity of PF is $O(Nlog(N))$ [41],[266].

6.7 Application of Estimation Methods to Industrial Systems Control

An application area of the previously analyzed estimation methods is in *sensorless control* or control with a reduced number of sensors for electric motors. Measuring the motor's angle or current it is possible to estimate through filtering the rest of the parameters of motor's state vector and consequently to implement a state feedback control scheme. For linear electric motor models subject to Gaussian measurement or process noise the Kalman Filter is the optimal state estimator, since it results in minimization of the trace of the estimation error's covariance matrix. For nonlinear electric motor models, subject to Gaussian noise one can use the Extended Kalman Filter. However, since the Extended Kalman Filter is based on a linearization of the system dynamics using a first order Taylor expansion, there is neither a proof of its convergence, nor a proof that the estimation succeeded by the EKF satisfies optimality criteria. Finally, to overcome the limitations of KF and of EKF, Sigma Point Kalman Filters (and particularly the Unscented Kalman Filter) can be used.

6.7.1 Kalman Filter-Based Control of Electric Motors

The control law of Eq. (4.3) is used to make the motor track the desirable trajectory, however the state vector x used in the control law is estimated through Kalman

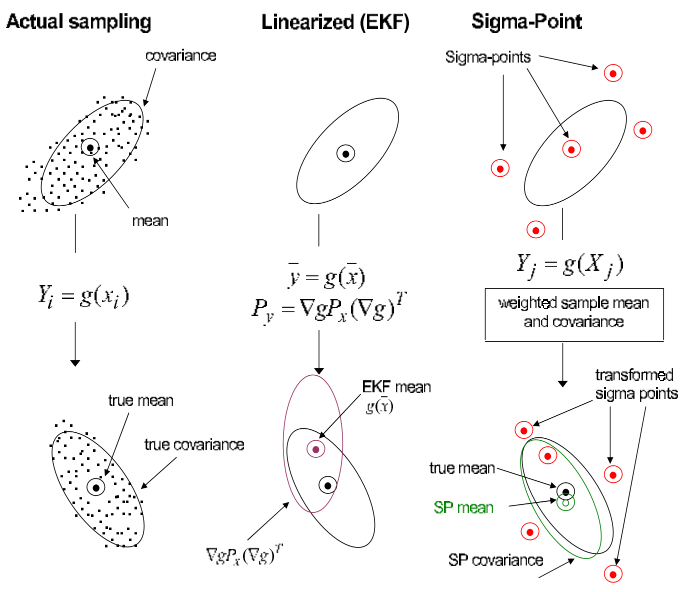
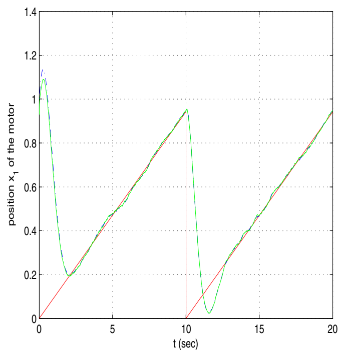


Fig. 6.7 Example of approximation of a 2D distribution by the Sigma-Point Kalman Filtering approach

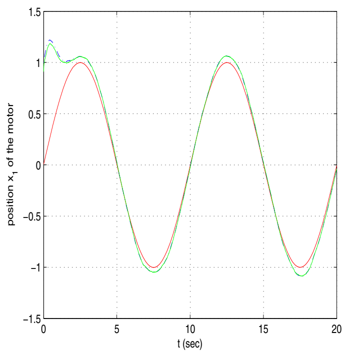
Filtering, as described in Eq. (6.11)-Eq. (6.12). Next, the same feedback control law is applied to the DC motor but this time the state vector is reconstructed through encoder measurements.

To simulate Gaussian noise Matlab's function *randn()* can be used. The process and measurement noises are considered to be uncorrelated. The process noise covariance matrix was taken to be $E\{w(i)w^T(j)\} = Q\delta(i-j)$, with diagonal elements with diagonal elements $w_{ii} = 10^{-3}$. The covariance matrix of the measurement noise was defined $E\{v(i)v^T(j)\} = R\delta(i-j)$, with diagonal elements $r_{ii} = 10^{-2}$. The estimation error covariance matrix $P \in R^{3\times 3}$ and the KF gain $K \in R^{3\times 1}$ were used in Eq. (6.11)-Eq. (6.12).

To perform sensorless control only measurements of the rotor's angle θ need to be used. The sampling period is taken to be $T_s = 0.01$sec. The tracking performance of the Kalman Filter-based control loop, in the case of a see-saw and a sinusoidal setpoint are depicted in Fig. 6.8, Fig. 6.9 and Fig. 6.10. The reference setpoint in denoted by the red line, the state vector variables of the motor are denoted by the blue line, while the estimated state vector elements are described by the green line.



(a)



(b)

Fig. 6.8 Parameter x_1 of the state vector in state estimation with use of the Kalman Filter (a) when tracking a see-saw set-point (b) when tracking a sinusoidal setpoint

6.7.2 Extended Kalman Filter-Based Control of Electric Motors

The induction motor model, given in Eq. (4.58) was considered. It was assumed that the torque T_L is due to the rotation of a rigid link, i.e. $T_L = mglsin(\theta)$. The model's state variables were taken to be $x_1 = \theta$ and $x_2 = \dot{\theta}$. The *measurement update* of the EKF is given by Eq. (7.3), while the time update of the EKF is given by Eq. (7.4). The measured state variable was supposed to be the rotor's angle θ. The

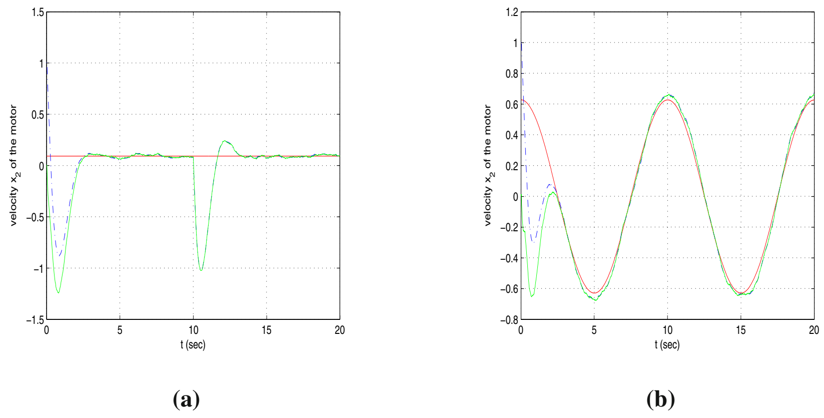


Fig. 6.9 Parameter x_2 of the state vector in state estimation with use of the Kalman Filter (a) when tracking a see-saw set-point (b) when tracking a sinusoidal setpoint

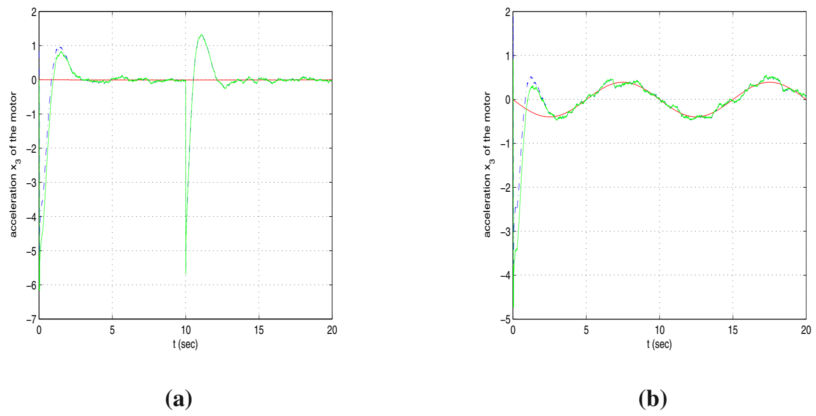
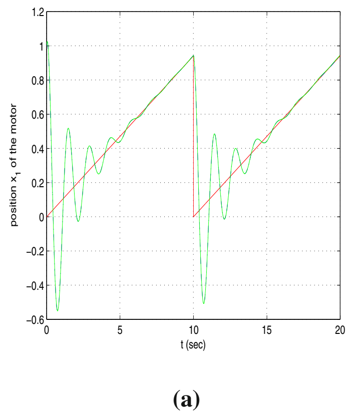


Fig. 6.10 Parameter x_3 of the state vector in estimation was performed with use of the Kalman Filter (a) when tracking a see-saw setpoint (b) when tracking of a sinusoidal set-point

Jacobian $J_\phi(x)$ is the 2×2 Jacobian of ϕ calculated through the expansion $\phi(x(k)) = \phi(\hat{x}^-(k)) + J_\phi[x(k) - \hat{x}^-(k)] + \cdots$, and is given by

$$J_\phi = \frac{\partial\phi}{\partial x}|_{x=\hat{x}^-(k)} = \begin{pmatrix} \frac{\partial\phi_1}{\partial x_1} & \frac{\partial\phi_1}{\partial x_2} \\ \frac{\partial\phi_2}{\partial x_1} & \frac{\partial\phi_2}{\partial x_2} \end{pmatrix} = \begin{pmatrix} 1 & T \\ -\frac{mglT}{J'}cos(x_1(k)) & \frac{1-bT}{J'} \end{pmatrix} \tag{6.39}$$

where T is the sampling period. Likewise, γ is expanded about $\hat{x}^-(k)$ as $\gamma(x(k)) = \gamma(\hat{x}^-(k)) + J_\gamma[x(k) - \hat{x}^-(k)] + \cdots$, where $\hat{x}^-(k)$ is the estimation of the state vector $x(k)$ before measurement at the k-th instant to be received and $\hat{x}(k)$ is the updated



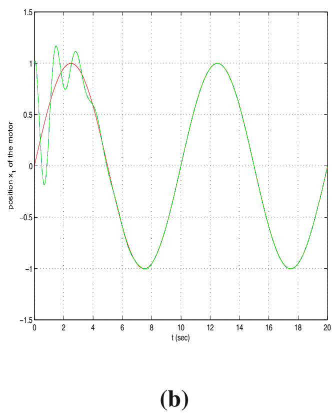
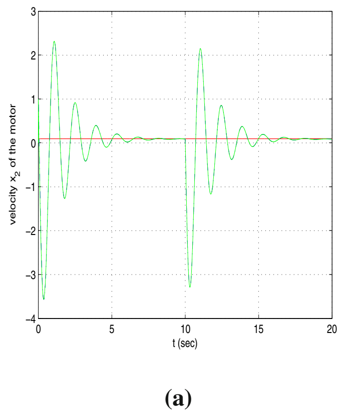


Fig. 6.11 Parameter x_1 of the state vector in estimation with use of the Extended Kalman Filter (a) when tracking a see-saw set-point (b) when tracking a sinusoidal setpoint



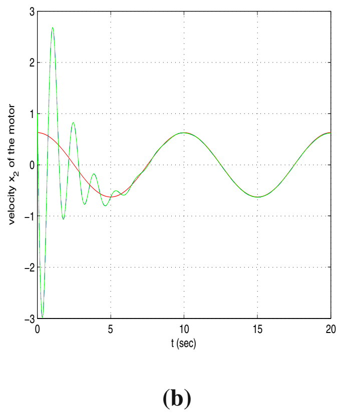


Fig. 6.12 Parameter x_2 of the state vector in state estimation with use of the Extended Kalman Filter (a) when tracking a see-saw set-point (b) when tracking a sinusoidal setpoint

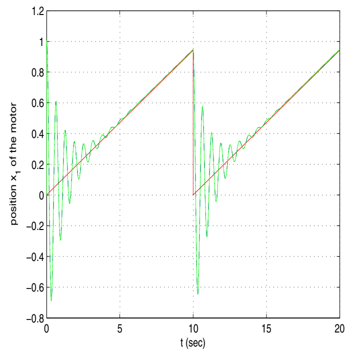
estimation of the state vector after measurement at the k-th instant has been received. The 1×2 Jacobian $J_\gamma(x)$ is

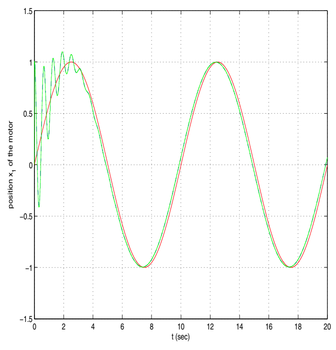
$$J_\gamma(x) = \frac{\partial\gamma}{\partial x}|_{x=\hat{x}^-(k)} = \begin{pmatrix} \frac{\partial\gamma_1}{\partial x_1} & \frac{\partial\gamma_1}{\partial x_2} \end{pmatrix} = \begin{pmatrix} 1 & 0 \end{pmatrix} \tag{6.40}$$

To implement sensorless control for the decoupled field-oriented induction motor model described in subsection 4.3.5,only measurements of the rotor's angle θ have to be used. As it can be seen in Fig. 6.11 and Fig. 6.12 the sensorless controller succeeded asympotic elimination of the tracking error despite abrupt changes in the reference trajectory, or the existence of process and measurement noises.

6.7.3 Unscented Kalman Filter-Based Control of Electric Motors

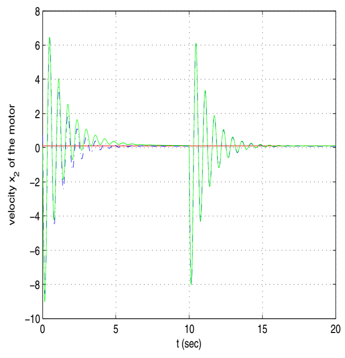
In Unscented Kalman Filter-based control a set of suitably chosen weighted sample points (sigma points) were propagated through the nonlinear system and used to approximate the true value of the system's state vector and of the state vector's covariance matrix. As explained in Section 6.5 the UKF algorithm consists of

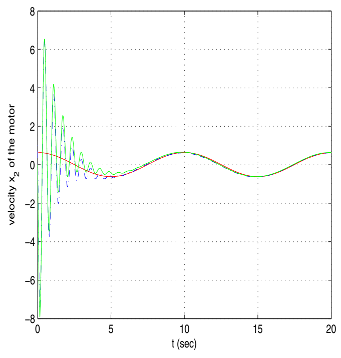




(a) **(b)**

Fig. 6.13 Parameter x_1 of the state vector in estimation was performed with use of the Unscented Kalman Filter (a) when tracking a see-saw setpoint (b) when tracking of a sinusoidal setpoint





(a) **(b)**

Fig. 6.14 Parameter x_2 of the state vector in estimation was performed with use of the Unscented Kalman Filter (a) when tracking a see-saw setpoint (b) when tracking of a sinusoidal setpoint

two-stages, the *time update* and the *measurement update*, which can be summarized as follows:

The *time update* of the UKF is

$$x_k^i = \phi(x_{k-1}^i) + L(k-1)U(k-1),\ i = 0, 1, \cdots, 2n$$

$$\hat{x}_k^- = \sum_{i=0}^{2n} w_i x_{k^-}^i$$

$$P_{xxk^-} = P_{xxk-1} + Q_k$$

The *measurement update* of the UKF is

$$z_k^i = h(x_{k^-}^i, u_k) + r_k,\ i = 0, 1, \cdots, 2n$$

$$\hat{z}_k = \sum_{i=0}^{2n} w_i z_k^i$$

$$P_{zz_k} = \sum_{i=0}^{2n} w_i [z_k^i - \hat{z}_k][z_k^i - \hat{z}_k]^T + R_k$$

$$P_{xz_k} = \sum_{i=0}^{2n} w_i [x_{k^-}^i - \hat{x_{k^-}}][z_k^i - \hat{z}_k]^T$$

$$K_k = P_{xz_k} {P_{zz_k}}^{-1}$$

$$\hat{x}_k = \hat{x}_{k^-} + K_k [z_k - \hat{z}_k]$$

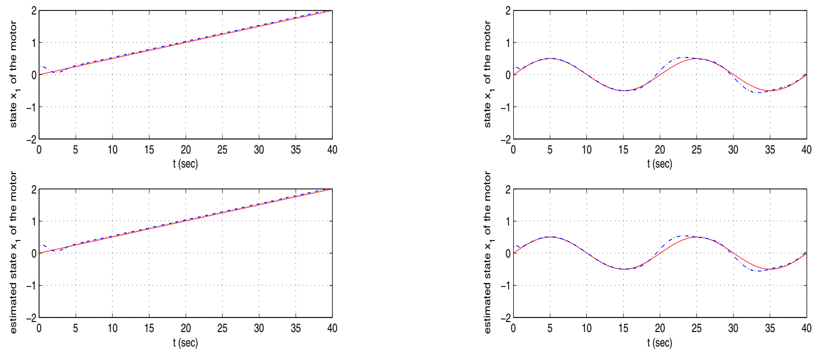
$$P_{xxk} = P_{k^-} - K_k P_{zzk} K_k^T$$

The simulation experiments of Fig. 6.13 and 6.14 show the good tracking performance of the UKF-based control loop, in the case of time varying setpoints (such as see-saw and sinusoidal reference trajectories).

6.7.4 Particle Filter-Based Control of Electric Motors

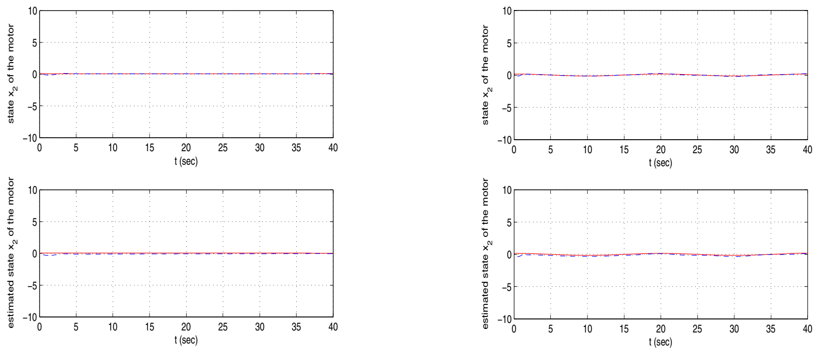
The Particle Filter is also applied for estimating the state vector of a DC motor by processing measurements only of the motor's output (angle) θ. The DC motor model was analyzed in subsection 4.3.2. The estimated state vector is again used by the control law of Eq. (4.3) so as to make the motor's state vector track desirable set-points.

From the simulation experiments it can be deduced that the particle filtering algorithm has satisfactory performance and results in accurate estimations of the motor's state vector . Of course the number of the particles influences the performance of the algorithm. The accuracy of the estimation succeeded by the particle filter algorithm improves as the number of particles increase. Indicative results about the PF's tracking accuracy, for $N = 2500$ particles and for a sinusoidal setpoint, are given in Table 6.1.



(a) **(b)**

Fig. 6.15 Parameter x_1 of the state vector in estimation with use of the Particle Filter under Rayleigh noise (a) when tracking a see-saw set-point (b) when tracking a sinusoidal set-point

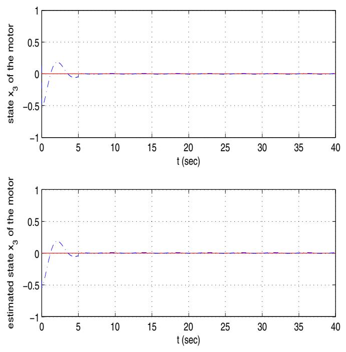


(a) **(b)**

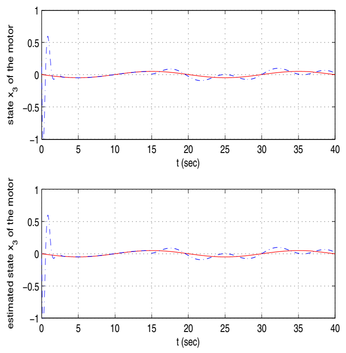
Fig. 6.16 Parameter x_2 of the state vector in state estimation with use of the Particle Filter under Rayleigh noise (a) when tracking a see-saw set-point (b) when tracking a sinusoidal setpoint

Table 6.1 KF and PF variance for $N = 2500$

parameter	θ	$\dot{\theta}$	$\ddot{\theta}$
KF	0.0543	0.2796	1.1596
PF	0.0003	0.0346	0.0678



(a)



(b)

Fig. 6.17 Parameter x_3 of the state vector in state estimation with use of the Particle Filter under Rayleigh noise (a) when tracking a see-saw set-point (b) when tracking a sinusoidal setpoint

Chapter 7
Sensor Fusion-Based Control for Industrial Systems

Abstract. Sensor fusion with the use of filtering methods is studied and state estimation of nonlinear systems based on the fusion of measurements from distributed sources is proposed for the implementation of stochastic control loops for industrial systems. Extended Kalman and Particle Filtering are first proposed for estimating, through multi-sensor fusion, the state vector of an industrial robotic manipulator and the state vector of a mobile robot. Moreover, sensor fusion with the use of Kalman and Particle Filtering is proposed for the reconstruction from output measurements the state vector of a ship which performs dynamic positioning.

7.1 Sensor Fusion-Based Control of Industrial Robots

7.1.1 The Sensor Fusion Problem

The next issue to be analyzed is that of fusion of measurements coming from distributed sensors, with the use of nonlinear filtering methods. Sensor fusion enables more accurate estimation of the state vector variables as well as robustness to failure of sensors and measuring devices. Moreover, it enables to substitute measurements coming from costly and difficult to install or maintain sensors by measurements given from low cost sensors or sensors which are suitable for harsh industrial environments. Thus, sensor fusion-based control finally results in more reliable and fault tolerant control loops. The fused data are used to reconstruct the state vector of an industrial robot, such as the one depicted in Fig. 1.1, and the estimated state vector is in turn used in a control loop. The robot's end effector (tool) has to follow accurately a specified trajectory expressed in cartesian coordinates. Then, through the solution of the inverse kinematic problem, this information is mapped to the vector of the angles of the robot's joints and an appropriate control signal is generated.

However, measurements of the joint angles from the encoders of the robot's motors do not suffice always, because they do not coincide with the angles of the joints when the flexibility of the joints cannot be neglected (see Fig. 7.1). To avoid inaccuracies in joints measurements provided by the encoders, the position of the

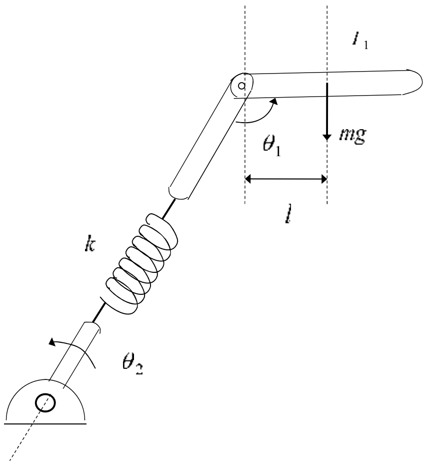


Fig. 7.1 A flexible joint of an industrial robotic manipulator

end-effector is measured, often using a laser tracker, which is a rather expensive sensor. Alternatively a sensor fusion approach can be followed to make the robot's tool track a predefined trajectory. This approach is as follows: (i) the laser tracker is substituted by a low cost accelerometer mounted on the robot's end effector and acceleration measurements in Cartesian coordinates are collected, (ii) measurements from the encoders of the robot's motors are also collected, (iii) the robot's state vector is estimated through fusion of the measurements coming from the accelerometer and the encoders, (iv) the estimated state vector is used to generate a suitable control signal.

With reference to the 3-DOF industrial robotic manipulator of Fig. 1.1 the goal is to estimate the joints positions θ^i, and the joints angular velocities $\dot{\theta}^i$, where i denotes the i-th robot joint, by measuring the motors angles θ_m^i and the Cartesian acceleration of the robot's end effector. The measurement obtained by the motors' encoders is related to the angle of the joint θ through

$$\theta_m = \frac{1}{r_g}\{\theta + \frac{1}{k}[D(\theta)\ddot{\theta} + h(\theta,\dot{\theta}) + G(\theta)]\} \tag{7.1}$$

Denoting by J the Jacobian of the robot, the cartesian acceleration is related to the state vector through

$$\ddot{p}(t) = J(\theta)\ddot{\theta}(t) + (\sum_{i=1}^{3}\frac{\partial J(\theta)}{\partial \theta_i}\dot{\theta}_i)\dot{\theta} \tag{7.2}$$

Thus in the proposed sensor fusion approach the measurement vector is given by $h(t) = [\theta_m(t), \ddot{p}(t)]^T$, where $\theta_m(t)$ is the vector of the measured motor angles θ_m^i, $i = 1,\cdots,3$ and $\ddot{p}$ is the cartesian acceleration vector in the accelerometer frame.

7.1.2 Application of EKF and PF for Sensor Fusion

7.1.2.1 EKF-Based State Estimation for Sensor Fusion of the Industrial Robot Model

Next, an overview of the Extended Kalman Filtering for the nonlinear state-measurement model is given. The EKF loop is depicted in Fig. 6.1.

Given initial conditions $\hat{x}(0)$ and $\hat{P}^-(0)$ the recursion proceeds as:

- *Measurement update*. Acquire $z(k)=[\theta_m(k),\ddot{p}(k)]^T$ and compute:

$$\begin{aligned}K(k)&=P^-(k)J_\gamma^T(\hat{x}^-(k))[J_\gamma(\hat{x}^-(k))P^-(k)J_\gamma^T(\hat{x}^-(k))+R(k)]^{-1}\\ \hat{x}(k)&=\hat{x}^-(k)+K(k)[z(k)-\gamma(\hat{x}^-(k))]\\ P(k)&=P^-(k)-K(k)J_\gamma(\hat{x}^-(k))P^-(k)\end{aligned} \tag{7.3}$$

- *Time update*. Compute:

$$\begin{aligned}P^-(k+1)&=J_\phi(\hat{x}(k))P(k)J_\phi^T(\hat{x}(k))+Q(k)\\ \hat{x}^-(k+1)&=\phi(\hat{x}(k))\end{aligned} \tag{7.4}$$

The application of the EKF algorithm for sensor fusion and state estimation in the 3-DOF industrial robot is described by Eq. (1.1) and next by Eq. (1.11). The measurement relations given by Eq. (7.1) and Eq. (7.2) result in intensive numerical computations. The measurement equations are nonlinear and are given by Eq. (7.1) and Eq. (7.2). Thus, the time update of the EKF is based on linear relations and uses $\phi(\hat{x}(k))=A{\cdot}\hat{x}$. However, the measurement update is based on nonlinear relations and needs extensive algebraic operations.

The *measurement update* of the EKF is given by Eq. (7.3), while the time update of the EKF is given by Eq. (7.4). The measured quantities are

$$\gamma(\theta,\dot{\theta})=\begin{pmatrix}\gamma_A(\theta,\dot{\theta})\\ \gamma_B(\theta,\dot{\theta})\end{pmatrix}=\begin{pmatrix}\theta_m\\ \ddot{p}\end{pmatrix}=\begin{pmatrix}\frac{1}{r_g}\{\theta+\frac{1}{k}[D(\theta)\ddot{\theta}+h(\theta,\dot{\theta})+G(\theta)]\\ J(\theta)\ddot{\theta}(t)+(\sum_{i=1}^3\frac{\partial J(\theta)}{\partial\theta_i}\dot{\theta}_i)\dot{\theta}\end{pmatrix} \tag{7.5}$$

To use the sensor measurements in Eq. (7.5), the Jacobian $J{\in}R^{3\times3}$ that relates the cartesian coordinates of the end-effector to the angles of the joints has to be calculated. To apply the EKF equations one has to calculate the Jacobian $J_\gamma{\in}R^{2\times6}$.

$$J=\begin{pmatrix}\frac{\partial x}{\partial\theta_1}&\frac{\partial x}{\partial\theta_2}&\frac{\partial x}{\partial\theta_3}\\ \frac{\partial y}{\partial\theta_1}&\frac{\partial y}{\partial\theta_2}&\frac{\partial y}{\partial\theta_3}\\ \frac{\partial z}{\partial\theta_1}&\frac{\partial z}{\partial\theta_2}&\frac{\partial z}{\partial\theta_3}\end{pmatrix},\ J_\gamma=\begin{pmatrix}\frac{\partial\gamma_A}{\partial\theta_1}&\frac{\partial\gamma_A}{\partial\theta_2}&\frac{\partial\gamma_A}{\partial\theta_3}&\frac{\partial\gamma_A}{\partial\dot{\theta}_1}&\frac{\partial\gamma_A}{\partial\dot{\theta}_2}&\frac{\partial\gamma_A}{\partial\dot{\theta}_3}\\ \\ \frac{\partial\gamma_B}{\partial\theta_1}&\frac{\partial\gamma_B}{\partial\theta_2}&\frac{\partial\gamma_B}{\partial\theta_3}&\frac{\partial\gamma_B}{\partial\dot{\theta}_1}&\frac{\partial\gamma_B}{\partial\dot{\theta}_2}&\frac{\partial\gamma_B}{\partial\dot{\theta}_3}\end{pmatrix} \tag{7.6}$$

Moreover, to calculate $\gamma_B(\theta,\dot{\theta})$, the partial derivatives $\partial J/\partial\theta_i$, and $\partial J/\partial\dot{\theta}_i$, $i=1,2,3$ have to be computed. Thus, it becomes clear that the application of the EKF algorithm for sensor fusion in the case of the 3-DOF robotic manipulator ends at complicated calculations.

7.1.2.2 Overview of PF-Based State Estimation

The Particle Filtering algorithm approximates the probability density function pdf(x_k) that describes the probability to locate the state vector x_k at time instant k in a certain region of the state space. After k output measurements $z_1, z_2, ..., z_k$ generated by the input sequence $u_1, u_2, ..., u_k$ a state vector $x_k^{[m]}$ belongs to the set of particles X_k that approximate accurately the real state vector x_k with probability given by the Bayes filter $x_k^{[m]} \sim p(x_k|z_{1:k}, u_{1:k})$. The stages of Particle Filtering, previously described in Section 6.6 are summarized in Table 7.1.

Table 7.1 Particle Filtering Stages

```
1.Algorithm of Particle Filtering$(X_{k-1}, u_k, z_k)$
2.  initialize$X_k = \emptyset$
3.  for $m = 1$ to $M$ do
4.      sample $x_k^{[m]} \sim p(x_k|u_k, x_{k-1}^{[m]})$          $\rightarrow$ time update
5.      $w_k^{[m]} = p(z_k|x_k^{[m]})$                            $\rightarrow$ measurement update
6.      $X_k = X_k + \{x_k^{[m]}, w_k^{[m]}\}$
7.  endfor
8.  for $m = 1$ to $M$ do                                  $\rightarrow$ resampling
9.      draw $x_k^{[i]}$ with probability $\propto w_k^i$
10.      add $x_k^{[i]}$ to $X_k$
11.  endfor
12.  return $X_k$
```

Recalling also Fig. 6.3, it can be observed that as in the case of the Kalman Filter or the Extended Kalman Filter, the Particles Filter consists of the measurement update (correction stage) and the time update (prediction stage).

1. The *prediction stage* (time update): The prediction stage calculates the probability $p(x(k)|Z^-)$ where $Z^- = \{z(1), z(2), \cdots, z(k-1)\}$. It holds that $p(x(k-1)|Z^-) = \sum_{i=1}^N w_{k-1}^i \delta_{\xi_{k-1}^i}(x(k-1))$, while using Bayes formula one finally obtains

$$p(x(k)|Z^-) = \sum_{i=1}^N w_{k-1}^i \delta_{\xi_{k^-}^i}(x(k)) \text{ with } \xi_{k^-}^i \sim p(x(k)|x(k-1) = \xi_{k-1}^i) \qquad (7.7)$$

As already analyzed, the meaning of Eq. (7.7) is that the state equation of the nonlinear system (e.g. described by Eq. (1.1) or Eq. (1.11) is executed N times, starting from the N previous values of the state vectors $x(k-1) = \xi_{k-1}^i$. Consequently, the mean value of the state vector is given from Eq. (7.7).

2. The *correction stage* (measurement update): A new position measurement $z(k)$ is obtained and the objective is to calculate the corrected probability density $p(x(k)|Z)$,

where $Z = z(1), z(2), ..., z(k)$. As discussed in Section 6.6 and from Bayes law one gets [50],[95],[284],[303].

$$p(x(k)|Z) = \sum_{i=1}^{N} w_k^i \delta_{\xi_{k^-}^i}(x(k))$$
$$\text{where } w_k^i = \frac{w_{k^-}^i p(z(k)|x(k)=\xi_{k^-}^i)}{\sum_{j=1}^{N} w_{k^-}^j p(z(k)|x(k)=\xi_{k^-}^j)} \tag{7.8}$$

3. *Resampling* for substitution of the degenerated particles is performed to keep the particles which are closer to the optimal estimation of the robot's state vector (see Fig. 7.2).

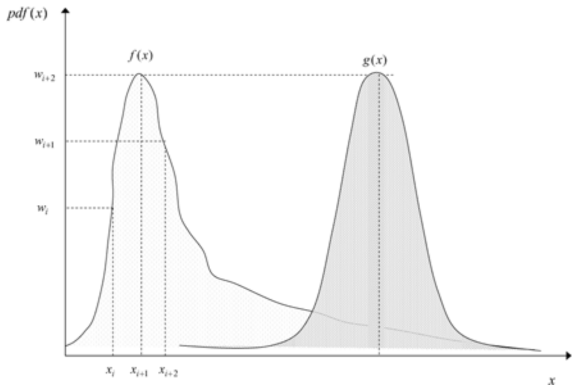


Fig. 7.2 Particle filtering-based state estimation: convergence from an initial p.d.f. $g(x)$ to the desirable final p.d.f. $f(x)$ through resampling

7.1.3 Simulation of EKF and PF-Based Sensor Fusion for Industrial Robot Control

Simulation experiments based on the industrial robotic manipulator model of Chapter 1 enable to study: (i) state estimation through sensor fusion with the EKF algorithm, and (ii) state estimation through sensor fusion with the PF algorithm, in terms of accuracy and computational effort. The parameters of the robotic manipulator described in Eq.(1.2) to Eq. (1.4), were given the values $m_1 = 10$, $m_2 = 7$, and $m_3 = 5$, while the associated lengths of the robot's links were $l_1 = 1.5$, $l_2 = 1.2$ and $l_3 = 1.0$ m. The measurement units of the monitored parameters of the robotic model were: (i) *rad* for motors' angles and joints angles, (ii) m/sec^2 for cartesian accelerations of the robot's end effector.

EKF for fusing the data that come from the accelerometer and the motor encoders is expected to be efficient under Gaussian distribution of the measurements' noise, while under non-Gaussian noise the performance of EKF cannot be assured. On the other hand, PF is a nonparametric state estimator and is expected be unaffected by the distribution of the measurements' noise. In most cases the combination of measurements from different sensors improves the accuracy of state estimation.

In state estimation of the 3-DOF robot with the use of the EKF and the PF algorithm, Gaussian measurement noise was first assumed. To generate the Gaussian noise Matlab's function *randn()* was used, while to generate Rayleigh noise function *raylrnd()* was used. In the presented results, the variance of the measurement noise considered for the cartesian accelerations was 0.1, while the variance of the measurement noise of the motor angles' was 0.05. Usually the process and measurement noises are taken to be uncorrelated. The process noise covariance matrix was $E\{w(i)w^T(j)\} = Q\delta(i-j) = 0$ while the covariance matrix of the measurement noise was taken to be diagonal $E\{v(i)v^T j\} = R\delta(i-j)$. In the simulation experiments that follow no process noise was considered, thus the process noise covariance matrix was taken to be $Q = 0_{6\times 6}$. The covariance matrix of the output measurement noise was taken to be $R = diag[r_{ii}]$, with $r_{ii} = 10^{-4}$ $i = 1, \cdots, 6$. The estimation error covariance matrix $P \in R^{6\times 6}$ and the EKF gain $K \in R^{6\times 6}$ were initialized and used in Eq. (7.3) and (7.4), while the Jacobian J_γ was calculated from Eq. (7.6). The estimated state vector was used in the control law of Eq. (1.8) to make the robot joints reach the desirable position and velocity set-points.

The performance of the EKF of the PF algorithm was also tested in the case of non-Gaussian measurement noise. In the latter case, the Rayleigh noise p.d.f. was assumed for the measurements received from the encoders and the accelerometer. Rayleigh noise can be induced to measurements transmitted over a network and in that case the effectiveness of the well-established Gaussian state-estimators is questionable. Results on networked control of dynamical systems under Rayleigh noise is given in [66],[234],[282].

The simulation experiments for the PF algorithm were carried out assuming a sufficiently large number of particles equal to $N = 2500$. Again the dynamic model of Eq. (1.1), and the measurement relations of Eq. (7.1) and Eq. (7.2) were used. As described in subsection 7.1.2 the estimation of the state vector of the robot was based on measurement and time update. The measurement vector $z(k)$ at time instant k is given by $z(k) = [\theta_1^m, \theta_2^m, \theta_3^m, \ddot{p}_x, \ddot{p}_y, \ddot{p}_z]$. The measurement and the time update of the particle filter were given by Eq. (7.7) and Eq. (7.8), respectively.

At each run of the time update of the PF, the state vector estimation $\hat{x}^-(k+1)$ is calculated N times, starting each time from a different value of the state vector ξ_k^i. The mean value of the state vector can thus be found. The estimated state vector is used in the measurement equations Eq. (7.1) and Eq. (7.2) and the outcome is compared to the measurements obtained from the sensors. The recorded difference (residual) is used to correct the weights associated with the N particles. It should be noted that the particle filter can utilize the non-linear model directly, whereas the EKF must use a linearized version. There is also no need to use a Gaussian approximation of the measurements probability density function.

In Fig. 7.3 a constant reference trajectory was considered and the performance of the control loop based on the state vector that was estimated either with the use of the EKF or the PF is presented. In Fig. 7.3(a) the convergence of the angle of the first joint of the robot to the desirable setpoint is demonstrated, when the measurements obtained by the robot's sensors (motor encoders and accelerometers) are subject to Gaussian noise. In Fig. 7.3(b) the simulation results about convergence of the angle of the first joint of the robot to the reference trajectory are obtained considering that the sensor measurements were affected by Rayleigh noise.

In Fig. 7.4 a constant reference trajectory was considered and simulation results about the convergence of the angle of the second robot joint to the aforementioned setpoint were presented. The control signal was generated again using the state vector that was estimated with the use of the EKF or the PF algorithm. In Fig. 7.4(a) the measurements obtained by the robot's sensors (motor encoders and accelerometers)

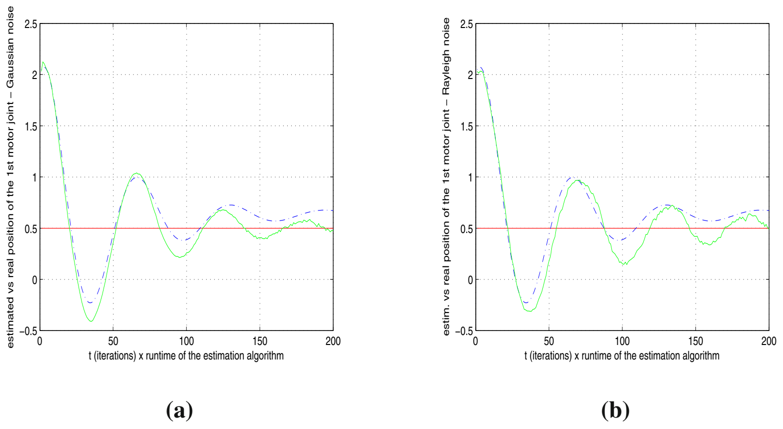


Fig. 7.3 Desirable trajectory (constant) for the first joint of the robot, trajectory succeeded when the state vector was estimated with the use of EKF (dashed line) and trajectory succeeded when the state vector was estimated with the use of PF (continuous line): (a) under Gaussian measurement noise, (b) under Rayleigh measurement noise

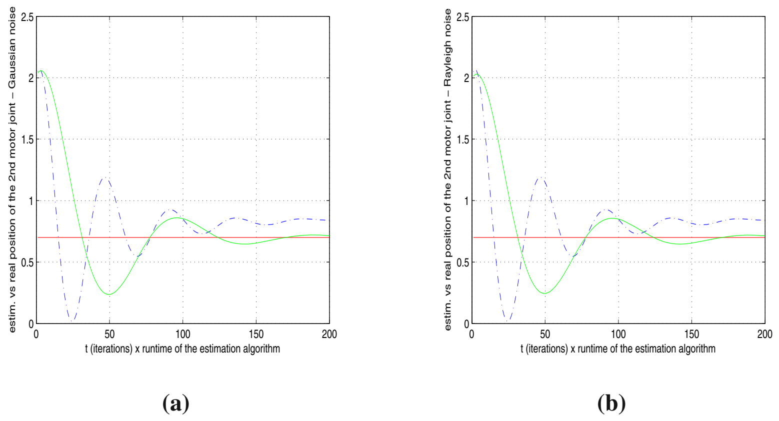
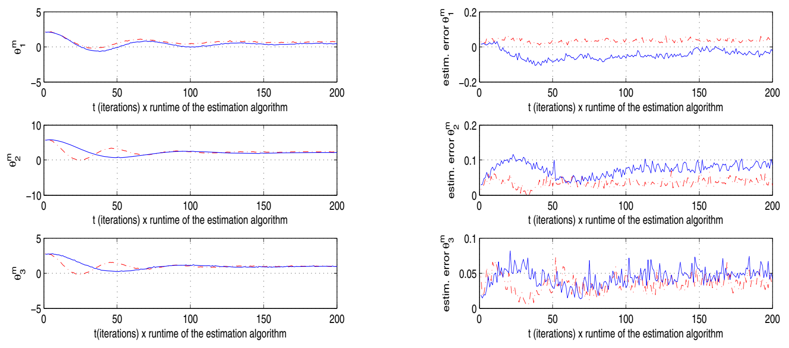
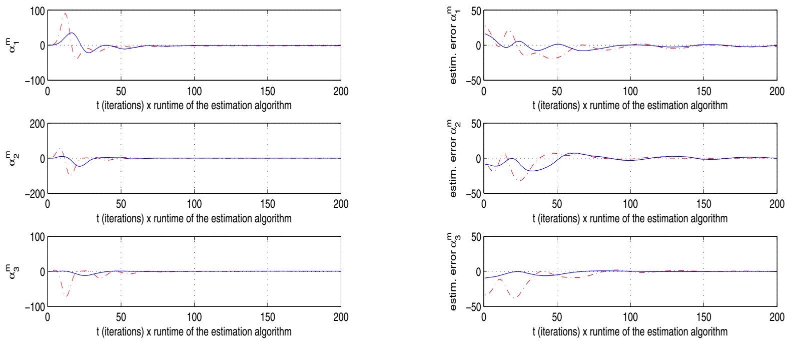


Fig. 7.4 Desirable trajectory (constant) for the second joint of the robot, trajectory succeeded when the state vector was estimated with the use of EKF (dashed line) and trajectory succeeded when the state vector was estimated with the use of PF (continuous line): (a) under Gaussian measurement noise, (b) under Rayleigh measurement noise



(a) **(b)**

Fig. 7.5 (a) Real values of the motor angles of the robot when tracking a constant setpoint, (b) estimation error of the robot's motor angles, when the state vector was estimated with the use of EKF (dashed line) and with the use of PF (continuous line)



(a) **(b)**

Fig. 7.6 (a) Real values of the cartesian accelerations of the robot's end effector when tracking a constant setpoint, (b) estimation error of the robot's accelerations, when the state vector was estimated with the use of EKF (dashed line) and with the use of PF (continuous line)

are subject to Gaussian noise, while in Fig. 7.4(b) the simulation results are obtained considering that the sensor measurements were affected by Rayleigh noise.

In Fig. 7.5(a), three different plots are given, depicting the angles of the joints' motors as the control algorithm advances in time, and convergence to a constant reference trajectory is pursued. In Fig. 7.5(b), the estimation errors of the three joints' motors are depicted. Results on the estimation error of the EKF and the PF algorithm are contained in each subplot. In Fig. 7.6(a), three different plots are given again, depicting the variation of the end effectors cartesian acceleration as the control

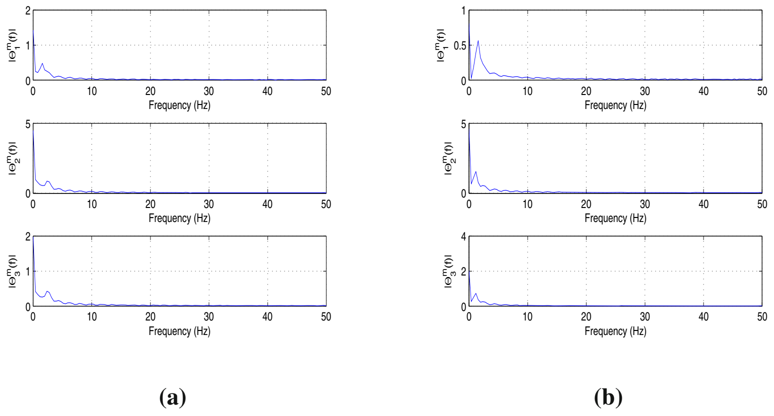


Fig. 7.7 (a) Frequency analysis of the signal that describes the motors' angles when tracking a constant setpoint and (a) the state vector of the robot was estimated with the use of EKF, (b) the state vector of the robot was estimated with the use of PF

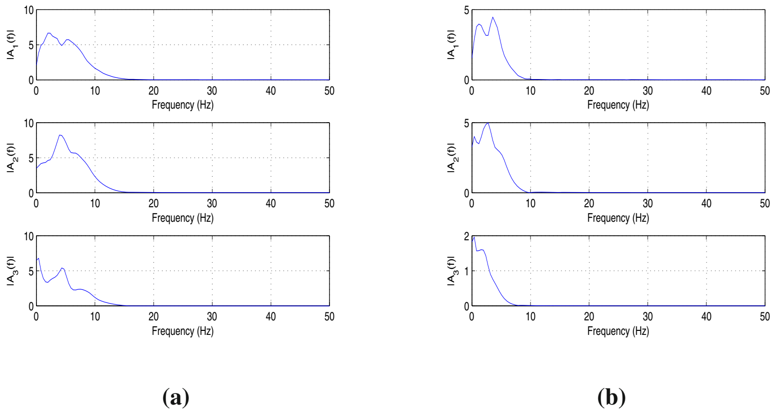
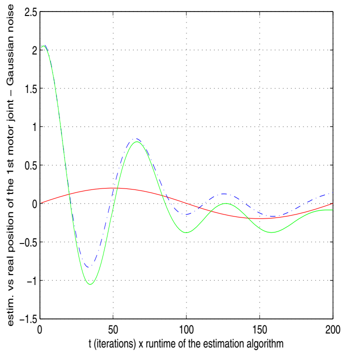
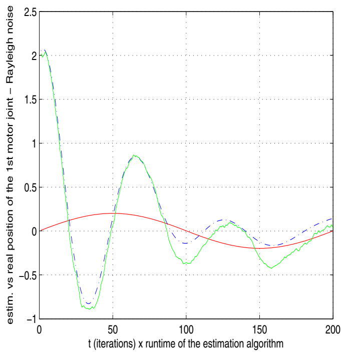


Fig. 7.8 Frequency analysis of the signal that describes the end-effector's accelerations when tracking a constant setpoint when (a) the state vector of the robot was estimated with the use of EKF, (b) the state vector of the robot was estimated with the use of PF

algorithm advances in time. In Fig. 7.6(b), the estimation errors of the end-effector's cartesian accelerations are depicted. Results on the estimation error of the EKF and the PF algorithm are again contained in each subplot.

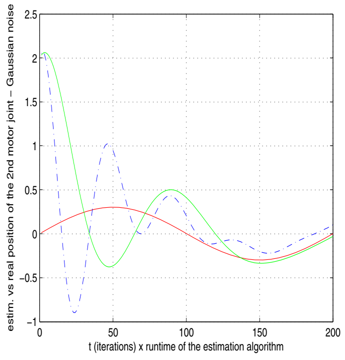
In Fig. 7.7 the frequency characteristics of the signals measured by the motor's encoders are depicted, when the feedback control pursues convergence to a constant setpoint. The frequency analysis of the three motor angles, e.g. based on the (Fast Fourier Transform) FFT algorithm, can provide information which is useful in fault diagnosis tasks. In Fig. 7.7(a) the frequency analysis of the estimated motor angles is based on the use of the state vector calculated by the EKF algorithm, while in

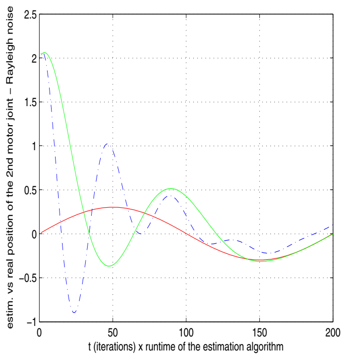




(a) **(b)**

Fig. 7.9 Desirable trajectory (sinusoidal) for the first joint of the robot, trajectory succeeded when the state vector was estimated with the use of EKF (dashed line) and trajectory succeeded when the state vector was estimated with the use of PF (continuous line): (a) under Gaussian measurement noise, (b) under Rayleigh measurement noise



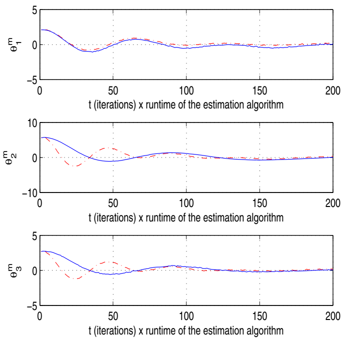


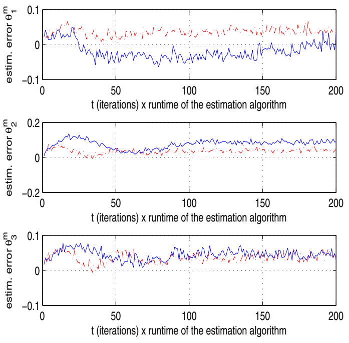
(a) **(b)**

Fig. 7.10 Desirable trajectory (sinusoidal) for the second joint of the robot, trajectory succeeded when the state vector was estimated with the use of EKF (dashed line) and trajectory succeeded when the state vector was estimated with the use of PF (continuous line): (a) under Gaussian measurement noise, (b) under Rayleigh measurement noise

Fig. 7.7(b) the estimated motor angles come from the state vector vector calculated by the PF algorithm. Frequency analysis can be also carried out for the signals measured by the cartesian accelerometer, which is mounted to the robot's end-effector, as shown in Fig. 7.8.

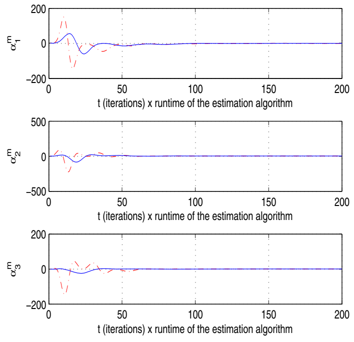
Next, tracking of a sinusoidal reference trajectory was considered. In Fig. 7.9 a sinusoidal reference trajectory was considered and the performance of the control loop based on the state vector that was estimated either with the use of the EKF or

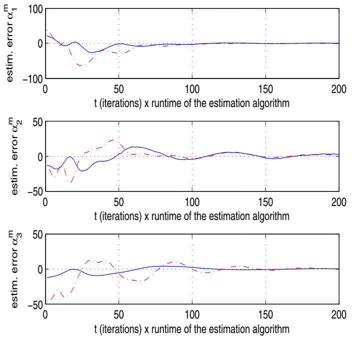




(a) **(b)**

Fig. 7.11 (a) Real values of the motor angles of the robot when tracking a sinusoidal setpoint, (b) estimation error of the robot's motor angles, when the state vector was estimated with the use of EKF (dashed line) and with the use of PF (continuous line)





(a) **(b)**

Fig. 7.12 (a) Real values of the cartesian accelerations of the robot's end effector when tracking a sinusoidal setpoint, (b) estimation error of the robot's accelerations, when the state vector was estimated with the use of EKF (dashed line) and with the use of PF (continuous line)

the PF is presented. In Fig. 7.9(a) the convergence of the angle of the first joint of the robot to the desirable trajectory is demonstrated, when the measurements obtained by the robot's sensors (motor encoders and accelerometers) are subject to Gaussian noise. In Fig. 7.9(b) the simulation results about convergence of the angle of the first joint of the robot to the reference trajectory are obtained considering that the sensor measurements were affected by Rayleigh noise.

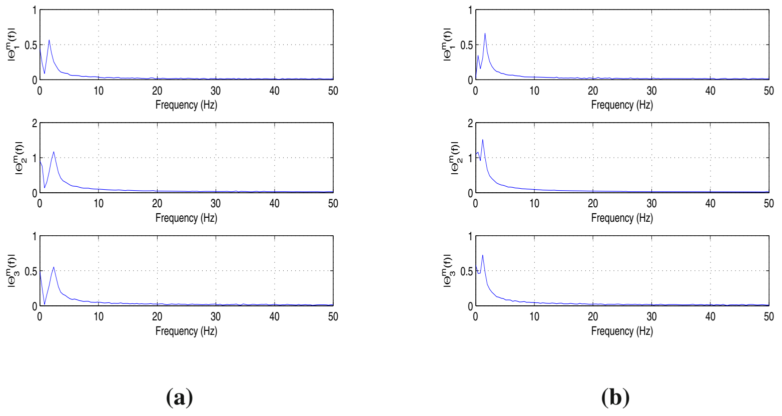


Fig. 7.13 Frequency analysis of the signal that describes the motors' angles when tracking a sinusoidal setpoint in case that (a) the state vector of the robot was estimated with the use of EKF, (b) the state vector of the robot was estimated with the use of PF

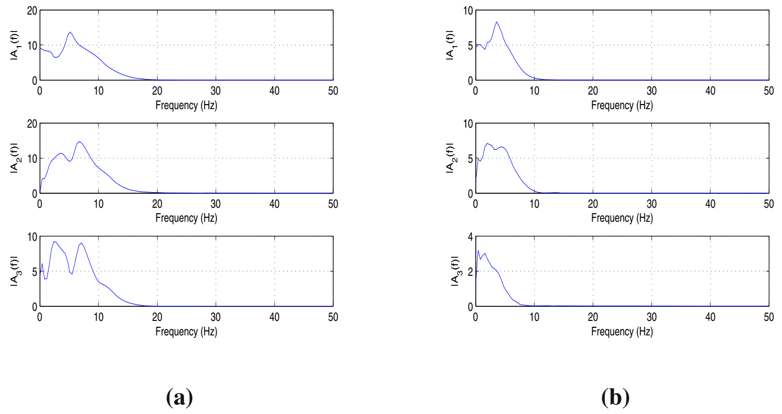


Fig. 7.14 Frequency analysis of the signal that describes the end-effector's accelerations when tracking a sinusoidal setpoint in case that (a) the state vector of the robot was estimated with the use of EKF, (b) the state vector of the robot was estimated with the use of PF

In Fig. 7.10 a sinusoidal reference trajectory was considered again and simulation results about the convergence of the angle of the second robot joint to the aforementioned setpoint were presented. The control signal was generated again using the state vector that was estimated with the use of the EKF or the PF algorithm. In Fig. 7.10(a) the measurements obtained by the robot's sensors (motor encoders and accelerometers) are subject to Gaussian noise. while in Fig. 7.10(b) the

Table 7.2 Variance of estimation using EKF and PF for $N = 2500$

parameter	θ_1	θ_2	θ_3	$\dot{\theta}_1$	$\dot{\theta}_2$	$\dot{\theta}_3$
variance EKF	0.0123	0.0120	0.0122	0.0455	0.0364	0.0455
variance PF	0.0029	0.0051	0.0071	0.0433	0.0346	0.0447

simulation results are obtained considering that the sensor measurements were affected by Rayleigh noise.

In Fig. 7.11(a), three different plots are given, depicting the angles of the joints' motors as the control algorithm advances in time, and the joints' angles convergence to sinusoidal reference trajectories. In Fig. 7.11(b), the estimation errors of the three joints' motors are depicted. Results on the estimation error of the EKF and the PF algorithm are contained in each subplot. In Fig. 7.12(a), three different plots are given again, depicting the variation of the end effectors cartesian acceleration as the control algorithm advances in time. In Fig. 7.12(b), the estimation errors of the end-effector's cartesian accelerations are depicted. Results on the estimation error of the EKF and the PF algorithm are again contained in each subplot.

In Fig. 7.13 the frequency characteristics of the signals measured by the motor's encoders are depicted, when the feedback control pursues convergence to a sinusoidal setpoint. In Fig. 7.13(a) the spectral analysis of the estimated motor angles is based on the use of the state vector calculated by the EKF algorithm, while in Fig. 7.13(b) the estimated motor angles come from the state vector vector calculated by the PF algorithm. As mentioned above, spectral analysis can be extended to the signals measured by the cartesian accelerometer which is mounted to the robot's end-effector, as shown in Fig. 7.14. Monitoring the changes of the spectrum of certain signals provided by the sensors of the industrial robotic manipulator is useful in fault diagnosis and preventive maintenance tasks. Alternatively, fault detection and isolation for the robotic manipulator can be performed through statistical processing of measurements in the time domain, as described in [16],[161].

From the simulation experiments it can be deduced that the particle filter performs better than the EKF and results in better estimations of the robot's state vector. Of course the number of the particles influences the performance of the algorithm. It can be observed that the accuracy of the estimation succeeded by the particle filter algorithm improves as the number of particles increases. The improved performance of the PF is summarized in Table 7.2, where the presented results were obtained considering equal noise levels for the EKF and PF simulation while the number of particles used in the PF simulation was $N = 2500$. Table 7.2, provides the variance of the estimation succeeded by the EKF and the PF for the parameters θ_i and $\dot{\theta}_i$, $i = 1,2,3$ of the robotic model.

Moreover, in Table 7.3, the maximum EKF and PF estimation error for the aforementioned parameters of the robotic manipulator, is presented. The improved performance of the PF algorithm can be noticed again.

In Table 7.4, an example is given on how the variance of the estimated parameter θ_1 is affected by the number of particles N. To test the performance of the particle filter with respect to the size of the particles set, the PF algorithm was run for

Table 7.3 Max estimation error using EKF and PF for $N = 2500$

parameter	θ_1	θ_2	θ_3	$\dot{\theta}_1$	$\dot{\theta}_2$	$\dot{\theta}_3$
max est. error PF	0.0912	0.0498	0.796	0.0803	0.0249	0.0467
max est. error EKF	0.1513	0.1498	0.1498	0.0843	0.0794	0.0839

Table 7.4 Particle number, simulation time and variance of $\hat{\theta}_1$

No Particles	250	300	400	500	750	1000
Simulation time (sec)	33.76	45.38	56.42	78.00	129.42	192.10
Variance	0.0761	0.0504	0.0220	0.0114	0.0070	0.0067

different numbers of particles ranging between 200 and 3000. The number of efficient particles used in the resampling procedure was set to $N^{\text{eff}} = 0.85{\cdot}N$. As expected, a larger number of particles resulted in smaller estimation error (variance), however this also caused an increase of the simulation time of the algorithm. The number of particles that gives the optimal trade-off between the estimation error and the run-time of the PF algorithm has to be chosen ad-hoc.

The simulation time of the particle filtering algorithm with respect to the number of particles, using the Matlab platform on a PC with 2GHz processor is depicted in Table 7.4. Optimization of code (for instance the functions used in the re-sampling of particle filtering) can improve to some extent the speed of the algorithm. It can be seen that the time needed for calculations is a critical parameter for the suitability of the PF algorithm for real-time applications. When it is necessary to use more particles, improved hardware, and parallel processing available to embedded systems, enable the PF to be implemented in real-time systems [41],[266],[458].

Remark 1: A comparison between the EKF and the PF-based sensor fusion is outlined as follows:

(i) The Extended Kalman Filter is based on a linearization of the systems' dynamics according to a first order Taylor expansion, and thus there is neither a proof of its convergence, nor a proof that the estimation succeeded by the EKF satisfies optimality criteria.
(ii) The Particle Filter is a nonparametric state estimator since it is not dependant on assumptions about the p.d.f. of the process and measurement noises and can function equally well for Gaussian and non-Gaussian noise distributions.
(iii) The Particle Filter is not based on any linearization of the system dynamics and can be very efficient in state estimation for nonlinear dynamical systems.
(iv) It is preferable to apply PF in sensor fusion for the industrial robot model since linearizations and Gaussian approximation may result in low performance of the EKF. The EKF algorithm needs the calculation of the Jacobians defined in Eq. (7.5) and (7.6). Moreover, the calculation of the Jacobian's derivatives $\frac{\partial J}{\partial \theta_i}$ in Eq. (7.5) requires extensive computations which for a many DOF robot, make the design of EKF a tedious task.

(v) Complicated calculations are not needed in the PF design. The prediction and correction stages consist of repetitive computations of the state estimates which are based on the nonlinear model of the robot. These are much simpler than the calculations of the Jacobians defined in Eq. (7.5) and (7.6).

Remark 2: The simulation tests have shown that the Particle Filter results in more accurate state estimations that the EKF provided that a sufficient number of particles is used. Moreover, PF avoids the calculations associated with the Jacobians that appear in the EKF equations. One could also consider other state estimators which are suitable for nonlinear systems and which may have improved performance and less complicated calculations comparing to EKF (no analytic derivation or Jacobians), such as Sigma-Point Kalman Filters (SPKF) and in particular the Unscented Kalman Filter (UKF) [177]. Comparing to PF the SPKF methods require less sample points to approximate the state distribution. However, the SPKF state estimators are still based on the assumption of a Gaussian process and measurement noise, while the PF is a nonparametric filter which can be applied to any type of state distribution.

7.2 Sensor Fusion-Based Control for Mobile Robots

7.2.1 Simulation of EKF-Based Control for Mobile Robots

Another example of state estimation-based control is the use of nonlinear filtering for fusing the data that come from different sensors of a mobile robot [51],[168],[228],[244]. The model of the unicycle robot, already described in Eq. (3.5), is considered again. Its continuous-time kinematic equation is:

$$\dot{x}(t) = v(t)cos(\theta(t)),\ \dot{y}(t) = v(t)sin(\theta(t)),\ \dot{\theta}(t) = \omega(t) \tag{7.9}$$

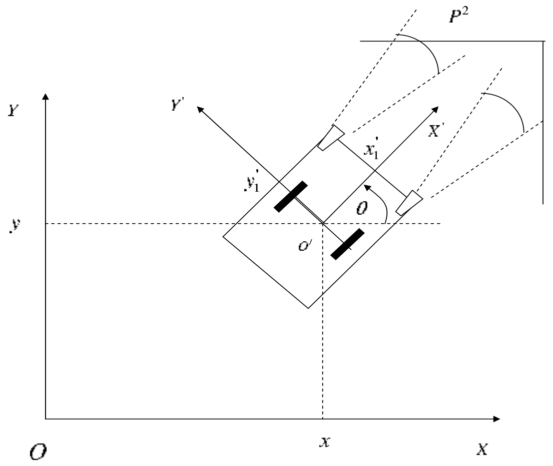


Fig. 7.15 Mobile robot with odometric and sonar sensors

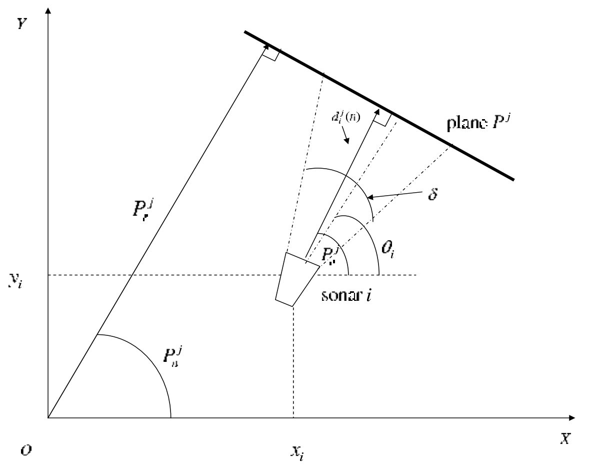


Fig. 7.16 Orientation of the sonar i

which is a simplified model of a car-like robot studied in [330]. Encoders placed on the driving wheels, provide a measure of the incremental angles over a sampling period T. These odometric sensors are used to obtain an estimation of the displacement and the angular velocity of the vehicle $v(t)$ and $\omega(t)$, respectively. These encoders introduce incremental errors, which result in an erroneous estimation of the vehicle's orientation θ. To improve the accuracy of the vehicle's localization, measurements from sonars can be used. The distance measure of sonar i from a neighboring surface P_j is thus taken into account (see Fig. 7.15 and Fig. 7.16). Sonar measurements may be affected by white Gaussian noise as well as by crosstalk interferences and multiples echoes.

The inertial coordinates system OXY is defined. Furthermore the coordinates system $O'X'Y'$ is considered (see Fig. 7.15). $O'X'Y'$ results from OXY if it is rotated by an angle θ (Fig. 7.15). The coordinates of the center of the wheels axis with respect to OXY are (x,y), while the coordinates of the sonar i that is mounted on the vehicle, with respect to $O'X'Y'$ are $x_i^{'}, y_i^{'}$. The orientation of the sonar with respect to $OX'Y'$ is $\theta_i^{'}$. Thus the coordinates of the sonar with respect to OXY are (x_i, y_i) and its orientation is θ_i, and are given by:

$$\begin{array}{c} x_i(k) = x(k) + x_i^{'} sin(\theta(k)) + y_i^{'} cos(\theta(k)) \\ y_i(k) = y(k) - x_i^{'} cos(\theta(k)) + y_i^{'} sin(\theta(k)) \\ \theta_i(k) = \theta(k) + \theta_i \end{array} \tag{7.10}$$

Each plane P^j in the robot's environment can be represented by P_r^j and P_n^j (Fig. 7.16), where (i) P_r^j is the normal distance of the plane from the origin O, (ii) P_n^j is the angle between the normal line to the plane and the x-direction.

The sonar i is at position $x_i(k), y_i(k)$ with respect to the inertial coordinates system OXY and its orientation is $\theta_i(k)$. Using the above notation, the distance of the sonar i, from the plane P^j is represented by P_r^j, P_n^j (see Fig. 7.16):

$$d_i^j(k) = P_r^j - x_i(k)cos(P_n^j) - y_i(k)sin(P_n^j) \tag{7.11}$$

where P_n^j ε $[\theta_i(n) - \delta/2, \theta_i(n) + \delta/2]$, and δ is the width of the sonar beam. Assuming a constant sampling period $\Delta t_k = T$ the measurement equation is $z(k+1) = \gamma(x(k)) + v(k)$, where $z(k)$ is the vector containing sonar and odometer measures and $v(k)$ is a white noise sequence $\sim N(0, R(kT))$. The dimension p_k of $z(k)$ depends on the number of sonar sensors. The measure vector $z(k)$ can be decomposed in two sub-vectors

$$\begin{aligned} z_1(k+1) &= [x(k) + v_1(k), y(k) + v_2(k), \theta(k) + v_3(k)] \\ z_2(k+1) &= [d_1^j(k) + v_4(k), \cdots, d_{n_s}^j(k) + v_{3+n_s}(k)] \end{aligned} \tag{7.12}$$

with $i = 1, 2, \cdots, n_s$, where n_s is the number of sonars, $d_i^j(k)$ is the distance measure with respect to the plane P^j provided by the i-th sonar and $j = 1, \cdots, n_p$ where n_p is the number of surfaces. By definition of the measurement vector one has that the output function $\gamma(x(k))$ is given by $\gamma(x(k)) = [x(k), y(k), \theta(k), d_1^1(k), d_2^2(k), \cdots, d_{n_s}^{n_p}]^T$. The robot state is $[x(k), y(k), \theta(k)]^T$ and the control input is denoted by $U(k) = [v(k), \omega(k)]^T$.

In the simulation tests, the number of sonar is taken to be $n_s = 1$, and the number of planes $n_p = 1$, thus the measurement vector becomes $\gamma(x(k)) = [x(k), y(k), \theta(k), d_1^1]^T$. To obtain the Extended Kalman Filter (EKF), the kinematic model of the vehicle is linearized about the estimates $\hat{x}(k)$ and $\hat{x}^-(k)$ the control input $U(k-1)$ is applied.

The *measurement update* of the EKF is

$$\begin{aligned} &K(k) = P^-(k)J_\gamma^T(\hat{x}^-(k))[J_\gamma(\hat{x}^-(k))P^-(k)J_\gamma^T(\hat{x}^-(k)) + R(k)]^{-1} \\ &\hat{x}(k) = \hat{x}^-(k) + K(k)[z(k) - \gamma(\hat{x}^-(k))] \\ &P(k) = P^-(k) - K(k)J_\gamma^T P^-(k) \end{aligned}$$

The *time update* of the EKF is

$$\begin{aligned} &P^-(k+1) = J_\phi(\hat{x}(k))P(k)J_\phi^T(\hat{x}(k)) + Q(k) \\ &\hat{x}^-(k+1) = \phi(\hat{x}(k)) + L(k)U(k) \end{aligned}$$

where

$$L(k) = \begin{pmatrix} Tcos(\theta(k)) & 0 \\ Tsin(\theta(k)) & 0 \\ 0 & T \end{pmatrix}$$

and

$$J_\phi(\hat{x}(k)) = \begin{pmatrix} 1 & 0 & -v(k)sin(\theta)T \\ 0 & 1 & -v(k)cos(\theta)T \\ 0 & 0 & 1 \end{pmatrix}$$

while $Q(k) = diag[\sigma^2(k), \sigma^2(k), \sigma^2(k)]$, with $\sigma^2(k)$ chosen to be 10^{-3} and $\phi(\hat{x}(k))$ $= [\hat{x}(k), \hat{y}(k), \hat{\theta}(k)]^T$, $\gamma(\hat{x}(k)) = [, \hat{x}(k), \hat{y}(k), \hat{\theta}(k), d(k)]^T$, i.e.

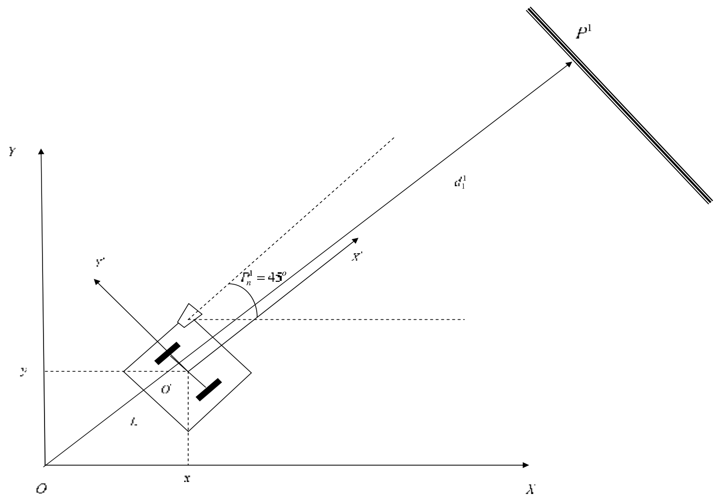


Fig. 7.17 Desirable trajectory of the autonomous vehicle i

$$\gamma(\hat{x}(k)) = \begin{pmatrix} \hat{x}(k) \\ \hat{y}(k) \\ \hat{\theta}(k) \\ P_r^j - x_i(k))cos(P_n^j) - y_i(k)sin(P_n^j) \end{pmatrix} \tag{7.13}$$

Assuming one sonar $n_s = 1$, and one plane P^1, $n_p = 1$ in the mobile robot's neighborhood one gets $J_\gamma^T(\hat{x}^-(k)) = [J_{\gamma_1}(\hat{x}^-(k)), J_{\gamma_2}(\hat{x}^-(k)), J_{\gamma_3}(\hat{x}^-(k)), J_{\gamma_4}(\hat{x}^-(k))]^T$, i.e.

$$J_\gamma^T(\hat{x}^-(k)) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ -cos(P_n^j) & -sin(P_n^j) & \{x_i' cos(\theta - P_n^j) - y_i' sin(\theta - P_n^j)\} \end{pmatrix} \tag{7.14}$$

The vehicle is steered by a dynamic feedback linearization control algorithm which is based on flatness-based control [295] and which was described in detail in Chapter 3:

$$\begin{aligned} u_1 &= \ddot{x}_d + K_{p_1}(x_d - x) + K_{d_1}(\dot{x}_d - \dot{x}) \\ u_2 &= \ddot{y}_d + K_{p_2}(y_d - y) + K_{d_2}(\dot{y}_d - \dot{y}) \\ \dot{\xi} &= u_1 cos(\theta) + u_2 sin(\theta) \\ v = \xi, & \ \ \omega = \frac{u_2 cos(\theta) - u_1 sin(\theta)}{\xi} \end{aligned} \tag{7.15}$$

Under the control law of Eq. (12.3) the dynamics of the tracking error finally becomes

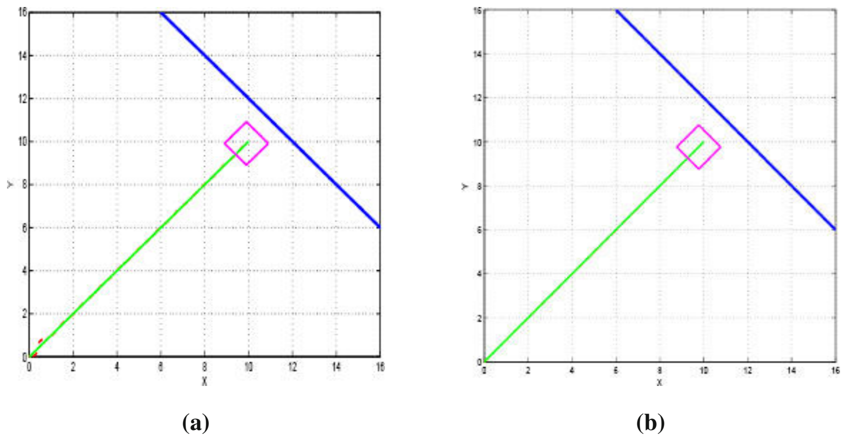


Fig. 7.18 (a) Desirable trajectory (continuous line) and obtained trajectory using EKF fusion based on odometric and sonar measurements (−.), (b) Desirable trajectory (continuous line) and obtained trajectory using PF fusion based on odometric and sonar measurements (−.)

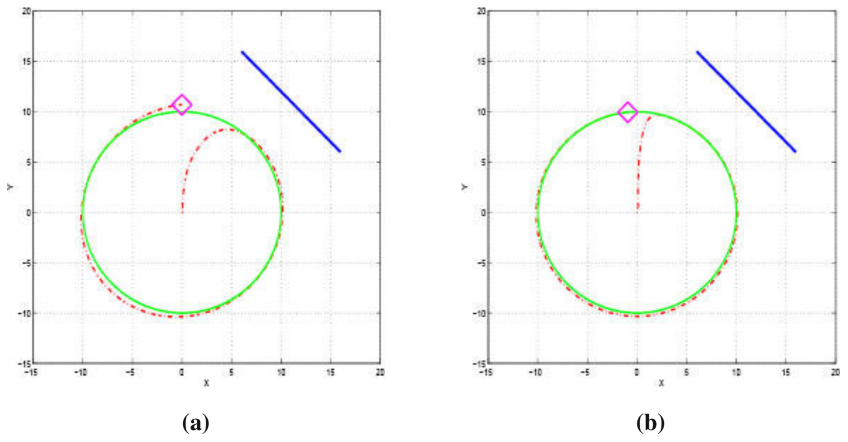


Fig. 7.19 (a) The trajectory of the mobile robot (dashed line) tracks the reference circular path (continuous line) when the robot's state vector is estimated with the use of Extended Kalman Filtering (b) The trajectory of the mobile robot (dashed line) tracks the reference circular path (continuous line) when the robot's state vector is estimated with the use of Particle Filtering.

$$\begin{aligned} \ddot{e}_x + K_{d_1}\dot{e}_x + K_{p_1}e_x = 0 \\ \ddot{e}_y + K_{d_2}\dot{e}_x + K_{p_2}e_y = 0 \end{aligned} \tag{7.16}$$

where $e_x = x - xd$ and $e_y = y - y_d$. The proportional-derivative (PD) gains are chosen as K_{p_1} and K_{d_1}, for $i = 1,2$. The dynamic compensator of Eq. (12.3) has a potential singularity at $\xi = v = 0$, i.e. when the vehicle is not moving. The

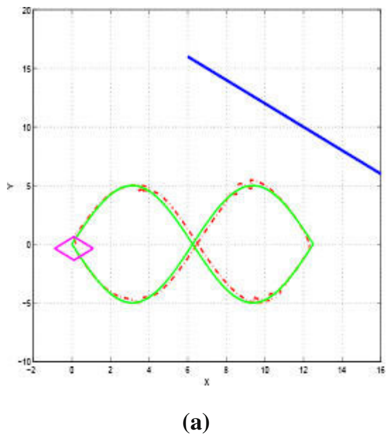


Fig. 7.20 (a) The trajectory of the mobile robot (dashed line) tracks the reference eight-shaped path (continuous line) when the robot's state vector is estimated with the use of Extended Kalman Filtering (b) The trajectory of the mobile robot (dashed line) tracks the reference eight-shaped path (continuous line) when the robot's state vector is estimated with the use of Particle Filtering.

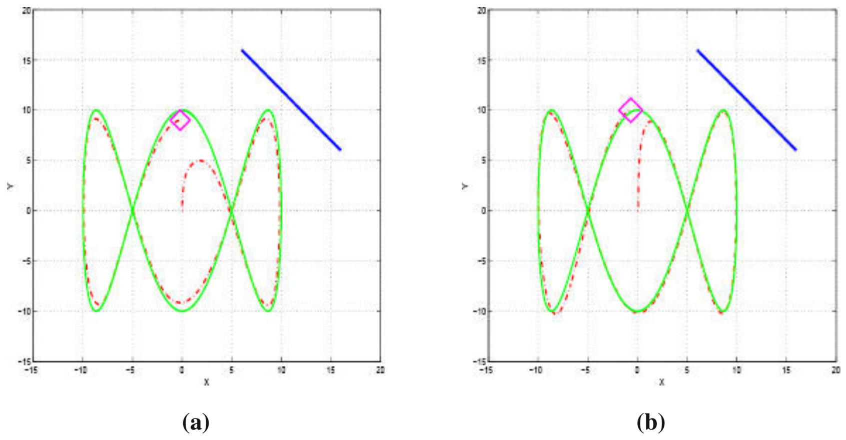


Fig. 7.21 (a) The trajectory of the mobile robot (dashed line) tracks the reference curved-shaped path (continuous line) when the robot's state vector is estimated with the use of Extended Kalman Filtering (b) The trajectory of the mobile robot (dashed line) tracks the reference curved-shaped path (continuous line) when the robot's state vector is estimated with the use of Particle Filtering.

occurrence of such a singularity is structural for non-holonomic systems. However, singularities are avoided under the assumption that the vehicle follows a smooth trajectory $(x_d(t), y_d(t))$ which is persistent, i.e. for which the nominal velocity

$v_d = (\dot{x}_d^2 + \dot{y}_d^2)^{1/2}$ along the trajectory never goes to zero (and thus singularities are avoided).

$$min_{t \geq 0} || \begin{pmatrix} \dot{x}_d(t) \\ \dot{y}_d(t) \end{pmatrix} || \geq \begin{pmatrix} \dot{\varepsilon}_x^0 \\ \dot{\varepsilon}_y^0 \end{pmatrix} \tag{7.17}$$

with $\dot{\varepsilon}_x^0 = \dot{\varepsilon}_x(0) \neq 0$ and $\dot{\varepsilon}_y^0 = \dot{\varepsilon}_y(0) \neq 0$ then the singularity $\xi = 0$ is never met.

In the simulation experiments appearing in Fig.7.18 to Fig.7.20 the following initialization is assumed (see Fig. 8.7):

- vehicle's initial position in OXY: $x(0) = 0m$, $y(0) = 0m$, $\theta(0) = 45.0^o$.
- position of the sonar in $O'X'Y'$: $x_1' = 0.5m$, $y_1' = 0.5m$, $\theta_1' = 0^o$.
- position of the plane P^1: $P_r^1 = 15.5m$, $P_n^1 = 45^o$.
- state noise $w(k) = 0$, $\hat{P}(0) = diag[0.1, 0.1, 0.1]$, and $R = diag[10^{-3}, 10^{-3}, 10^{-3}, 10^{-3}]$.
- Kalman Gain $K(k)\ \varepsilon\ R^{3 \times 4}$.

The use of the EKF for fusing the data that come from odometric and sonar sensors provides an estimation of the state vector $[x(t), y(t), \theta(t)]$ and enables the successful application of nonlinear steering control of Eq. (12.3). For the case of motion along a straight line on the 2D-plane, the obtained resutls are depicted in Fig. 7.18(a). Moreover, results on the tracking of a circular reference path are given in Fig. 7.19(a), while the case of tracking of an eight-shaped reference path is depicted in Fig. 7.20(a) Tracking experiments for EKF-based state estimation were completed in the case of a curved path as the one shown in Fig. 7.21(a).

7.2.2 Simulation of Particle Filter-Based Mobile Robot Control

The particle filter can also provide solution to the sensor fusion problem. The mobile robot model described in Eq. (12.2), and the control law given in Eq. (12.3) are used again. The number of particles was set to $N = 1000$.

The *measurement update* of the PF is

$$\begin{aligned} &p(x(k)|Z) = \textstyle\sum_{i=1}^N w_k^i \delta_{\xi_{k^-}^i}(x(k)) \\ &\text{with } w_k^i = \frac{w_{k^-}^i p(z(k)|x(k) = \xi_{k^-}^i)}{\sum_{j=1}^N w_k^j p(z(k)|x(k) = \xi_{k^-}^j)} \end{aligned} \tag{7.18}$$

where the measurement equation is given by $\hat{z}(k) = z(k) + v(k)$. The elements of the state vector are $z(k) = [x(k), y(k), \theta(k), d(k)]^T$, while $v(k)$ is the measurement noise.

The *time update* of the PF is

$$\begin{aligned} &p(x(k+1)|Z) = \textstyle\sum_{i=1}^N w_k^i \delta_{\xi_k^i}(x(k)) \\ &where\ \xi_k^i \sim p(x(k+1)|x(k) = \xi_{k^-}^i) \end{aligned} \tag{7.19}$$

and the state equation is $\hat{x}^- = \phi(x(k)) + L(k)U(k)$, where $\phi(x(k))$, $L(k)$, and $U(k)$ are defined in subsection 7.2. At each run of the time update of the PF, the state vector estimation $\hat{x}^-(k+1)$ is calculated N times, starting each time from a different value of the state vector ξ_k^i.

The measurement noise distribution was assumed to be Gaussian. As the number of particles increases, the performance of the particle filter-based tracking algorithm also improves, but this happens at the demand for computational resources. Control of the diversity of the particles through the tuning of the resampling procedure may also affect the performance of the algorithm. The obtained results are given in Fig. 7.18(b) for the case of motion along a straight line on the 2D plane, Additionally, results on the tracking of a circular reference path are given in Fig. 7.19(b), while the case of tracking of an eight-shaped reference path is depicted in Fig. 7.20(b). Tracking experiments for PF-based state estimation were completed in the case of a curved path as the one shown in Fig. 7.21(b).

7.2.3 Simulation of EKF and PF-Based Parallel Parking Control

Finally the efficiency of the proposed sensor fusion-based control scheme is presented in the case of a different application example, which is the automated parallel parking. The parallel parking task has attracted considerable attention in the area of mobile robotics since it has practical significance for the automotive industry [44],[127],[140],[438],[439]. A rectangular parking area is considered and a reference frame is defined so as its axes to coincide with the rectangle's side (see Fig. 7.22). The vehicle starts from an initial position out of the rectangle and moves backwards. After performing appropriate maneuvers the vehicle should be found completely inside the parking area with its longitudinal axis to be aligned with the horizontal axis of the reference system [329]. Collisions with the boundaries of the parking area are not allowed, while several other constraints about the geometry of the vehicle's trajectory and the vehicle's velocity can be imposed. To perform the parallel parking task usually three stages can be distinguished: (i) detection of a suitable parking area, (ii) path planning, i.e. selection of the trajectory to be followed by the vehicle's center of symmetry while moving towards the final position, (iii) control of the vehicle's motion. Parallel parking can be performed (a) in direct motion, i.e. the vehicle moves only backwards, along a single continuous path, towards its final position (b) in recursive motion, i.e. the vehicle moves backwards and forwards and performs sequential maneuvers along different paths with different velocity sign, before reaching its goal point [127],[329]. In Fig. 7.22 results on state estimation-based nonlinear control for direct-motion parallel parking are presented.

A robotic unicycle and a rectangular parking area are considered and a collision-free path that connects the initial to the final position of the vehicle is defined. The objective is to steer the vehicle along the reference path when the vehicle's state vector is not directly available but is estimated by fusing measurements coming from distributed sensors. Again, two different types of measurements are considered: (i) measurements of the distance covered by the robot which are provided by the robot's encoders and (ii) measurements of the distance from a reference surface (blue line in Fig. 7.22) which can be provided by a sonar. It can be observed that EKF or PF-based sensor fusion for estimating the state vector of the mobile loop enables tracking of

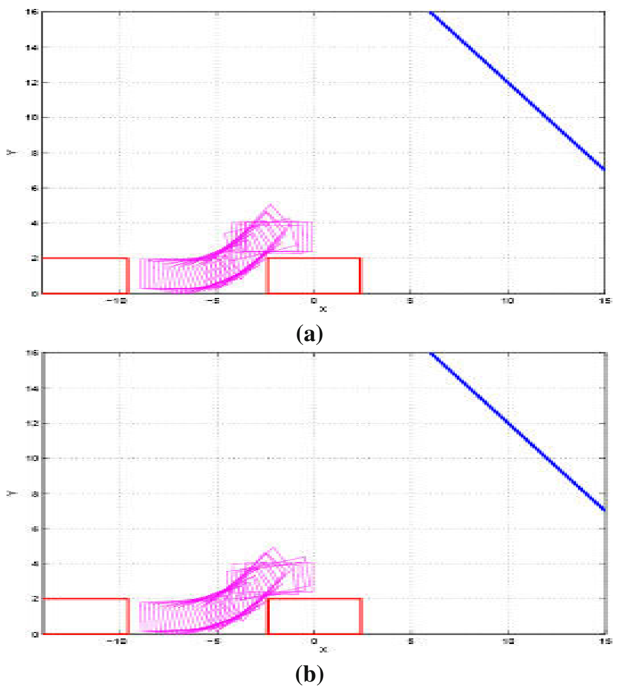


Fig. 7.22 Maneuvers performed by the autonomous vehicle for the completion of parallel parking when the vehicle's state vector is estimated using (a) Extended Kalman Filtering (b) Particle Filtering.

the reference path and convergence of the vehicle to its goal point inside the parking area. From this last simulation experiment it can be deduced again that for a sufficiently large number of particles, the particle filter can have good performance, in the problem of estimation of the state vector of the mobile robot, without being subject in the constraint of Gaussian distribution for the obtained measurements. The initialization of the particles, (state vector estimates) may also affect the convergence of the PF towards the real value of the state vector of the monitored system.

7.2.4 Performance Analysis of EKF and PF-Based Mobile Robot Control

Finally about the convergence and stability of the Extended Kalman and the Particle Filter-based control loops the following can be noted:

(i) For the closed-loop system which consists of the mobile robot, the nonlinear controller and the state-vector estimator (EKF or PF), a nonlinear stochastic control problem is formulated. The problem of stochastic control for linear dynamical systems affected by Gaussian noise, can be solved under certain assumptions following the Linear Quadratic Gaussian (LQG) methodology. The stability proof in the LQG approach results in the requirement for simultaneous solution of two Riccati equations: one Riccati equation is associated with the optimal controller that uses feedback of the estimated state vector, while the second Riccati equation is associated with the state estimator (Kalman Filter) which provides an approximation of the state vector using output measurements. Certain assumptions are made on the process and measurement noise (they are taken to be uncorrelated, of zero mean and with known covariance matrices). The stochastic control for nonlinear dynamical systems subject to non-Gaussian process/measurement noise is definitely a more difficult problem and the current chapter contributes to the study of such a control loop performance. The separation principle is also followed for the closed-loop of Eq. (12.3), and a controller is designed independently from the observer using the estimated vector in place of the real state vector [333],[341].

(ii) Demonstration of simultaneous convergence of the estimated state vector and of the real state vector to the desirable state vector (i.e. $\hat{x}{\rightarrow}x_d$ and $x{\rightarrow}x_d$) would complete a stability proof for the nonlinear stochastic control problem described above. Stability issues for EKF-based control of a class of systems have been discussed in [3], while the application of EKF to closed-loop control of robotic manipulators has been also studied in [39] and [164] . Convergence proof for the particle filter when the PF is used as observer in a stochastic control loop is an open research problem. In [75] necessary conditions have been stated for the convergence of the particle filtering algorithm, however these concern the case in which the PF is not included in a control loop. In case that the PF becomes part of the control loop it is anticipated that a particle filter with a sufficiently large number of particles, bounded weights, and a standard resampling scheme can provide an accurate estimation of the mobile robot's state vector thus enabling the controller also to converge to the desirable setpoint.

From the simulation experiments it can be noticed that the Particle Filter has improved performance comparing to EKF and that it results in better estimates of the vehicle's state vector, even when the process and measurement noises are taken to be Gaussian. Table 7.5 presents results on the variance of the state vector estimates, when considering equal noise levels for the EKF and the PF simulation, and assuming that the number of particles used by the EKF was $N = 1200$.

Table 7.5 Variance using EKF and PF for N=1200

state variable	x	y	θ
EKF	0.086	0.084	$16.59{\cdot}10^{-3}$
PF	0.061	0.079	$1.40{\cdot}10^{-3}$

The cycle time (runtime) of the Particle Filter with respect to the number of particles, using the Matlab platform on a PC with a 2GHz Intel Core Duo processor, is depicted in Table 7.6. Optimization of code of the resampling procedure can improve to some extent the speed of the algorithm. As mentioned, when it is necessary to use more particles, improved hardware and parallel processing available to embedded systems enable real-time implementation of the PF algorithm [41],[266],[458].

Table 7.6 Particle number, simulation time and variance of $\hat{\theta}$

No Particles	800	900	1000	1100	1200	1300
PF cycle time (sec)	0.219	0.264	0.317	0.371	0.432	0.490
Variance	0.073	0.068	0.066	0.066	0.061	0.059

When sorting of particles is not performed in the resampling procedure the run-time of Particle Filtering increases linearly with respect to the number of particles [370].

7.3 Sensor Fusion-Based Dynamic Ship Positioning

7.3.1 EKF and PF-Based Sensor Fusion for the Ship Model

7.3.1.1 Sensor Fusion Using Extended Kalman Filtering

Next, it will be shown how the state vector of a ship can be estimated with the use of the Extended Kalman and the Particle Filter. It will also be explained how the estimated state vector can be used in a nonlinear control loop for performing dynamic positioning of the vessel.

The ship's state space model described in Eq. (3.55) and in Eq. (3.56) is suitable form applying the EKF [102],[104],[124],[335],[342],[362],[405]:

$$\begin{aligned} x(k+1) &= \phi(x(k)) + B(k)u(k) + w(k) \\ z(k) &= \gamma(x(k)) + v(k), \end{aligned} \tag{7.20}$$

where $x \in R^{m\times 1}$ is the system's state vector and $z \in R^{p\times 1}$ is the system's output, while $w(k)$ and $v(k)$ are uncorrelated, Gaussian zero-mean noise processes with covariance matrices $Q(k)$ and $R(k)$ respectively.

Again the application of EKF requires a linearization procedure around the current estimate of the state vector $\hat{x}(k)$, which results to the linearized version of the system: $x(k+1) = \phi(\hat{x}(k)) + J_\phi(\hat{x}(k))[x(k) - \hat{x}(k)] + w(k)$, $z(k) = \gamma(\hat{x}^-(k)) + J_\gamma(\hat{x}^-(k))[x(k) - \hat{x}^-(k)] + v(k)$, where $J_\phi(x)$ and $J_\gamma(x)$ are the Jacobians of ϕ and γ respectively, calculated at $\hat{x}(k)$. As already analyzed, the EKF recursion is as follows:

Measurement update. Acquire $z(k)$ and compute:

$$\begin{aligned} &K(k) = P^-(k)J_\gamma^T(\hat{x}^-(k))\cdot[J_\gamma(\hat{x}^-(k))P^-(k)J_\gamma^T(\hat{x}^-(k)) + R(k)]^{-1} \\ &\hat{x}(k) = \hat{x}^-(k) + K(k)[z(k) - \gamma(\hat{x}^-(k))] \\ &P(k) = P^-(k) - K(k)J_\gamma(\hat{x}^-(k))P^-(k) \end{aligned} \tag{7.21}$$

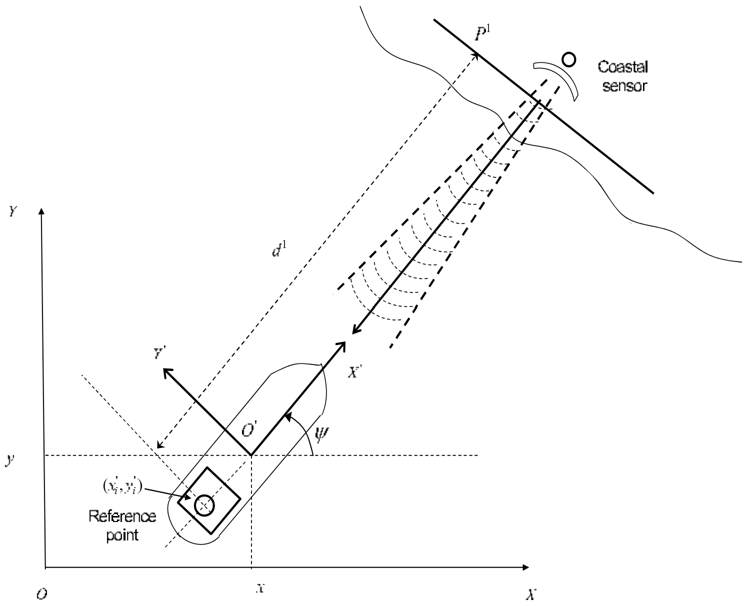


Fig. 7.23 Estimation of the ship's state vector by fusing the measurements of its position and orientation with the measurement of its distance from the coast provided by a coastal sensor (e.g. radar)

Time update. Compute:

$$\begin{aligned} P^-(k+1) &= J_\phi(\hat{x}(k))P(k)J_\phi^T(\hat{x}(k)) + Q(k) \\ \hat{x}^-(k+1) &= \phi(\hat{x}(k)) + B(k)u(k) \end{aligned} \tag{7.22}$$

It is noted that for the previously analyzed ship model (see Chapter 3) it holds $\phi(x) = Ax$ and $J_\phi(\hat{x}) = A$ therefore the time update part of the state estimation algorithm is the same with the time update of the standard Kalman Filter.

The coordinates of the center of symmetry of the ship with respect to OXY (inertial coordinates system) are (x, y), while the coordinates of a reference point i of the ship (e.g. bridge), with respect to $O'X'Y'$ (body-fixed coordinates system) are x_i', y_i' (Fig. 7.23). The orientation of the ship's reference point with respect to $O'X'Y'$ is ψ_i'. Thus the coordinates of the reference point i with respect to OXY are (x_i, y_i) and its orientation is ψ_i, and are given by $x_i(k) = x(k) + x_i' sin(\psi(k)) + y_i' cos(\psi(k))$, $y_i(k) = y(k) - x_i' cos(\psi(k)) + y_i' sin(\psi(k))$ and $\psi_i(k) = \psi(k) + \psi_i$. Each reference plane P^j on the coast can be represented by P_r^j and P_n^j (Fig. 7.23), where (i) P_r^j is the normal distance of the plane from the origin O, (ii) P_n^j is the angle between the normal line to the plane and the x-direction. Using the above notation, the distance of the ship's reference point i (e.g. bridge), from the reference plane P^j on the coast depends on P_r^j, P_n^j (see Fig. 7.23): $d^1(k) = P_r^j - x_i(k)cos(P_n^j) - y_i(k)sin(P_n^j)$ [342].

By definition of the measurement vector one has that the output function $\gamma(x(k))$ is given by $\gamma(x(k)) = [x(k), y(k), \psi(k), d^1(k)]^T$. To obtain the Extended Kalman Filter (EKF), the model of the ship is linearized about the estimates $\hat{x}(k)$ and $\hat{x}^-(k)$ as described in the previous subsection. The process noise covariance matrix $Q(k)\in R^{12\times 12}$ and the measurement noise matrix $R\in R^{4\times 4}$ are taken to be diagonal. The Kalman Filter gain is $K\in R^{12\times 4}$. For matrix γ appearing in the ship's output equation it holds $\gamma(\hat{x}(k)) = [\hat{x}(k), \hat{y}(k), \hat{\psi}(k), P_r^j - x_i(k))cos(P_n^j) - y_i(k)sin(P_n^j)]^T$. The Jacobian of the ship model's output γ with respect to the state vector $x(k)$ is thus,

$$J_\gamma^T(\hat{x}^-(k)) = \begin{pmatrix} 1 & 0 & 0 & 0_{1\times 9} \\ 0 & 1 & 0 & 0_{1\times 9} \\ 0 & 0 & 1 & 0_{1\times 9} \\ \alpha_{41} & \alpha_{42} & \alpha_{43} & 0_{1\times 9} \end{pmatrix} \tag{7.23}$$

where $\alpha_{41} = -cos(P_n^j)$, $\alpha_{42} = -sin(P_n^j)$ and $\alpha_{43} = \{x_i^{'}cos(\psi - P_n^j) - y_i^{'}sin(\psi - P_n^j)\}$.

As analyzed in Chapter 3, the ship can be steered along the reference trajectory using the estimated state vector and control based on feedback linearization of the ship's dynamic model. Alternatively nonlinear backstepping control or robust control can be used [32],[182].

7.3.1.2 Sensor Fusion Using Particle Filtering

As in the EKF case, the Particles Filter for the ship dynamic positioning problem consists also of the measurement update (correction stage) and the time update (prediction stage) [226],[409],[472]. The prediction stage calculates $p(x(k)|Z^-)$ where $Z^- = \{z(1), z(2), , z(k-1)\}$ are output measurements up to time instant $k-1$. As in has been shown in Section 6.6 it holds that $p(x(k-1)|Z^-) = \sum_{i=1}^N w_{k-1}^i \delta_{\xi_{k-1}^i}(x(k-1))$, while from Bayes formula it holds $p(x(k)|Z^-) = \int p(x(k)|x(k-1))p(x(k-1)|Z^-)dx$. From the above one finally obtains: $p(x(k)|Z^-) = \sum_{i=1}^N w_{k-1}^i \delta_{\xi_{k^-}^i}(x(k))$, with $\xi_{k^-}^i \sim p(x(k)|x(k-1) = \xi_{k-1}^i)$. The previous relation means that the state equation of the system is executed N times, starting from the N previous values of the state vectors $x(k-1) = \xi_{k-1}^i$. Consequently, the value of the state vector which is calculated in the prediction stage is the result of the weighted averaging of the state vectors of the ship which were computed after running the state equation, starting from the N previous values of the state vectors ξ_{k-1}^i.

Again, following the analysis of Section 6.6, the a-posteriori probability density is found as follows: a new position measurement $z(k)$ is obtained and the corrected probability density $p(x(k)|Z)$ is calculated, where $Z = \{z(1), z(2), \cdots, z(k)\}$. From Bayes law it holds that $p(x(k)|Z) = p(Z|x(k))p(x(k))/p(Z)$ which can be also written as

$$p(x(k)|Z) = p(z(k)|x(k))p(x(k)|Z^-)/\int p(z(k)|x(k), Z^-)p(x(k)|Z^-)dx$$

After intermediate calculations one finally obtains $p(x(k)|Z) = \sum_{i=1}^{N} w_k^i \delta_{\xi_{k^-}^i}(x(k))$, where it holds $w_k^i = w_{k^-}^i p(z(k)|x(k) = \xi_{k^-}^i)/\sum_{j=1}^{N} w_{k^-}^j p(z(k)|x(k) = \xi_{k^-}^j)$. The previous equation denotes the corrected value for the state vector. The recursion of the PF proceeds in a way similar to the update of the EKF, i.e.:

Measurement update: Acquire $z(k)$ and compute the new value of the state vector

$$\begin{array}{l} p(x(k)|Z) = \sum_{i=1}^{N} w_k^i \delta_{\xi_{k^-}^i}(x(k)) \\ \text{with corrected weights } w_k^i = \frac{w_{k^-}^i p(z(k)|x(k)=\xi_{k^-}^i)}{\sum_{j=1}^{N} w_{k^-}^i p(z(k)|x(k)=\xi_{k^-})^i} \\ \text{and } \xi_k^i = \xi_{k^-}^i \end{array} \tag{7.24}$$

Resampling: Substitute the degenerated particles. The particles of low weight factors are removed and their place is occupied by duplicates of the particles with high weight factors.

Time update: compute state vector $x(k+1)$ according to the pdf

$$\begin{array}{l} p(x(k+1)|Z) = \sum_{i=1}^{N} w_k^i \delta_{\xi_k^i}(x(k)) \\ \text{where } \xi_k^i \sim p(x(k+1)|x(k) = \xi_k^i). \end{array} \tag{7.25}$$

Knowing the measured value of the ship's position and orientation $[x, y, \psi]$, one can assign a weight to each particle (estimate of the state vector $[\hat{x}, \hat{y}, \hat{\psi}]_i$), according to how closely the particle approaches the measured state vector. Similarly knowing the distance d^1 from the coastal reference surface, and calculating an estimation of this distance $\hat{d}^1$ for every particle $[\hat{x}, \hat{y}, \hat{\psi}]_i$, one can assign a weight to the particle according to the accuracy of estimation of the distance d^1. Further averaging of these two weight values associated with each particle provides the aggregate particle's weight which is used in the Particle Filter's iteration.

7.3.2 Simulation of EKF and PF-Based Ship Dynamic Positioning

The number of particles used by the PF was $N = 1000$. The use of Extended Kalman and Particle Filtering for fusing the data that come from the ship's navigation instruments with the measurements that come from coastal sensors provides an estimation of the state vector $[x(t), y(t), \theta(t)]$ and enables the successful application of nonlinear steering control, as shown in Fig 7.24 to Fig. 7.28 and in Fig 7.29 to Fig. 7.33, respectively. Moreover, from the simulation experiments it can be observed that the Extended Kalman Filter and the Particle Filter provide accurate estimations of the external disturbances. Thus, an auxiliary control term based on the disturbances estimation can be included in the right hand side of Eq. (3.51), and can compensate for the disturbances' effects.

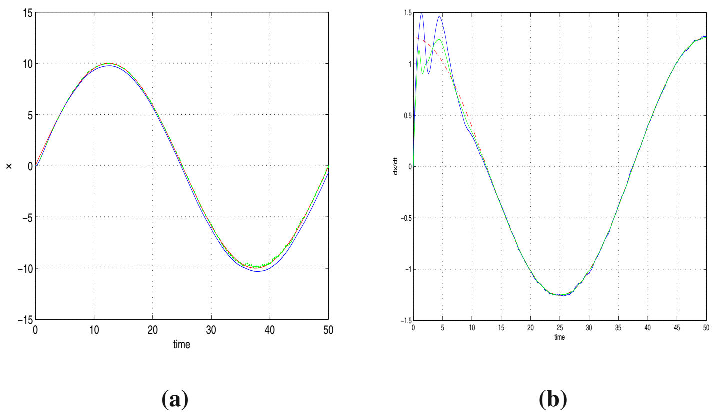


Fig. 7.24 (a) EKF-based estimation of the ship's position along the *x*-axis (continuous line) and desirable *x*-axis position (dashed line), (b) EKF-based estimation of the ship's velocity along the *x*-axis (continuous line) and desirable *x*-axis velocity).

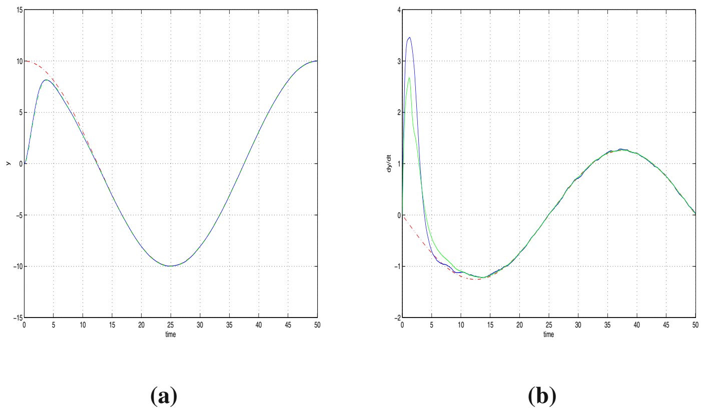


Fig. 7.25 (a) EKF-based estimation of the ship's position along the *y*-axis (continuous line) and desirable *y*-axis position (dashed line), (b) EKF-based estimation of the ship's velocity along the *y*-axis (continuous line) and desirable *y*-axis velocity).

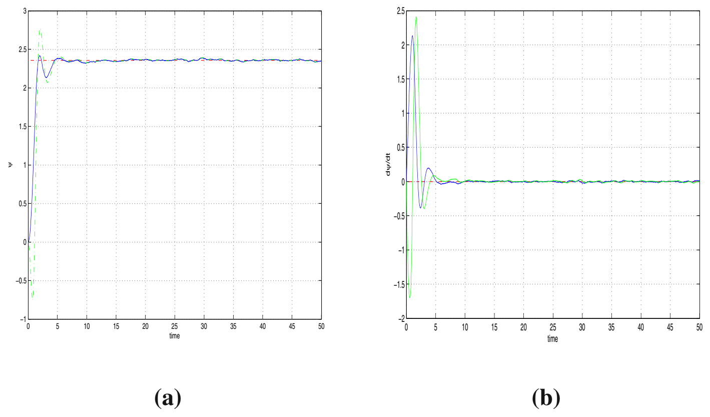


Fig. 7.26 (a) EKF-based estimation of the ship's angle round the z-axis (continuous line) and desirable z-axis rotation angle (dashed line), (b) EKF-based estimation of the ship's angular velocity round the z-axis (continuous line) and desirable angular velocity).

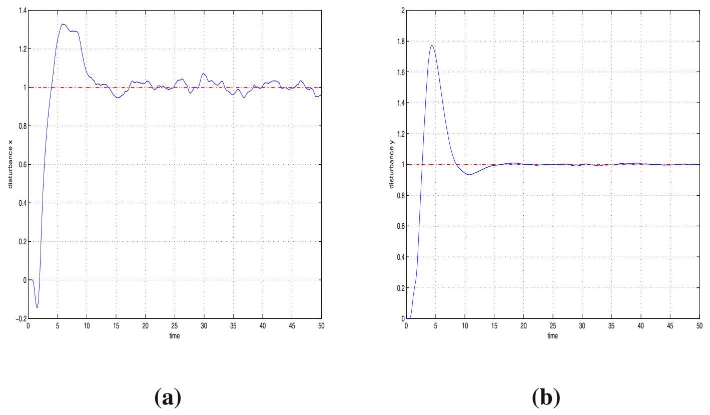
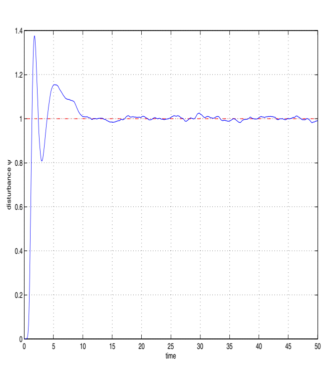
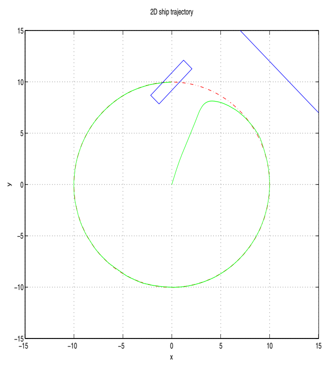


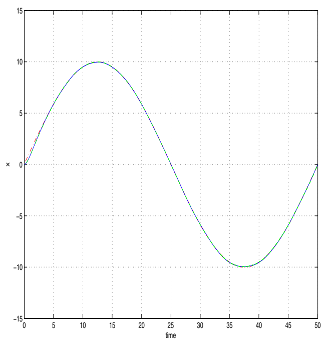
Fig. 7.27 (a) EKF-based estimation of the disturbance along the x-axis (continuous line) and real value of the x-axis disturbance (dashed line), (b) EKF-based estimation of the disturbance along the y-axis (continuous line) and real value of the y-axis disturbance (dashed line).

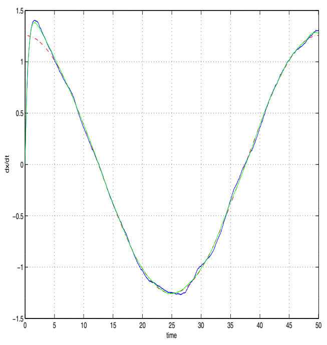




(a) **(b)**

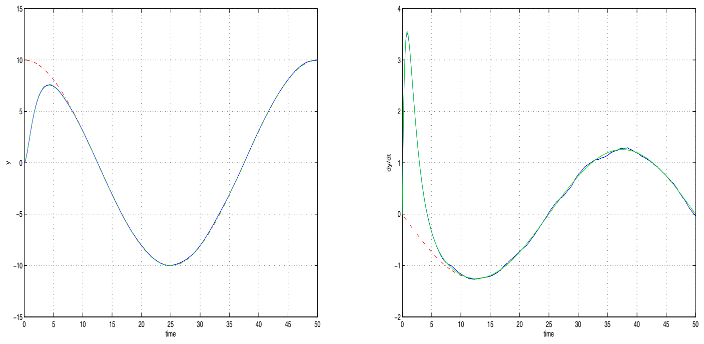
Fig. 7.28 (a) EKF-based estimation of the disturbance torque round the *z*-axis (continuous line) and real value of the *z*-axis disturbance torque (dashed line), (b) Trajectory of the ship on the *xy*-plane (continuous line) and desirable ship trajectory (dashed line) in the case of EKF-based state estimation.





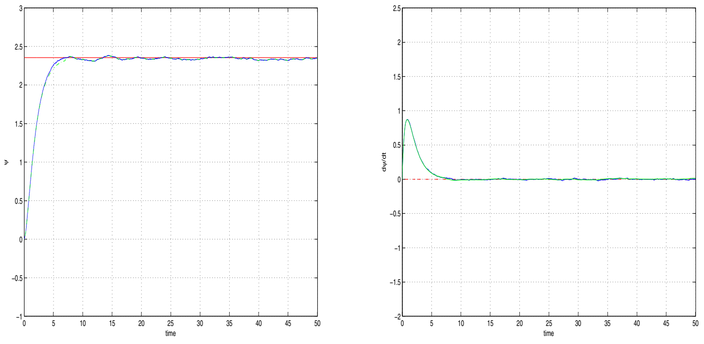
(a) **(b)**

Fig. 7.29 (a) PF-based estimation of the ship's position along the *x*-axis (continuous line) and desirable *x*-axis position (dashed line), (b) PF-based estimation of the ship's velocity along the *x*-axis (continuous line) and desirable *x*-axis velocity).



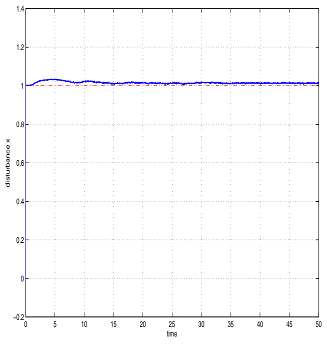
(a) (b)

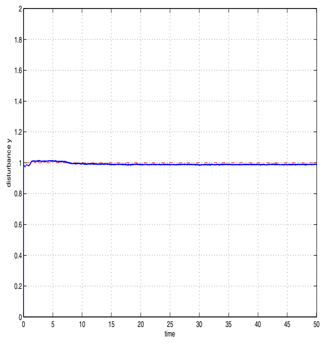
Fig. 7.30 (a) PF-based estimation of the ship's position along the *y*-axis (continuous line) and desirable *y*-axis position (dashed line), (b) PF-based estimation of the ship's velocity along the *y*-axis (continuous line) and desirable *y*-axis velocity).



(a) (b)

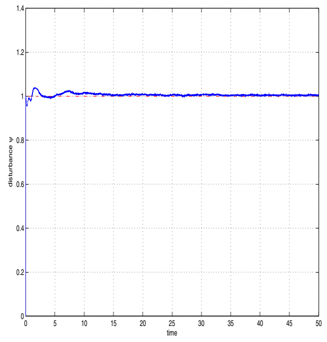
Fig. 7.31 (a) PF-based estimation of the ship's angle round the *z*-axis (continuous line) and desirable *z*-axis rotation angle (dashed line), (b) PF-based estimation of the ship's angular velocity round the *z*-axis (continuous line) and desirable angular velocity).

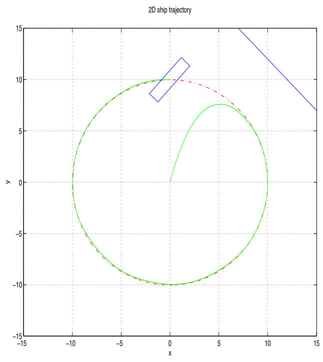




(a) **(b)**

Fig. 7.32 (a) PF-based estimation of the disturbance along the *x*-axis (continuous line) and real value of the *x*-axis disturbance (dashed line), (b) PF-based estimation of the disturbance along the *y*-axis (continuous line) and real value of the *y*-axis disturbance (dashed line).





(a) **(b)**

Fig. 7.33 (a) PF-based estimation of the disturbance torque round the *z*-axis (continuous line) and real value of the *z*-axis disturbance torque (dashed line), (b) Trajectory of the ship on the *xy*-plane (continuous line) and desirable ship trajectory (dashed line) in the case of PF-based state estimation.

Chapter 8
Distributed Filtering and Estimation for Industrial Systems

Abstract. Distributed Filtering and estimation methods for industrial systems are studied. Such methods are particularly useful in case that measurements about the industrial system are collected and processed by *n* different monitoring stations. The overall concept is that at each monitoring station a filter is used to track the state of the system by fusing measurements which are provided by various sensors, while by fusing the state estimates from the distributed local filters an aggregate state estimate for the industrial system is obtained. In particular, the chapter proposes first the Extended Information Filter (EIF) and the Unscented Information Filter (UIF) as possible approaches for fusing the state estimates provided by the local monitoring stations, under the assumption of Gaussian noises. The EIF and UIF estimated state vector can, in turn, be used by nonlinear controllers which can make the system's state track desirable setpoints. Moreover, the Distributed Particle Filter (DPF) is proposed for fusing the state estimates provided by the local monitoring stations (local filters). The motivation for using DPF is that it is well-suited to accommodate non-Gaussian measurements. The DPF estimated state vector can again be used by a nonlinear controller to make system converge to desirable setpoints. The performance of the Extended Information Filter, of the Unscented Information Filter and of the Distributed Particle Filter is evaluated through simulation experiments in the case of a 2-UAV (unmanned aerial vehicle) model monitored and remotely navigated by two local stations.

8.1 The Problem of Distributed State Estimation over Sensor Networks

State estimation and control over sensor networks is a problem met in several applications such as surveillance and condition monitoring of large-scale systems, multi-robot systems and cooperating UAVs. In sensor networks the simplest kind of architecture is centralized. Distributed sensors send measurement data to a central processing unit which provides the state estimate for the monitored system. Such an approach has several weaknesses: (i) it lacks fault tolerance: if the central processing unit is subject to a fault then state estimation becomes impossible, (ii) communication overhead often prohibits proper functioning in case of a large number of distributed measurement units. On the other hand decentralized architectures are based on the communication between neighboring measurement units [245],[318]. This assures scalability for the network since the number of messages received or sent by each measurement unit is independent of the total number of measurement units in the system. It has been shown that scalable decentralized state estimation can be achieved for linear Gaussian models, when the measurements are linear functions of the state and the associated process and measurement noise models follow a Gaussian distribution [283]. A solution to decentralized sensor fusion over sensor networks with the use of distributed Kalman Filtering has been proposed in [115],[290],[291],[408],[449]. Distributed state estimation in the case of non-Gaussian models has been studied in [354] where decentralized sensor fusion with the use of distributed particle filters has been proposed in several other research works [77],[245],[247].

In this chapter autonomous navigation of UAVs will be examined and a solution to this problem will be first attempted with the use of the Extended Information Filter and the Unscented Kalman filter [217],[218], [379],[435]. Comparatively, autonomous UAV navigation with the use of the Distributed Particle Filter will be studied. This problem belongs to the wider area of multi-source multi-target tracking [73],[74],[151],[157],[272]. Subproblems to be solved for succeeding autonomous navigation of the UAVs are: (i) implementation of sensor fusion with the use of distributed filtering. In this approach the goal is to consistently combine the local particle distribution with the communicated particle distribution coming from particle filters running on nearby measurement stations [47]. It is assumed that each local measurement station runs its own local filter and communicates information to other measurement stations close to it. The motivation for using particle filters is that they can represent almost arbitrary probability distributions, thus becoming well-suited to accommodate the types of uncertainty and nonlinearities that arise in the distributed estimation [335],[350] (ii) nonlinear control of the UAVs based on the state estimates provided by the particle filtering algorithm. Various approaches have been proposed for the UAV navigation using nonlinear feedback control [23],[322],[385]. The chapter proposes flatness-based control for the UAV models. Flatness-based control theory is based on the concept of differential flatness and has been successfully applied to several nonlinear dynamical systems. Flatness-based control for a UAV helicopter-like model has been developed in [216], assuming that the UAV performs manoeuvres at a constant altitude.

The chapter proposes first the Extended Information Filter (EIF) and the Unscented Information Filter (UIF) as possible approaches for fusing the state estimates provided by the local monitoring stations, under the assumption of Gaussian noises. The EIF and UIF estimated state vector is in turn used by a flatness-based controller that makes the UAV follow the desirable trajectory. The Extended Information Filter is a generalization of the Information Filter in which the local filters do not exchange raw measurements but send to an aggregation filter their local information matrices (local inverse covariance matrices which can be also associated to the Fisher Information Matrices) and their associated local information state vectors (products of the local information matrices with the local state vectors) [217],[379]. In the case of the Unscented Information Filter there is no linearization of the UAVs observation equation. However the application of the Information Filter algorithm is possible through an implicit linearization which is performed by approximating the Jacobian matrix of the system's output equation with the product of the inverse of the state vector's covariance matrix (information matrix) with the cross-correlation covariance matrix between the system's state vector and the system's output [218], [435]. Again, the local information matrices and the local information state vectors are transferred to an aggregation filter which produces the global estimation of the system's state vector.

Next, the Distributed Particle Filter (DPF) is proposed for fusing the state estimates provided by the local monitoring stations (local filters). The motivation for using DPF is that it is well-suited to accommodate non-Gaussian measurements. A difficulty in implementing distributed particle filtering is that particles from one particle set (which correspond to a local particle filter) do not have the same support (do not cover the same area and points on the samples space) as particles from another particle set (which are associated with another particle filter) [293], [294]. This can be resolved by transforming the particles sets into Gaussian mixtures, and defining the global probability distribution on the common support set of the probability density functions associated with the local filters. The state vector which is estimated with the use of the DPF is used again by a flatness-based controller to make each UAV follow a desirable flight path.

8.2 Distributed Extended Kalman Filtering

8.2.1 Calculation of Local Extended Kalman Filter Estimations

Again the discrete-time nonlinear system of Eq. (8.1) is considered.

$$\begin{aligned} x(k+1) &= \phi(x(k)) + L(k)u(k) + w(k) \\ z(k) &= \gamma(x(k)) + v(k) \end{aligned} \tag{8.1}$$

The Extended Information Filter (EIF) performs fusion of the local state vector estimates which are provided by the local Extended Kalman Filters, using the *Information matrix* and the *Information state vector* [217],[218],[252],[435]. The Information Matrix is the inverse of the state vector covariance matrix, and can be also associated to the Fisher Information matrix [344]. The Information state vector is the product between the Information matrix and the local state vector estimate

$$\begin{aligned} Y(k) &= P^{-1}(k) = I(k) \\ \hat{y}(k) &= P^{-}(k)^{-1}\hat{x}(k) = Y(k)\hat{x}(k) \end{aligned} \tag{8.2}$$

The update equations for the Information Matrix and the Information state vector are given by

$$\begin{aligned} Y(k) &= P^{-}(k)^{-1} + J_{\gamma}^{T}(k)R^{-1}(k)J_{\gamma}(k) \\ &= Y^{-}(k) + I(k) \end{aligned} \tag{8.3}$$

$$\begin{aligned} \hat{y}(k) &= \hat{y}^{-}(k) + J_{\gamma}^{T}R(k)^{-1}[z(k) - \gamma(x(k)) + J_{\gamma}\hat{x}^{-}(k)] \\ &= \hat{y}^{-}(k) + i(k) \end{aligned} \tag{8.4}$$

where

$$\begin{aligned} I(k) &= J_{\gamma}^{T}(k)R^{(}k)^{-1}J_{\gamma}(k) \text{ is the associated information matrix and} \\ i(k) &= J_{\gamma}^{T}R^{(}k)^{-1}[(z(k) - \gamma(x(k))) + J_{\gamma}\hat{x}^{-}(k)] \text{ is the information state contribution} \end{aligned} \tag{8.5}$$

The predicted information state vector and Information matrix are obtained from

$$\begin{aligned} \hat{y}^{-}(k) &= P^{-}(k)^{-1}\hat{x}^{-}(k) \\ Y^{-}(k) = P^{-}(k)^{-1} &= [J_{\phi}(k)P^{-}(k)J_{\phi}(k)^{T} + Q(k)]^{-1} \end{aligned} \tag{8.6}$$

The Extended Information Filter is next formulated for the case that multiple local sensor measurements and local estimates are used to increase the accuracy and reliability of the estimation. It is assumed that an observation vector $z^{i}(k)$ is available for N different sensor sites $i = 1, 2, \cdots, N$ and each sensor observes a common state according to the local observation model, expressed by

$$z^{i}(k) = \gamma(x(k)) + v^{i}(k),\ i = 1, 2, \cdots, N \tag{8.7}$$

where the local noise vector $v^{i}(k) \sim N(0, R^{i})$ is assumed to be white Gaussian and uncorrelated between sensors. The variance of a composite observation noise vector v_k is expressed in terms of the block diagonal matrix

$$R(k) = diag[R(k)^{1}, \cdots, R^{N}(k)]^{T} \tag{8.8}$$

The information contribution can be expressed by a linear combination of each local information state contribution i^{i} and the associated information matrix I^{i} at the i-th sensor site

$$\begin{aligned} i(k) &= \textstyle\sum_{i=1}^{N} J_{\gamma}^{i\,T}(k)R^{i}(k)^{-1}[z^{i}(k) - \gamma^{i}(x(k)) + J_{\gamma}^{i}(k)\hat{x}^{-}(k)] \\ I(k) &= \textstyle\sum_{i=1}^{N} J_{\gamma}^{i\,T}(k)R^{i}(k)^{-1}J_{\gamma}^{i}(k) \end{aligned} \tag{8.9}$$

Using Eq. (8.9) the update equations for fusing the local state estimates become

$$\begin{aligned} \hat{y}(k) &= \hat{y}^{-}(k) + \textstyle\sum_{i=1}^{N} J_{\gamma}^{i\,T}(k)R^{i}(k)^{-1}[z^{i}(k) - \gamma^{i}(x(k)) + J_{\gamma}^{i}(k)\hat{x}^{-}(k)] \\ Y(k) &= Y^{-}(k) + \textstyle\sum_{i=1}^{N} J_{\gamma}^{i\,T}(k)R^{i}(k)^{-1}J_{\gamma}^{i}(k) \end{aligned} \tag{8.10}$$

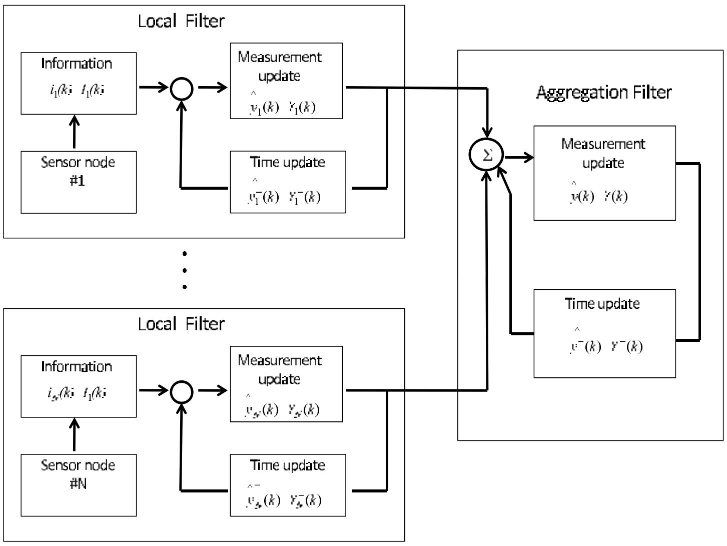


Fig. 8.1 Fusion of the distributed state estimates with the use of the Extended Information Filter

It is noted that in the Extended Information Filter an aggregation (master) fusion filter produces a global estimate by using the local sensor information provided by each local filter.

As in the case of the Extended Kalman Filter the local filters which constitute the Extended information Filter can be written in terms of *time update* and a *measurement update* equation.

Measurement update: Acquire $z(k)$ and compute

$$\begin{gathered} Y(k)=P^-(k)^{-1}+J_\gamma^T(k)R(k)^{-1}J_\gamma(k) \\ \text{or } Y(k)=Y^-(k)+I(k) \text{ where } I(k)=J_\gamma^T(k)R^{-1}(k)J_\gamma(k) \end{gathered} \tag{8.11}$$

$$\begin{gathered} \hat{y}(k)=\hat{y}^-(k)+J_\gamma^T(k)R(k)^{-1}[z(k)-\gamma(\hat{x}(k))+J_\gamma\hat{x}^-(k)] \\ \text{or } \hat{y}(k)=\hat{y}^-(k)+i(k) \end{gathered} \tag{8.12}$$

Time update: Compute

$$Y^-(k+1)=P^-(k+1)^{-1}=[J_\phi(k)P(k)J_\phi(k)^T+Q(k)]^{-1} \tag{8.13}$$

$$y^-(k+1)=P^-(k+1)^{-1}\hat{x}^-(k+1) \tag{8.14}$$

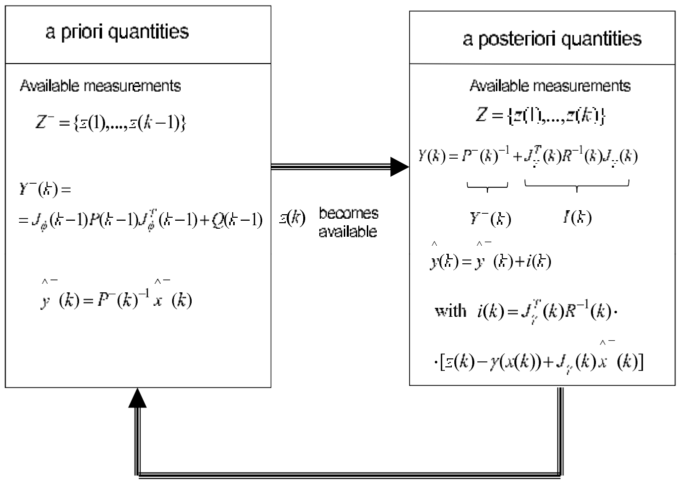

a priori quantities
Available measurements
$Z^- = \{z(1),\dots,z(k-1)\}$
$Y^-(k) = J_\phi(k-1)P(k-1)J_\phi^T(k-1)+Q(k-1)$
$\hat{y}^-(k) = P^-(k)^{-1}\hat{x}^-(k)$
$z(k)$ becomes available
a posteriori quantities
Available measurements
$Z = \{z(1),\dots,z(k)\}$
$Y(k) = \underbrace{P^-(k)^{-1}}_{Y^-(k)} + \underbrace{J_\gamma^T(k)R^{-1}(k)J_\gamma(k)}_{I(k)}$
$\hat{y}(k) = \hat{y}^-(k)+i(k)$
with $i(k) = J_\gamma^T(k)R^{-1}(k)\cdot[z(k)-\gamma(x(k))+J_\gamma(k)\hat{x}^-(k)]$

Fig. 8.2 Schematic diagram of the Extended Information Filter loop

8.2.2 *Extended Information Filtering for State Estimates Fusion*

In the Extended Information Filter each one of the local filters operates independently, processing its own local measurements. It is assumed that there is no sharing of measurements between the local filters and that the aggregation filter (Fig. 8.1) does not have direct access to the raw measurements feeding each local filter. The outputs of the local filters are treated as measurements which are fed into the aggregation fusion filter [217], [218], [435]. Then each local filter is expressed by its respective error covariance and estimate in terms of information contributions given in Eq.(8.6)

$$\begin{gathered} P_i^{-1}(k) = P_i^-(k)^{-1} + J_\gamma^T R^(k)^{-1} J_\gamma(k) \\ \hat{x}_i(k) = P_i(k)(P_i^-(k)^{-1}\hat{x}_i^-(k)) + J_\gamma^T R(k)^{-1}[z^i(k) - \gamma^i(x(k)) + J_\gamma^i(k)\hat{x}_i^-(k)] \end{gathered} \tag{8.15}$$

It is noted that the local estimates are suboptimal and also conditionally independent given their own measurements. The global estimate and the associated error covariance for the aggregate fusion filter can be rewritten in terms of the computed estimates and covariances from the local filters using the relations

$$\begin{gathered} J_\gamma^T(k)R(k)^{-1}J_\gamma(k) = P_i(k)^{-1} - P_i^-(k)^{-1} \\ J_\gamma^T(k)R(k)^{-1}[z^i(k) - \gamma^i(x(k)) + J_\gamma^i(k)\hat{x}^-(k)] = P_i(k)^{-1}\hat{x}_i(k) - P_i(k)^{-1}\hat{x}_i(k-1) \end{gathered} \tag{8.16}$$

For the general case of N local filters $i = 1,\cdots,N$, the distributed filtering architecture (aggregate covariance matrix and aggregate state vector estimate) is described by the following equations

$$\begin{array}{c} P(k)^{-1} = P^{-}(k)^{-1} + \sum_{i=1}^{N}[P_i(k)^{-1} - P_i^{-}(k)^{-1}] \\ \hat{x}(k) = P(k)[P^{-}(k)^{-1}\hat{x}^{-}(k) + \sum_{i=1}^{N}(P_i(k)^{-1}\hat{x}_i(k) - P_i^{-}(k)^{-1}\hat{x}_i^{-}(k))] \end{array} \quad (8.17)$$

It is noted that the global state update equation in the above distributed filter can be written in terms of the information state vector and of the information matrix

$$\begin{array}{c} \hat{y}(k) = \hat{y}^{-}(k) + \sum_{i=1}^{N}(\hat{y}_i(k) - \hat{y}_i^{-}(k)) \\ \hat{Y}(k) = \hat{Y}^{-}(k) + \sum_{i=1}^{N}(\hat{Y}_i(k) - \hat{Y}_i^{-}(k)) \end{array} \quad (8.18)$$

The local filters provide their own local estimates and repeat the cycle at step $k+1$. In turn the global filter can predict its global estimate and repeat the cycle at the next time step $k+1$ when the new state $\hat{x}(k+1)$ and the new global covariance matrix $P(k+1)$ are calculated. From Eq. (16.63) it can be seen that if a local filter (processing station) fails, then the local covariance matrices and the local state estimates provided by the rest of the filters will enable an accurate computation of the system's state vector.

8.3 Distributed Sigma-Point Kalman Filtering

8.3.1 Calculation of Local Unscented Kalman Filter Estimations

It is also possible to estimate the state vectors of distributed industrial systems (such as distributed UAVs constituting a multi-UAV swarm) through the fusion of the estimates provided by local Sigma-Point Kalman Filters. This can be succeeded using the Distributed Sigma-Point Kalman Filter, also known as *Unscented Information Filter* (UIF) [217], [218]. First, the functioning of the local Sigma-Point Kalman Filters will be explained. Each local Sigma-Point Kalman Filter generates an estimation of the state vector by fusing measurements from distributed sensors (e.g. in the UAV case such sensors can be the IMU and the GPS), Sigma-Point Kalman Filtering is proposed [170], [171], [368]. As explained in Section 6.5 the Sigma-Point Kalman Filter overcomes some of the flaws of Extended Kalman Filtering. Unlike EKF, in SPKF no analytical Jacobians of the system equations need to be calculated. This makes the sigma-point approach suitable for application in "black-box" models where analytical expressions of the system dynamics are either not available or not in a form which allows for easy linearization. This is achieved through a different approach for calculating the posterior 1st and 2nd order statistics of a random variable that undergoes a nonlinear transformation. The state distribution is represented again by a Gaussian Random Variable but is now specified using a minimal set of deterministically chosen weighted sample points.

The Unscented Information Filter is derived by introducing a linear error propagation based on the unscented transformation into the Extended Information Filter structure. First, an augmented state vector $x_{\alpha}{}^{-}(k)$ is considered, along with the process noise vector, and the associated covariance matrix is introduced

$$\hat{x}_{\alpha}^{-}(k) = \begin{pmatrix} \hat{x}^{-}(k) \\ \hat{w}^{-}(k) \end{pmatrix}, \ P^{\alpha-}(k) = \begin{pmatrix} P^{-}(k) & 0 \\ 0 & Q^{-}(k) \end{pmatrix} \quad (8.19)$$

As in the case of local (lumped) Unscented Kalman Filters, a set of weighted sigma points $X_{\alpha}^{i\,-}(k)$ is generated as

$$\begin{aligned} X_{\alpha,0}^{-}(k) &= \hat{x}_{\alpha}^{-}(k) \\ X_{\alpha,i}^{-}(k) &= \hat{x}_{\alpha}^{-}(k) + [\sqrt{(n_\alpha+\lambda)P_{\alpha}^{-}(k-1)}]_i,\ i=1,\cdots,n \\ X_{\alpha,i}^{-}(k) &= \hat{x}_{\alpha}^{-}(k) + [\sqrt{(n_\alpha+\lambda)P_{\alpha}^{-}(k-1)}]_i,\ i=n+1,\cdots,2n \end{aligned} \tag{8.20}$$

where $\lambda=\alpha^2(n_\alpha+\kappa)-n_\alpha$ is a scaling, while $0\leq\alpha\leq 1$ and κ are constant parameters. The corresponding weights for the mean and covariance are defined as in the case of the lumped Unscented Kalman Filter

$$\begin{aligned} W_0^{(m)} &= \tfrac{\lambda}{n_\alpha+\lambda} & W_0^{(c)} &= \tfrac{\lambda}{(n_\alpha+\lambda)+(1-\alpha^2+\beta)} \\ W_i^{(m)} &= \tfrac{1}{2(n_\alpha+\lambda)},\ i=1,\cdots,2n_\alpha & W_i^{(C)} &= \tfrac{1}{2(n_\alpha+\lambda)},\ i=1,\cdots,2n_\alpha \end{aligned} \tag{8.21}$$

where β is again a constant parameter. The equations of the prediction stage (measurement update) of the information filter, i.e. the calculation of the information matrix and the information state vector of Eq. (8.6) now become

$$\begin{aligned} \hat{y}^{-}(k) &= Y^{-}(k)\Sigma_{i=0}^{2n_\alpha}W_i^{m}X_i^{x}(k) \\ Y^{-}(k) &= P^{-}(k)^{-1} \end{aligned} \tag{8.22}$$

where X_i^x are the predicted state vectors when using the sigma point vectors X_i^w in the state equation $X_i^x(k+1)=\phi(X_i^{w-}(k))+L(k)U(k)$. The predicted state covariance matrix is computed as

$$P^{-}(k)=\sum_{i=0}^{2n_\alpha}W_i^{(c)}[X_i^x(k)-\hat{x}^{-}(k)][X_i^x(k)-\hat{x}^{-}(k)]^T \tag{8.23}$$

As noted, the equations of the Extended Information Filter (EIF) are based on the linearized dynamic model of the system and on the inverse of the covariance matrix of the state vector. However, in the equations of the Unscented Kalman Filter (UKF) there is no linearization of the system dynamics, thus the UKF cannot be included directly into the EIF equations. Instead, it is assumed that the nonlinear measurement equation of the system given in Eq. (16.48) can be mapped into a linear function of its statistical mean and covariance, which makes possible to use the information update equations of the EIF. Denoting $Y_i(k)=\gamma(X_i^x(k))$ (i.e. the output of the system calculated through the propagation of the i-th sigma point X^i through the system's nonlinear equation) the observation covariance and its cross-covariance are approximated by

$$\begin{aligned} P_{YY}^{-}(k) &= E[(z(k)-\hat{z}^{-}(k))(z(k)-\hat{z}^{-}(k))^T] \\ &\simeq J_\gamma(k)P^{-}(k)J_\gamma(k)^T \end{aligned} \tag{8.24}$$

$$\begin{aligned} P_{XY}^{-}(k) &= E[(x(k)-\hat{x}(k)^{-})(z(k)-\hat{z}(k)^{-})^T] \\ &\simeq P^{-}(k)J_\gamma(k)^T \end{aligned} \tag{8.25}$$

where $z(k)=\gamma(x(k))$ and $J_\gamma(k)$ is the Jacobian of the output equation $\gamma(x(k))$. Next, multiplying the predicted covariance and its inverse term on the right side of the information matrix Eq. (8.5) and replacing $P(k)J_\gamma(k)^T$ with $P_{XY}^-(k)$ gives the following representation of the information matrix [217], [218], [435]

$$\begin{aligned} I(k) &= J_\gamma(k)^T R(k)^{-1} J_\gamma(k) \\ &= P^-(k)^{-1} P^-(k) J_\gamma(k)^T R(k)^{-1} J_\gamma^-(k) P^-(k)^T (P^-(k)^{-1})^T \\ &= P^-(k)^{-1} P_{XY}(k) R(k)^{-1} P_{XY}(k)^T (P^-(k)^{-1})^T \end{aligned} \tag{8.26}$$

where $P^-(k)^{-1}$ is calculated according to Eq. (8.23) and the cross-correlation matrix $P_{XY}(k)$ is calculated from

$$P_{XY}^-(k)=\sum_{i=0}^{2n_\alpha} W_i^{(c)} [X_i^x(k)-\hat{x}^-(k)][Y_i(k)-\hat{z}^-(k)]^T \tag{8.27}$$

where $Y_i(k)=\gamma(X_i^x(k))$ and the predicted measurement vector $\hat{z}^-(k)$ is obtained by $\hat{z}^-(k)=\sum_{i=0}^{2n_\alpha} W_i^{(m)} Y_i(k)$. Similarly, the information state vector i_k can be rewritten as

$$\begin{aligned} i(k) &= J_\gamma(k)^T R(k)^{-1}[z(k)-\gamma(x(k))+J_\gamma(k)^T \hat{x}^-(k)] \\ &= P^-(k)^{-1} P^-(k) J_\gamma(k)^T R(k)^{-1}[z(k)-\gamma(x(k))+J_\gamma(k)^T (P^-(k))^T (P^-(k)^{-1})^T \hat{x}^-(k)] \\ &= P^-(k)^{-1} P_{XY}^-(k) R(k)^{-1}[z(k)-\gamma(x(k))+P_{XY}^-(k)(P^-(k)^{-1})^T \hat{x}^-(k)] \end{aligned} \tag{8.28}$$

To complete the analogy to the information contribution equations of the EIF a "measurement" matrix $H^T(k)$ is defined as

$$H(k)^T = P^-(k)^{-1} P_{XY}^-(k) \tag{8.29}$$

In terms of the "measurement" matrix $H(k)$ the information contributions equations are written as

$$\begin{aligned} i(k) &= H^T(k) R(k)^{-1}[z(k)-\gamma(x(k))+H(k)\hat{x}^-(k)] \\ I(k) &= H^T(k) R(k)^{-1} H(k) \end{aligned} \tag{8.30}$$

The above procedure leads to an implicit linearization in which the nonlinear measurement equation of the system given in Eq. (8.1) is approximated by the statistical error variance and its mean

$$z(k)=\gamma(x(k)) \simeq H(k)x(k)+\bar{u}(k) \tag{8.31}$$

where $\bar{u}(k)=\gamma(\hat{x}^-(k))-H(k)\hat{x}^-(k)$ is a measurement residual term. (8.31).

Next, the local estimations provided by distributed (local) Unscented Kalmans filters will be expressed in terms of the information contributions (information matrix I and information state vector i) of the Unscented Information Filter, which were defined in Eq. (8.30) [217], [218], [435]. It is assumed that the observation vector $\bar{z}_i(k+1)$ is available from N different sensors, and that each sensor observes a common state according to the local observation model, expressed by

$$\bar{z}_i(k) = H_i(k)x(k) + \bar{u}_i(k) + v_i(k) \tag{8.32}$$

where the noise vector $v_i(k)$ is taken to be white Gaussian and uncorrelated between sensors. The variance of the composite observation noise vector v_k of all sensors is written in terms of the block diagonal matrix $R(k) = diag[R_1(k)^T, \cdots, R_N(k)^T]^T$. Then one can define the local information matrix $I_i(k)$ and the local information state vector $i_i(k)$ at the i-th sensor, as follows

$$\begin{array}{c} i_i(k) = H_i^T(k)R_i(k)^{-1}[z_i(k) - \gamma^i(x(k)) + H_i(k)\hat{x}^-(k)] \\ I_i(k) = H_i^T(k)R_i(k)^{-1}H_i(k) \end{array} \tag{8.33}$$

Since the information contribution terms have group diagonal structure in terms of the innovation and measurement matrix, the update equations for the multiple state estimation and data fusion are written as a linear combination of the local information contribution terms

$$\begin{array}{c} \hat{y}(k) = \hat{y}^-(k) + \sum_{i=1}^N i_i(k) \\ Y(k) = Y^-(k) + \sum_{i=1}^N I_i(k) \end{array} \tag{8.34}$$

Then using Eq. (8.22) one can find the mean state vector for the multiple sensor estimation problem.

As in the case of the Unscented Kalman Filter, the Unscented Information Filter running at the i-th measurement processing unit can be written in terms of *measurement update* and *time update* equations:

Measurement update: Acquire measurement $z(k)$ and compute

$$\begin{array}{c} Y(k) = P^-(k)^{-1} + H^T(k)R^{-1}(k)H(k) \\ \text{or } Y(k) = Y^-(k) + I(k) \text{ where } I(k) = H^T(k)R^{-1}(k)H(k) \end{array} \tag{8.35}$$

$$\begin{array}{c} \hat{y}(k) = \hat{y}^-(k) + H^T(k)R^{-1}(k)[z(k) - \gamma(\hat{x}(k)) + H(k)\hat{x}^-(k)] \\ \text{or } \hat{y}(k) = \hat{y}^-(k) + i(k) \end{array} \tag{8.36}$$

Time update: Compute

$$\begin{array}{c} Y^-(k+1) = (P^-(k+1))^{-1} \\ \text{where } P^-(k+1) = \sum_{i=0}^{2n_\alpha} W_i^{(c)}[X_i^x(k+1) - \hat{x}^-(k+1)][X_i^x(k+1) - \hat{x}^-(k+1)]^T \end{array} \tag{8.37}$$

$$\begin{array}{c} \hat{y}(k+1) = Y(k+1)\sum_{i=0}^{2n_\alpha} W_i^{(m)} X_i^x(k+1) \\ \text{where } X_i^x(k+1) = \phi(X_i^w(k)) + L(k)U(k) \end{array} \tag{8.38}$$

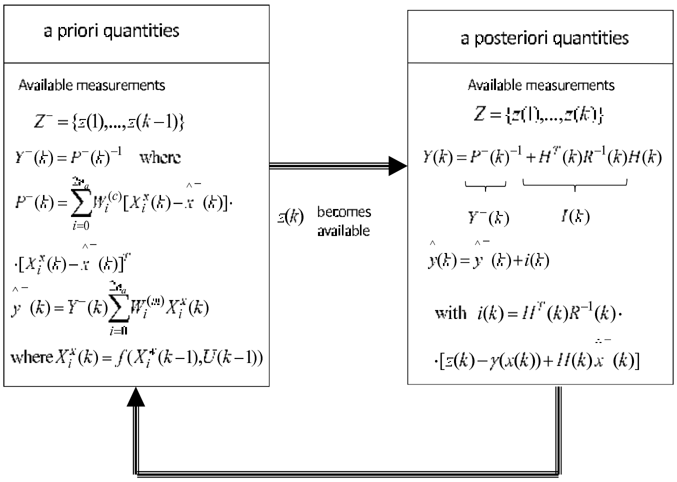

a priori quantities
Available measurements
$Z^- = \{z(1),\ldots,z(k-1)\}$
$Y^-(k) = P^-(k)^{-1}$ where
$P^-(k) = \sum_{i=0}^{2n_a} W_i^{(c)}[X_i^x(k) - \hat{x}^-(k)] \cdot [X_i^x(k) - \hat{x}^-(k)]^T$
$\hat{y}^-(k) = Y^-(k) \sum_{i=0}^{2n_a} W_i^{(m)} X_i^x(k)$
where $X_i^x(k) = f(X_i^x(k-1), U(k-1))$
$z(k)$ becomes available
a posteriori quantities
Available measurements
$Z = \{z(1),\ldots,z(k)\}$
$Y(k) = \underbrace{P^-(k)^{-1}}_{Y^-(k)} + \underbrace{H^T(k)R^{-1}(k)H(k)}_{I(k)}$
$\hat{y}(k) = \hat{y}^-(k) + i(k)$
with $i(k) = H^T(k)R^{-1}(k) \cdot [z(k) - \gamma(x(k)) + H(k)\hat{x}^-(k)]$

Fig. 8.3 Schematic diagram of the Unscented Information Filter loop

8.3.2 Unscented Information Filtering for State Estimates Fusion

It has been shown that the update of the aggregate state vector of the Unscented Information Filter architecture can be expressed in terms of the local information matrices I_i and of the local information state vectors i_i, which in turn depend on the local covariance matrices P and cross-covariance matrices P_{XY}. Next, it will be shown that the update of the aggregate state vector can be also expressed in terms of the local state vectors $x_i(k)$ and in terms of the local covariance matrices $P_i(k)$ and cross-covariance matrices $P^i_{XY}(k)$. It is assumed that the local filters do not have access to each other row measurements and that they are allowed to communicate only their information matrices and their local information state vectors. Thus each local filter is expressed by its respective error covariance and estimate in terms of the local information state contribution i_i and its associated information matrix I_i at the i-th filter site. Then using Eq. (8.22) one obtains

$$\begin{aligned} P_i(k)^{-1} &= P_i^-(k)^{-1} + H_i^T(k)R_i(k)^{-1}H_i(k) \\ \hat{x}_i &= P_i(k)(P_i^-(k)\hat{x}_i^-(k) + H_i^T(k)R_i(k)^{-1}[z_i(k) - \gamma^i(x(k)) + H_i(k)\hat{x}^-(k)]) \end{aligned} \tag{8.39}$$

Using Eq. (8.39), each local information state contribution i_i and its associated information matrix I_i at the i-th filter are rewritten in terms of the computed estimates and covariances of the local filters

$$\begin{aligned} H_i^T(k)R_i(k)^{-1}H_i(k) &= P_i^{-1}(k) - P_i^-(k)^{-1} \\ H_i^T(k)R_i(k)^{-1}[z_i(k) - \gamma^i(x(k)) + H_i(k)\hat{x}^-(k)]) &= P_i(k)^{-1}\hat{x}_i(k) - P_i^-(k)^{-1}\hat{x}_i^-(k) \end{aligned} \tag{8.40}$$

where according to Eq.(8.29) it holds $H_i(k) = P_i^-(k)^{-1}P_{XY,i}^-(k)$. Next, the aggregate estimates of the distributed Unscented Information Filtering are derived for a number of N local filters $i = 1,\cdots,N$ and sensor measurements, first in terms of covariances [217], [218],[435]

$$\begin{array}{c} P(k)^{-1} = P^-(k)^{-1} + \sum_{i=1}^N [P_i(k)^{-1} - P_i^-(k)^{-1}] \\ \hat{x}(k) = P(k)[P^-(k)^{-1}\hat{x}^-(k) + \sum_{i=1}^N (P_i(k)^{-1}\hat{x}_i(k) - P_i^-(k)^{-1}\hat{x}_i^-(k))] \end{array} \quad (8.41)$$

and also in terms of the information state vector and of the information state covariance matrix

$$\begin{array}{c} \hat{y}(k) = \hat{y}^-(k) + \sum_{i=1}^N (\hat{y}_i(k) - \hat{y}_i^-(k)) \\ Y(k) = Y^-(k) + \sum_{i=1}^N [Y_i(k) - Y_i^-(k)] \end{array} \quad (8.42)$$

State estimation fusion based on the Unscented Information Filter (UIF) is fault tolerant. From Eq. (8.41) it can be seen that if a local filter (processing station) fails, then the local covariance matrices and local estimates provided by the rest of the filters will enable a reliable calculation of the system's state vector. Moreover, it is and computationally efficient comparing to centralized filters and results in enhanced estimation accuracy.

8.4 Distributed Particle Filter

8.4.1 Distributed Particle Filtering for State Estimation Fusion

The Distributed Particle Filter performs fusion of the state vector estimates which are provided by the local Particle Filters. This is succeeded by fusing the discrete probability density functions of the local Particle Filters into a common probability distribution of the system's state vector. Without loss of generality fusion between two estimates which are provided by two different probabilistic estimators (particle filters) is assumed. This amounts to a multiplication and a division operation to remove the common information, and is given by [293], [294]

$$p(x(k)|Z_A \bigcup Z_B) \propto \frac{p(x(k)|Z_A)p(x(k)|Z_B)}{p(x(k)|Z_A \bigcap Z_B)} \quad (8.43)$$

where Z_A is the sequence of measurements associated with the i-th processing unit and Z_B is the sequence of measurements associated with the j-th measurement unit. In the implementation of distributed particle filtering, the following issues arise:

1. Particles from one particle set (which correspond to a local particle filter) do not have the same support (do not cover the same area and points on the samples space) as particles from another particle set (which are associated with another particle filter). Therefore a point-to-point application of Eq. (8.43) is not possible.
2. The communication of particles representation (i.e. local particle sets and associated weight sets) requires significantly more bandwidth compared to other representations, such as Gaussian mixtures.

Fusion of the estimates provided by the local particle filters (located at different processing units) can be performed through the following stages. First, the discrete particle set of Particle Filter A (Particle Filter B) is transformed into a continuous distribution by placing a Gaussian kernel over each sample (Fig. 8.4) [277]

$$K_h(x) = h^2 K(x) \tag{8.44}$$

where $K()$ is the rescaled Kernel density and $h > 0$ is the scaling parameter. Then the continuous distribution A (B) is sampled with the other particles set B (A) to obtain the new importance weights, so that the weighted sample corresponds to the numerator of Eq. (8.43) (Fig. 8.5). Such a conversion from discrete particle probability distribution functions $\sum_{i=1}^{N} w_A^{(i)} \delta(x_A^{(i)})$ ($\sum_{i=1}^{N} w_B^{(i)} \delta(x_B^{(i)})$) into continuous distributions is denoted as

$$\sum_{i=1}^{N} w_A^{(i)} \delta(x_A^{(i)}) \rightarrow p_A(x) \quad (\sum_{i=1}^{N} w_B^{(i)} \delta(x_B^{(i)}) \rightarrow p_B(x)) \tag{8.45}$$

The common information appearing in the processing units A and B should not be taken into account in the joint probability distribution which is created after fusing the local probability densities of A and B. This means that in the joint p.d.f. one should sample with importance weights calculated according to Eq. (8.43). The objective is then to create an importance sampling approximation for the joint distribution that will be in accordance to Eq. (8.43). A solution to this can be obtained through Monte Carlo sampling and suitable selection of the so-called ”proposal distribution” [293], [294].

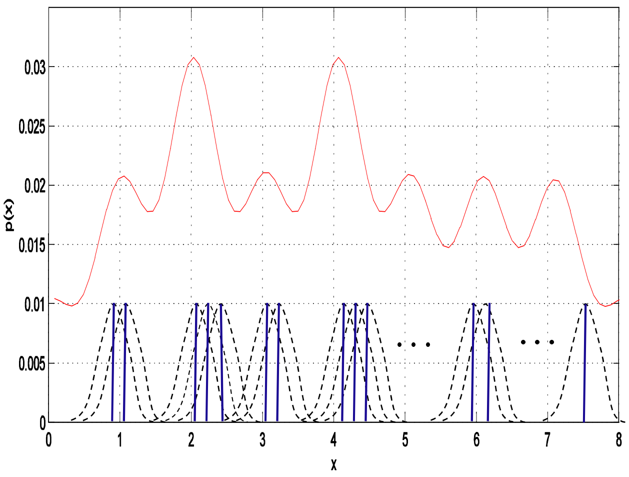


Fig. 8.4 Conversion of the particles discrete probability density function to a continuous distribution, after allocating a Gaussian kernel over each particle

8.4.2 Fusion of the Local Probability Density Functions

According to the above, for the joint distribution the idea behind Monte Carlo sampling is to draw N i.i.d samples from the associated probability density function $p(x)$, such that the target density is approximated by a point-mass function of the form

$$p(x) \simeq \sum_{i=1}^{N} w_k^{(i)} \delta(x_k^{(i)}) \tag{8.46}$$

where $\delta(x_k^{(i)})$ is a Dirac delta mass located at $x_k^{(i)}$. Then the expectation of some function $f(x)$ with respect to the pdf $p(x)$ is given by

$$I(f) = E_{p(x)}[f(x)] = \int f(x)p(x)dx \tag{8.47}$$

the Monte-Carlo approximation of the integral with samples is then

$$I_N(f) = \frac{1}{N}\sum_{i=1}^{N} f(x^{(i)}) \tag{8.48}$$

where $x^{(i)} \simeq p(X)$ and $I_N(f) \to I(f)$ for $N \to \infty$. since, the true probability distribution $p(x)$ is hard to sample from, the concept of importance sampling is to select a proposal distribution $\bar{p}(x)$ in place of $p(x)$, with the assumption that $\bar{p}(x)$ includes the support space of $p(x)$. Then the expectation of function $f(x)$, previously given in Eq. (8.47), is now calculated as

$$I(f) = \int f(x)\frac{p(x)}{\bar{p}(x)}\bar{p}(x)dx = \int f(x)w(x)\bar{p}(x)dx \tag{8.49}$$

where $w(x)$ are the importance weights

$$w(x) = \frac{p(x)}{\bar{p}(x)} \tag{8.50}$$

Consequently, the Monte-Carlo estimation of the mean value of function $f(x)$ becomes

$$I_N(f) = \sum_{i=1}^{N} f(x^{(i)})w(x^{(i)}) \tag{8.51}$$

For the division operation, the desired probability distribution is

$$p(x^{(i)}) = \frac{p_A(x^{(i)})}{p_B(x^{(i)})} \tag{8.52}$$

In that case the important weights of the fused probability density functions become

$$w(x^{(i)}) = \frac{p_A(x^{(i)})}{p_B(x^{(i)})\bar{p}(x^{(i)})} \tag{8.53}$$

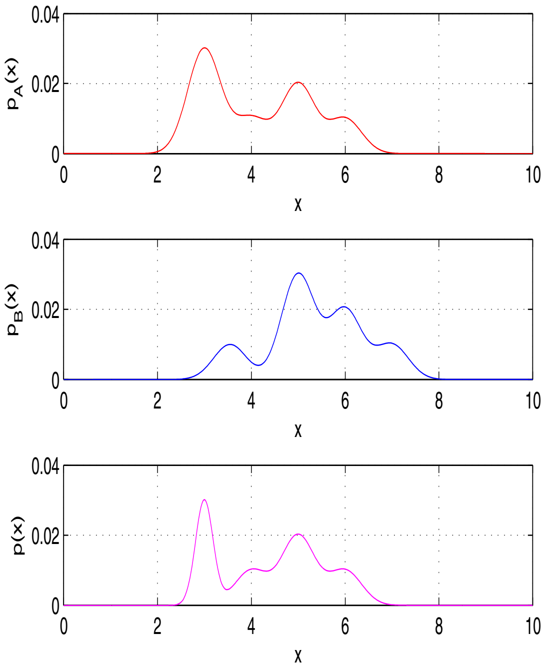


Fig. 8.5 Fusion of the probability density functions produced by the local particle filters

which is then normalized so that $\sum_{i=1}^{N} w(x^{(i)}) = 1/N$, where N is the number of particles. The next step is to decide what will be the form of the proposal distribution $\bar{p}(x)$. A first option is to take $\bar{p}(x)$ to be a uniform distribution, with a support that covers both of the support sets of the distributions A and B.

$$\bar{p}(x) = U(x) \tag{8.54}$$

Then the sample weights $\bar{p}(x^{(i)})$ are all equal at a constant of value C. Hence the importance weights are

$$w(x^{(i)}) = \frac{p_A(x^{(i)})}{p_B(x^{(i)})C} \tag{8.55}$$

Another suitable proposal distribution that takes more into account the new information received (described as the probability distribution of the second processing unit) is given by

$$\bar{p}(x) = p_B(x) \tag{8.56}$$

and the important weights are then adjusted to be

$$w(x^{(i)}) = \frac{p_A(x^{(i)})}{p_B(x^{(i)})^2} \tag{8.57}$$

8.5 Simulation Tests

8.5.1 Multi-UAV Control with Extended Information Filtering

It was assumed that $m = 2$ helicopter models were monitored by $n = 2$ different ground stations. At each ground station an Extended Kalman Filter was used to track each UAV. By fusing the measurements provided by the sensors mounted on each UAV, each local EKF was able to provide an estimation of a UAV's motion. Next, the state estimates obtained by the pair local EKFs associated with each UAV was fused with the use of the Extended Information Filter. This fusion-based state estimation scheme is depicted in Fig 8.6. As explained in described in Section 8.2 the weighting of the state estimates of the local EKFs was performed using the local information matrices. The distributed fitering architecture is shown in Fig. 8.6

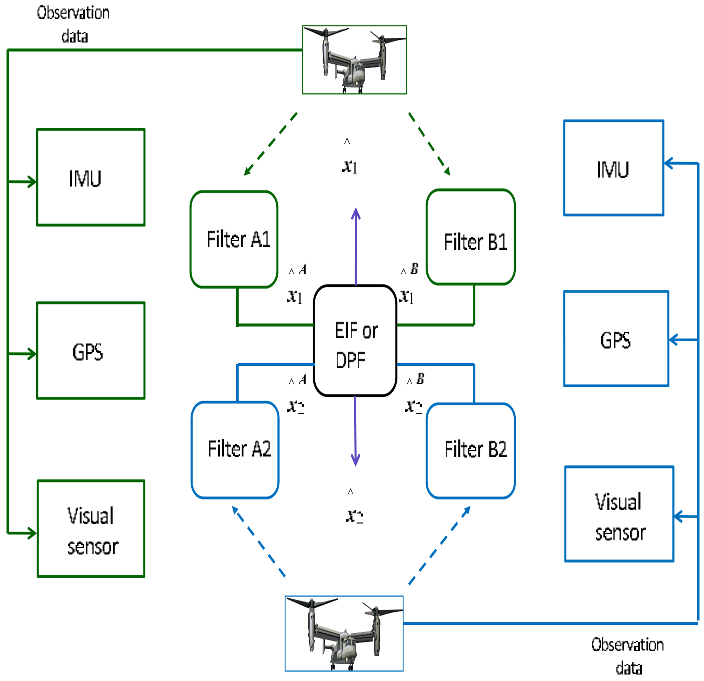


Fig. 8.6 Distributed Filtering over WSN

Next, some details will be given about the local EKF design for the UAV model of Eq. (3.5). Modeling of the UAV kinematics has been studied in [53], [181],[307]. Assuming that the UAV performs maneuvers at a constant altitude, its continuous-time kinematic equation is :

$$\begin{aligned}\dot{x}(t) &= v(t)cos(\theta(t))\\ \dot{y}(t) &= v(t)sin(\theta(t))\\ \dot{\theta}(t) &= \omega(t)\end{aligned} \tag{8.58}$$

The IMU system provides measurements or the UAV's position $[x, y]$ and the UAV's orientation angle θ over a sampling period T. These sensors are used to obtain an estimation of the displacement and the angular velocity of the UAV $v(t)$ and $\omega(t)$, respectively. The IMU sensors can introduce incremental errors, which result in an erroneous estimation of the orientation θ. To improve the accuracy of the UAV's localization, measurements from the GPS can be used. On the other hand, the GPS on this own is not always reliable since its signal can be intermittent. Therefore, to succeed accurate localization of the UAV it is necessary to fuse the GPS measurements with the IMU measurements of the UAV or with measurements from vision sensors (visual odometry) [117].

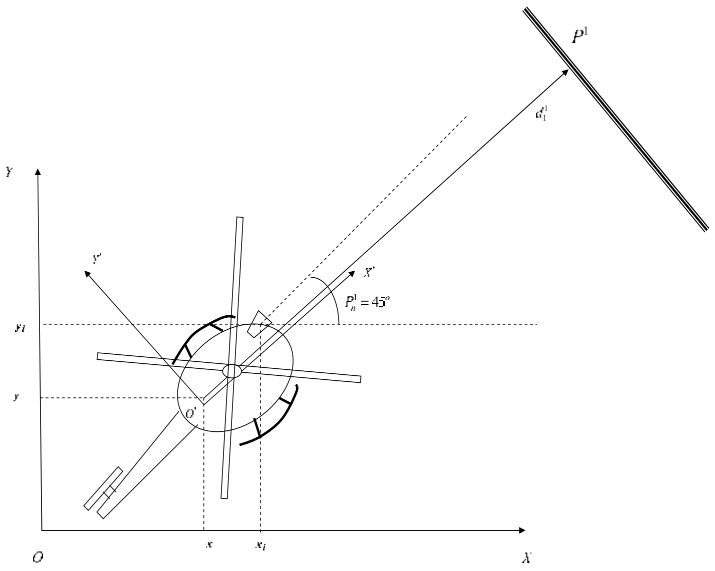


Fig. 8.7 Reference frames for the UAV

The inertial coordinates system OXY is defined. Furthermore the coordinates system $O'X'Y'$ is considered (Fig. 8.7). $O'X'Y'$ results from OXY if it is rotated by an angle θ. The coordinates of the center of symmetry of the UAV with respect to OXY are (x, y), while the coordinates of the GPS or vision sensor that is mounted on the UAV, with respect to $O'X'Y'$ are $x_i^{'}, y_i^{'}$. The orientation of the GPS (or vision sensor) with respect to $OX'Y'$ is $\theta_i^{'}$. Thus the coordinates of the GPS or vision sensor with respect to OXY are (x_i, y_i) and its orientation is θ_i, and are given by:

$$\begin{aligned} x_i(k) &= x(k) + x_i^{'} sin(\theta(k)) + y_i^{'} cos(\theta(k)) \\ y_i(k) &= y(k) - x_i^{'} cos(\theta(k)) + y_i^{'} sin(\theta(k)) \\ \theta_i(k) &= \theta(k) + \theta_i \end{aligned} \tag{8.59}$$

For manoeuvres at constant altitude the GPS measurement (or the vision sensor measurement) can be considered as the measurement of the distance from a reference surface P^j. A reference surface P^j in the UAVs 2D flight area can be represented by P_r^j and P_n^j, where (i) P_r^j is the normal distance of the plane from the origin O, (ii) P_n^j is the angle between the normal line to the plane and the x-direction.

The GPS sensor (or vision sensor i) is at position $x_i(k), y_i(k)$ with respect to the inertial coordinates system OXY and its orientation is $\theta_i(k)$. Using the above notation, the distance of the GPS (or vision sensor i), from the plane P^j is represented by P_r^j, P_n^j (see Fig. 8.7):

$$d_i^j(k) = P_r^j - x_i(k)cos(P_n^j) - y_i(k)sin(P_n^j) \tag{8.60}$$

Assuming a constant sampling period $\Delta t_k = T$ the measurement equation is $z(k+1) = \gamma(x(k)) + v(k)$, where $z(k)$ is the vector containing GPS (or vision sensor) and IMU measures and $v(k)$ is a white noise sequence $\sim N(0, R(kT))$.

By definition of the measurement vector one has that the output function is $\gamma(x(k)) = [x(k), y(k), \theta(k), d(k)]^T$. The UAV state is $[x(k), y(k), \theta(k)]^T$ and the control input is denoted by $U(k) = [v(k), \omega(k)]^T$. To obtain the Extended Kalman Filter (EKF), the kinematic model of the UAV is linearized about the estimates $\hat{x}(k)$ and $\hat{x}^-(k)$ the control input $U(k-1)$ is applied.

The *measurement update* of the EKF is

$$\begin{aligned}
&K(k) = P^-(k)J_\gamma^T(\hat{x}^-(k))[J_\gamma(\hat{x}^-(k))P^-(k)J_\gamma^T(\hat{x}^-(k)) + R(k)]^{-1}\\
&\hat{x}(k) = \hat{x}^-(k) + K(k)[z(k) - \gamma(\hat{x}^-(k))]\\
&P(k) = P^-(k) - K(k)J_\gamma^T P^-(k)
\end{aligned}$$

The *time update* of the EKF is

$$\begin{aligned}
&P^-(k+1) = J_\phi(\hat{x}(k))P(k)J_\phi^T(\hat{x}(k)) + Q(k)\\
&\hat{x}^-(k+1) = \phi(\hat{x}(k)) + L(k)U(k)
\end{aligned}$$

$$\text{where } L(k) = \begin{pmatrix} Tcos(\theta(k)) & 0 \\ Tsin(\theta(k)) & 0 \\ 0 & T \end{pmatrix} \text{ and } J_\phi(\hat{x}(k)) = \begin{pmatrix} 1 & 0 & -v(k)sin(\theta)T \\ 0 & 1 & -v(k)cos(\theta)T \\ 0 & 0 & 1 \end{pmatrix} \tag{8.61}$$

while $Q(k) = diag[\sigma^2(k), \sigma^2(k), \sigma^2(k)]$, with $\sigma^2(k)$ chosen to be 10^{-3} and $\phi(\hat{x}(k)) = [\hat{x}(k), \hat{y}(k), \hat{\theta}(k)]^T$, $\gamma(\hat{x}(k)) = [, \hat{x}(k), \hat{y}(k), \hat{\theta}(k), d(k)]^T$, i.e.

$$\gamma(\hat{x}(k)) = \begin{pmatrix} \hat{x}(k) \\ \hat{y}(k) \\ \hat{\theta}(k) \\ P_r^j - x_i(k))cos(P_n^j) - y_i(k)sin(P_n^j) \end{pmatrix} \tag{8.62}$$

In the calculation of the observation equation Jacobian one gets

$$J_\gamma^T(\hat{x}^-(k)) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ -cos(P_n^j) & -sin(P_n^j) & \{x_i^{'}cos(\theta - P_n^j) - y_i^{'}sin(\theta - P_n^j)\} \end{pmatrix} \tag{8.63}$$

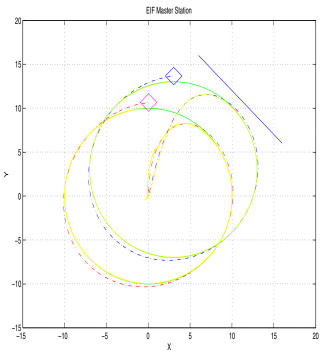
The UAV is steered by a dynamic feedback linearization control algorithm which is based the flatness-based control analyzed in Chapter 3.

$$\begin{array}{c} u_1 = \ddot{x}_d + K_{p_1}(x_d - x) + K_{d_1}(\dot{x}_d - \dot{x}) \\ u_2 = \ddot{y}_d + K_{p_2}(y_d - y) + K_{d_2}(\dot{y}_d - \dot{y}) \\ \dot{\xi} = u_1 cos(\theta) + u_2 sin(\theta) \\ v = \xi, \ \ \omega = \frac{u_2 cos(\theta) - u_1 sin(\theta)}{\xi} \end{array} \tag{8.64}$$

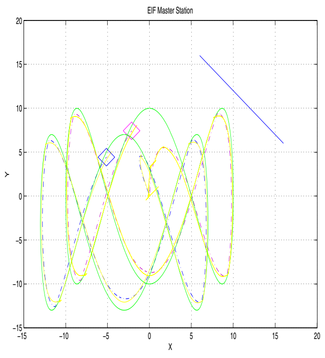
Under the control law of Eq. (8.64) the dynamics of the tracking error finally becomes

$$\begin{array}{c} \ddot{e}_x + K_{d_1}\dot{e}_x + K_{p_1}e_x = 0 \\ \ddot{e}_y + K_{d_2}\dot{e}_x + K_{p_2}e_y = 0 \end{array} \tag{8.65}$$

where $e_x = x - xd$ and $e_y = y - y_d$. The proportional-derivative (PD) gains are chosen as K_{p_i} and K_{d_i}, for $i = 1, 2$.



(a)



(b)

Fig. 8.8 Autonomous navigation of the multi-UAV system when the UAVs state vector is estimated with the use of the Extended Information Filter (a) tracking of circular reference trajectory (b) tracking of a curve-shaped reference trajectory.

Results on the performance of the Extended Information Filter in estimating the state vectors of multiple UAVs when observed by distributed processing units is given in Fig. 8.8. Using distributed EKFs and fusion through the Extended Information Filter is more robust comparing to the centralized EKF since (i) if a local processing unit is subject to a fault then state estimation is still possible and can be used for accurate localization of the UAV, as well as for tracking of desirable flight paths, (ii) communication overhead remains low even in the case of a large number of distributed measurement units, because the greatest part of state estimation is performed locally and only information matrices and state vectors are communicated between the local processing units, (iii) the aggregation performed on the local EKF also compensates for deviations in state estimates of local filters (which can be due to linearization errors).

The simulation experiments were repeated using the Unscented Information Filter in the place of the Extended Information Filter. The UIF-based state estimation of the UAVs was again used by a flatness-based controller which made the UAVs track desirable trajectories. The simulation results show that the Unscented Information Filter is a reliable method for performing fusion of distributed state estimates (see Fig. 8.9). Moreover, unlike the EIF, the UIF is not based on the calculation of Jacobians and avoids cumulative linearization errors which are due to Taylor expansion used in the local EKFs computation.

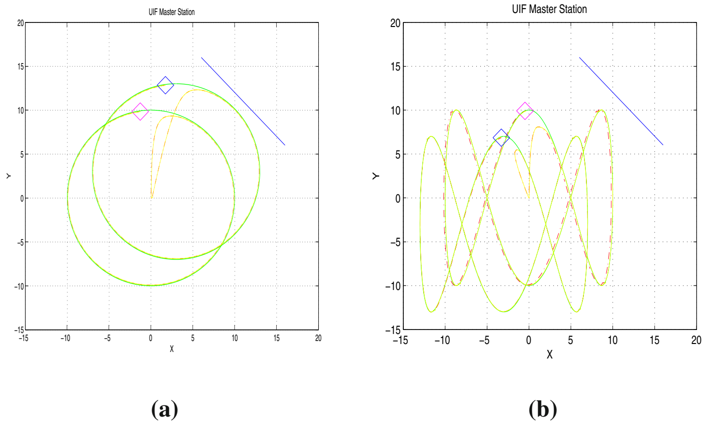


Fig. 8.9 Autonomous navigation of the multi-UAV system when the UAVs state vector is estimated with the use of the Unscented Information Filter (a) tracking of circular reference trajectory (b) tracking of a curve-shaped reference trajectory.

8.5.2 Multi-UAV Control with Distributed Particle Filtering

Details on the implementation of the local particle filters are given first. Each local particle filter provides an estimation of the UAVs state vector using sensor fusion. The UAV model described in Eq. (12.2), and the control law given in Eq. (12.3) are used again.

The *measurement update* of the PF is

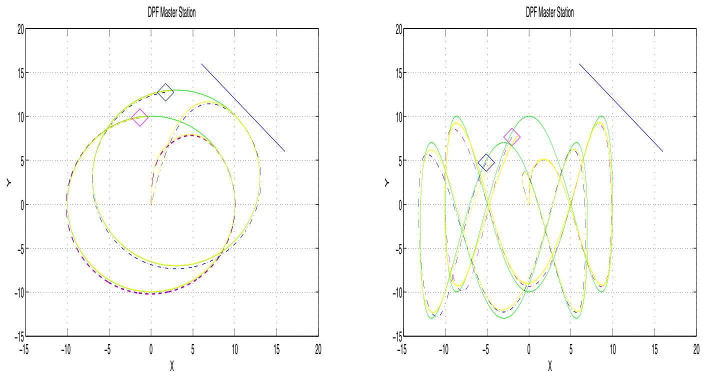
$$p(x(k)|Z) = \Sigma_{i=1}^{N} w_k^i \delta_{\xi_{k-}^i}(x(k))$$
$$\text{with } w_k^i = \frac{w_{k-}^i p(z(k)|x(k)=\xi_{k-}^i)}{\Sigma_{j=1}^{N} w_k^j p(z(k)|x(k)=\xi_{k-}^j)}$$

where the measurement equation is given by $\hat{z}(k) = z(k) + v(k)$. The measurements vector for the considered UAV model is $z(k) = [x(k), y(k), \theta(k), d(k)]^T$, and $v(k)$ is the measurement noise.

The *time update* of the PF is

$$p(x(k+1)|Z) = \sum_{i=1}^{N} w_k^i \delta_{\xi_k^i}(x(k))$$
$$\text{where } \xi_k^i \sim p(x(k+1)|x(k) = \xi_{k^-}^i)$$

and the state equation is $\hat{x}^- = \phi(x(k)) + L(k)U(k)$, where $\phi(x(k))$, $L(k)$, and $U(k)$ are defined in subsection 8.5.1. At each run of the time update of the PF, the state vector estimation $\hat{x}^-(k+1)$ is calculated N times, starting each time from a different value of the state vector ξ_k^i. Although the Distributed Particle Filter can function under any noise distribution in the simulation experiments the measurement noise was assumed to be Gaussian. The obtained results are given in Fig. 8.10.



(a) **(b)**

Fig. 8.10 Autonomous navigation of the multi-UAV system when the UAVs state vector is estimated with the use of the Distributed Particle Filter (a) tracking of circular reference trajectory (b) tracking of a curve-shaped reference trajectory.

In the simulation experiments it was observed that the Distributed Particle Filter, for $N = 1000$ particles, succeeded accurate state estimation (small variance) and consequently enabled accurate tracking of the desirable trajectories by the UAVs. An advantage of the DPF and UIF over the EIF is due to the fact that the local EKFs that constitute the EIF introduce cumulative errors due to the EKF linearization assumption (truncation of higher order terms in the Taylor expansion of Eq. (6.14) and Eq. (6.16)). Comparing to the Extended Information Filter and the Unscented Information Filter, the Distributed Particle Filter demands more computation resources and its computation cycle is longer. However, the computation cycle of PF can be drastically reduced on a computing machine with a fast processor or with parallel processors. Other significant issues that should be taken into account in the design of the Distributed Particle Filter are the consistency of the fusion performed between the probability density functions of the local filters and the communication overhead between the local filters.

The simulation results presented in Fig. 8.10 show the efficiency of the Distributed Particle Filtering in providing accurate localization for the multi-UAV system, as well as for implementing state estimation-based control schemes. The advantages of using Distributed Particle Filtering are summarized as follows: (i) there is robust state estimation which is not constrained by the assumption of Gaussian noises. The fusion performed between the local probability density functions enables to remove outlier particles thus resulting in an aggregate state distribution that confines with accuracy the real state vector of each UAV. If a local processing unit (local filter) fails, the reliability of the aggregate state estimation will be preserved (ii) computation load can be better managed comparing to a centralized particle filtering architecture. The greatest part of the necessary computations is performed at the local filters. Moreover the advantage of communicating state posteriors over raw observations is bandwidth efficiency, which is particularly useful for control over a wireless sensor network.

Chapter 9
Fault Detection and Isolation for Industrial Systems

Abstract. The chapter analyzes a fault detection and isolation approach for efficient condition monitoring of industrial systems. As shown, two main issues in statistical methods for fault diagnosis are residuals generation and fault threshold selection. For residuals generation an accurate model of the system in the fault-free condition is needed. Such models can be obtained through nonlinear identification techniques or through nonlinear state estimation and filtering methods. On the other hand the fault threshold should enable both diagnosis of incipient faults and minimization of the false alarms rate.

9.1 Fault Diagnosis with Statistical Methods

9.1.1 Residual Generation through Nonlinear System Modelling

A fault is an undetermined deviation of at least one characteristic property (feature) of the system from its acceptable standard condition. Frequently, faults are difficult to detect particularly if they are small or hidden. Faults may develop abruptly (stepwise) or incipiently (driftwise) (see Fig. 9.1). The task of fault diagnosis consists in determining the type, size and location of the most possible fault in an industrial or robotic system, as well as its time of detection. Fault diagnosis consists of (i) *fault detection* which is the decision on the existence of the industrial or robotic system in an abnormal state, and (ii) *fault isolation* i.e. finding the parameter or component of the industrial or robotic system that is responsible for the abnormal functioning.

Model-based methods of fault detection use the relation between several measured variables to extract information on possible changes caused by faults. The relations between the input and output signals are represented by a mathematical model. Such models can be obtained through nonlinear identification techniques or through nonlinear state estimation and filtering methods. Decisions on the existence and cause of a fault are usually based on the comparison between the observed system outputs with their nominal values. The differences between the nominal and the observed system features are also called *residuals*.

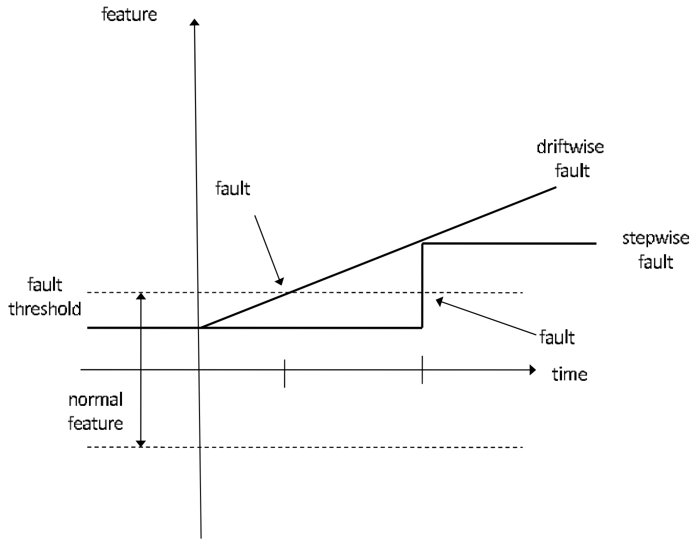


Fig. 9.1 Evolution of faults in industrial and robotic systems: driftwise and stepwise faults

Many industrial or robotic systems (for instance [246]) can be written in terms of differential equations and thus provide us with a *physical model*. However, the physical model is not always available or it can be too complex, or knowledge about it maybe incomplete. In that case, the exact model of the system can be represented in the form of a black-box model (i.e. a neural network, a wavelets network or a fuzzy rule base, trained with the use of input and output measurements of the physical system). Nevertheless, it should be noted that there is no one to one correspondence between the parameters of the physical model and the parameters of a black-box model. Thus, if the Fault Detection and isolation (FDI) method indicates a change in a parameter of the black-box model this cannot be mapped clearly to a failure of a certain component of the physical system. The physical model is usually of higher dimension (parameter vector ϕ) than the black-box model (parameter vector θ) and thus the transformation of the physical model into the black-box model implies dimension reduction [22],[474]. In such cases, in place of the psysical model it is possible to use the so-called *exact model*. In simple words the *exact model* model is the model that describes the fault-free system up to random noise, or the model within the model set that approximates best the system (see Fig. 9.2) [344].

Neuro-fuzzy networks have shown a good performance in problems of modeling and identification for nonlinear dynamical systems. Fig. 9.4 shows the architecture of a feed-forward neural network which can estimate the output $y(k)$ of a dynamical system, when receiving as input the regressor vector that contains past values of the output and past values of the systems inputs. In that case, the input vector of the NN is the regressor vector, $z(k) = [y(k-1),\cdots,y(k-n),u(k),u(k-1),\cdots,u(k-m)]$.

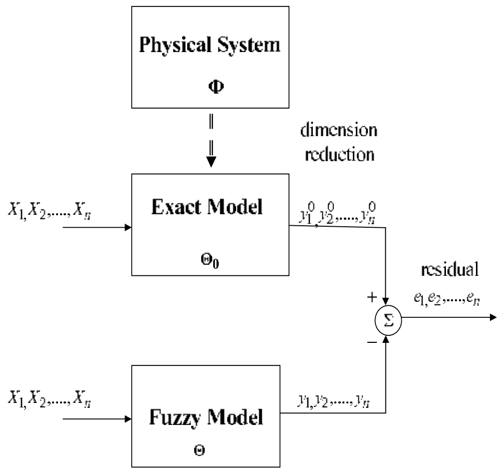


Fig. 9.2 Residual between the neuro-fuzzy model and the exact model (the exact model is the neuro-fuzzy model that is extracted from input/output data of the physical system when the latter is in normal condition)

Identification of the unknown dynamical system can be also carried out with the use of a diagonal recurrent neural network such as the one shown in Fig. 1.6.

9.1.2 Determination of the Nonlinear Model's Structure

9.1.2.1 Determination of the Number of Fuzzy Rules

When constructing a neural-fuzzy model one has to decide on the optimal number and the type of the fuzzy rules that will be used in this model. Two common approaches for the selection of the number of the fuzzy rules are the following [56]: (i) The input space partition, (ii) The input dimension (grid) partition (see Fig. 9.3).

Input space partitioning can be the result of a clustering procedure, as shown in Fig. 9.3(a). Two popular clustering algorithms are the fuzzy c-means (FCM) algorithm and the Gustafson-Kessel (GK) algorithm. Furthermore, clustering can be the outcome of an optimization procedure (selection of the centers and spreads of the fuzzy sets from numerical data through a nonlinear least-squares approach). The problem with all clustering methods is that the projections of the obtained partitions on the axes of the input variables x_i may overlap, which implies loss of the interpretability of the fuzzy rules. In input space partition the membership of the i-th rule can be for example a Gaussian membership function, i.e.

$$\mu_{r^l}(x) = e^{-\sum_{i=1}^{n}\left(\frac{(x_i - c_i^l)}{v_i^l}\right)^2} \tag{9.1}$$

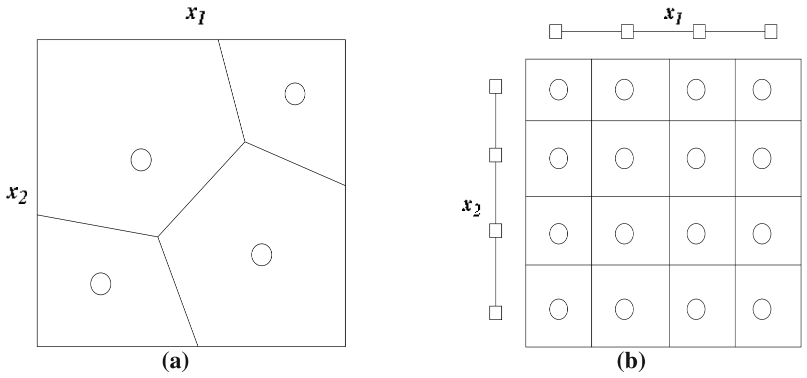


Fig. 9.3 (a) Input space partition (b) Input dimension (grid) partition.

where n is the number of the fuzzy membership functions of the i-th rule , while c_i^l denote the i-th center (spread) in the antecedent part of the l-th rule.

Input dimension partitioning can be the result of the so-called "grid" approach. If the partition of the patterns space is carried out following the "grid" approach the interpretability of the fuzzy rules is maintained. However an increased number of fuzzy rules may be obtained and some of the rules maybe practically inactive because they will be associated with empty areas of the patterns space. Therefore, refinement of the fuzzy rule base has to be carried out at a second stage. In input dimension partition, dimension x_i is split into consecutive segments, each one described by a membership function of the form: $\mu_{R^l}(x_i) = e^{(\frac{-x_i - c_i^l}{v_i^l})^2}$, where c_i^l denotes the center of the i-th fuzzy set in the l-th fuzzy rule, and v_i^l is the associated variance.

When the input variables x_i are split into equal membership functions the input space is covered by the grid shown in Fig. 9.3(b). This is the uniform grid partition. It should be noted that the input space partition is equivalent to that of an input dimension (grid) partition. The reason is that Eq. (9.1) can be rewritten as $\mu_{R^l}(x) = e^{-(x-c)^T \Lambda^{-2}(x-c)}$, where $x = [x_1, x_2, \cdots, x_n]^T$, $c^l = [c_1^l, c_2^l, \cdots, c_n^l]^T$ and $\Lambda = diag(v_1^l, v_2^l, \cdots, v_n^l)$. For different Λ^l's, multi-dimensional Gaussian membership functions with different spreads are derived, thus defining a non-uniform grid in the input space. To get a partition similar to the one depicted in Fig. 9.3(a), the spread matrix Λ has to be non-diagonal.

In *input dimension (grid) partition*, the data space is split into $N_d = \Pi_{i=1}^n p_i$, where p_i is the number of partitions of each one of the n input dimensions. This is equal to the number of centers (spreads) that have to be tuned. If *input space partition* is applied, the number of centers to be tuned is $N_s = n{\cdot}p_s$, where n is again the number of dimensions and p_s is the number of the partitions of the input space. Usually $N_s < N_d$. The input space partition can succeed the same approximation accuracy with input dimension partition but with less tunable parameters. The drawback of input space partition is that it may result in redundant rule bases which are difficult to be linguistically interpreted [125].

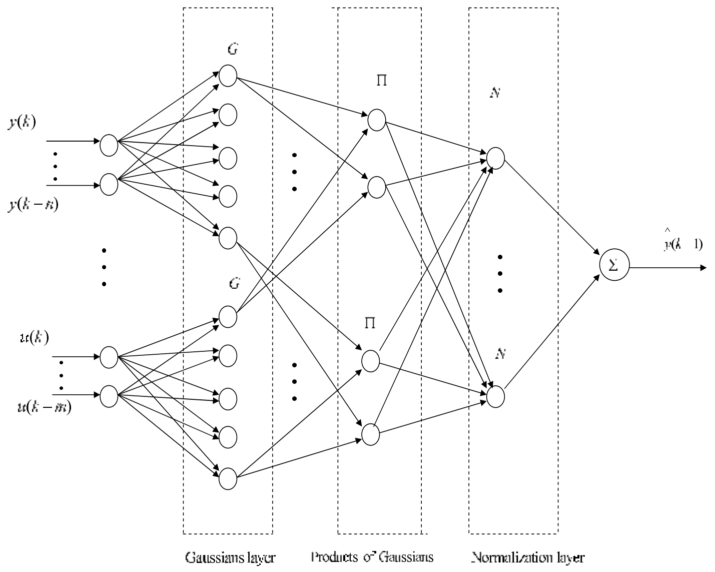


Fig. 9.4 Neural-fuzzy approximator

9.1.2.2 Determination of the Type of Fuzzy Rules

Once the number of rules has been determined one has to select the type of the rules that will be employed by the model. There are several types of fuzzy rules such as linguistic (Mamdani's) rules, relational rules and rules of the Takagi-Sugeno type [401],[447]. Linguistic fuzzy rules of the Mamdani type can be found in early versions of fuzzy models and are obtained from experts' knowledge. Moreover, the construction of fuzzy relational matrices is also a matter of human knowledge. In both approaches the question that arises is how reliable can be the numerical variables contained in the fuzzy models. These approaches seem at most as reliable as the expert-system method of asking an expert to give condition-action rules with numerical uncertainty weights. On the other hand the neurofuzzy models studied in this chapter are extracted from numerical data. The rules are obtained using optimization criteria thus assuring accuracy and objectiveness of the obtained rule base. The more generic type of fuzzy rules that numerically extracted models could contain is the Takagi-Sugeno one.

In the sequel fuzzy rules of the Takagi-Sugeno type will be considered. These have the form:

$$\begin{aligned} R_l \ : \ & IF\, x_1 \ is\ A_1^l AND\ x_2\ is\ A_2^l\ AND\ \cdots AND\ x_n\ is\ A_n^l \\ & THEN\, \bar{y}^l = \sum_{i=1}^{n} w_i^l x_i + b^l \ \ l = 1,2,\cdots,L \end{aligned} \tag{9.2}$$

where R^l is the l-th rule, $x = [x_1, x_2, \cdots, x_n]^T$ is the input (antecedent) variable, $\bar{y}^l$ is the output (consequent) variable, and w_i^l, b^l are the parameters of the local linear models. The above model is a Takagi-Sugeno model of order 1. Setting $w_i^l = 0$ results in the zero order Takagi-Sugeno model [125]. The output of the Takagi-Sugeno model is given by the weighted average of the rules consequents (Fig. 9.4):

$$\hat{y} = \frac{\sum_{l=1}^{L} \bar{y}^l \prod_{i=1}^{n} \mu_{A_i^l}(x_i)}{\sum_{l=1}^{L} \prod_{i=1}^{n} \mu_{A_i^l}(x_i)} \tag{9.3}$$

where $\mu_{A_i^l}(x_i): R \rightarrow [0,1]$ is the membership function of the fuzzy set A_i^l in the antecedent part of the rule R^l. In the case of a zero order TS system the output of the l-th local model is $\bar{y}^l = b^l$, while in the case of a first order TS system the output of the l-th local model is given by $\bar{y}^l = \sum_{l=1}^{L} w_i^l x_i + b^l$.

If the numerically extracted neural-fuzzy model does not approximate efficiently the monitored physical system then a refinement of the partitioning of the patterns space may be required [160],[281].

9.1.3 Stages of Nonlinear Systems Modeling

The individual steps of data-driven neuro-fuzzy modeling for nonlinear identification are discussed in [56],[159],[165],[375],[376]. These stages are demonstrated in Fig. 9.5.

1. *Initialization of the neural-fuzzy model*: As explained in Subsection 9.1.2 the creation of the initial neural-fuzzy model comprises two elements: (i) partition of the data space and formation of data clusters [376],[375]. When clustering is

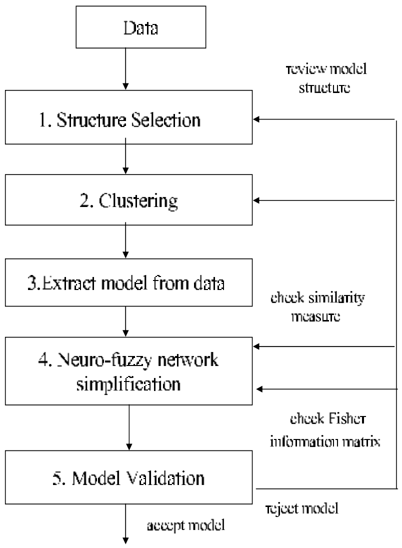


Fig. 9.5 General scheme of data-driven neural-fuzzy modelling

complete a collection of L clusters $C = (c_1, c_2, \cdots, c_L)$ is produced. Each cluster is associated with a fuzzy rule, (ii) Selection of the type of the fuzzy rules. As already mentioned, the fuzzy model may consist of linguistic (Mamdani) rules, relational rules or fuzzy rules of the Takagi-Sugeno type.

2. *Parameter Optimization*: To improve the model performance and to achieve higher modelling accuracy, the parameters of the initial model should be optimized against a certain performance index, such as the RMSE. The centers and spreads of the membership functions and the weights of the local models are exctracted through the recursive solution of a nonlinear least squares problem [453]. This can be carried out with the use, for instance, of Extended Kalman Filtering (EKF) [448], the Gauss-Newton or the Levenberg-Marquardt method [33],[34],[292],[382],[383]. If only the linear weights of the neural-fuzzy system are of interest then the problem reduces to linear least squares and can be efficiently solved by e.g. applying the LMS algorithm.

3. *Model Simplification*: The initial rule-base obtained from data is often redundant and unnecessarily complex since it is generated from unconstrained optimization [85],[147]. The absence of constraints on the nonlinear parameters (centers and spreads of the Gaussian basis functions) may result in a redundant (non interpretable) fuzzy rule base. The complexity is reduced when similarity analysis is used to identify fuzzy sets that represent conflicting or redundant concepts. By merging the redundant fuzzy sets, a more comprehensible fuzzy rule base can be derived [374],[376],[463].

4. *Neural-fuzzy model validation*: Using a validation technique the final model is either accepted as suitable for describing the real system or it is rejected and a new training procedure is evoked. Model validation helps to decide which parameters of the neural-fuzzy model need tuning thus avoiding global retraining [214],[344].

9.2 Fault Threshold Selection with the Generalized Likelihood Ratio

9.2.1 The Local Statistical Approach to Fault Diagnosis

Apart from residuals generation through modeling of the monitored system's function in the fault-free condition, another significant problem for fault detection and isolation is optimal selection of the fault threshold. The value of the fault threshold should be that small faults can be detected at their early stages and also the rate of false alarms is minimized. A solution to the problem of fault threshold selection in fault diagnosis systems has been provided by the *Local Statistical Approach* to change detection [16],[474]. The Local Statistical Approach has been successfully applied to several FDI problems, such as [19],[20],[28]: (i) Vibration monitoring in linear dynamic systems: this application concerns power plants (rotating machines, core and pipes of nuclear power plants), civil engineering (large buildings subject to earthquakes, bridges, dams and offshore structures), aeronautics (wings and other structures subject to strength), automobile and train transportation, (ii) Nonlinear static systems, such as the gas turbine: this example refers to the detection of faults in the combustion chamber of gas turbines, which are widely used as industrial motors, in electric power generators, and aircraft engines, (iii) Nonlinear dynamic systems, such as the catalyst used in vehicles: the latter application concerns the detection of faults in the catalytic converter and the oxygen sensors of the automobiles [17],[21],[470].

Based on a small parametric disturbance assumption, the Local Statistical Approach to fault diagnosis aims at transforming complex detection problems concerning a parameterized stochastic process into the problem of monitoring the mean of a Gaussian vector. The local statistical approach consists of two stages : i) the global test which indicates the existence of a change in some parameters of the fuzzy model, ii) the diagnostics tests (sensitivity or min-max) which isolates the parameter affected by the change. The method's stages are analyzed next, following closely the method presented in [16],[19].

9.2.2 Fault Detection with the Local Statistical Approach

First, the dynamics of the monitored system in the fault-free condition is learned by a non-linear model, as described in subsection 9.1.3. At each time instant the model's output y_i^m is compared to the real condition of the system y_i. The difference between the real condition of the power system and the output of the model is the previously defined residual. The statistical processing of a sufficiently large number of residuals provides an index-variable that is compared against a fault threshold and which can give early indication about deviation of the monitored system from the normal operating conditions. Therefore alarm launching can be activated at the early stages of a fault's occurence, and restoration measures can be taken. Under certain conditions (detectability of changes) the proposed FDI method enables also fault isolation, i.e. to identify the source of fault within the monitored system.

The partial derivative of the residual square is:

$$H(\theta, y_i) = \frac{\partial e_i^2}{\partial \theta} = e_i \frac{\partial \hat{y}_i}{\partial \theta} \tag{9.4}$$

where θ is the vector of model's parameters. The vector H having as elements the above $H(\theta, y_i)$ is called primary residual. In the case of neuro-fuzzy models the gradient of the output with respect to the consequent parameters w_i^l is given by

$$\frac{\partial \hat{y}}{\partial w_i^l} = \frac{x_i \mu_{R^l}(x)}{\sum_{l=1}^{L} \mu_{R^l}(x)} \tag{9.5}$$

The gradient with respect to the center c_i^l is

$$\frac{\partial \hat{y}}{\partial c_i^l} = \sum_{l=1}^{L} \frac{\overline{y}^l \frac{2(x_i - c_i^l)}{v_i^l} \mu_{R^l}(x_i)[\sum_{j=1}^{L} \mu_{R^j}(x_i) - \mu_{R^l}(x_i)]}{[\sum_{l=1}^{L} \mu_{R^l}(x_i)]^2} \tag{9.6}$$

The gradient with respect to the spread v_i^l is

$$\frac{\partial \hat{y}}{\partial v_i^l} = \sum_{l=1}^{L} \frac{\overline{y}^l \frac{2(x_i - c_i^l)^2}{{v_i^l}^3} \mu_{R^l}(x_i)[\sum_{j=1}^{L} \mu_{R^j}(x_i) - \mu_{R^l}(x_i)]}{[\sum_{l=1}^{L} \mu_{R^l}(x_i)]^2} \tag{9.7}$$

Next, having calculated the partial derivatives of Eq.(9.5), Eq.(9.6) and Eq.(9.7), the rows of the Jacobian matrix J are found by

$$J(\theta_0, y_k) = \left. \frac{\partial \hat{y}_k(\theta)}{\partial \theta} \right|_{\theta = \theta_0} \tag{9.8}$$

The problem of change detection with the χ^2 test consists of monitoring a change in the mean of the Gaussian variable which for the one-dimensional parameter vector θ is formulated as

$$X = \frac{1}{\sqrt{N}} = \sum_{i=1}^{N} e_k \frac{\partial \hat{y}_k}{\partial \theta} \sim N(\mu, \sigma^2) \tag{9.9}$$

where $\hat{y}_k$ is the output of the neural model generated by the input pattern x_k, e_k is the associated residual and θ is the vector of the model's parameters. For a multivariable parameter vector θ should hold $X \sim (M\eta, \Sigma)$. In order to decide if the power system is in fault-free operating conditions, a given set of data of N measurements, let θ_* be the value of the parameters vector μ minimizing the RMSE. The notation is introduced only for the convenience of problem formulation, and its actual value does not need to be known. Then the fault detection problem amounts to make a decision between the two hypotheses:

$$\begin{aligned} H_0 &: \ \theta_* = \theta_0 \\ H_1 &: \ \theta_* = \theta_0 + \tfrac{1}{\sqrt{N}} \delta\theta \end{aligned} \tag{9.10}$$

where $\delta\theta \neq 0$. It is known from the central limit theorem that for a large data sample, the normalized residual given by Eq.(9.9) asymptotically follows a Gaussian distribution when $N \rightarrow \infty$ [16,19]. More specifically, the hypothesis that has to be tested is:

$$\begin{aligned} H_0 &: X \sim N(0, S) \\ H_1 &: X \sim N(M\eta, S) \end{aligned}$$

where M is the sensitivity matrix (see Eq. (9.11)), η is the parameters' vector and S is the convariance matrix (see Eq. (9.12)). The product $M\eta$ denotes the new center of the monitored Gaussian variable X, after a change on the system's parameter θ. The sensitivity matrix M of $\frac{1}{\sqrt{N}}X$ is defined as the mean value of the partial derivative with respect to θ of the primary residual defined in Eq. 9.4, i.e. $E\{\frac{\partial}{\partial\theta}H(\theta,y_k)\}$ and is approximated by [8]:

$$M(\theta_0)\simeq\frac{\partial}{\partial\theta}\frac{1}{N}\sum_{k=1}^{N}H(\theta_0,y_k)\simeq\frac{1}{N}J^TJ \tag{9.11}$$

The covariance matrix S is defined as $E\{H(\theta,y_k)H^T(\theta,y_{k+m})\}$, $m=0,\pm1,\cdots$ and is approximated by [19]:

$$\begin{aligned}S=&\simeq\sum_{k=1}^{N}[H(\theta_0,y_k)H^T(\theta_0,y_k)]+\\&+\sum_{m=1}^{I}\frac{1}{N-m}\sum_{k=1}^{N-m}[H(\theta_0,y_k)H^T(\theta_0,y_{k+m})+H(\theta_0,y_{k+m})H^T(\theta_0,y_k)]\end{aligned} \tag{9.12}$$

where an acceptable value for I is 3. The decision tool is the likelihood ratio $s(X)=ln\frac{p_{\theta_1(x)}}{p_{\theta_0(x)}}$, where $p_{\theta_1}(X)=e^{[X-\mu(X)]^TS^{-1}[X-\mu(X)]}$ and $p_{\theta_0}(X)=e^{X^TS^{-1}X}$. The center of the Gaussian distribution of the changed system is denoted as $\mu(X)=M\eta$ where η is the parameters vector. The *Generalized Likelihood Ratio* (GLR) is calculated by maximizing the likelihood ratio with respect to η [19]. This means that the most likely case of parameter change is taken into account. This gives the global χ^2 test t, defined as:

$$t=X^TS^{-1}M(M^TS^{-1}M)^{-1}M^TS^{-1}X \tag{9.13}$$

Since X asymptotically follows a Gaussian distribution, the statistics defined in Eq. (9.13) follows a χ^2 distribution with n degrees of freedom. Mapping the change detection problem to this χ^2 distribution enables the choice of the change threshold [367]. Assume that the desired probability of false alarm is α then the change threshold λ should be chosen from the relation

$$\int_{\lambda}^{\infty}\chi_n^2(s)ds=\alpha \tag{9.14}$$

where $\chi_n^2(s)$ is the probability density function (p.d.f.) of a variable that follows the χ^2 distribution with n degrees of freedom.

9.2.3 Fault Isolation with the Local Statistical Approach

9.2.3.1 The Sensitivity Test

Fault isolation is needed to identify the source of faults in the monitored system. A first approach to change isolation is to focus only on a subset of the parameters while considering that the rest of the parameters remain unchanged [19]. The parameters vector η can be written as $\eta=[\phi,\psi]^T$, where ϕ contains those parameters to be subject to the isolation test ,while ψ contains those parameters to be excluded from the isolation test. M_ϕ contains the columns of the sensitivity matrix M which are associated with the parameters subject to the isolation test. Similarly M_ψ contains the columns of M that are associated with the parameters to be excluded from the sensitivity test.

Assume that among the parameters η, it is only the subset ϕ that is suspected to have undergone a change. Thus η is restricted to $\eta = [\phi, 0]^T$. The associated columns of the sensitivity matrix are given by M_ϕ and the mean of the Gaussian to be monitored is $\mu = M_\phi \phi$, i.e.

$$\mu = MA\phi, \ \ A = [0, I]^T \tag{9.15}$$

Matrix A is used to select the parameters that will be subject to the fault isolation test. The rows of A correspond to the total set of parameters while the columns of A correspond only to the parameters selected for the test. Thus the fault diagnosis (χ^2) test of Eq. (9.13) can be restated as:

$$t_\phi = X^T S^{-1} M_\phi (M_\phi^T S^{-1} M_\phi)^{-1} M_\phi^T S^{-1} X \tag{9.16}$$

9.2.3.2 The Min-Max Test

In this approach the aim is to find a statistic that will be able to detect a change on the part ϕ of the parameters vector η and which will be robust to a change in the non observed part ψ [19]. Assume the vector partition $\eta = [\phi, \psi]^T$. The following notation is used:

$$M^T S^{-1} M = \begin{pmatrix} I_{\varphi\varphi} & I_{\varphi\psi} \\ I_{\psi\varphi} & I_{\psi\psi} \end{pmatrix} \tag{9.17}$$

$$\gamma = \begin{pmatrix} \varphi \\ \psi \end{pmatrix}^T \cdot \begin{pmatrix} I_{\varphi\varphi} & I_{\varphi\psi} \\ I_{\psi\varphi} & I_{\psi\psi} \end{pmatrix} \cdot \begin{pmatrix} \varphi \\ \psi \end{pmatrix} \tag{9.18}$$

where S is the previously defined covariance matrix. The min-max test aims to minimize the non-centrality parameter γ with respect to the parameters that are not suspected for change. It is noted that matrix I is also known as Fisher Information matrix and provides indices on the diagnosability of nonlinear models [109],[110],[111].

The minimum of γ with respect to ψ is given for:

$$\psi^* = \arg\min_{\psi} \gamma = \varphi^T (I_{\varphi\varphi} - I_{\varphi\psi} I_{\psi\psi}^{-1} I_{\psi\varphi})\varphi \tag{9.19}$$

and is found to be

$$\begin{aligned} \gamma^* &= \min_{\psi} \gamma = \varphi^T (I_{\varphi\varphi} - I_{\varphi\psi} I_{\psi\psi}^{-1} \, I_{\psi\varphi})\varphi = \\ &= \begin{pmatrix} \varphi \\ -I_{\psi\psi}^{-1} I_{\psi\varphi}\varphi \end{pmatrix}^T \begin{pmatrix} I_{\varphi\varphi} & I_{\varphi\psi} \\ I_{\psi\varphi} & I_{\psi\psi} \end{pmatrix} \begin{pmatrix} \varphi \\ -I_{\psi\psi}^{-1} I_{\psi\varphi}\varphi \end{pmatrix} \end{aligned} \tag{9.20}$$

which results in

$$\gamma^* = \varphi^T \{[I, -I_{\varphi\psi} I_{\psi\psi}^{-1}] M^T \Sigma^{-1}\} \, \Sigma^{-1} \{\Sigma^{-1} M [I, -I_{\varphi\psi} I_{\psi\psi}^{-1}]\} \varphi \tag{9.21}$$

The following linear transformation of the observations is considered :

$$X_{\phi}^{*} = [I, -I_{\varphi\psi}I_{\psi\psi}^{-1}]M^{T}\Sigma^{-1}X \tag{9.22}$$

The transformed variable X_{ϕ}^{*} follows a Gaussian distribution $N(\mu_{\phi}^{*}, I_{\phi}^{*})$ with mean:

$$\mu_{\varphi}^{*} = I_{\varphi}^{*}\varphi \tag{9.23}$$

and with covariance :

$$I_{\varphi}^{*} = I_{\varphi\varphi} - I_{\varphi\psi}I_{\psi\psi}^{-1}I_{\psi\varphi} \tag{9.24}$$

The min-max test decides between the hypotheses :

$$H_0^{*} : \mu^{*} = 0$$
$$H_1^{*} : \mu^{*} = I_{\varphi}^{*}\varphi$$

and is described by :

$$\tau_{\varphi}^{*} = X_{\varphi}^{*T} I_{\varphi}^{*-1} X_{\varphi}^{*} \tag{9.25}$$

The stages of fault detection and isolation (FDI) with the use of the local statistical approach are summarized in the following table:

Table 9.1 Stages of the local statistical approach for FDI

1. Generate the residuals partial derivative given by Eq.(9.4)
2. Calculate the Jacobian matrix J given by Eq.(9.8)
3. Calculate the sensitivity matrix M given by Eq.(9.11)
4. Calculate the covariance matrix S given by Eq.(9.12)
5. Apply the χ^2 test for change detection of Eq.(9.13)
6. Apply the change isolation tests of Eq. (9.16) or Eq.(9.25)

9.2.4 Fault Threshold for Residuals of Unknown Distribution

9.2.4.1 Residuals Generation Using Kalman Filtering

Apart from neuro-fuzzy models, statistical filters such as the Kalman Filter can be used for estimating the output of the monitored system in the fault-free condition and subsequently for generating residuals [399],[417]. As already analyzed in Chapter 6, the discrete-time Kalman Filter is an optimal state estimator for linear dynamical systems of the form:

$$\begin{aligned} x(k+1) &= \Phi x(k) + L(k)u(k) + w(k) \\ y(k) &= Cx(k) + v(k) \end{aligned} \tag{9.26}$$

In the state model of Eq. (9.26), the state $x(k)$ is a N-vector, $w(k)$ is a N-element process noise vector, Φ is a $N \times N$ real matrix, C is an N-element row vector of

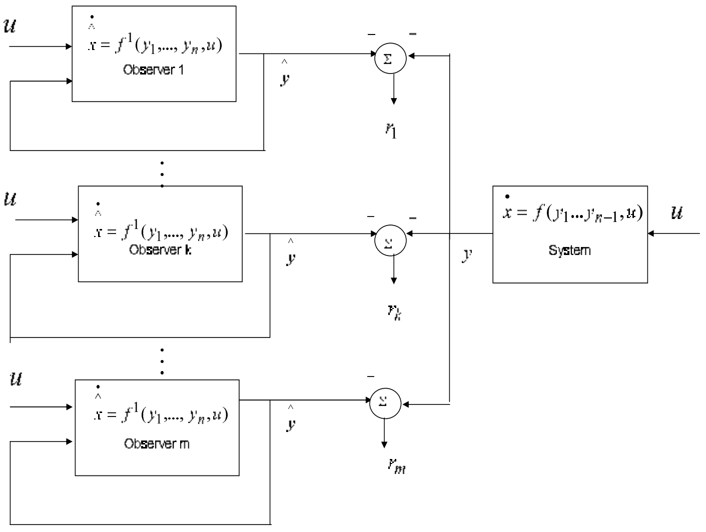

u
$\dot{\hat{x}} = f^1(y_1,\dots,y_n,u)$
Observer 1
$\hat{y}$
Σ
r_1
u
$\dot{\hat{x}} = f^1(y_1,\dots,y_n,u)$
Observer k
$\hat{y}$
r_k
$\dot{x} = f(y_1\dots y_{n-1},u)$
System
y
u
u
$\dot{\hat{x}} = f^1(y_1,\dots,y_n,u)$
Observer m
$\hat{y}$
r_m

Fig. 9.6 Fault detection and isolation based on filters (observers)

real numbers, and $v(k)$ is the measurement noise. It is assumed that $w(k)$ and $v(k)$ are uncorrelated. The process and output noise are Gaussian zero-mean and their covariance matrices are given by: $Q = E[w(i)w^T(j)]$ and $R = E[v(i)v^T(j)]$. The initial value of the state vector $x(0)$, and the initial value of the error covariance matrix $P(0)$ is unknown and an estimation of them is considered, i.e. $\hat{x}(0)$ = a guess of $E[x(0)]$, and $\hat{P}(0)$ = a guess of $Cov[x(0)]$. For the initialization of matrix P one can set $\hat{P}(0) = \lambda I$, with $\lambda > 0$ [175],[341].

According to the analysis given in Chapter 6, the Kalman filter consists of two stages: (i) time update, and (ii) measurement update. The first part employs an estimate $\hat{x}^-(k)$ of the state vector $x(k)$ made before the output measurement $y(k)$ is available (a priori estimate), i.e. when the set of the measurements sequence is given by $Y^- = \{y(1),\cdots,y(k-1)\}$. The second part provides an estimate $\hat{x}(k)$ of $x(k)$ after $y(k)$ has become available (a posteriori estimate), when the set of measurements becomes $Y = \{y(1),\cdots,y(k)\}$. The discrete Kalman filter equations, initially given in Eq. (6.12) and Eq. (6.11), are:

measurement update:

$$\begin{array}{c} K(k) = P^-(k)C^T[C{\cdot}P^-(k)C^T + R]^{-1} \\ \hat{x}(k) = \hat{x}^-(k) + K(k)[z(k) - C\hat{x}^-(k)] \\ P(k) = P^-(k) - K(k)CP^-(k) \end{array} \tag{9.27}$$

time update:

$$\begin{array}{c} P^-(k+1) = \Phi(k)P(k)\Phi^T(k) + Q(k) \\ \hat{x}^-(k+1) = \Phi(k)\hat{x}(k) + L(k)u(k) \end{array} \tag{9.28}$$

Thus, in the case of Gaussian process and measurement noise the Kalman Filter is an optimal model (in terms of estimation's accuracy) of the fault-free functioning of the system and can be used for residuals generation (see Fig. 9.6). Such residuals can be processed by the local statistical approach for detecting the existence and the cause of the fault within the system.

9.2.4.2 Residuals Generation Using Particle Filtering

In [172],[226],[472] the likelihood ratio has been combined with Particle Filter (PF) and a Particle Filtering-based LR method for FDI suitable for nonlinear non-Gaussian systems has been presented. As mentioned, the monitored system is governed by one of the $M+1$ models based on Eq. (4.49) and including faults as additional inputs. The normal operation situation is denoted by $m=0$, while $m=1,\cdots,M$ corresponds to one of the M faulty situations. For each one of the faults m that may take place $m=1,\cdots,M$, a particle filter is used to approximate the posterior probability density function of the state vector of these models. The main concept for FDI is (i) to use the M particle filters to calculate the pdf of the M system models of Eq. (4.49) for $m{\neq}0$, (ii) to generate residuals by comparing the output of the monitored system to the output of the particle filters, (iii) to perform M likelihood-ratio tests using the M generated residuals.

It is assumed that the measurement noise v has the same dimensionality as the measurement y and for each model given the state x_j^m and the measurement y^j the measurement noise v is defined by an observation-error function $v_j^{(m)} = g_j^{(m)}(y_j,x_j^{(m)})$, where $g_j^{(m)}(.,.)$ has a Jacobian defined by $\frac{\partial g_j^{(m)}}{\partial y_j}$. The likelihood ratio for the m-th hypothesized model is

$$S_j^k(m) = \sum_{j=r}^k ln\frac{p(y_j|H_m,Y_{j-1})}{p(y_j|H_0,Y_{j-1})} \tag{9.29}$$

where the likelihood of the observation y_j gives its past values Y_{j-1}, i.e. the probability function $p(y_j|H_m,Y_{j-1})$, $m=0,1,\cdots,M$ is the output estimation based on hypothesis H_m, which is defined by the m-th measurement model and the known statistics of the observation error function $e_j^{(m)} = g_j^{(m)}(y_j,x_j^{(m)})$. If the pdf of $e_j^{(m)}$ is denoted by $q_j^{(m)}(e_j^{(m)})$, then the probability $p(y_j|H_m,Y_{j-1})$ can be expressed in a way analogous to the LR residual

$$s_j^{(m)} = p(y_j|H_m,Y_{j-1}) = q_j^{(m)}(e_j^{(m)})|\frac{\partial e_j^{(m)}}{\partial y_j}| \tag{9.30}$$

where $m=0,1,\cdots,M$ and $x_{j|j-1}^{(m)}$ is the output estimation of the m-th model, given Y_{k-1}. It can be verified that in the Gaussian case, the quantity defined by Eq. (9.30) is just the innovation likelihood, which can be derived from the Kalman Filter equations, based on the m-th model. For the general nonlinear non-Gaussian model there are no analytical means to perform the calculation. However, with the particle filter this quantity can be estimated using the complete pdf information of the predicted state $x_{j|j-1}^{(m)}$ represented by the swarm of particles. Since $x_{j|j-1}^{(m)}\ :\ i=1,\cdots,N$ can be considered as the samples from $p(x_j|H_m,Z_{j-1})$ for the m-th model, the required quantity can be computed via the Monte-Carlo integration as follows:

$$s_j^{(m)}=p(y_j|H_m,Y_{j-1})\simeq\tfrac{1}{N}\textstyle\sum_{i=1}^{N}p(y_j|x_{j|j-1}^{(m)}) \tag{9.31}$$

This means that taking for N particles, the likelihood ratio corresponding to the m-th fault model (filter) is given at time instant i by

$$s_j^{(m)}(i)=\frac{1}{N}\sum_{j=1}^{N}w_j^{(m)}(i) \tag{9.32}$$

where $w_j^{(m)}(i)$ are the unnormalized particle weights, of the m-th particle estimator.

Chapter 10
Application of Fault Diagnosis to Industrial Systems

Abstract. Applications of statistical methods for fault diagnosis are presented. First, the problem of early diagnosis of cascading events in the electric power grid is considered. Residuals are generated with the use of a nonlinear model of the distributed electric power system and the fault threshold is determined with the use of the generalized likelihood ratio assuming that the residuals follow a Gaussian distribution. Next, the problem of fault detection and isolation in electric motors is analyzed. It is proposed to use nonlinear filters for the generation of residuals and to derive a fault threshold from the generalized likelihood ratio without prior knowledge of the residuals statistical distribution.

10.1 Fault Diagnosis of the Electric Power System

10.1.1 Cascading Events in the Electric Power Grid

Modern large-scale power systems have commonly experienced adverse impacts on the system operation and cascading events due to the under-damped low-frequency oscillation. Low-frequency oscillations is a main cause for instability in power systems. Such oscillations are classified into two modes: (i) local modes representing oscillations between one generator and the rest of the power grid or oscillations among several adjacent synchronous power generators, and (ii) inter-area modes representing swings among different power grids interconnected through tie-lines [195],[206]. Monitoring of these oscillations can help to diagnose at its early stages the risk of power system destabilization [173],[365],[404],. To succeed failure diagnosis and to enable restoration action, models of the power system have been developed.

The behavior of the electric power transmission system is usually described by a multi-area multi-machine power system model. Slow and fast modes can be distinguished. For instance slow modes correspond to the weak connection between areas. Early detection of the system's tendency to become unstable permits the activation of automatic voltage regulators which can damp-out inter-area oscillations. Thus cascading events (black-outs) can be prevented. The chapter presents results on modeling of the dynamics of power transmission systems, and on the application of the local statistical approach for the detection of faults at their early stages. Thus, through FDI it is possible to take measures for fault restoration and actions for power system stabilization.

Linear models or NARX representations (nonlinear autoregressive models with exogenous inputs) and NARMAX representations (nonlinear autoregressive moving average models with exogenous inputs) have shown in several cases for deficiencies for modeling the complete dynamics of power systems. On the other hand, neural-fuzzy networks have demonstrated a good performance in problems of modeling the nonlinear dynamics of electric power systems [48],[114],[158],[341],[423],[437] and [13],[87],[253],[406], [407],[427]. In this chapter, modeling of the multi-machine multi-area power system dynamics (in fault-free operating conditions) will be carried out with a neuro-fuzzy network (alternatively, one can use neural networks that are able to capture the multi-frequency characteristics of the power system dynamics, such as wavelet networks, or feed-forward neural networks with Hermite polynomial activation functions [112],[347],[471]). Comparing the measurements from the power system with the output of the neural model the early detection of cascading events (black-outs) is possible. To this end, the statistical fault detection and isolation (FDI) algorithm analyzed in [16] is used. If the deviation exceeds a threshold, which is defined by the aforementioned FDI algorithm, then an alarm can be launched [18],[19],[473]. In several cases fault isolation can be also performed, i.e. the sources of fault in the power transmission system can be also identified. The use of pattern recognition methods for FDI in the electric power system is also possible [161],[349].

Many research efforts aim at finding ways to prevent or mitigate cascading events: study of the cascade model, dynamic decision event tree analysis, wide area backup protection, relay hidden failure analysis, special protection scheme, self-healing system with the aid of multi-agent technology, etc. [52],[62],[88],[237],[404]. The above mentioned techniques are still far from being an established practice in solving the cascading event problem. In general, cascading events can be prevented by in-time operator actions.

Various methods have tried to analyze the oscillations in real-time, such as peak-value detecting [142], oscillation decay time detecting [452], FFT analysis and wavelet transform [143],[144]. However, enough detailed information about the oscillations from real-time measurements cannot be acquired by using these methods. The detailed oscillatory modes can be instead obtained with the stochastic subspace identification method [123] and the Kalman filtering method [197]. However none of them is suitable for online applications as the former is only appropriate for small disturbance responses and the latter requires a high computation time. The Prony algorithm, provided with fairly good performance [365], has been widely used in power systems [9],[132],[148],[464],[467],[476] as it can be used for online monitoring of system behavior and prediction of system instability. Nevertheless, it cannot ensure accurate mode identification and deal with the distorted real-time signals.

Normally there are two stages of a cascading event. First, there is a stage of slowly evolving consecutive events that can be approximated with steady state analysis. Several new disturbances following one another can make the system operating conditions worse. Then, as a consequence of a succession of several major disturbances a fast transient process takes place that results in cascading events, and finally the system collapses. When the total system collapse starts, normally it is too late to stop it. Nevertheless, much can be done during the slow steady state successions at the first stage and early proper control actions at the steady state stage can prevent the possible cascading event [390]. Analysis of blackouts of power systems has shown that these outages are caused by a cascading sequence of events involving line outages, overloading of certain lines in the transmission system, malfunctions of protection systems, power oscillations and voltage problems and finally system separation and collapse. A dynamical system model of the failure phenomena in the power transmission network has been presented in [79] while a probabilistic model of the propagation of failures in a power system has been given in [321].

Apart from low frequency oscillations, some examples of basic patterns of cascading events are: (i) line tripping due to overloading (a line fault can cause power flows to be rerouted, leading to overload of other lines. As a result the overloaded lines may be tripped due to a depressed voltage and /or high current resulting in a low apparent impedance seen by the impedance relay), (ii) generator tripping due to over-excitation (line outages may cause low voltages and high reactive power demand on generators nearby, leading to tripping of these generator units by over-excitation protection equipment), (iii) line tripping due to loss of synchronism (system disturbances such as faults, line outages and generator tripping can cause oscillation in machine rotor angles and system bus voltages leading to a power flow swing. Furthermore, when the angle difference between two buses is sufficiently large due to a power swing, the impedance seen by a relay may trigger line tripping), (iv) generator tripping due to abnormal voltage and frequency system condition (a loss of a generator may affect the system voltage and frequency and sustained low voltage may cause other generator tripping. Furthermore, high voltage on some part of the system can also contribute to generator tripping. Additionally, under-frequency or over-frequency can trigger relays thus provoking generator tripping), (v) under-frequency / voltage load shedding (automatic under-frequency load shedding or under-voltage load-shedding may be triggered when the system voltage or frequency in some areas falls below a pre-specified value) [6],[2],[46],[156],[249], [288],[455].

The relation between the major network problems and the time available to control actions is shown in Table 10.1.

Cascading events can be stopped by a number of complementary and corrective actions, including load rejection, controlled separation, load shedding and low-frequency isolation scheme.

Table 10.1 Relationship between network faults and the time available to control actions

Major network problems	Time available to control action	Blackout prevention by operation actions	Blackout prevention by emergency controls
overloads	second to minutes	partially possible	possible
low transmission voltage	second to minutes	partially possible	possible
low frequency	0.1 seconds to seconds	impossible	possible
loss of synchronism	milliseconds to seconds	impossible	possible

10.1.2 Electric Power Systems Dynamics

To analyze the various stability problems, power system dynamics are usually modeled into the following four time scales: (i) Long-term dynamics (several minutes and slower): Boiler dynamics, daily load cycles, etc, (ii) Mid-term dynamics (1-5 min): Load Tap Changers (LTC), Automatic Generation Control (AGC), thermostat-controlled loads, generator over-excitation limiters, etc, (iii) Transient dynamics (seconds): Generators, Automatic Voltage Regulators (AVR), governors, induction motors, HVDC controllers, etc. (iv) Practically instantaneous (less than a msec): Electromagnetic and network transients, various electronically controlled loads, etc.

A power system can be modeled as a set of nonlinear differential equations and algebraic equations. Usually in control and fault diagnosis problems it is aimed to obtain a model representation of the monitored system in linear state-space form, as knowledge of the system states is required by several established methodologies for detection of fault detection and isolation (FDI). If the dynamic system is described in linear state-space form, i.e. $\dot{x} = Ax + Bu$, then the modes of the power system are the eigenvalues of matrix A. In that case the dynamic behavior of the power system is a linear combination of the modes of its state-space description. Slow and fast modes can be distinguished. For instance slow modes correspond to the weak connection between areas. Faults can be associated with eigenstructure change.

In the more general case, a nonlinear model of the power system can be extracted from input-ouput data which makes possible the representation of the system in input-output form. A nonlinear representation of the nonlinear dynamical system usually takes the form [137]:

$$\begin{aligned} x(t+1) &= g(x(t), u(t)) \\ y(t) &= h(x(t), u(t)) \end{aligned} \tag{10.1}$$

As in all dynamical systems, in electric power systems the property of *stability* is of primary importance [206].

Definition: A system with state vector x is said to be *globally stable* about the equilibrium point 0 if for any initial state vector x_0 (x_0 can be arbitrarily large) it holds that $\forall\ \varepsilon \geq 0,\ \exists \delta(\varepsilon) > 0$ such that if $||x_0|| < \delta$ then $||x(t)|| < \varepsilon$, for all $t > t_0$.

If the above property holds for small $||x_0|| < \delta$ then the system is *locally stable* (or the system is said to be stable in the small), which is the case to be examined after linearization of a nonlinear system about an operating point 0. The system is said to be *asymptotically stable* if $\forall\ \varepsilon \geq 0,\ \exists \delta(\varepsilon) > 0$ such that if $||x_0|| < \delta$ then $\lim_{t \to \infty} ||x(t)|| = 0$.

It should be noted that the dynamics of power generation and transmission systems is actually nonlinear and that power systems are described by nonlinear differential equations. At certain operating points linearization can be performed thus resulting in locally valid linear models. Small signal analysis of power systems assumes that the power system can be described by a linear model. Linear modeling can be sufficient for small signal stability analysis of the power system as well as for performing fault diagnosis in case of small perturbations of the model's parameters. A problem also remains to define the order of the linear models if the associated linear differential equations are unknown. On the other hand, working with the nonlinear model gives more reliable stability analysis of the power system. Furthermore, when the differential equations of the power system are not available and the power system has to be modeled with a black-box representation (using input and output training data), it is better to a develop a nonlinear model since this is applicable to a wide range of operating points. Nonlinear modeling in model-based fault-diagnosis outperforms the use of local linear models which have to be substituted according to the change of the operating point.

10.1.3 The Multi-area Multi-machine Electric Power System

Kundur's multi-area multi-machine power system is considered here. Fig. 10.1 shows Kundur's 2-area 4-machine power system model consisting of two symmetrical areas interconnected by a 220km two-looped weak tie-line. G1-G4 are synchronous generators with rating of 900MVA and 20kV which consist two power areas - Area 1 and Area 2 respectively. The transmission system nominal voltage is 230kV. The local loads of the two areas and the active power transmitted on tie-line are also shown in Fig. 10.1.

A parametric representation of the linearized dynamics of the single-machine power system is given in the form [206]:

$$\begin{pmatrix} \Delta\dot{\omega}_r \\ \Delta\dot{\delta} \\ \Delta\dot{\psi}_{fd} \\ \Delta\dot{\psi}_{1d} \\ \Delta\dot{\psi}_{1q} \\ \Delta\dot{\psi}_{2q} \end{pmatrix} = \begin{pmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} & \alpha_{14} & \alpha_{15} & \alpha_{16} \\ \alpha_{21} & 0 & 0 & 0 & 0 & 0 \\ 0 & \alpha_{32} & \alpha_{33} & \alpha_{34} & \alpha_{35} & \alpha_{36} \\ 0 & \alpha_{42} & \alpha_{43} & \alpha_{44} & \alpha_{45} & \alpha_{46} \\ 0 & \alpha_{52} & \alpha_{53} & \alpha_{54} & \alpha_{55} & \alpha_{56} \\ 0 & \alpha_{62} & \alpha_{63} & 0 & \alpha_{65} & \alpha_{66} \end{pmatrix} \cdot \begin{pmatrix} \Delta\omega_r \\ \Delta\delta \\ \Delta\psi_{fd} \\ \Delta\psi_{1d} \\ \Delta\psi_{1q} \\ \Delta\psi_{2q} \end{pmatrix} + \begin{pmatrix} b_{11} & 0 \\ 0 & 0 \\ 0 & b_{32} \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} \Delta T_m \\ \Delta E_{fd} \end{pmatrix} \tag{10.2}$$

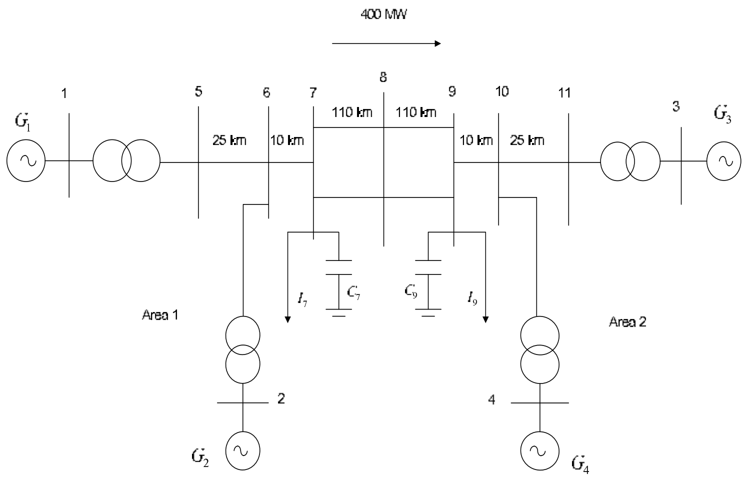


Fig. 10.1 Four-machine two-area model

Using the $d-q$ reference frame to express the magnetic flux and assuming that the model includes one d-axis amortissuer and two q-axis amortisseurs, the components of the state vector are defined as follows: $\Delta\omega_r$ is the rotor's speed difference, and $\Delta\delta$ is the rotor's angle difference. The rest of the parameters of the state vector are defined as $\Delta\psi_{fd}$ is the difference of the field (stator) flux along axis d, $\Delta\psi_{1d}$, is the difference of the rotor's flux of the d-axis amortisseur, and $\Delta\psi_{1q}$ is the difference of the rotor's flux of the first amortisseur of the q-axis, $\Delta\psi_{2d}$ is the difference of the rotor's flux of the second amortisseur of the q-axis. Moreover, the components of the control input vector are defined as follows: ΔT_m is the difference of the rotor's input torque and ΔE_{fd} is the difference of the excitation (stator's) field along axis d. The parameters a_{ij}, $i=1,\cdots,6$ and $j=1,\cdots,6$ and b_{ij} $i=1,\cdots,6$ and $j=1,\cdots,2$ are described in [206].

For the 2-area four-machine model depicted in Fig. 10.1 the state vector of the dynamic model is $x \in R^{24\times 1}$. The state vector can be written as $x=[x^1,x^2,x^3,x^4]$, where x^i, $i=1,\cdots,4$ is the state vector of the i-th generator given by $x^i=[\Delta\dot{\omega}_r,\Delta\dot{\delta},\Delta\psi_{fd},\Delta\psi_{1d},\Delta\psi_{1q},\Delta\psi_{2q}]^T$. Low frequency oscillation modes in Fig. 10.1 include two local modes and one inter-area mode. The two local oscillation modes are the swing between G1 and G2, and the swing between G3 and G4, respectively. The inter-area mode is the oscillation between Area 1 and Area 2 connected by transmission tie line. Indicative values of the system modes, when a linear model of the power system is considered, are given in Table 10.2. [195],[206],[267].

To stabilize the multi-area, multi-machine power system the distributed generators G_i are equipped with Automatic Voltage Regulators (AVR). For instance in Fig. 10.1 G1 and G2 are equipped with local Power System Stabilizer (PSS) in order to restrain the poorly damped local oscillation mode between G1 and G2. In order to eliminate poorly damped inter-area oscillation between Area 1 and

Table 10.2 Modes (eigenvalues) for the two-area four-machines power system

No.	Eig. Real	Eig. Imag.	Frequency	Damping Ratio	Dominant States
1,2	$-0.76E-3$	$\pm0.22E-2$	0.0003	0.331	$\Delta\omega$, $\Delta\delta$ of G_1, G_2, G_3, G_4
3	$-0.96E-1$	–	–	–	”
4,5	−0.111	±3.43	0.545	0.032	”
6	−0.117	–	–	–	”
7	−0.265	–	–	–	$\Delta\psi_{fd}$ of G_3 and G_4
8	−0.276	–	–	–	$\Delta\psi_{fd}$ of G_1 and G_2
9,10	−0.492	±6.82	1.087	0.072	$\Delta\omega$, $\Delta\delta$ of G_1, G_2
11,12	−0.506	±7.02	1.117	0.072	$\Delta\omega$, $\Delta\delta$ of G_3, G_4
13	−3.428	–	–	–	d, q amortisseur flux linkages
14	−4.139	–	–	–	”
15	−5.287	–	–	–	”
16	−5.303	–	–	–	”
17	−31.03	–	–	–	”
18	−32.45	–	–	–	”
19	−34.07	–	–	–	”
20	−35.53	–	–	–	”
21,22	−37.89	±0.142	0.023	≃1.0	”
23,24	−38.01	$\pm0.38E-1$	0.006	≃1.0	”

Area 2, the following variables can be measured and used by algorithm for control of supplementary excitation: (1) Remote/global signals of area speed difference $\Delta\omega_1 + \Delta\omega_2 - \Delta\omega_3 - \Delta\omega_4$, (2) Local signals at a generator G_i, such as shaft speed ω_i, terminal voltage V_{t_i}, excitation voltage E_{fd_i} and accelerating power P_{α_i}.

10.1.4 Nonlinear Modeling of the Electric Power System

Because of linear models or NARX representations and NARMAX representations (nonlinear autoregressive moving average models with exogenous inputs) have been proved in several cases inadequate to model the complete dynamics of power systems, neural-fuzzy network models have been considered in this paper. Artificial neural networks (ANNs) proved to be suitable at identifying and controlling nonlinear systems with complex dynamic and transient processes and they can easily identify the interactions between the system's inputs and outputs for multivariable applications [121],[153],[279],[437]. It has been shown that the complex and nonlinear dynamics of a single machine infinite bus configuration can be identified with sufficient accuracy by a Multilayer Perceptron neural network using deviation signals as inputs [437].

ANNs have also been used for dynamic security assessment [13],[43],[87],[253] an ANN is used to predict system stability in [43] and to estimate maximum generators swing angles in [87],[437]. The transient energy margins are used for contingencies screening and ranking in [253]. In [13], recurrent radial basis function, and multilayer perceptron ANN schemes are used for dynamic system modeling and generators' angles and angular velocities prediction for multi-machine power systems. Transient stability is assessed based on monitoring generators' angles and angular velocities with time, and checking whether they exceed the specified limits for system stability or not. In [406],[407] a new method based on neural networks for eigenvalue predictions of critical stability modes of power systems is proposed, the interest is focused on inter-area oscillations in the European interconnected power system.

Neural-fuzzy modeling using the data of the power system is described as follows: a collection of N data in a $n+1$-dimensional space is considered, and a generic neuro-fuzzy model is presented as a set of fuzzy rules in the following form:

$$R^l\ :\ IF\, x_1\ \ is\, A_1^l\ \ AND\ \ x_2\ is\ \ A_2^l\, AND \cdots AND\, x_n\ is\ \ A_n^l\ THEN\, \bar{y}^l = z^l(x) \quad (10.3)$$

where $x = [x_1, x_2, \cdots, x_n] \in U$ are input variables, $\bar{y}^l \in V$ is the output variable, A_i^l are the fuzzy sets in which $U^i \in R$ is divided and $z^l(x)$ is a function of the input variables.

The modeling procedure begins with the model initialization stage that includes data processing, prior knowledge utilization, and initial rule-base generation. The optimal number of fuzzy rules is determined and the appropriate type of fuzzy rules is selected. Model optimization follows which includes parameter learning and rule-base simplification. Finally, the acquired neurofuzzy model is validated under certain performance indexes. If the model performance is not satisfactory, further modication including structure and parameter optimization is required [56],[59],[137],[138],[447].

In the sequel fuzzy rules of the Takagi-Sugeno type will be considered. These have the form:

$$\begin{array}{rl} R_l\ :& IF\, x_1\ is\ A_1^l AND\, x_2\ is\ A_2^l\ AND\ \cdots AND\, x_n\ is\ A_n^l \\ & THEN\, \bar{y}^l = \sum_{i=1}^n w_i^l x_i + b^l\ l = 1, 2, \cdots, L \end{array} \quad (10.4)$$

where R^l is the l-th rule, $x = [x_1, x_2, \cdots, x_n]^T$ is the input (antecedent) variable, $\bar{y}^l$ is the output (consequent) variable, and w_i^l, b^l are the parameters of the local linear models. The above model is a Takagi-Sugeno model of order 1. Setting $w_i^l = 0$ results in the zero order Takagi-Sugeno model [125],[160],[281],. The output of the Takagi-Sugeno model is given by the weighted average of the rules consequents (Fig. 9.4):

$$\hat{y} = \frac{\sum_{l=1}^L \bar{y}^l \prod_{i=1}^n \mu_{A_i^l}(x_i)}{\sum_{l=1}^L \prod_{i=1}^n \mu_{A_i^l}(x_i)} \quad (10.5)$$

where $\mu_{A_i^l}(x_i)\ :\ R \rightarrow [0,1]$ is the membership function of the fuzzy set A_i^l in the antecedent part of the rule R^l. In the case of a zero order TS system the output of the l-th local model is $\bar{y}^l = b^l$, while in the case of a first order TS system the output of the l-th local model is given by $\bar{y}^l = \sum_{l=1}^L w_i^l x_i + b^l$.

The individual steps of data-driven nonlinear modeling, and particularly of neuro-fuzzy modeling, are discussed in [159],[165],[376],[375]. Methods for optimization of neuro-fuzzy models parameters has been extensively discussed in bibliography [33],[85],[308],[374],[382],[383],[463]. These issues were analyzed in detail in Chapter 9.

10.2 Fault Diagnosis Tests for the Electric Power System

10.2.1 Parameters of the Nonlinear Power System Model

Neuro-fuzzy modelling of the presented power transmission system has been carried out. The power system dynamics can be modelled using a neural network for each generator, having as output the rotor's speed difference $\Delta\omega_r(k)$ and as inputs $\Delta\omega_r(k-1)$, $\Delta\omega_r(k-2)$ and $\Delta\delta(k-1)$. Neural models with a different output, such as $\Delta\delta$ and a larger number of inputs could be also considered. The input space was segmented using the input dimension (grid) partition (additionally, a neural network with Hermite basis functions was used to model the dynamics of the power system. In the latter case the Hermite basis functions can capture with increased accuracy the multi-frequency characteristics of the oscillatory behavior of the power transmission system).

For a fault-free condition, a comparison between the neural model output and the exact model output is shown in Fig. 10.2, for generator G_1. The exact model has been simulated by using Kundur's power system model of PSAT [267]. The neural model has been identified considering both a neural network with Hermite polynomial basis functions and a neuro-fuzzy network of the Takagi-Sugeno type.

As pointed out in Section 10.2, the neural model is used to simulate the physical power system in a stable undistorted state and the real power system is simulated

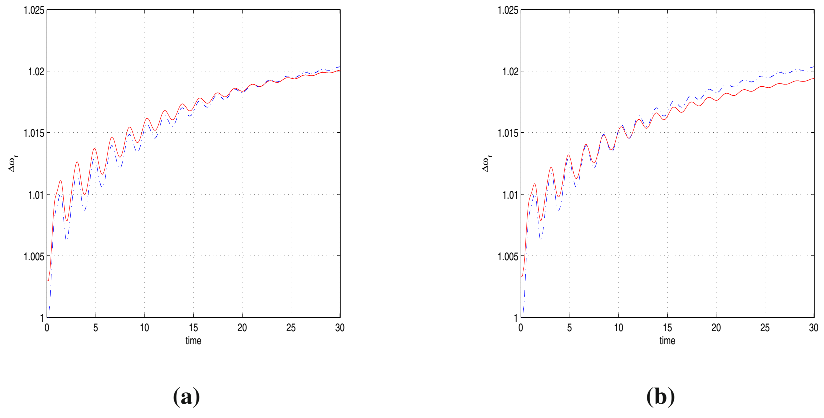


Fig. 10.2 (a) Approximation of the oscillatory behavior of the electric power system (dashed line) by a neural network with Hermite polynomial basis functions(continuous line) (b) Approximation of the oscillatory behavior of the electric power system (dashed line) by a neuro-fuzzy network of the Takagi-Sugeno type (continuous line)

.

by the exact model (a neural model extracted from input/output data of the power system). Due to parameters variation in the exact model, its output differs from the output of the neural model, and thus the residuals described in Section 10.2 have been calculated.

Regarding the general description of the training set used to identify both the neural model and the general description of the exact model, this is given by:

$$\begin{pmatrix} u(k-m) & \cdots & u(k-1) & y(k-n) & \cdots & y(k-1) & \rightarrow y(k) \\ u(k-m-1) & \cdots & u(k-2) & y(k-n-1) & \cdots & y(k-2) & \rightarrow y(k-1) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u(k) & \cdots & u(m-1) & y(1) & \cdots & y(n-1) & \rightarrow y(n) \end{pmatrix} \tag{10.6}$$

An indicative model consists of rules of the form:

$$R^l : IF\ y(k-3)\ is\ A_i\ AND\ y(k-2)\ is\ B_j\ AND\ y(k-1)\ is\ C_k\ \ THEN\ y(k)\ is\ D_m \tag{10.7}$$

with $i,j,k,m = 1,2,3,4$. The size of the training set was 3000. The LMS (Least Mean Square) algorithm was used for the adaptation of the linear weights $w_1^{(l)}$. The rule base consists of 64 rules (3 input variables partitioned in 4 fuzzy subsets each). All fuzzy sets where assumed to have the same spread. Inactive rules have been removed from the fuzzy rule base.

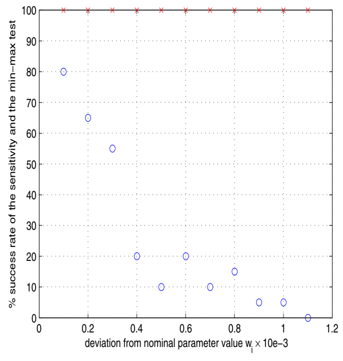
The reduced-size rule base contained 22 rules. Therefore, the dimension of the parameters vector is 34, (22 linear weights in the antecedent part of the rules, and 12 nonlinear centers). Using Eq.(9.14) and setting the false alarm rate $\alpha = 0.5$, from the table of the χ^2 distribution one obtains that the value of the change threshold λ should be set to $\eta = 34$. In the case of absence of change to the parameters of the rule base the global χ^2 test was (averaging over 10 trials) $t = 38.6216$. The condition of the Fisher Information Matrix was $cond(M^T\Sigma^{-1}M) = 4.4314\times10^9$.

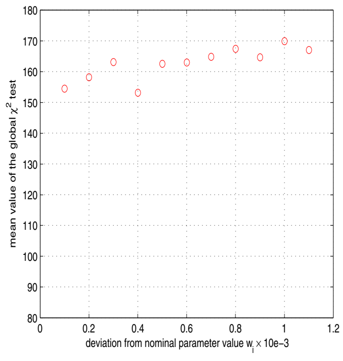
10.2.2 Efficiency of the Fault Diagnosis Method

Next, faults were imposed to the linear and the nonlinear parameters of the exact model that describes the power system dynamics. Detailed diagrams on the performance of the change detection and isolation tests, in the case of a change in parameter $w_1^{(10)} = 1.195923$ of the exact model, are given in Fig. 10.3. The success rate of the two tests is depicted in Fig. 10.3(a). The mean value of the global χ^2 is depicted in Fig. 10.3(b).

Detailed diagrams on the performance of the change detection and isolation tests, in the case of a change in parameter $c_1^{(1)} = 1.954000$ of the exact model, are given in Fig. 10.4. The parameters vector contained only the centers of the fuzzy model. The success rate of the two tests is depicted in Fig. 10.4(a).

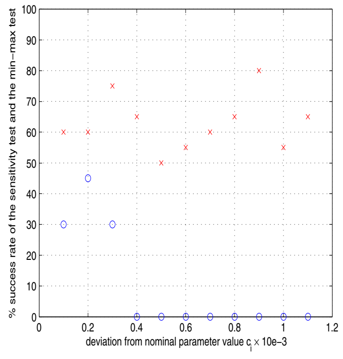
The mean value of the global χ^2 is depicted in Fig. 10.4(b). It is clear that when $c_1^{(1)}$ is close to its nominal value the χ^2 test takes a value close to the change

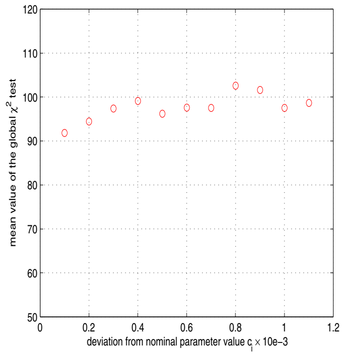




(a) **(b)**

Fig. 10.3 Success rate of the sensitivity ($\times$) and the min-max (o) tests in case of a change (fault) in parameter $w_1^{(10)}$ of the exact model of the electric power transmission system. (b) mean value of the global χ^2 test.





(a) **(b)**

Fig. 10.4 Success rate of the sensitivity ($\times$) and the min-max (o) tests in case of a change (fault) in parameter $c_1^{(1)}$ of the exact model of the electric power transmission system. (b) mean value of the global χ^2 test.

threshold $\eta = 34$. On the other hand when the deviation of $c_1^{(1)}$ from its nominal value increases the value of the χ^2 test grows significantly.

To increase the success rate of the change isolation tests one had better perform the tests separately to the linear and the nonlinear parameters of the fuzzy model [344]. This is summarized as follows: (i) perform the diagnostic test to the linear parameters. If it finds a fault then stop, otherwise (ii) perform the test to the nonlinear parameters.

The success rate of the change isolation tests is also affected by the following factors [473]: (a) the magnitude of the parameter change: a parameter change close to the nominal value results in increased success of the sensitivity test, (b) the size of the data set used for the statistical tests : if the number of data used in the statistical tests is large then the success rate is high, (c) the signal-to-noise ratio.

The changes on the parameters of the power system model that were used in the simulation experiments are summarized in Table 10.4:

Table 10.4 % parameter change and % success rate of fault isolation tests

$w_1^{(10)} = 1.195931$	sensitivity	min-max	$c_1^{(1)} = 1.954000$	sensitivity	min-max
0.0083	100	80	0.0083	60	30
0.0166	100	65	0.0166	60	45
0.0249	100	55	0.0249	75	30
0.0332	100	20	0.0332	65	0
0.0415	100	10	0.0415	50	0
0.0498	100	20	0.0498	55	0
0.0581	100	10	0.0581	60	0
0.0664	100	15	0.0664	65	0
0.0747	100	5	0.0747	80	0
0.0830	100	5	0.0830	55	0

10.3 Fault Diagnosis of Electric Motors

10.3.1 Failures in Rotating Electrical Machines

Next, the problem of fault detection and isolation in electric motors is analyzed. The reasons behind failures in rotating electrical machines have their origin in design,manufacturing tolerance, assembly, installation, working environment, nature of load and schedule of maintenance. Motor faults can be classified in two types: mechanical and electrical. Mechanical faults in the rotor are identified as eccentricity (static or dynamic) and misalignment, while stator eccentricity and core slacking are the main type of mechanical faults in the stator [274]. Moreover, bearing fault, which may also cause rotor eccentricity, is the common mechanical fault in the induction motors. Other mechanical faults, such as rotor rubbing, stator and rotor fatigue etc., are the consequence of the previously mentioned faults. Winding faults, such as turn to turn, phase to phase and winding to earth faults are the roots of electrical faults in the rotor [96],[150],[196],[276],[384]. The causes of electrical faults in the squirrel cage rotor are bars crack, bars slack and bad connection with the end rings. In addition, short circuit of rotor laminations is a common fault. Stator is subject to some types of fault, such as winding faults and core faults. Winding faults are due to turn to turn, phase to phase or winding to earth short circuit, while core faults are due to core slacking, laminations short circuit and rotor strike. Statistics on electric motor faults are given in the diagrams depicted in Fig. 10.5 [384].

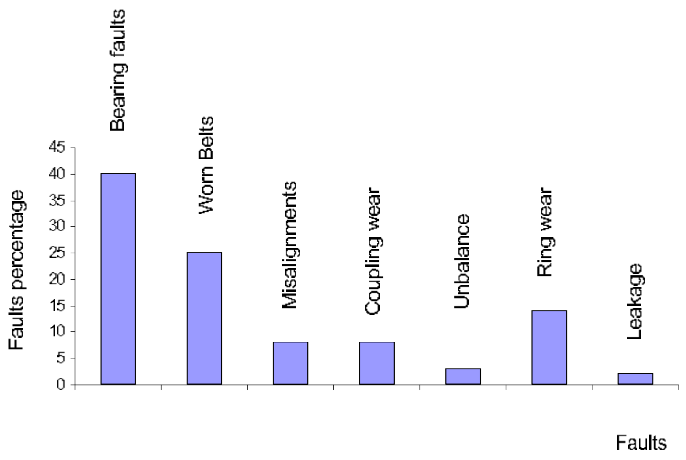


Fig. 10.5 Classification of the major faults in electric motors

It is important to spot faults in time because they can lead to the total destruction of the motor. Stator winding faults can be classified as follows: (i) turn to turn short circuits within a coil (inter-turn short circuit), (ii) short circuits between coils, (iii) coils to earth short circuits, (iv) open circuits within coils.

10.3.2 Faults in the DC Motor Control Loop

In Fig. 10.6 the DC motor control loop and the associated disturbances (faults) are given: e is the input to the controller, f_c is the controller's fault, u_c is a disturbance to the system's input, f_α is the motor's fault, η_p is the disturbance to the motor's output, and η_s is a disturbance to the sensor measuring the motor's angle. In [161] the following indicative faults have been examined: F_1: offset sensor signal, F_2: increased input-output gain, F_3: increased actuator friction, F_4: offset in the motor's output, F_6: increased sensor gain.

10.3.3 Residual Generation with the Use of Kalman Filtering

To model the motor dynamics in the fault-free condition Kalman Filtering is used. As explained in Chapter 9 the output of the model produced by the Kalman Filter is compared to the real output and thus residuals are generated. The statistical processing of the residuals sequence enables to decide on the existence of a fault and to distinguish its cause. To perform fault detection one can consider only one Kalman filter, which provides an estimation of the motor's state vector (see Fig. 10.6). This in turn is compared to the real state vector so as to generate a residual and to decide on whether the motor functions in normal mode or not. To perform fault isolation

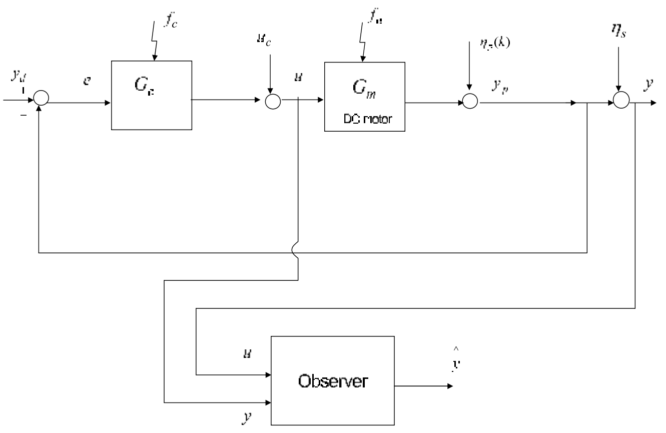


Fig. 10.6 Control loop with variables and fault influences

one should consider M Kalman filters, each one designed to estimate the motor's state vector in the case of the m-th fault (see Fig. 9.6).

Fault diagnosis takes place in-the-loop. The controller assures convergence of the system's state vector to the reference state vector despite changes in the model's parameters. This in turn means that despite the existence of parametric changes to the model, a typical residual (e.g. the difference between the output that is associated with the Kalman Filter state vector and the output of the monitored system) will hardly differ from the residual of the normal system, thus making fault diagnosis a not-so-easy task. On the other hand faults in sensors out of the loop, as denoted by disturbance η_s in Fig. 10.6, can be detected by processing the residual with simple decision criteria (e.g. root mean square error or likelihood ratio). The results obtained are given in Fig. 10.7 to 10.11. In the case of Kalman Filtering the process and measurement noise covariance matrices were taken to be diagonal with non-zero elements equal to 10^{-3}.

10.3.4 Residual Generation with the Use of Particle Filtering

Again the residual is the difference between the output that is associated with the Particle Filter state vector and the output of the monitored system. Fault diagnosis is again performed in-the-loop and as explained in the case of the Kalman Filter, change decision criteria (which are based on the RMSE and the likelihood ratio) will not make distinguishable changes in the model's structure. On the other hand it is possible to visualize with the aforementioned residuals and change detection criteria (e.g root mean square error or likelihood ratio) faults affecting sensors out of the loop. Here, the fault η_s shown in Fig. 10.6 is considered. The results obtained from the fault diagnosis tests are shown in Fig. 10.8 to 10.12. Convergence problems in Particle Filter algorithms for FDI have been analyzed in [76],[472]. In the

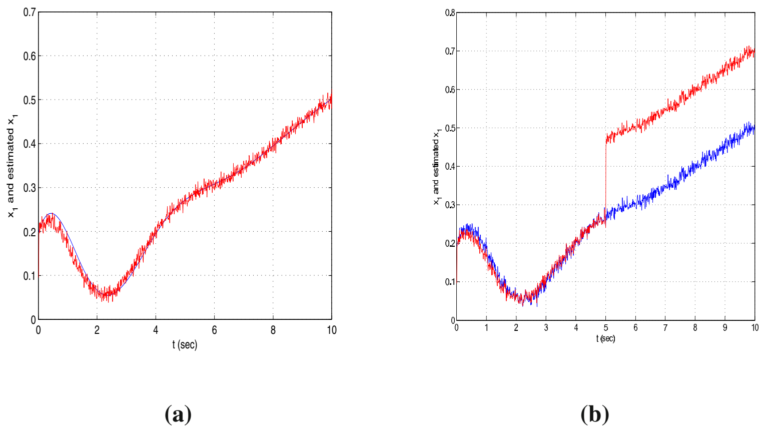


Fig. 10.7 State variable x_1 of the real system (blue line) and estimated state variable $\hat{x}_1$ (red line) provided by the Kalman Filter when the motor operates in (a) normal mode (b) faulty mode.

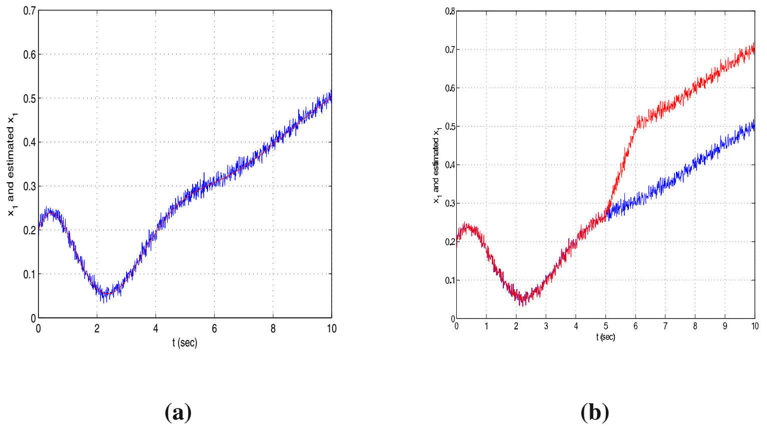


Fig. 10.8 State variable x_1 of the real system (blue line) and estimated state variable $\hat{x}_1$ (red line) provided by the Particle Filter when the motor operates in (a) normal mode (b) faulty mode.

simulation experiments the number of particles in the PF state estimation was set equal to 1000. It is also noted that the computation cycle of Particle Filter is significantly longer than the computation cycle of the Kalman Filter. This means that the simulation time of the Particle Filter is given by $(\frac{t}{T_s}) \times$ computation cycle, where t is the time value appearing in the simulation diagrams, and T_s is the sampling period. The computation time of the PF depends linearly on the number of particles

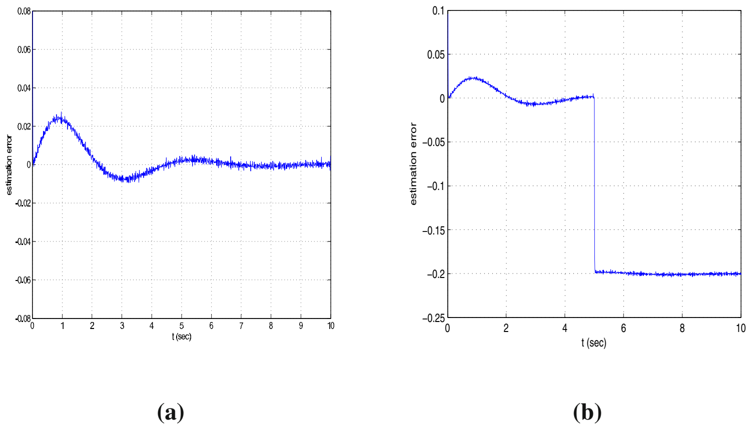


Fig. 10.9 Estimation error between x_1 and $\hat{x}_1$ when using the Kalman Filter and the motor operates in (a) normal mode (b) faulty mode.

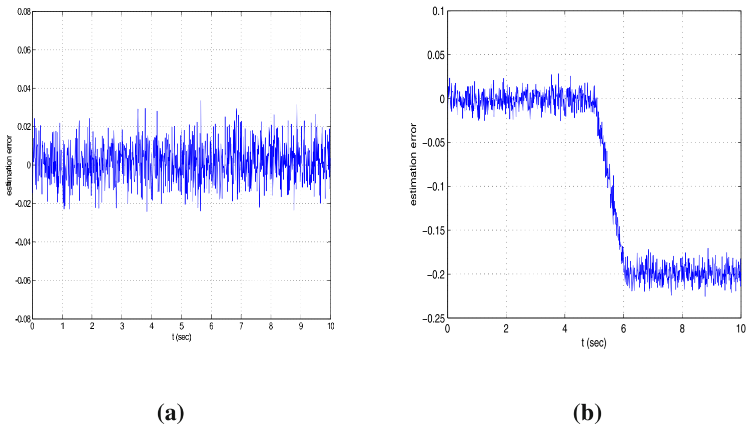
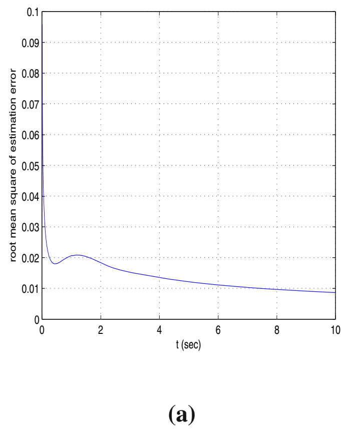


Fig. 10.10 Estimation error between x_1 and $\hat{x}_1$ when using the Particle Filter and the motor operates in (a) normal mode (b) faulty mode.

(when no sorting is performed in the resampling procedure). Detailed results on the relation between computation time of the PF and number of particles can be found in [333],[335].

10.3.5 Fault Diagnosis in Control Loops

As it can be observed from the simulation experiments, fault detection and isolation in the control loop is more difficult than fault diagnosis in the open loop. Small faults in the actuator, be they additive or multiplicative are usually compensated by



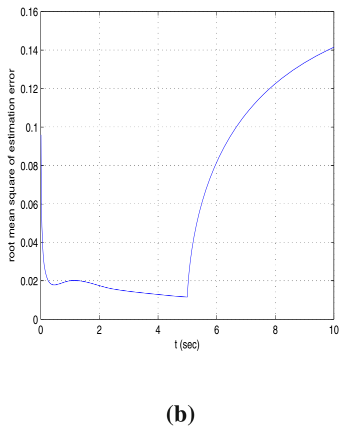
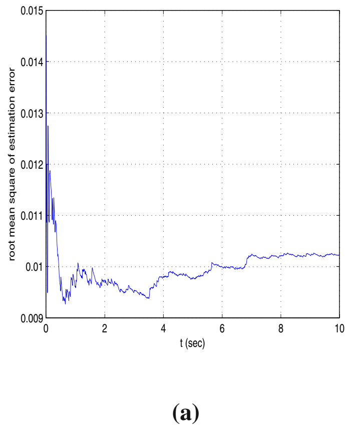


Fig. 10.11 Root of the mean square of the estimation error S_e when using the Kalman Filter the motor operates in (a) normal mode (b) faulty mode.



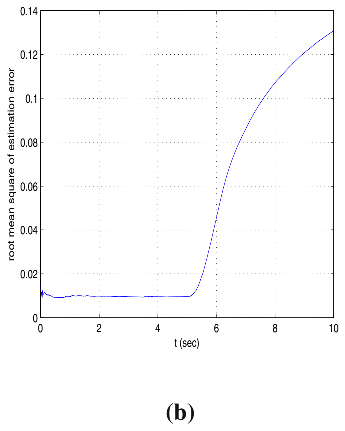


Fig. 10.12 Root of the mean square of the estimation error when using the Particle Filter S_e and the motor operates in (a) normal mode (b) faulty mode.

the feedback controller and they are not detectable by considering a residual based on the output's error $e(k)$ and on the output $y(k)$, as long as the control deviation turns back to approximately zero [161]. Also faults due to small sensor offsets (in the measurement of $x_1(k)$, $x_2(k)$ and $x_3(k)$) will not be detected. The controller will just make the wrong sensor signal equal to the reference input $x_d(k)$. Only by a redundant sensor or other redundant information for the controlled variable, the offset fault can usually be detected.

Since the root-mean square error, the likelihood ratio or the performance criteria mentioned above do not give a clear view of the system's condition when fault diagnosis is performed in the closed loop, one has to search for more elaborated statistical change detection criteria [81]. Such a change detection criterion is provided by the *local statistical approach* to FDI, which finally results to the χ^2 test analyzed in Chapter 9. The aforementioned statistical method for FDI can detect slight changes in the motor's dynamic model, which are due to small changes in the model's parameters. Therefore, this FDI method is more efficient in performing in-the-loop fault diagnosis for the motor's model [474].

Chapter 11
Optimization Methods for Motion Planning of Multi-robot Systems

Abstract. Optimization through nonlinear programming techniques, such as gradient algorithms, can be an efficient approach for solving various problems in the design of intelligent robots, e.g. motion planning for multi-robot systems. A distributed gradient algorithm is proposed for enabling coordinated convergence of an ensemble of mobile robots towards a goal state, and at the same time for assuring avoidance of collisions between the robots as well as avoidance of collisions with obstacles in the motion plane. The stability of the multi-robot system is proved with Lyapunov's theory and particularly with LaSalle's theorem. Motion planning with the use of distributed gradient is compared to motion planning based on particle swarm optimization.

11.1 Distributed Gradient for Motion Planning of Multi-robot Systems

11.1.1 Approaches to Multi-robot Motion Planning

In recent years there has been growing interest in multi-robot systems since swarms of cooperating robots can perform complicated tasks that a single robot can not carry out [477]. As the cost of robotic vehicles goes down and their size becomes more compact the number of military and industrial applications of mobile multi-robot systems increases. Possible industrial applications of multi-robot systems include hazardous inspection, underwater or space exploration, assembling and transportation, search and rescue, and underground exploitation of energy resources [130]. Some examples of military applications are guarding, escorting, patrolling (surface surveillance) and strategic behaviors, such as stalking and attacking.

Control of cooperating robotic vehicles has been extensively studied in both the behavior-based and the system-theoretic approach. Behavior-based approaches for multi-robot systems have the advantage of being flexible, easy to implement and update, while they require no explicit models of the vehicle/robot and its environment [329]. These approaches are well suited to domains in which mathematical representation of tasks are difficult to obtain, and models are not available, too complex for computation, or time-varying [37]. On the other hand, system-theoretic approaches have provable performance and are applicable in cases where tasks can be parameterized, but require models of the vehicles and their environment. In the system-theoretic approach the stability of cooperative motion can been analyzed using Lyapunov theory through which suitable control functions for the steering of the individual mobile agents can be also found [145]. In the latter case, distributed control laws for multi-robot systems have been derived and have made possible motion planning through obstacles and convergence of mobile agents to targeted regions [131],[386]. To implement this cooperative behavior issues related to distributed sensing, measurements fusion and communication between the individual robots have also to be taken into account [299],[372]

This chapter studies multi-robot swarms following principally the system-theoretic point of view, i.e. it is assumed that an explicit mathematical model of the robots and their interaction with the environment is available. The objective is to succeed motion planning of the multi-robot system in a workspace that contains obstacles. A usual approach for doing this is the *potential fields theory*, in which the individual robots are steered towards an equilibrium by the gradient of an harmonic potential [185],[259],[320],[351]. Variances of this method use non-linear anisotropic harmonic potential fields which introduce to the robots' motion directional and regional avoidance constraints [259].

The novelty introduced is the so-called *distributed gradient* algorithm. There are M robots which emanate from arbitrary positions in the 2D space and the potential of each robot consists of two terms: (i) the cost V^i due to the distance of the i-th robot from the goal state, (ii) the cost due to the interaction with the other $M-1$ robots. Moreover, a repulsive field, generated by the proximity to obstacles, is taken into account. The gradient of the aggregate potential provides the kinematic model for each robot, and defines a path towards the equilibrium. Thus, it is proved that the update of the position of each robot is described by a gradient algorithm which contains an interaction term with the gradient algorithms defining the motion of the rest $M-1$ robots. Distributed gradient assures simultaneous convergence of the individual robots towards the equilibrium, and this convergence is analytically proved with the use of Lyapunov stability theory and LaSalle's theorem. It is shown that the mean position of the multi-robot system reaches precisely the goal state x^* while each robot stays in a bounded area close to x^*. The distributed gradient algorithm is an original result for the area of stochastic approximations and adaptive systems and can have several engineering applications. Moreover, it is of interest for the field of nanorobotics since it approximates the Brownian motion and simulates the diffusion of nanoparticles [68].

An alternative solution to multi-robot motion planning proposed by this paper is based on *swarm intelligence*. Previous applications of the particle swarm optimization algorithm for the steering towards desirable final positions and the simulation-solution of diffusion systems can be found in [167]. This method works by searching iteratively in regions defined by each robot's best previous move and the best previous move of its neighbors. Swarm intelligence is evident in biological systems and has been also studied in statistical physics, where collective behavior of self-propelled particles has been observed [220]. The method is useful for the avoidance of local minima. A swarm, which is a collection of robots, can converge to a wide range of distributions, while no individual robot is aware of the distribution it is working to realize. The dynamic behavior of the robots under the particle swarm algorithm can be analyzed with the use of ordinary differential equations [65]. It can be shown that appropriate tuning of the differential equation's coefficients can prevent explosion, i.e. the robots velocity is kept within certain bounds.

11.1.2 The Distributed Gradient Algorithm

In the following sections it will be shown how motion planning of multi-robot systems can be solved with the use of *distributed stochastic search* algorithms. These can be multiple gradient algorithm's that start from different points in the solutions space and interact with each other while moving towards the goal position. Distributed gradient algorithms, stem from stochastic search algorithms treated in [29],[86] if an interaction term is added:

$$x^i(t+1) = x^i(t) + \gamma^i(t)[h(x^i(t)) + e^i(t)] + \sum_{j=1, j\neq i}^{M} g(x^i - x^j),\ i = 1, 2, \cdots, M \quad (11.1)$$

The term $h(x(t)^i) = -\nabla_{x^i} V^i(x^i)$ indicates a local gradient algorithm, i.e. motion in the direction of decrease of the cost function $V^i(x^i) = \frac{1}{2} e^i(t)^T e^i(t)$. The term $\gamma^i(t)$ is the algorithm's step while the stochastic disturbance $e^i(t)$ enables the algorithm to escape from local minima. The term $\sum_{j=1, j\neq i}^{M} g(x^i - x^j)$ describes the interaction between the i-th and the rest $M-1$ stochastic search algorithms. Convergence analysis based on the Lyapunov stability theory can be stated in the case of distributed gradient algorithms. This is important for the problem of multi-robot motion planning.

11.1.3 Kinematic Model of the Multi-robot System

The objective is to lead a swarm of M mobile robots, with different initial positions on the 2-D plane, to a desirable final position. The position of each robot in the 2-D space is described by the vector $x^i \in R^2$. The motion of the robots is synchronous, without time delays, and it is assumed that at every time instant each robot i is aware about the position and the velocity of the other $M-1$ robots. The cost function that describes the motion of the i-th robot towards the goal state is denoted as $V(x^i) : R^n \rightarrow R$. The value of $V(x^i)$ is high on hills, small in valleys, while it holds $\nabla_{x^i} V(x^i) = 0$ at the goal position and at local optima. The following conditions must hold:

(i) The cohesion of the swarm should be maintained, i.e. the norm $||x^i - x^j||$ should remain upper bounded $||x^i - x^j|| < \varepsilon^h$,

(ii) Collisions between the robots should be avoided, i.e. $||x^i - x^j|| > \varepsilon^l$,
(iii) Convergence to the goal state should be succeeded for each robot through the negative definiteness of the associated Lyapunov function $\dot{V}^i(x^i) = \dot{e}^i(t)^T e^i(t) < 0$ [86].

The interaction between the i-th and the j-th robot is

$$g(x^i - x^j) = -(x^i - x^j)[g_a(||x^i - x^j||) - g_r(||x^i - x^j||)] \tag{11.2}$$

where $g_a()$ denotes the attraction term and is dominant for large values of $||x^i - x^j||$, while $g_r()$ denotes the repulsion term and is dominant for small values of $||x^i - x^j||$. Function $g_a()$ can be associated with an attraction potential, i.e. $\nabla_{x_i} V_a(||x^i - x^j||) = (x^i - x^j) g_a(||x^i - x^j||)$. Function $g_r()$ can be associated with a repulsion potential, i.e. $\nabla_{x_i} V_r(||x^i - x^j||) = (x^i - x^j) g_r(||x^i - x^j||)$. A suitable function $g()$ that describes the interaction between the robots is given by [113]

$$g(x^i - x^j) = -(x^i - x^j)(a - be^{\frac{||x^i - x^j||^2}{\sigma^2}}) \tag{11.3}$$

where the parameters a, b and c are suitably tuned. It holds that $g_a(x^i - x^j) = -a$, i.e. attraction has a linear behavior (spring-mass system) $||x^i - x^j|| g_a(x^i - x^j)$. Moreover, $g_r(x^i - x^j) = be^{\frac{-||x^i - x^j||^2}{\sigma^2}}$ which means that $g_r(x^i - x^j)||x^i - x^j|| \leq b$ is bounded. Applying Newton's laws to the i-th robot yields

$$\dot{x}^i = v^i, \;\; m^i \dot{v}^i = U^i \tag{11.4}$$

where the aggregate force is $U^i = f^i + F^i$. The term $f^i = -K_v v^i$ denotes friction, while the term F^i is the propulsion. Assuming zero acceleration $\dot{v}^i = 0$ one gets $F^i = K_v v^i$, which for $K_v = 1$ and $m^i = 1$ gives $F^i = v^i$. Thus an approximate kinematic model is

$$\dot{x}^i = F^i \tag{11.5}$$

According to the Euler-Langrange principle, the propulsion F^i is equal to the derivative of the total potential of each robot, i.e.

$$\begin{aligned}
&F^i = -\nabla_{x^i}\{V^i(x^i) + \tfrac{1}{2}\sum_{i=1}^{M}\sum_{j=1,j\neq i}^{M}[V_a(||x^i - x^j|| + V_r(||x^i - x^j||)]\} \Rightarrow \\
&F^i = -\nabla_{x^i}\{V^i(x^i)\} + \sum_{j=1,j\neq i}^{M}[\nabla_{x^i} V_a(||x^i - x^j||) - \nabla_{x^i} V_r(||x^i - x^j||)] \Rightarrow \\
&F^i = -\nabla_{x^i}\{V^i(x^i)\} + \sum_{j=1,j\neq i}^{M}[-(x^i - x^j) g_a(||x^i - x^j||) - (x^i - x^j) g_r(||x^i - x^j||)] \Rightarrow \\
&F^i = -\nabla_{x^i}\{V^i(x^i)\} - \sum_{j=1,j\neq i}^{M} g(x^i - x^j)
\end{aligned}$$

Substituting in Eq. (11.5) one gets Eq. (11.1), i.e. $x^i(t+1) = x^i(t) + \gamma^i(t)[-\nabla_{x^i} V^i(x^i) + e^i(t+1)] - \sum_{j=1,j\neq i}^{M} g(x^i - x^j), \; i = 1, 2, \cdots, M$, with $\gamma^i(t) = 1$, which verifies that the kinematic model of a multi-robot system is equivalent to a distributed gradient search algorithm.

11.1.4 Cohesion of the Multi-robot System

The behaviour of the multi-robot system is determined by the behaviour of its center (mean of the vectors x^i) and of the position of each robot with respect to this center. The center of the multi-robot system is given by

$$\begin{array}{c}\bar{x} = E(x^i) = \frac{1}{M}\sum_{i=1}^{M} x^i \Rightarrow \dot{\bar{x}} = \frac{1}{M}\sum_{i=1}^{M}\dot{x}^i \Rightarrow \\ \dot{\bar{x}} = \frac{1}{M}\sum_{i=1}^{M}[-\nabla_{x^i}V^i(x^i) - \sum_{j=1, j\neq i}^{M}(g(x^i - x^j))]\end{array} \tag{11.6}$$

From Eq. (11.3) it can be seen that $g(x^i - x^j) = -g(x^j - x^i)$, i.e. $g()$ is an odd function. Therefore, it holds that $\frac{1}{M}(\sum_{j=1,j\neq i}^{M} g(x^i - x^j)) = 0$, and

$$\dot{\bar{x}} = \frac{1}{M}\sum_{i=1}^{M}[-\nabla_{x^i}V^i(x^i)] \tag{11.7}$$

Denoting the goal position by x^* , and the distance between the i-th robot and the mean position of the multi-robot system by $e^i(t) = x^i(t) - \bar{x}$ the objective of distributed gradient for robot motion planning can be summarized as follows:

(i) $lim_{t\to\infty}\bar{x} = x^*$, i.e. the center of the multi-robot system converges to the goal position,
(ii) $lim_{t\to\infty}x^i = \bar{x}$, i.e. the i-th robot converges to the center of the multi-robot system,
(iii) $lim_{t\to\infty}\dot{\bar{x}} = 0$, i.e. the center of the multi-robot system stabilizes at the goal position.

If conditions (i) and (ii) hold then $lim_{t\to\infty}x^i = x^*$. Furthermore, if condition (iii) also holds then all robots will stabilize close to the goal position.

It is known that the stability of local gradient algorithms can be proved with the use of Lyapunov theory [29]. A similar approach can be followed in the case of the distributed gradient algorithms given by Eq. (11.1). The following simple Lyapunov function is considered for each gradient algorithm [113]:

$$V_i = \frac{1}{2}{e^i}^T e^i \Rightarrow V_i = \frac{1}{2}||e_i||^2 \tag{11.8}$$

Thus, one gets

$$\dot{V}^i = {e^i}^T \dot{e}^i \Rightarrow \dot{V}^i = (\dot{x}^i - \dot{\bar{x}})e^i \Rightarrow$$

$$\dot{V}^i = [-\nabla_{x^i}V^i(x^i) - \sum_{j=1,j\neq i}^{M} g(x^i - x^j) + \frac{1}{M}\sum_{j=1}^{M}\nabla_{x^j}V^j(x^j)]e^i.$$

Substituting $g(x^i - x^j)$ from Eq. (11.3) yields

$$\begin{array}{c}\dot{V}_i = [-\nabla_{x^i}V^i(x^i) - \sum_{j=1,j\neq i}^{M}(x^i - x^j)a + \\ +\sum_{j=1,j\neq i}^{M}(x^i - x^j)g_r(||x^i - x^j||) + \frac{1}{M}\sum_{j=1}^{M}\nabla_{x^j}V^j(x^j)]e^i\end{array}$$

which gives,

$$\dot{V}_i = -a[\sum_{j=1,j\neq i}^{M}(x^i - x^j)]e^i + \\ +\sum_{j=1,j\neq i}^{M} g_r(||x^i - x^j||)(x^i - x^j)^T e^i - [\nabla_{x^i} V^i(x^i) - \tfrac{1}{M}\sum_{j=1}^{M} \nabla_{x^j} V^j(x^j)]^T e^i$$

It holds that $\sum_{j=1}^{M}(x^i - x^j) = Mx^i - M\frac{1}{M}\sum_{j=1}^{M} x^j = Mx^i - M\bar{x} = M(x^i - \bar{x}) = Me^i$, therefore

$$\dot{V}_i = -aM||e^i||^2 + \sum_{j=1,j\neq i}^{M} g_r(||x^i - x^j||)(x^i - x^j)^T e^i - [\nabla_{x^i} V^i(x^i) - \frac{1}{M}\sum_{j=1}^{M} \nabla_{x^j} V^j(x^j)]^T e^i \tag{11.9}$$

It assumed that for all x^i there is a constant $\bar{\sigma}$ such that

$$||\nabla_{x^i} V^i(x^i)|| \leq \bar{\sigma} \tag{11.10}$$

Eq. (11.10) is reasonable since for a robot moving on a 2-D plane, the gradient of the cost function $\nabla_{x^i} V^i(x^i)$ is expected to be bounded. Moreover it is known that the following inequality holds:

$$\sum_{j=1,j\neq i}^{M} g_r(x^i - x^j)^T e^i \leq \sum_{j=1,j\neq i}^{M} b e^i \leq \sum_{j=1,j\neq i}^{M} b||e^i||.$$

Thus the application of Eq. (11.9) gives:

$$\dot{V}^i \leq aM||e^i||^2 + \sum_{j=1,j\neq i}^{M} g_r(||x^i - x^j||)||x^i - x^j|| \cdot ||e^i|| + \\ +||\nabla_{x^i} V^i(x^i) - \tfrac{1}{M}\sum_{j=1}^{M} \nabla_{x^j} V^j(x^j)||||e^i||$$

$$\Rightarrow \dot{V}^i \leq aM||e^i||^2 + b(M-1)||e^i|| + 2\bar{\sigma}||e^i||$$

where it has been taken into account that

$$\sum_{j=1,j\neq i}^{M} g_r(||x^i - x^j||)^T ||e^i|| \leq \sum_{j=1,j\neq i}^{M} b||e^i|| = b(M-1)||e^i||,$$

and from Eq. (11.10),

$$||\nabla_{x^i} V^i(x^i) - \tfrac{1}{M}\sum_{j=1}^{M} \nabla_{x^i} V^j(x^j)|| \leq ||\nabla_{x^i} V^i(x^i)|| + \tfrac{1}{M}||\sum_{j=1}^{M} \nabla_{x^i} V^j(x^j)|| \\ ||\nabla_{x^i} V^i(x^i) - \tfrac{1}{M}\sum_{j=1}^{M} \nabla_{x^i} V^j(x^j)|| \leq \bar{\sigma} + \tfrac{1}{M} M\bar{\sigma} \leq 2\bar{\sigma}.$$

Thus, one gets

$$\dot{V}^i \leq aM||e^i|| \cdot [||e^i|| - \frac{b(M-1)}{aM} - 2\frac{\bar{\sigma}}{aM}] \tag{11.11}$$

The following bound ε is defined:

$$\varepsilon = \frac{b(M-1)}{aM} + \frac{2\bar{\sigma}}{aM} = \frac{1}{aM}(b(M-1) + 2\bar{\sigma}) \tag{11.12}$$

Thus, when $||e^i|| > \varepsilon$, $\dot{V}_i$ will become negative and consequently the error $e^i = x^i - \bar{x}$ will decrease. Therefore the error e^i will remain in an area of radius ε i.e. the position x^i of the i-th robot will stay in the cycle with center $\bar{x}$ and radius ε.

11.1.5 Convergence to the Goal Position

The case of a convex quadratic cost function is examined, for instance

$$V^i(x^i) = \frac{A}{2}||x^i - x^*||^2 = \frac{A}{2}(x^i - x^*)^T(x^i - x^*) \tag{11.13}$$

where $x^* = [0,0]$ is a minimum point $V^i(x^i = x^*) = 0$. The distributed gradient algorithm is expected to converge to x^*. The robotic vehicles will follow different trajectories on the 2-D plane and will end at the goal position.

Using Eq.(11.13) yields $\nabla_{x^i}V^i(x^i) = A(x^i - x^*)$. Moreover, the assumption $\nabla_{x^i}V^i(x^i) \leq \bar{\sigma}$ can be used, since the gradient of the cost function remains bounded. The robotic vehicles will concentrate round $\bar{x}$ and will stay in a radius ε given by Eq. (11.12). The motion of the mean position $\bar{x}$ of the vehicles is

$$\begin{array}{c} \dot{\bar{x}} = -\frac{1}{M}\sum_{i=1}^{M}\nabla_{x^i}V^i(x^i) \Rightarrow \dot{\bar{x}} = -\frac{A}{M}(x^i - x^*) \Rightarrow \\ \dot{\bar{x}} - \dot{x}^* = -\frac{A}{M}x^i + \frac{A}{M}x^* \Rightarrow \dot{\bar{x}} - \dot{x}^* = -A(\bar{x} - x^*) \end{array} \tag{11.14}$$

The variable $e_\sigma = \bar{x} - x^*$ is defined, and consequently

$$\dot{e}_\sigma = -Ae_\sigma \Rightarrow \varepsilon_\sigma(t) = c_1 e^{-At} + c_2, \tag{11.15}$$

with $c_1 + c_2 = e_\sigma(0)$. Eq. (11.15) is an homogeneous differential equation, which for $A > 0$ results into $lim_{t\to\infty}e_\sigma(t) = 0$, thus $lim_{t\to\infty}\bar{x}(t) = x^*$. It is left to make more precise the position to which each robot converges.

11.1.6 Stability Analysis Using La Salle's Theorem

It has been shown that $lim_{t\to\infty}\bar{x}(t) = x^*$ and from Eq. (11.11) that each robot will stay in a cycle C of center $\bar{x}$ and radius ε given by Eq. (11.12). The Lyapunov function given by Eq. (11.8) is negative semi-definite, therefore asymptotic stability cannot be guaranteed. It remains to make precise the area of convergence of each robot in the cycle C of center $\bar{x}$ and radius ε. To this end, La Salle's theorem can be employed [113],[184].

La Salle's Theorem: Assume the autonomous system $\dot{x} = f(x)$ where $f : D \to R^n$. Assume $C \subset D$ a compact set which is positively invariant with respect to $\dot{x} = f(x)$, i.e. if $x(0) \in C \Rightarrow x(t) \in C \ \forall \ t$. Assume that $V(x) : D \to R$ is a continuous and differentiable Lyapunov function such that $\dot{V}(x) \leq 0$ for $x \in C$, i.e. $V(x)$ is negative semi-definite in C. Denote by E the set of all points in C such that $\dot{V}(x) = 0$. Denote by M the largest invariant set in E and its boundary by L^+, i.e. for $x(t) \in E$: $lim_{t\to\infty}x(t) = L^+$, or in other words L^+ is the positive limit set of E. Then every solution $x(t) \in C$ will converge to M as $t \to \infty$.

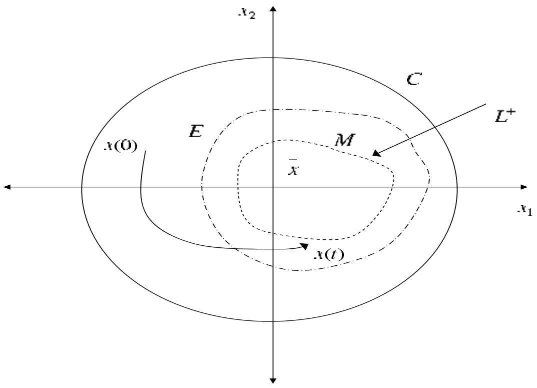


Fig. 11.1 LaSalle's theorem: C: invariant set, $E \subset C$: invariant set which satisfies $\dot{V}(x)=0$, $M \subset E$: invariant set, which satisfies $\dot{V}(x)=0$, & and which contains the limit points of $x(t) \in E$, L^+ the set of limit points of $x(t) \in E$

La Salle's theorem is applicable in the case of the multi-robot system and helps to describe more precisely the area round $\bar{x}$ to which the robot trajectories x^i will converge. A generalized Lyapunov function is introduced which is expected to verify the stability analysis based on Eq. (11.11). It holds that

$$\begin{array}{l} V(x)=\sum_{i=1}^M V^i(x^i)+\frac{1}{2}\sum_{i=1}^M\sum_{j=1,j\neq i}^M\{V_a(||x^i-x^j||-V_r(||x^i-x^j||)\} \Rightarrow \\ V(x)=\sum_{i=1}^M V^i(x^i)+\frac{1}{2}\sum_{i=1}^M\sum_{j=1,j\neq i}^M\{a||x^i-x^j||-V_r(||x^i-x^j||) \end{array}$$

and

$$\begin{array}{l} \nabla_{x^i}V(x)=[\sum_{i=1}^M\nabla_{x^i}V^i(x^i)]+\frac{1}{2}\sum_{i=1}^M\sum_{j=1,j\neq i}^M\nabla_{x^i}\{a||x^i-x^j||-V_r(||x^i-x^j||)\} \Rightarrow \\ \nabla_{x^i}V(x)=[\sum_{i=1}^M\nabla_{x^i}V^i(x^i)]+\sum_{j=1,j\neq i}^M(x^i-x^j)\{g_a(||x^i-x^j||)-g_r(||x^i-x^j||)\} \Rightarrow \\ \nabla_{x^i}V(x)=[\sum_{i=1}^M\nabla_{x^i}V^i(x^i)]+\sum_{j=1,j\neq i}^M(x^i-x^j)\{a-g_r(||x^i-x^j||)\} \end{array}$$

and using Eq. (11.1) with $\gamma^i(t)=1$ yields $\nabla_{x^i}V(x)=-\dot{x}^i$, and

$$\dot{V}(x)=\nabla_x V(x)^T\dot{x}=\sum_{i=1}^M\nabla_{x^i}V(x)^T\dot{x}^i \Rightarrow \dot{V}(x)=-\sum_{i=1}^M||\dot{x}^i||^2\leq 0 \tag{11.16}$$

Therefore, in the case of a quadratic cost function it holds $V(x)>0$ and $\dot{V}(x)\leq 0$ and the set $C=\{x:V(x(t))\leq V(x(0))\}$ is compact and positively invariant. Thus, by applying La Salle's theorem one can show the convergence of $x(t)$ to the set $M\subset C$, $M=\{x:\dot{V}(x)=0\}\Rightarrow M=\{x:\dot{x}=0\}$.

11.2 Particle Swarm Theory for Multi-robot Motion Planning

11.2.1 *The Particle Swarm Theory*

It has been shown that the distributed gradient algorithm can have satisfactory performance for the motion planning problem of multi-robot systems in the case of quadratic cost functions. An alternative method of distributed search for the goal position is the particle swarm algorithm which belongs to derivative-free optimization techniques [65].

The similarity between the particle models and the distributed gradient algorithms is noteworthy. Particle models consist of M particles with mass m^i, position x^i and velocity v^i. Each particle has a self-propelling force F^i. To prevent the particles from reaching large speeds, a friction force with coefficient K_v is introduced. In addition, each particle is subject to an attractive force which is affected by the proximity σ to other particles. This force is responsible for swarming. To prevent particle collisions a shorter-range repulsive force is introduced. In analogy to Eq. (11.3), the potential of the particles is given by

$$V_a - V_r = \sum_{j=1,j\neq i}^{M} a e^{-\frac{(|x^i - x^j|)^2}{\sigma^2}} - \sum_{j=1,j\neq i}^{M} b e^{-\frac{(|x^i - x^j|)^2}{\sigma^2}}$$

where a and b determine the strength of the attractive and the repulsive force respectively. Thus, the motion equations for each particle are [220]:

$$\begin{aligned} m_i \frac{\partial}{\partial t} v^i &= F^i - K_v v^i - \nabla(V_a - V_r) \\ \frac{\partial}{\partial t} x^i &= v^i \end{aligned} \tag{11.17}$$

The particle swarm algorithm evolves in the search space by modifying the trajectories of the independent vectors $x^i(t)$ which are called *particles*. Considering each robot as a particle, the new position of each robot $x^i(t+1)$ is selected taking into account the moves of the robot from its current position $x^i(t)$ and the best moves of the rest $M-1$ robots from their positions at time instant t, i.e. $x^j(t)\ j = 1, \cdots, M \vee j \neq i$.

Assume a set of M robots which is initialized at random positions $x^i(0)$ and which have initial velocities $v^i(0)$. The cost function of the i-th robot is denoted again by $V^i(x^i)$. The following parameters are defined (Fig. 11.2):

(i) $x^i(t)$ is the position vector of the i-th robot at time instant t,
(ii) $p^i(t)$ is the best position (according to V^i) to which the i-th robot can move, starting from its current position $x^i(t)$,
(iii) $p^g(t)$ is the best position (according to V^i) to which the neighbors of the i-th robot can move, starting from their current positions $x^j(t)\ j = 1, \cdots, M \vee j \neq i$.

Figure 11.2 describes the Von Neumann region round each mobile robot. The 2D-plane is divided into a grid of square cells and at the time instant k the robot is assumed to be cell at $c_{i,j}$. Then at time instant $k+1$ the robot can make one of the following moves: $c_{i+1,j}$, $c_{i+1,j-1}$,$c_{i+1,j+1}$, $c_{i-1,j}$, $c_{i-1,j-1}$, $c_{i-1,j+1}$, $c_{i,j-1}$, and $c_{i,j+1}$.

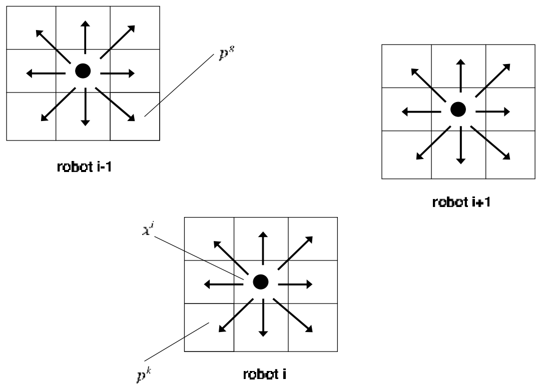


Fig. 11.2 Von Neumann region round each robot in the particle swarm algorithm

11.2.2 Stability of the Particle Swarm Algorithm

The primary concerns of the particle swarm theory are: (i) Each particle i should move in the direction of cost function decrease (negative gradient), taking into account the directions already examined by the neighboring particles, (ii) The velocity of each particle should approach 0 as time goes to infinity. To this end, the dynamic behavior of the particle swarm can be studied with the use of ordinary differential equations, following the analysis given in [65]. The position and the velocity update of the i-th particle is:

$$v^i(t+1) = v^i(t) + \phi_1(p^k - x^i) + \phi_2(p^g - x^i) \tag{11.18}$$

$$x^i(t+1) = x^i(t) + v^i(t+1) \tag{11.19}$$

Variable p^k denotes the best possible move of the k-th individual robot, taking into account that the Von-Neumann motion pattern of the robot permits movement to 8 neighboring cells. p^g is the best possible move of the neighboring robots, i.e. the movement that results in the largest decrease of the cost function. It holds that

$$\begin{aligned}
&v^i(t+1) = v^i(t) + \phi_1(p^k - x^i) + \phi_2(p^g - x^i) \Rightarrow \\
&v^i(t+1) = v^i(t) + \tfrac{\phi_1 p^i + \phi_2 p^g}{\phi_1 + \phi_2}(\phi_1 + \phi_2) - x^i(\phi_1 + \phi_2) \Rightarrow \\
&v^i(t+1) = v^i(t) + \phi\bar{p} - \phi x^i \Rightarrow v^i(t+1) = v^i(t) + \phi(\bar{p} - x^i)
\end{aligned}$$

The parameter ϕ_1 determines the contribution to the update of the position of the i-th robot of the term $p^k - x^i$ which denotes the distance between robot's position x^i and the position reached after its best possible move p^k. The parameter ϕ_2 determines the contribution to the update of the position of the i-th robot of the term $p^g - x^i$, which denotes the distance between the robot's position x^i and the position reached after the best possible move of its neighbors p^g. Through parameters ϕ_1 and

ϕ_2 the following parameters are defined: $\phi = \phi_1 + \phi_2$ and $\bar{p}^i = \frac{\phi_1 p^i + \phi_2 p^g}{\phi_1 + \phi_2}$.Thus, the following simplified equations can be derived

$$v^i(t+2) = v^i(t+1) + \phi(\bar{p} - x^i(t+1)) \tag{11.20}$$

$$x^i(t+1) = x^i(t) + v^i(t+1) \tag{11.21}$$

To refine the search in the solutions space, tuning through a constriction coefficient $\chi = \frac{\kappa}{\rho_2}$, $\kappa \in (0,1)$ is introduced in Eq. (11.19) and Eq. (11.18). In that case the particle swarm algorithm takes the following form:

$$\begin{aligned} v^i(t+1) &= \chi(v^i(t) + \phi(\bar{p}^i - x^i(t)) \\ x^i(t+1) &= \chi(x^i(t) + v^i(t+1)) \end{aligned} \tag{11.22}$$

It holds that

$$v^i(t+2) = v^i(t+1)(1-\phi) + \phi(\bar{p} - x^i(t)). \tag{11.23}$$

Subtracting Eq. (11.20) from Eq. (11.23) yields

$$v^i(t+2) + (\phi - 2)v^i(t+1) + v^i(t) = 0$$

Using the z-transform a frequency space expression of the above difference equation is $z^2 + (\phi - 2)z + 1 = 0$. Thus, the dynamic behavior of the particle depends on the roots of the polynomial $z^2 + (\phi - 2)z + 1$ which are $\rho_1 = 1 - \frac{\phi}{2} + \frac{\sqrt{\phi^2 - 4\phi}}{2}$ and $\rho_2 = 1 - \frac{\phi}{2} - \frac{\sqrt{\phi^2 - 4\phi}}{2}$. The general solution of the differential equation is

$$v^i(t) = c_1^i e^{\rho_1 t} + c_2^i e^{\rho_2 t} \tag{11.24}$$

The parameters c_1^i and c_2^i are random. In Eq. (11.24) the stability condition $lim_{t \to \infty} v^i(t) = 0$ is assured if $\phi \geq 4$ [65]. The pseudocode of the particle swarm algorithm is summarized as follows:

```
Initialize the robots population randomly: $x^i(0), v^i(0), i = 1,2,\cdots,M$

Do (until convergence to $x^*$)
{
    For $(i = 1; i < M; i++)$
     For all possible moves $p^i,\ i = 1,2,\cdots,N$ from $x^i$
     $p^k = arg\ min_{p^i}\{V^i(p^i)\}$
       For all particles $x^j,\ j = 1,2,\cdots,G$ in area $g$
      $p^g = arg\ min_{p^j}\{V^i(p^g)\}$;
        If $(V^i(p^k) < V^i(x^i))$
          $v^i(t+1) = v^i(t) + \phi_1(p^k - x^i) + \phi_2(p^g - x^i)$;
           $v^i(t+1) = sgn\{v^i(t+1) min[v^i(t+1), v_{max}]\}$;
           $x^i(t+1) = x^i(t) + v^i(t+1)$;
}
```

Parameters ϕ_1 and ϕ_2 are selected from a uniform distribution, taking into account the above mentioned convergence conditions. The robots velocity is bounded in the interval $\pm v_{max}$. Random weighting with the use of the parameters ϕ_1 and ϕ_2 helps to avoid local minima but can lead to explosion. It can be observed that:

• The term $\phi_1(p^k - x^i)$ stands for $\nabla_{x^i} V^i(x^i)$ of Eq. (11.1)
• The term $\phi_2(p^g - x^i)$ stands for the term $\sum_{j=1, j \neq i}^{M} g(x^i - x^j)$. It is assumed that in the neighborhood of the i-th particle the rest $M - 1$ particles are contained.
• The condition $V^i(p^k) < V^i(x^i)$ denotes movement in the direction of the negative gradient of the cost function $V^i(x^i)$.

11.3 Evaluation Tests for the Stochastic Search Algorithms

11.3.1 Convergence towards the Equilibrium

In the conducted simulation tests the multi-robot system consisted of 10 robots which were randomly initialized in the 2-D field. Theoretically, there is no constraint in the number of robots that constitute the robotic swarm. Of course, the number of robots can be increased and if it is sufficiently large then the obtained measurements will be also statistically significant. Two cases were distinguished: (i) motion in an obstacle-free environment (Fig. 11.3 -Fig. 11.4) and (ii) motion in an environment with obstacles (Fig. 11.5 - Fig. 11.6). The objective was to lead the robot swarm to the origin $[x_1, x_2] = [0, 0]$. To avoid obstacles, apart from the motion equations given in Sections 11.1 and 11.2 repulsive forces between the obstacles and the robots had to be taken into account.

Results about the motion of the robots in an obstacle-free 2D-plane were obtained. Fig. 3(a) describes the motion of the individual robots towards the goal state, in an obstacle-free environment, when the distributed gradient algorithm is applied, while Fig. 3(b) shows the motion of the robots in the same environment when particle swarm optimization is used to steer the robots. Fig. 4(a) demonstrates how the mean position of the multi-robot formation approaches the goal state when the motion takes place in an obstacle-free environment and the distributed-gradient algorithm is used to steer the robots. Fig. 4(b) shows the convergence of the average position of the robotic swarm to the equilibrium $[x^*, y^*] = [0, 0]$, in a 2D-plane without obstacles, when the steering of the robots is the result of particle swarm optimization. Next, motion of the robots in a 2D-plane that contains obstacles is studied. Fig. 5(a) demonstrates the motion of the individual robots towards the goal state, in an environment with obstacles, when the steering of the robots is the result of the distributed gradient algorithm. The robots are now also subject to attractive and repulsive forces due to the obstacles. Fig. 5(b) presents the convergence of the individual robots to the attractor $[x^*, y^*] = [0, 0]$, when the motion takes place again in an environment with obstacles and when the robots' steering is the result of the particle swarm optimization. Fig. 6(a) shows how the average position of the multi-robot formation reaches the equilibrium when the motion is performed in a

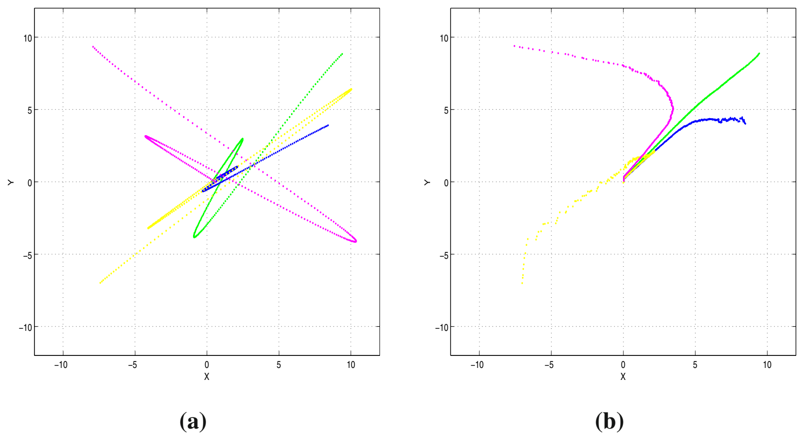


Fig. 11.3 (a) Distributed gradient and (b) Particle swarm with robots interaction in an obstacles-free environment, considering a quadratic cost function.

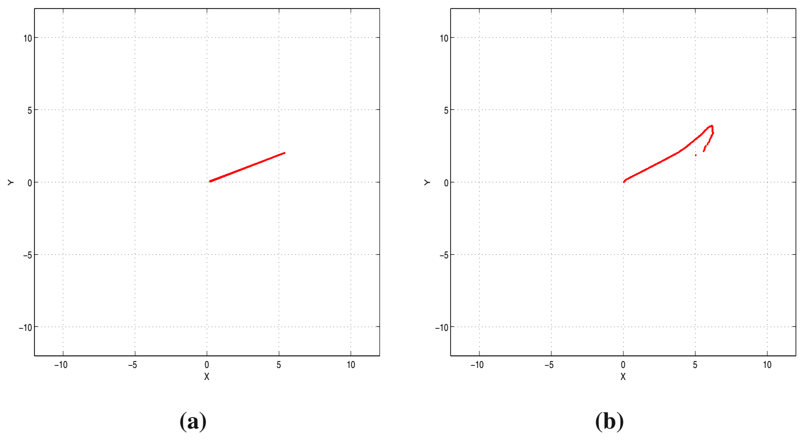


Fig. 11.4 (a) Distributed gradient and (b) Particle swarm with robots interaction in an obstacles-free environment: Trajectory of the mean of the multi-robot system.

2D-plane which contains obstacles and when the robots' trajectories are generated by the distributed gradient algorithm. Finally, in Fig.6(b) the motion of the mean of the multi-robot system towards the goal state is given, when the robots move again in an environment that contains obstacles, while their paths are generated by the particle swarm optimization algorithm.

When the multi-robot system evolved in an environment with obstacles, the interaction between the individual robots (attractive and repulsive forces) had to be loose, so as to give priority to obstacles avoidance. Therefore coefficients a and b in Eq. (11.3) were set to small values. The repulsive potential due to the obstacles was calculated by a relation similar to Eq. (11.3) after substituting x^j with x_o^j, where x_o^j was the center of the j-th obstacle.

In the case of distributed gradient the relative values of the parameters a and b that appear in the attractive and repulsive potential respectively, affected the performance of the algorithm. For $a > b$ the cohesion of the robotic swarm was maintained and abrupt displacements of the individual robots were avoided. In the particle swarm algorithm the area of possible moves round each robot was a Von Neumann one (Fig. 11.2). It was observed that the ratio $\lambda = \frac{\phi_1}{\phi_2}$ affected the performance of the algorithm. It was observed that large λ resulted in excessive wandering of the robots, while small λ led to the early formation of a robot cluster.

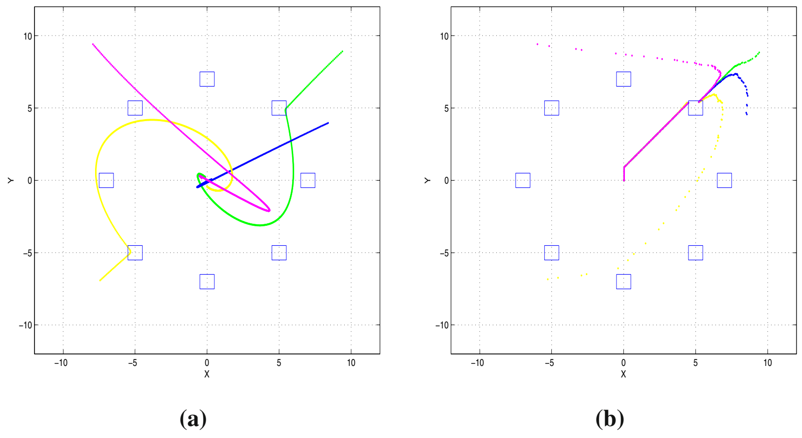


Fig. 11.5 (a) Distributed gradient and (b) Particle swarm with robots interaction in an environment with obstacles, considering a quadratic cost function.

Fig. 11.7 presents the variation of the Lyapunov functions of the robots when the motion takes place in an environment without obstacles, and the distributed gradient approach is applied to steer the robots towards the attractor. The Lyapunov function is given both in the case of the individual robots and in the case of the mean position of the multi-robot formation. It can be observed that the Lyapunov functions are not monotonous, and this change of the sign of the Lyapunov function's derivative is due to the fact that the robots' path encircle the target position several times before finally stabilizing at $[x^*, y^*] = [0, 0]$. This means that robots which were initially approaching fast the goal position had to make circles round the attractor in order to wait for those robots which had delayed. This maintained the cohesion of the multi-robot swarm.

Fig. 11.8 presents the variation of the Lyapunov function of the robots when the motion takes place in an environment with obstacles, and the distributed gradient approach is applied to steer the robots towards the attractor. The Lyapunov function is given again both in the case of the individual robots and in the case of the mean position of the multi-robot formation. In that case the Lyapunov functions tend to

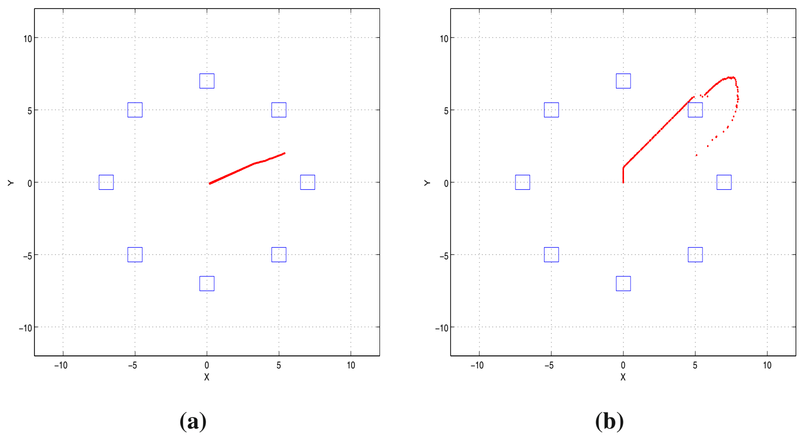


Fig. 11.6 (a) Distributed gradient and (b) Particle swarm with robots interaction in an obstacles-free environment: Trajectory of the mean of the multi-robot system.

become monotonous, i.e. continuously decreasing, and this is due to the fact that the interaction forces between the robots have been made weaker after suitable tuning of the coefficients α and β. This enables the robots to approach to the goal state following an almost linear trajectory, since the interaction forces that caused curving of the robots path and encircling of the attractor have now been diminished.

Fig. 11.9 shows the variation of the Lyapunov function of the robots and the Lyapunov function of the mean position of the multi-robot formation, when particle swarm optimization is applied to steer the robots towards the attractor and no obstacles are present in the 2D plane. The Lyapunov functions are monotonically decreasing which is due to the tuning of the interaction forces between the robots. Setting the interaction between the robots at low values enables almost linear convergence towards the goal state, which means that the quadratic error function keeps on decreasing as the motion of the robots continues.

Fig. 11.10 shows the variation of the Lyapunov function of the robots and of the Lyapunov function of the average position of the multi-robot system when particle swarm optimization is applied to steer the robots towards the attractor in the presence of obstacles. It can be observed again that the Lyapunov functions decrease monotonically until they become zero, and comparing to the motion in an obstacle-free environment the rate of approach towards the steady state $[e=0, \dot{e}=0]$ is larger. This is due to the fact that the interaction between the robots has become looser, thus the robots are enabled to approach rapidly the goal state without waiting for convergence of the rest of the swarm.

Regarding the significance of the mean and the variance of the multi-robot system for evaluating the behavior of the robotics swarm the following can be stated: although the average position of the multi-robot system is not always meaningful, for instance in case that the individual robots bypass a obstacle with equal probability from its left or right side, it is a useful parameter that helps to monitor this

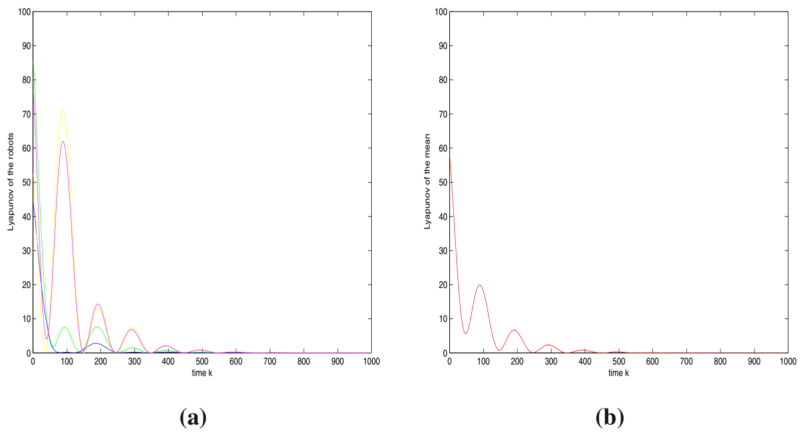


Fig. 11.7 Distributed gradient approach in an obstacles-free environment: (a) Lyapunov function of the individual robots and (b) Lyapunov function of the mean.

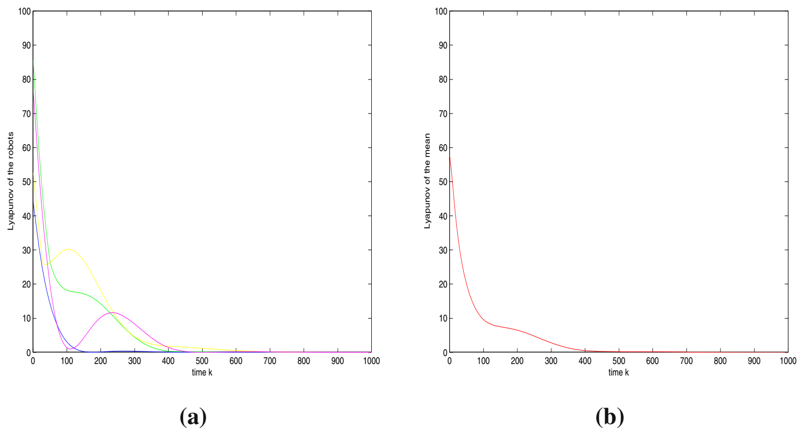


Fig. 11.8 Distributed gradient approach in an environment with obstacles: (a) Lyapunov function of the individual robots and (b) Lyapunov function of the mean.

many-body system. The mean and the variance of the multi-robot formation becomes particularly significant when the motion takes place at micro or nanoscale. In the latter case, all information about the behavior of the micro or nano particles swarm is contained in the mean position of the swarm and its variance.

Finally, simulation results about the motion of the multi-robot swarm in a workspace with polyhedric obstacles have been given. Thus additional evaluation for the performance of the proposed motion planning algorithms is obtained. The obstacles considered in the simulation experiments are not points but polyhedra which cover certain regions in the 2D plane. Therefore the attractive and repulsive

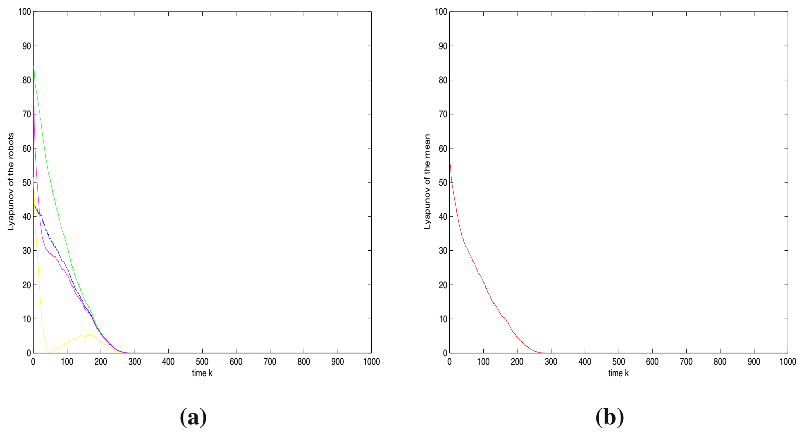


Fig. 11.9 Particle swarm approach in an obstacles-free environment: (a) Lyapunov function of the individual robots and (b) Lyapunov function of the mean.

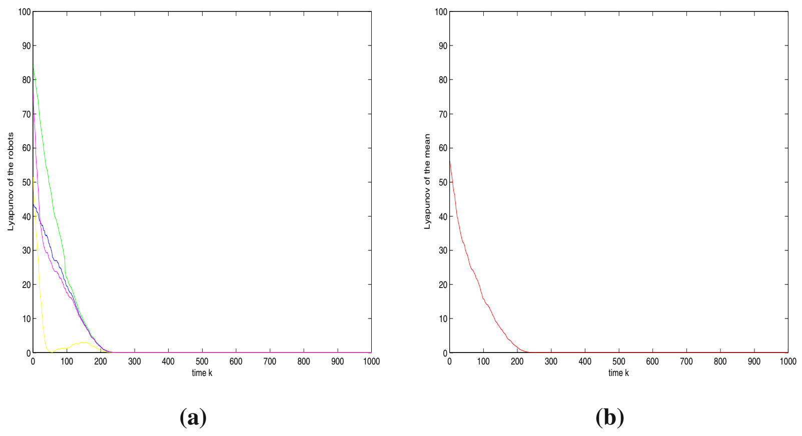


Fig. 11.10 Particle swarm approach in an environment with obstacles : (a) Lyapunov function of the individual robots and (b) Lyapunov function of the mean.

forces generated between the robots and obstacles affect the robots' trajectories and may also result in local minima. These results are depicted in Fig. 11.11 to Fig. 11.14.

It can be observed that in the case of motion in a 2D-plane with arbitrarily positioned polyhedric obstacles the distributed gradient algorithm results in smoother trajectories than the particle swarm optimization method. For even though the obstacles are not symmetrically placed in the 2D-plane and thus the aggregate force exerted on the robots in not zero, the multi-robot system in both cases converges

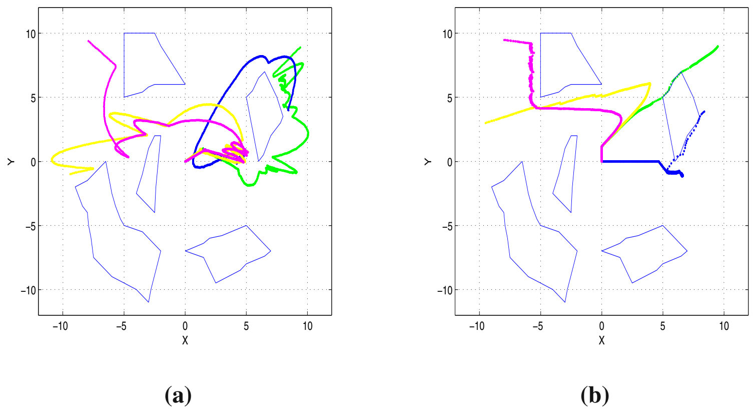


Fig. 11.11 (a) Distributed gradient and, (b) Particle swarm optimization for motion planning of the multi-robot system in a 2D-plane with polyhedric obstacles.

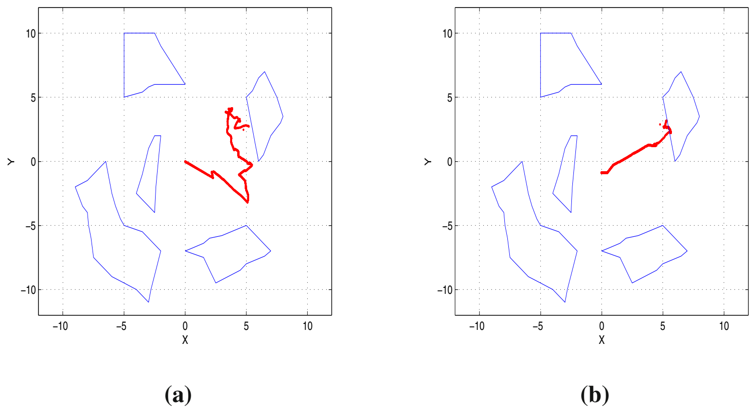


Fig. 11.12 (a) Distributed gradient, and (ii) Particle swarm optimization in a 2D-plane with polyhedric obstacles: trajectories of the mean of the multi-robot system.

to the origin $[x^* = 0, y^* = 0]$. The cost function is no longer a convex one. Local minima can be generated due to the proximity to obstacles and the attractive forces between the individual robots. For instance, local minima can be found in narrow passages between the obstacles. The inclusion of a stochastic term in the equations that describe the position update of the robots, i.e. in Eq. 11.1 and Eq. 11.19 may enable escape from minima. Moreover, suitable tuning of the attractive and repulsive forces that exist between the robots may also permit the robots to move away from local minima.

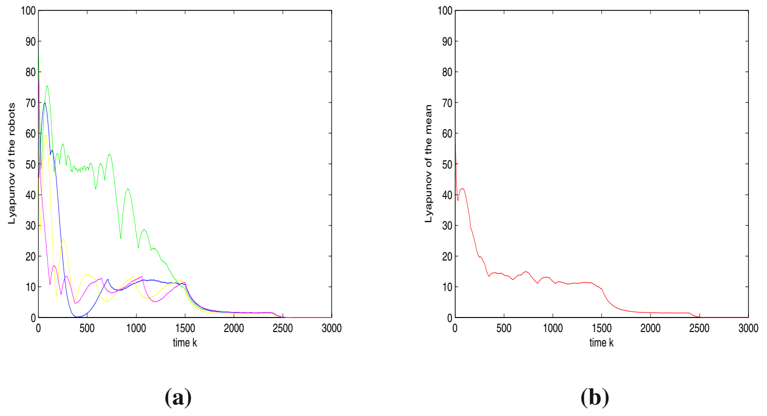


Fig. 11.13 Distributed gradient in a 2D-plane with polyhedric obstacles: (i) Lyapunov functions of the robots, (ii) Lyapunov function of the mean of the multi-robot formation.

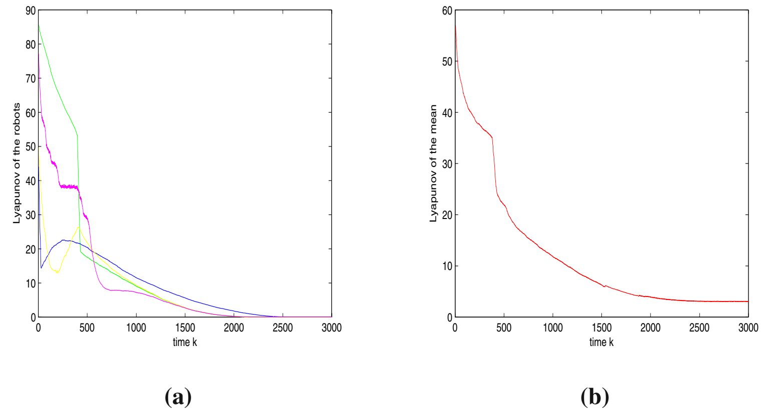


Fig. 11.14 Particle swarm optimization in a 2D-plane with polyhedric obstacles: (i) Lyapunov functions of the robots, (ii) Lyapunov function of the mean of the multi-robot formation.

11.3.2 Tuning of the Stochastic Search Algorithms

Regarding the tuning of the coefficients α and b which appear in Eq. (11.3) and which affect the trajectories of the individual robots the following should be noted: coefficient α describes the influence that the attractive potential $V_\alpha(x)$ has on the i-th robot (particle), and coefficient b describes the effect of the repulsive potential $V_r(x)$. These potential terms are respectively given by

$$V_\alpha(x) = \frac{1}{2}\alpha(x^i - x^j)^2,\ V_r(x) = \frac{1}{2}\sigma^2 b e^{||\frac{(x^i - x^{j2}}{\sigma^2}||} \tag{11.25}$$

The derivation of $V_\alpha(x)$ with respect to x^i generates an attractive spring force $F_\alpha(x)$ while the derivation of $V_r(x)$ with respect to x^i results in a repulsive force that contains a Gaussian term. These forces are explicitly given by:

$$F_\alpha(x) = \nabla V_\alpha(x) = \alpha(x^i - x^j),\ F_r(x) = \nabla V_b(x) = (x^i - x^j)be^{||\frac{(x^i-x^j)^2}{\sigma^2}||}$$

Similarly a repulsive force can describe the interaction between the i-th particle and the j-th obstacle, where the latter is assumed to be located at position x_o^j. Setting $\alpha \leq b$, or in general choosing the ratio α/b to be much smaller than 1 means that attractive forces between the individual robots are weaker than the repulsive forces. This may delay the convergence of the individual robots to the goal state $[x^*, y^*] = [0,0]$. One may also notice that the robots' paths become curved and encircle the goal state several times, and in that way the robots remain in sufficient distance.

Regarding the tuning of coefficient $\lambda = \phi_1/\phi_2$ which appears in the particle swarm optimization algorithm, this is performed ad hoc. Large values of λ resulted in excessive wandering of the robots, i.e. the robots encircled many times the goal state before finally converging to it. On the other hand small values of λ resulted in an early formation of a robots cluster, which may delay convergence to the goal state in a motion-plane with obstacles.

Regarding the convergence of the distributed gradient algorithm to the global minimum this can be sure only in the case of a convex cost function. The distributed gradient algorithm risks to be trapped to local minima, however the fact that the search in the solutions space is distributed and the existence of random terms to the individual gradient algorithms make possible the escape from these local minima. The tuning of the coefficients of the distributed gradient algorithm is performed ad-hoc and affects the paths in which the individual robots approach the goal state, as well as the shape of the trajectories the robots follow trying to deviate from obstacles.

Finally regarding the performance of the proposed motion planning algorithms, distributed gradient appears to be superior than particle swarm optimization for the following reasons: (i) distributed gradient stands for the mathematical formulation of a physical phenomenon (Brownian motion and interaction between diffusing particles [194]) while particle swarm optimization has no direct relation to physics laws, (ii) in distributed gradient, convergence to attractors is proved based on Lyapunov stability analysis. It is shown that the mean of the variables updated through the distributed gradient algorithm converges exactly to the equilibrium $[x^*, y^*] = [0,0]$, while the individual variables stay at a circle of radius ε round the attractor, as predicted by LaSalle's theorem. On the other hand there is no strict mathematical proof of the convergence of particle swarm optimization approach (iii) the fact that the particle swarm optimization method is a derivative-free optimization technique (while distributed gradient required the explicit calculation of derivatives) is moderated by the fact that particle-swarm optimization needs heuristic tuning of several parameters to converge to a fixed point, (iv) Regarding the capability to succeed global optimization, the distributed gradient algorithm is as powerful as the particle swarm optimization approach, since it can contain stochastic terms that enable escape from local minima.

In conclusion, both the distributed gradient and the particle swarm optimization method succeeded cooperative behavior of the robots without requirement for explicit coordination or communication. The performance of both methods was satisfactory, however distributed gradient was evaluated to have advantages over the particle swarm optimization mainly due to its sound convergence proof and the smooth motion patterns it produced in various environments.

Chapter 12
Optimization Methods for Target Tracking by Multi-robot Systems

Abstract. The chapter studies the two-fold optimization problem of distributed motion planning and distributed filtering for multi-robot systems. Tracking of a target by a multi-robot system is pursued assuming that the target's state vector is not directly measurable and has to be estimated by distributed filtering based on the target's cartesian coordinates and bearing measurements obtained by the individual mobile robots. The robots have to converge in a synchronized manner towards the target, while avoiding collisions between them and avoiding collisions with obstacles in the motion plane. To solve the overall problem, the following steps are followed: (i) distributed filtering, so as to obtain an accurate estimation of the target's state vector. This estimate provides the desirable state vector to be tracked by each one of the mobile robots, (ii) motion planning and control that enables convergence of the vehicles to the goal position and also maintains the cohesion of the vehicles swarm. The efficiency of the proposed distributed filtering and distributed motion planning scheme is tested through simulation experiments.

12.1 Distributed Motion Planning and Filtering in Multi-robot Systems

12.1.1 Target Tracking in Mobile Sensors Networks

The problem treated in this chapter is as follows: there are N mobile robots (unmanned ground vehicles) which pursue a moving target. The vehicles emanate from random positions in their motion plane. Each vehicle can be equipped with various sensors, such as odometric sensors, cameras and non-imaging sensors such as sonar, radar and thermal signature sensors. These vehicles can be considered as *mobile sensors* while the ensemble of the autonomous vehicles constitutes a *mobile sensor network* [91],[290],[291],[342]. At each time instant each vehicle can obtain a measurement of the target's cartesian coordinates and orientation. Additionally, each autonomous vehicle is aware of the target's distance from a reference surface measured in a cartesian coordinates system. Finally, each vehicle can be aware of the positions of the rest $N-1$ vehicles. The objective is to make the unmanned vehicles converge in a synchronized manner towards the target, while avoiding collisions between them and avoiding collisions with obstacles in the motion plane. To solve the overall problem, the following steps are necessary: (i) to perform distributed filtering, so as to obtain an estimate of the target's state vector. This estimate provides the desirable state vector to be tracked by each one of the unmanned vehicles, (ii) to design a suitable control law for the unmanned vehicles that will enable not only convergence of the vehicles to the goal position but will also maintain the cohesion of the vehicles ensemble.

Regarding the implementation of the control law that will allow the mobile robots to converge to the target in a coordinated manner, this can be based on the calculation of a cost (energy) function consisting of the following elements : (i) the cost due to the distance of the i-th mobile robot from the target's coordinates, (ii) the cost due to the interaction with the other $N-1$ vehicles, (iii) the cost due to proximity to obstacles or inaccessible areas in the motion plane. The gradient of the aggregate cost function defines the path each vehicle should follow to reach the target and at the same time assures the synchronized approaching of the vehicles to the target. In this way, the update of the position of each vehicle will be finally described by a gradient algorithm which contains an interaction term with the gradient algorithms that defines the motion of the rest $N-1$ mobile robots. A suitable tool for proving analytically the convergence of the vehicles' swarm to the goal state is Lyapunov stability theory and particularly LaSalle's theorem [325],[326].

Regarding the implementation of distributed filtering, the Extended Information Filter (EIF) and the Unscented Information Filter (UIF) are suitable approaches. Fusion of state estimates with the use of the Extended Information Filter and the Unscented Information Filter was analyzed in Chapter 8. In the Extended Information Filter there are local filters which do not exchange raw measurements but send to an aggregation filter their local information matrices (local inverse covariance matrices) and their associated local information state vectors (products of the local information matrices with the local state vectors) [341]. The Extended Information Filter performs fusion of the local state vector estimates which are provided by the local Extended Kalman Filters (EKFs), using the *Information matrix* and the *Information state vector* [217], [218], [252],[435]. The Information Matrix is the inverse of the state vector covariance matrix and can be also associated to the Fisher Information matrix [344]. The Information state vector is the product between the Information matrix and the local state vector estimate [379]. The Unscented Information Filter is a derivative-free distributed filtering approach which permits to calculate an aggregate estimate of the target's state vector by fusing the state estimates provided by Unscented Kalman Filters (UKFs) running at the mobile sensors. In the Unscented Information Filter an implicit linearization is performed through the approximation of the Jacobian matrix of the system's output equation by the product of the inverse of the estimation error covariance matrix with the cross-covariance matrix between the system's state vector and the system's output. Again the local information matrices and the local information state vectors are transferred to an aggregation filter which produces the global estimation of the system's state vector.

Using distributed EKFs and fusion through the Extended Information Filter or distributed UKFs through the Unscented Information Filter is more robust comparing to the centralized Extended Kalman Filter, or similarly the centralized Unscented Kalman Filter since, (i) if a local filter is subject to a fault then state estimation is still possible and can be used for accurate localization of the target, (ii) communication overhead remains low even in the case of a large number of distributed measurement units, because the greatest part of state estimation is performed locally and only information matrices and state vectors are communicated between the local filters, (iii) the aggregation performed also compensates for deviations in the state estimates of the local filters [342].

12.1.2 The Problem of Distributed Target Tracking

It is assumed that there are N mobile robots (unmanned vehicles) with positions $p_1, p_2, ..., p_N \in R^2$ respectively, and a target with position $x^* \in R^2$ moving on a plane (see Fig. 12.1). Each unmanned vehicle can be equipped with various sensors, cameras and non-imaging sensors, such as sonar, radar or thermal signature sensors. The unmanned vehicles can be considered as mobile sensors while the ensemble of the autonomous vehicles constitutes a mobile sensors network. The discrete-time target's kinematic model is in the form

$$\begin{aligned} x_t(k+1) &= \phi(x_t(k)) + L(k)u(k) + w(k) \\ z_t(k) &= \gamma(x_t(k)) + v(k) \end{aligned} \tag{12.1}$$

where $x \in R^{m \times 1}$ is the target's state vector and $z \in R^{p \times 1}$ is the measured output, while $w(k)$ and $v(k)$ are uncorrelated, zero-mean, Gaussian zero-mean noise processes with covariance matrices $Q(k)$ and $R(k)$ respectively. The operators $\phi(x)$ and $\gamma(x)$ are defined as $\phi(x) = [\phi_1(x), \phi_2(x), \cdots, \phi_m(x)]^T$, and $\gamma(x) = [\gamma_1(x), \gamma_2(x), \cdots, \gamma_p(x)]^T$, respectively.

At each time instant each mobile robot can obtain a measurement of the target's position. Additionally, each mobile robot is aware of the target's distance from a reference surface measured in an inertial coordinates system. Finally, each mobile sensor can be aware of the positions of the rest $N-1$ sensors. The objective is to make the mobile sensors converge in a synchronized manner towards the target, while avoiding collisions between them and avoiding collisions with obstacles in the motion plane. To solve the overall problem, the following steps are necessary: (i) to perform distributed filtering, so as to obtain an estimate of the target's state vector. This estimate provides the desirable state vector to be tracked by each one of the mobile robots, (ii) to design a suitable control law that will enable the mobile sensors not only convergence to the target's position but will also preserve the cohesion of the mobile sensors swarm (see Fig. 12.2).

The exact position and orientation of the target can be obtained through distributed filtering. Actually, distributed filtering provides a two-level fusion of the distributed sensor measurements. At the first level, local filters running at each mobile sensor provide an estimate of the target's state vector by fusing the cartesian

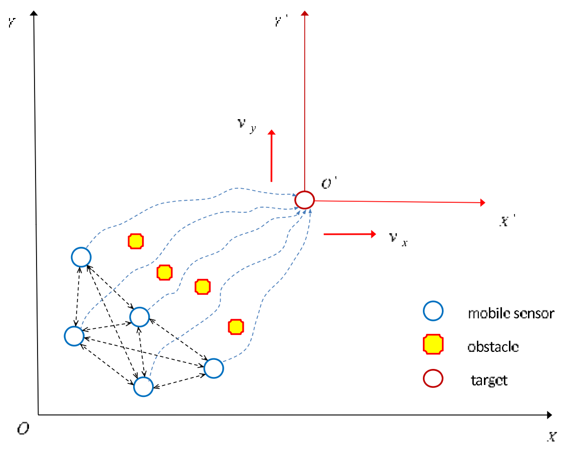


Fig. 12.1 Distributed target tracking in an environment with inaccessible areas.

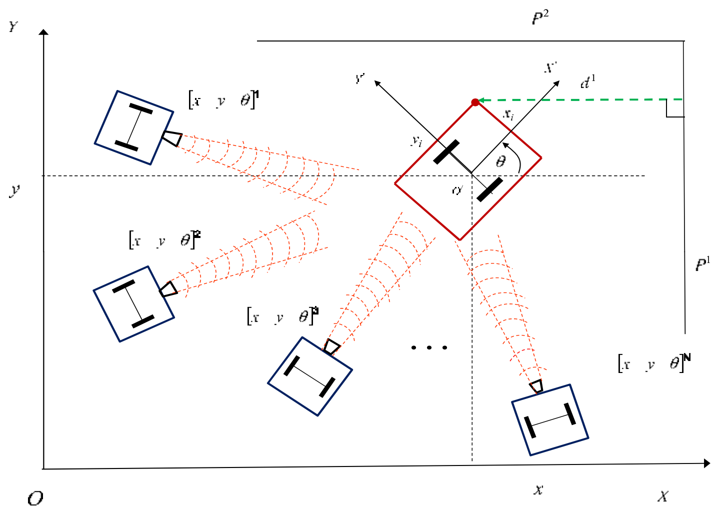


Fig. 12.2 Mobile robot providing estimates of the target's state vector, and the associated inertial and local coordinates reference frames

coordinates and bearing measurements of the target with the target's distance from a reference surface which is measured in an inertial coordinates system [440]. At a second level, fusion of the local estimates is performed with the use of the Extended Information Filter and the Unscented Information Filter. It is also assumed that the time taken in calculating the selection of data and in communicating between mobile robots is small, and that time delays, packet losses and out-of-sequence measurement problems in communication do not distort significantly the flow of the exchanged data.

Comparing to the traditional centralized or hierarchical fusion architecture, the network-centric architectures for the considered multi-robot system has the following advantages: (i) Scalability: since there are no limits imposed by centralized computation bottlenecks or lack of communication bandwidth, every mobile robot can easily join or quit the system, (ii) Robustness: in a decentralized fusion architecture no element of the system is mission-critical, so that the system is survivable in the event of on-line loss of part of its partial entities (mobile robots), (iii) Modularity: every partial entity is coordinated and does not need to possess a global knowledge of the network topology. However, these benefits are possible only if the sensor data can be fused and synthesized for distribution within the constraints of the available bandwidth.

12.1.3 Tracking of the Reference Path by the Target

The continuous-time target's kinematic model is assumed to be that of the unicycle robot described in Chapter 7 and is given by

$$\begin{array}{c}\dot{x}(t) = v(t)cos(\theta(t)) \\ \dot{y}(t) = v(t)sin(\theta(t)) \\ \dot{\theta}(t) = \omega(t)\end{array} \tag{12.2}$$

The target is steered by a dynamic feedback linearization control algorithm which is based on flatness-based control, as described in Chapter 3 [98],[216],[332],[438]:

$$\begin{array}{c}u_1 = \ddot{x}_d + K_{p_1}(x_d - x) + K_{d_1}(\dot{x}_d - \dot{x}) \\ u_2 = \ddot{y}_d + K_{p_2}(y_d - y) + K_{d_2}(\dot{y}_d - \dot{y}) \\ \dot{\xi} = u_1 cos(\theta) + u_2 sin(\theta) \\ v = \xi, \ \ \omega = \frac{u_2 cos(\theta) - u_1 sin(\theta)}{\xi}\end{array} \tag{12.3}$$

The dynamics of the tracking error is given by

$$\begin{array}{c}\ddot{e}_x + K_{d_1}\dot{e}_x + K_{p_1}e_x = 0 \\ \ddot{e_y} + K_{d_2}\dot{e}_x + K_{p_2}e_y = 0\end{array} \tag{12.4}$$

where $e_x = x - x_d$ and $e_y = y - y_d$. The proportional-derivative (PD) gains are chosen as K_{p_i} and K_{d_i}, for $i = 1,2$. The dynamic compensator of Eq. (12.3) has a potential singularity at $\xi = v = 0$, i.e. when the target is not moving. It has been noted that the occurrence of such a singularity is structural for non-holonomic systems. It is assumed that the target follows a smooth trajectory $(x_d(t), y_d(t))$ which is persistent, i.e. for which the nominal velocity $v_d = (\dot{x}_d^2 + \dot{y}_d^2)^{1/2}$ along the trajectory never goes to zero (and thus singularities are avoided). The conditions for avoiding singularities in the implementation of the mobile robot's control law have already been discussed in Chapter 7.

12.1.4 Convergence of the Multi-robot System to the Target

The kinematic model of the multi-robot system as well as its cohesion round the mean position of the robots ensemble was studied in Chapter 11, with the use of Lyapunov's theory and LaSalle's theorem. Now the convergence of the multi-robot system to the moving target's position is analyzed. The case of a convex quadratic cost function is examined, for instance

$$V^i(x^i) = \frac{A}{2}||x^i - x^*||^2 = \frac{A}{2}(x^i - x^*)^T(x^i - x^*) \tag{12.5}$$

where $x^* \in R^2$ denotes the target's position, while the associated Lyapunov function has a minimum at x^*, i.e. $V^i(x^i = x^*) = 0$. The distributed gradient algorithm is expected to converge to x^*. The robotic vehicles will follow different trajectories on the 2-D plane and will end at the target's position.

Using Eq.(12.5) yields $\nabla_{x^i}V^i(x^i) = A(x^i - x^*)$. Moreover, the assumption $\nabla_{x^i}V^i(x^i) \leq \bar{\sigma}$ can be used, since the gradient of the cost function remains bounded. The robotic vehicles will concentrate round $\bar{x}$ and will stay in a radius ε given by Eq. (11.12). The motion of the mean position $\bar{x}$ of the vehicles is

$$\begin{array}{c} \dot{\bar{x}} = -\frac{1}{N}\sum_{i=1}^N \nabla_{x^i}V^i(x^i) \Rightarrow \dot{\bar{x}} = -\frac{A}{N}\sum_{i=1}^N (x^i - x^*) \Rightarrow \\ \dot{\bar{x}} = -\frac{A}{N}\sum_{i=1}^N x^i + \frac{A}{N}Nx^* \Rightarrow \dot{\bar{x}} - \dot{x}^* = -A(\bar{x} - x^*) - \dot{x}^* \end{array} \tag{12.6}$$

The variable $e_\sigma = \bar{x} - x^*$ is defined, and consequently

$$\dot{e}_\sigma = -Ae_\sigma - \dot{x}^* \tag{12.7}$$

The following cases can be distinguished:

(i) The target is not moving, i.e. $\dot{x}^* = 0$. In that case Eq. (12.7) results in an homogeneous differential equation, the solution of which is given by

$$\varepsilon_\sigma(t) = \varepsilon_\sigma(0)e^{-At} \tag{12.8}$$

Knowing that $A > 0$ results into $lim_{t\to\infty}e_\sigma(t) = 0$, thus $lim_{t\to\infty}\bar{x}(t) = x^*$.

(ii) the target is moving at constant velocity, i.e. $\dot{x^*} = a$, where $a > 0$ is a constant parameter. Then the error between the mean position of the multi-robot formation and the target becomes

$$\varepsilon_\sigma(t) = [\varepsilon_\sigma(0) + \frac{a}{A}]e^{-At} - \frac{a}{A} \tag{12.9}$$

where the exponential term vanishes as $t \to \infty$.

(iii) the target's velocity is described by a sinusoidal signal or a superposition of sinusoidal signals, as in the case of function approximation by Fourier series expansion. For instance consider the case that $\dot{x^*} = b{\cdot}sin(at)$, where $a, b > 0$ are constant parameters. Then the nonhomogeneous differential equation Eq. (12.7) admits a sinusoidal solution. Therefore the distance $\varepsilon_{\sigma}(t)$ between the center of the multi-robot formation $\bar{x}(t)$ and the target's position $x^*(t)$ will be also a bounded sinusoidal signal.

12.2 Simulation Tests

12.2.1 Target Tracking Using Extended Information Filtering

The number of mobile robots used for target tracking in the simulation experiments was $N = 10$. However, since the mobile robots ensemble (mobile sensor network) is modular a larger number of mobile robot's could have been also considered. It is assumed that each mobile robot can obtain an estimation of the target's cartesian coordinates and bearing, i.e. the target's cartesian coordinates $[x,y]$ as well as the target's orientation θ. To improve the accuracy of the target's localization, the target's coordinates and bearing are fused with the distance of the target from a reference surface measured in an inertial coordinates system (see Fig. 12.2 and 12.3).

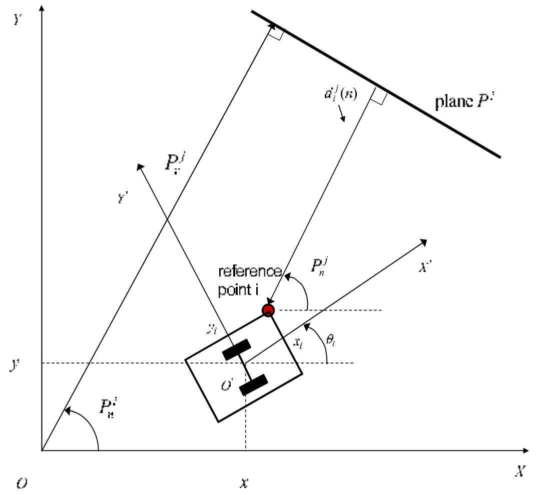


Fig. 12.3 Distance of the target's reference point i from the reference plane P^j, measured in the inertial coordinates system OXY

The inertial coordinates system OXY is defined. Furthermore the coordinates system $O'X'Y'$ is considered (Fig. 12.2). $O'X'Y'$ results from OXY if it is rotated by an angle θ (Fig. 12.2). The coordinates of the center of the wheels axis with respect to OXY are (x,y), while the coordinates of the reference point i that is mounted on the vehicle, with respect to $O'X'Y'$ are $x_i^{'}, y_i^{'}$. The orientation of the reference point with respect to $OX'Y'$ is $\theta_i^{'}$. Thus the coordinates of the reference point with respect to OXY are (x_i, y_i) and its orientation is θ_i, and are given by:

$$\begin{array}{c} x_i(k) = x(k) + x_i^{'} sin(\theta(k)) + y_i^{'} cos(\theta(k)) \\ y_i(k) = y(k) - x_i^{'} cos(\theta(k)) + y_i^{'} sin(\theta(k)) \\ \theta_i(k) = \theta(k) + \theta_i \end{array} \tag{12.10}$$

Each plane P^j in the robot's environment can be represented by P_r^j and P_n^j (Fig. 12.3), where (i) P_r^j is the normal distance of the plane from the origin O, (ii) P_n^j is the angle between the normal line to the plane and the x-direction.

The target's reference point i is at position $x_i(k), y_i(k)$ with respect to the inertial coordinates system OXY and its orientation is $\theta_i(k)$. Using the above notation, the distance of the reference point i, from the plane P^j is represented by P_r^j, P_n^j (see Fig. 12.3):

$$d_i^j(k) = P_r^j - x_i(k) cos(P_n^j) - y_i(k) sin(P_n^j) \tag{12.11}$$

Assuming a constant sampling period $\Delta t_k = T$ the measurement equation is $z(k+1) = \gamma(x(k)) + v(k)$, where $z(k)$ is the vector containing (i) target's coordinates and bearing estimates obtained from sensors on the mobile robot and (ii) the measurement of the target's distance to the reference surface. Variable $v(k)$ denotes a white noise sequence $\sim N(0, R(kT))$. The measure vector $z(k)$ can thus be written as

$$z(k) = [x(k) + v_1(k), y(k) + v_2(k), \theta(k) + v_3(k), d_1^j(k) + v_4(k)] \tag{12.12}$$

with $i = 1, 2, \cdots, n_s$, $d_i^j(k)$ to be the distance measure with respect to the plane P^j and $j = 1, \cdots, n_p$ to be the number of reference surfaces. By definition of the measurement vector one has that the output function $\gamma(x(k))$ is given by $\gamma(x(k)) = [x(k), y(k), \theta(k), d_1^1(k)]$.

To obtain the Extended Kalman Filter (EKF), the kinematic model of the target described in Eq. (12.2) is discretized and written in the discrete-time state-space form, as already described in Section 7.2 [332],[335].

The *measurement update* of the EKF is

$$\begin{array}{l} K(k) = P^-(k) J_\gamma^T(\hat{x}^-(k)) [J_\gamma(\hat{x}^-(k)) P^-(k) J_\gamma^T(\hat{x}^-(k)) + R(k)]^{-1} \\ \hat{x}(k) = \hat{x}^-(k) + K(k)[z(k) - \gamma(\hat{x}^-(k))] \\ P(k) = P^-(k) - K(k) J_\gamma^T P^-(k) \end{array}$$

The *time update* of the EKF is

$$\begin{array}{l} P^-(k+1) = J_\phi(\hat{x}(k)) P(k) J_\phi^T(\hat{x}(k)) + Q(k) \\ \hat{x}^-(k+1) = \phi(\hat{x}(k)) + L(k) U(k) \end{array}$$

where

$$L(k) = \begin{pmatrix} T cos(\theta(k)) & 0 \\ T sin(\theta(k)) & 0 \\ 0 & T \end{pmatrix}$$

and

$$J_\phi(\hat{x}(k)) = \begin{pmatrix} 1 & 0 & -v(k)sin(\theta)T \\ 0 & 1 & -v(k)cos(\theta)T \\ 0 & 0 & 1 \end{pmatrix}$$

while $Q(k) = diag[\sigma^2(k), \sigma^2(k), \sigma^2(k)]$, with $\sigma^2(k)$ chosen to be 10^{-3} and $\phi(\hat{x}(k)) = [\hat{x}(k), \hat{y}(k), \hat{\theta}(k)]^T$, $\gamma(\hat{x}(k)) = [\hat{x}(k), \hat{y}(k), \hat{\theta}(k), d(k)]^T$, i.e.

$$\gamma(\hat{x}(k)) = \begin{pmatrix} \hat{x}(k) \\ \hat{y}(k) \\ \hat{\theta}(k) \\ P_r^j - x_i(k))cos(P_n^j) - y_i(k)sin(P_n^j) \end{pmatrix} \tag{12.13}$$

The vector of the control input is given by $U(k) = [v(k), \omega(k)]^T$. Assuming one reference surface in the target's neighborhood one gets

$$J_\gamma^T(\hat{x}^-(k)) = [J_{\gamma_1}(\hat{x}^-(k)), J_{\gamma_2}(\hat{x}^-(k)), J_{\gamma_3}(\hat{x}^-(k)), J_{\gamma_4}(\hat{x}^-(k))]^T, \text{i.e.}$$

$$J_\gamma^T(\hat{x}^-(k)) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ -cos(P_n^j) & -sin(P_n^j) & \{x_i^{'}cos(\theta - P_n^j) - y_i^{'}sin(\theta - P_n^j)\} \end{pmatrix} \tag{12.14}$$

The use of EKF for fusing the target's coordinates and bearing measured by each mobile robot with the target's distance from a reference surface measured in an inertial coordinates system provides an estimation of the state vector $[x(t), y(t), \theta(t)]$ and enables the successful tracking of the target's motion by the individual mobile robots (mobile sensors).

The tracking of the target by the swarm of the autonomous vehicles was tested in the case of several reference trajectories, both for motion in an environment without obstacles as well as for motion on a plane containing obstacles. The proposed distributed filtering scheme enabled accurate estimation of the target's state vector $[x, y, \theta]^T$ through the fusion of the measurements of the target's coordinates and orientation obtained by each mobile robot with the measurement of the distance from a reference surface in an inertial coordinates frame. The state estimates provided by the Extended Kalman Filters running at each mobile sensor were fused into one single state estimate using Extended Information Filtering. The aggregate estimated coordinates of the target $(\hat{x}^*, \hat{y}^*)$ provided the reference setpoint for the distributed motion planning algorithm. Each mobile sensor was made to move along the path defined by $(\hat{x}^*, \hat{y}^*)$. The convergence properties of the distributed motion planning algorithm were described in subsection 12.1.4. The tracking of the target's trajectory by the mobile robots ensemble as well as the accuracy of the two-level sensor fusion-based estimation of the target's coordinates is depicted in Fig. 12.4 to Fig. 12.8. The target is marked as a thick-line rectangle and the associated reference trajectory is plotted as a thick line.

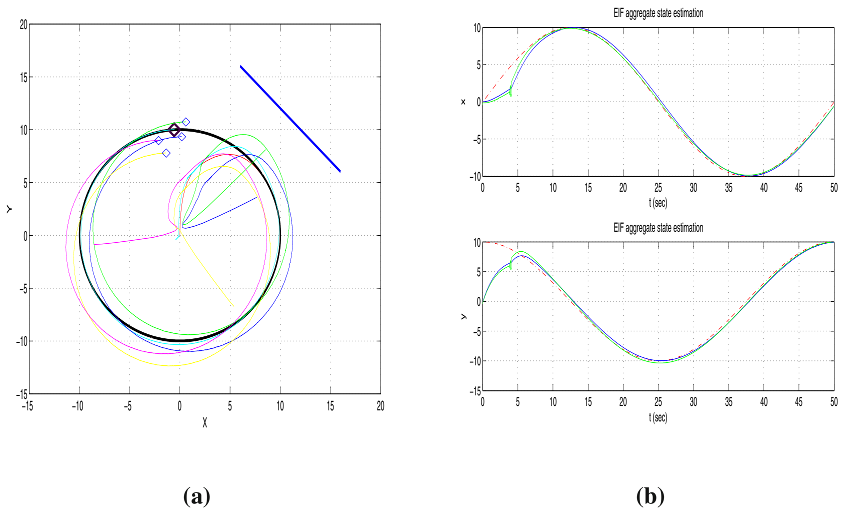
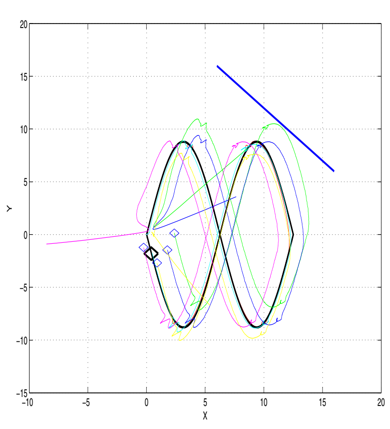


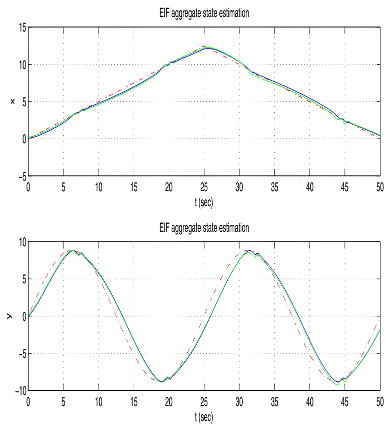
Fig. 12.4 (a) Distributed target tracking by an ensemble of autonomous vehicles when the target follows a circular trajectory in an obstacles-free motion space, (b) Aggregate estimation of the target's position with the use of Extended Information Filtering (continuous line) and target's reference path (dashed line).

It is noted that using distributed EKFs and fusion through the Extended Information Filter is more robust comparing to the centralized EKF since (i) if a local processing unit is subject to a fault then state estimation is still possible and can be used for accurate localization of the target, as well as for tracking of target's trajectory by the individual mobile sensors (autonomous vehicles), (ii) communication overhead remains low even in the case of a large number of distributed mobile sensors, because the greatest part of state estimation procedure is performed locally and only information matrices and state vectors are communicated between the local processing units, (iii) the aggregation performed on the local EKF also compensates for deviations in state estimates of local filters (which can be due to linearization errors).

12.2.2 Target Tracking Using Unscented Information Filtering

Next, the estimation of the target's state vector was performed using the Unscented Information Filter. Again, the proposed distributed filtering enabled precise estimation of the target's state vector $[x, y, \theta]^T$ through the fusion of measurements of the target's coordinates and bearing obtained by each mobile sensor with the distance of the target from a reference surface measured in an inertial coordinates system. The state estimates of the local Unscented Kalman Filters running at each mobile sensor (autonomous vehicle) were aggregated into a single estimation by the Unscented Information Filter. The estimated coordinates $[\hat{x}^*, \hat{y}^*]$ of the target were used to generate the reference path which was followed by the mobile sensors. The tracking





(a) **(b)**

Fig. 12.5 (a) Distributed target tracking by an ensemble of autonomous vehicles when the target follows an eight-shaped trajectory in an obstacles-free motion space, (b) Aggregate estimation of the target's position with the use of Extended Information Filtering (continuous line) and target's reference path (dashed line).

of the target's trajectory by the mobile robots ensemble as well as the accuracy of the two-level sensor fusion-based estimation of the target's position is shown in Fig. 12.9 to Fig. 12.13.

As previously analyzed, the Unscented Information Filter is a derivative-free distributed filtering approach in which the local Unscented Kalman Filters provide estimations of the target's coordinates using the update in-time of a number of suitably chosen sigma-points. Therefore, unlike the Extended Information Filter and the local Extended Kalman Filters contained in it, in the Unscented Information Filter there is no need to calculate Jacobians through the computation of partial derivatives. Additionally, unlike the case of local Extended Kalman Filters there is no truncation of higher order Taylor expansion terms and no linearization errors are introduced at the local estimators. In that sense the Unscented Information Filter provides a robust distributed state estimation and enables accurate tracking of the target by the mobile sensors (autonomous vehicles).

A conclusive remark is that using distributed EKFs or UKFs and fusion through the Extended Information Filter and the Unscented Kalman Filter respectively, is more robust comparing to the centralized EKF and centralized UKF since (i) if a local processing unit is subject to a fault then state estimation of the target's position is still possible and can be used for planning of the mobile robot's motion towards the target, (ii) communication overhead remains low even in the case of a large number of mobile robots, because the greatest part of state estimation is performed locally and only information matrices and state vectors are communicated between the local processing units.

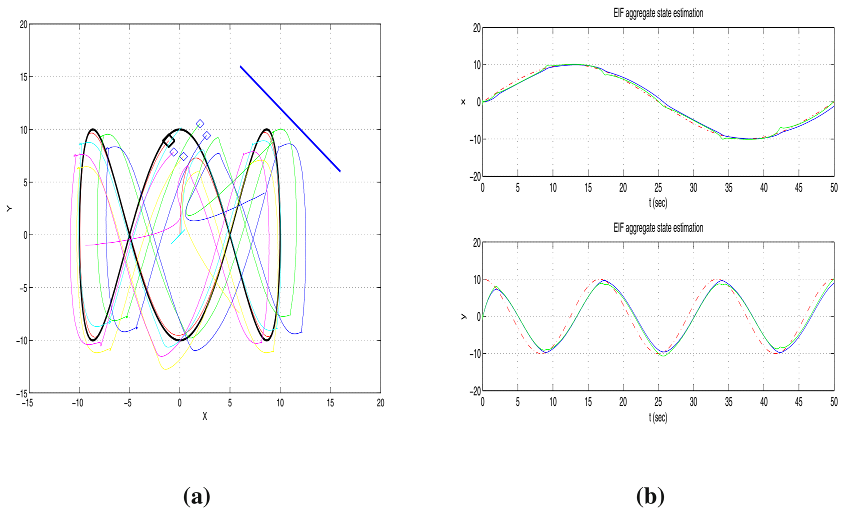


Fig. 12.6 (a) Distributed target tracking by an ensemble of autonomous vehicles when the target follows a curve-shaped trajectory in an obstacles-free motion space, (b) Aggregate estimation of the target's position with the use of Extended Information Filtering (continuous line) and target's reference path (dashed line).

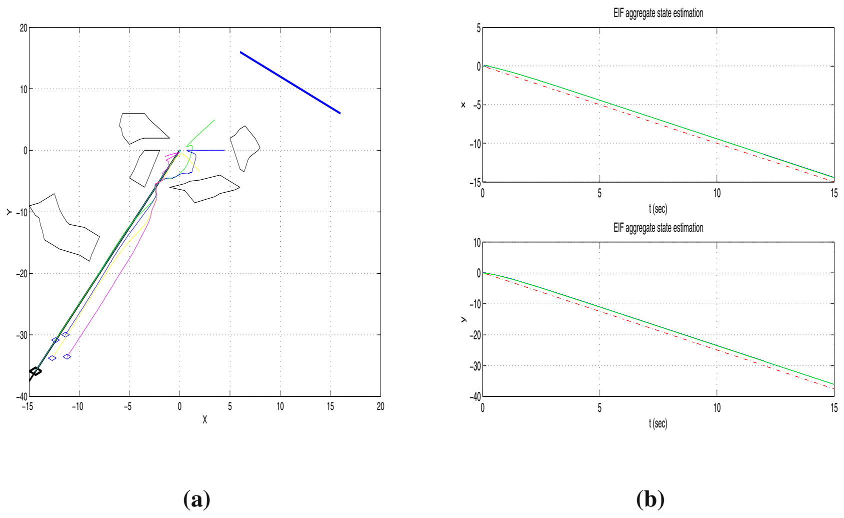
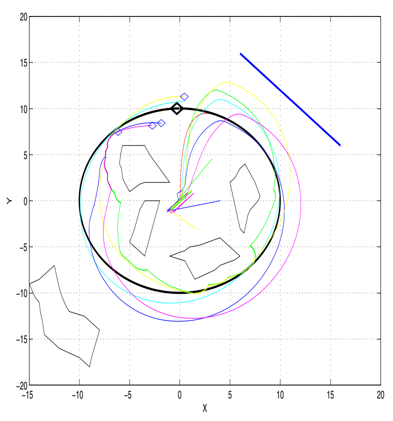
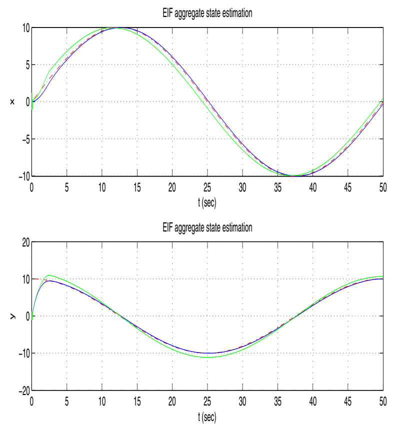


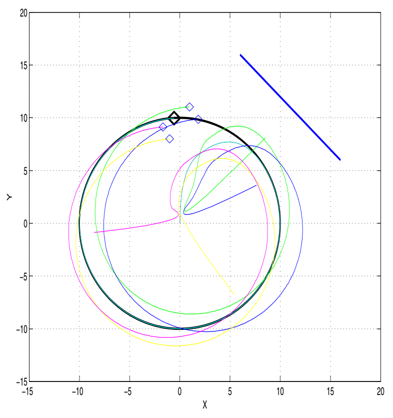
Fig. 12.7 (a) Distributed target tracking by an ensemble of autonomous vehicles when the target follows a line path in a motion space with obstacles, (b) Aggregate estimation of the target's position with the use of Extended Information Filtering (continuous line) and target's reference path (dashed line).

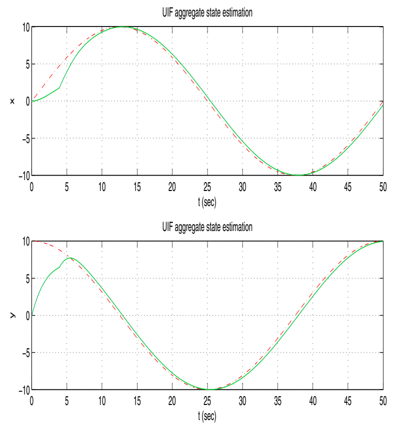




(a) **(b)**

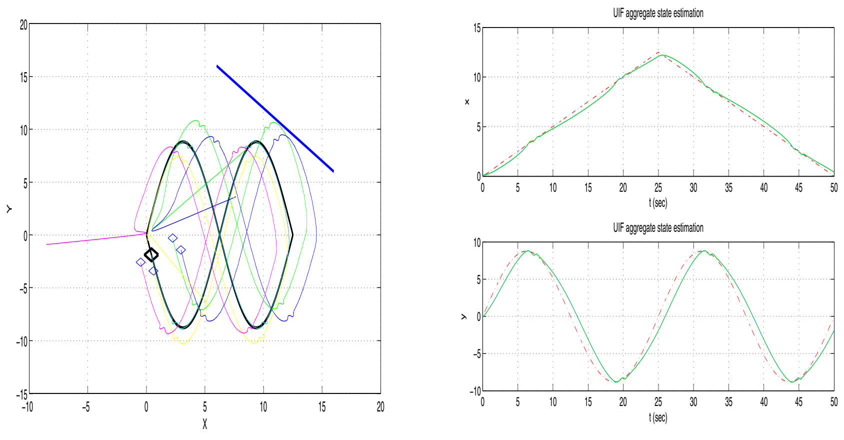
Fig. 12.8 (a) Distributed target tracking by an ensemble of autonomous vehicles when the target follows a circular path in a motion space with obstacles, (b) Aggregate estimation of the target's position with the use of Extended Information Filtering (continuous line) and target's reference path (dashed line).





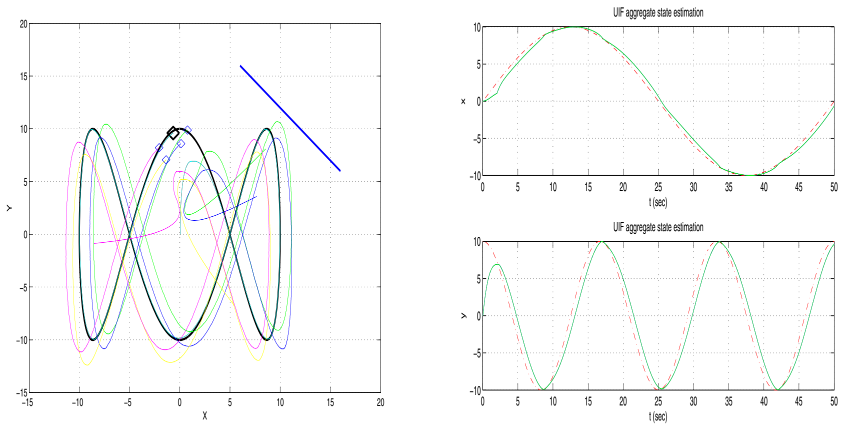
(a) **(b)**

Fig. 12.9 (a) Distributed target tracking by an ensemble of autonomous vehicles when the target follows a circular trajectory in an obstacles-free motion plane, (b) Aggregate estimation of the target's position with the use of Unscented Information Filtering (continuous line) and target's reference path (dashed line)



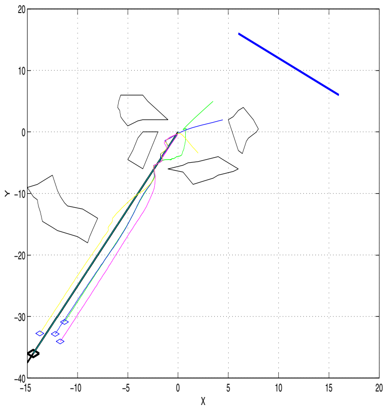
(a) (b)

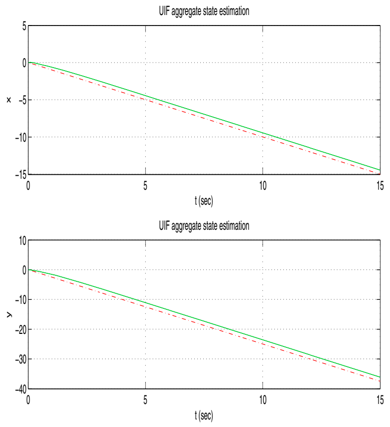
Fig. 12.10 (a) Distributed target tracking by an ensemble of autonomous vehicles when the target follows an eight-shaped trajectory in an obstacles-free motion plane, (b) Aggregate estimation of the target's position with the use of Unscented Information Filtering (continuous line) and target's reference path (dashed line).



(a) (b)

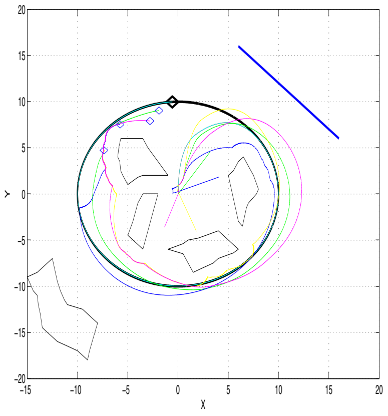
Fig. 12.11 (a) Distributed target tracking by an ensemble of autonomous vehicles when the target follows a curve-shaped trajectory in an obstacles-free motion plane, (b) Aggregate estimation of the target's position with the use of Unscented Information Filtering (continuous line) and target's reference path (dashed line).

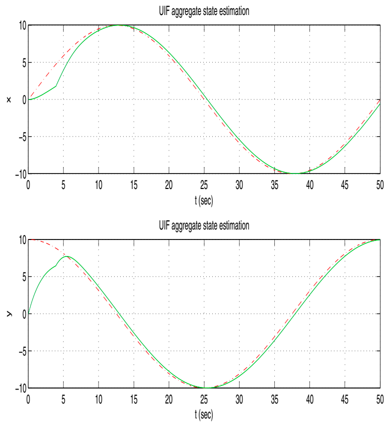




(a) **(b)**

Fig. 12.12 (a) Distributed target tracking by an ensemble of autonomous vehicles when the target follows a line path in a motion plane with obstacles, (b) Aggregate estimation of the target's position with the use of Unscented Information Filtering (continuous line).





(a) **(b)**

Fig. 12.13 (a) Distributed target tracking by an ensemble of autonomous vehicles when the target follows a circular path in a motion plane with obstacles, (b) Aggregate estimation of the target's position with the use of Unscented Information Filtering (continuous line) and target's reference path (dashed line).

Chapter 13
Optimization Methods for Industrial Automation

Abstract. Evolutionary algorithms are powerful optimization methods which in several cases can go beyond nonlinear programming optimization techniques and can be successfully applied to complex optimization problems. In this chapter, a genetic algorithm with a new crossover operator is developed to solve the warehouse replenishment problem. The automated warehouse management is a multi-objective optimization problem since it requires to satisfy goals and performance indexes that are usually conflicting with each other. The decisions taken must ensure optimized usage of resources, cost reduction and better customer service. It is shown that the proposed genetic algorithm produces Pareto-optimal permutations of the stored products.

13.1 Multi-objective Optimization for Industrial Automation

13.1.1 The Warehouse Replenishment Problem

Warehouse replenishment is the process of moving inventory from distant back-up locations, to locations that are easily accessible, i.e. the relocation of products closer to specific delivery points, or collection points or picking locations. Thus the problem is essentially to find which products would be most suitable to occupy the picking locations or be placed near the collection points,so that there is maximum gain in time and effort during running of the warehouse. The replenishment task can be carried out when the warehouse is idle and helps to reduce the time required for the products' shipment when the transactions start again. Furthermore, the warehouse operator can initiate a replenishment procedure each time he realises that inventory at the picking positions has fallen bellow a predefined level.

The replenishment task can be viewed as a *multi-objective optimization problem* (MOP) since it tries to allocate the inventory using objectives that can be conflicting with each other [57],[148],[162],[219],[479]. Such objectives are the distance from the delivery points, the distance from the collection points, the distance from the picking locations, the expiration date of the product and its seasonal demand [430]. The solution of a MOP is often given in the Pareto-optimality sense. In simple words a solution $x \in X$ is Pareto optimal if it cannot result in further improvement of an objective $f_i(x)$ without causing the degradation of another objective.

The aim is to find as many Pareto-optimal solutions as possible. The operator of the warehouse is going to select among these solutions according to his preferences, i.e. according to the significance he gives to each one of the objective functions $f_i(x)$ To this end the *weighting approach* is employed. A cost function $J(x) = \sum_{i=1}^{n} w_i f_i(x)$ is minimized with respect to x. The weights w_i represent the importance that is attributed to each objective $f_i(x)$ The minimum x^* of $J(x)$ is found through a genetic algorithm. For different weights w_i's a different optimum x^* is obtained. All these optimums are also Pareto-optimal solutions.

The choice of the weights w_i's can be done either in a stochastic or a deterministic way [468]. If the aim is to have Pareto-optimal solutions that are uniformly scattered in the cost functions space one has to perform a stochastic tuning of the weights w_i's as described in subsection 13.2.3. On the other hand if the warehouse administrator knows approximately the acceptable values of the objectives $f_i(x)$'s, a fuzzy rule-base can be used to choose the different values for the w_i's.

13.1.2 Multi-objective Optimization Problems

Multi-objective Optimization Problems (MOP) are an extension of the single-objective (scalar) optimization. In MOP, a number of conflicting objective functions have to be optimized simultaneously. An ideal solution, at which each objective function gets its optimal value usually does not exist due to the conflicting nature of objective functions. Thus, a different definition of optimality is required. The solution of a MOP is associated with the definition of Pareto-optimal solutions [57].

The concept of Pareto optimality was introduced at the turn of the previous century by the Swiss economist Pareto. A solution is said to be *Pareto optimal* (also referred in the literature as *non-inferior* or efficient or *non-dominated*) if the value of any objective function $f_i(x)$ cannot be improved without degrading at least one of the other objective functions. Two Pareto-optimal solutions are not comparable without some preference ordering (provided by the decision maker).

To find the best compromise solution of a MOP, two general stages are followed: (a) find the set or a representative subset of Pareto-optimal solutions, (b) appropriately apply the decision makers preference to choose the best compromise solution from the generated set. A genetic algorithm can be used to find a representative set of Pareto-optimal solutions while the tuning of the weighting vector of the objective function makes possible to enrich the optimal solutions set [148],[219],[479].

13.1.3 The Pareto-optimality Principles

A Multi-objective Optimization Problem is defined as [57]:

$$\min_{x \in X} f(x) \tag{13.1}$$

where $f(x) = (f_1(x), \dots, f_n(x))$ is a vector of n real-valued objective. Defining $U^N = R^N$ in case that x is a real-valued vector and $U^N = D^N$ in case that x is a discrete variable,the following notation is introduced: $x \in U^N$ is a vector of N-dimensional decision variables, $X = x | x \in U^N$ is the feasible solution set, $g_i(x)$ with $i = 1, 2, \dots, q$ and $x \in S$ is a function representing the ith constraint, S is a subset of U^N representing the definition set of the constraints.

An alternative x_1 is at least as preferable as x_2 if the value of each objective function at x_1 is not larger than that at x_2: Formally, an alternative x^* is said to be a *Pareto-optimal solution* of the MOP stated in Eq. 13.1, if and only if there exists no other feasible alternative $x \in X$ such that $f(x) \leq f(x^*)$, meaning that $f_j(x) \leq f_j(x^*)$ for all $j = 1, \dots, n$ n with strict inequality holding for at least one j. A solution x_1 is said to be *dominated or inferior or inefficient*, if there exists a $x \in X$ such that $f(x) \leq f(x_1)$.

In MOPs two decision vectors x_1 and x_2 have three possibilities regarding the $\geq$ relation (in contrast to single-objective optimization problems (SOPs)):

$$\begin{aligned} f(x_1) \geq f(x_2) \quad & (x_1 \ \ dominates \ \ x_2), \\ f(x_2) \geq f(x_1) \quad & (x_2 \ \ dominates \ \ x_1), \\ f(x_1) \ngeq f(x_2) \wedge f(x_2) \ngeq f(x_1) \quad & (x_1 \ \ is \ \ indifferent \ \ to \ \ x_2) \end{aligned} \tag{13.2}$$

In simple words, a MOP solution x is *Pareto optimal* in the sense that, it cannot be improved in any objective $f_i(x)$, without causing a degradation in at least another objective $f_j(x)$.

In Fig. 13.1 a 2-D MOP is depicted. Both objective functions $f_1(x)$ and $f_2(x)$ are assumed to improve when moving far from the origin. Since $f_1(B) > f_1(C)$ and $f_2(B) > f_2(C)$ the solution represented by point B is better than the solution represented by point C. On the other hand, it holds $f_1(B) > f_1(E)$ and $f_2(E) >$

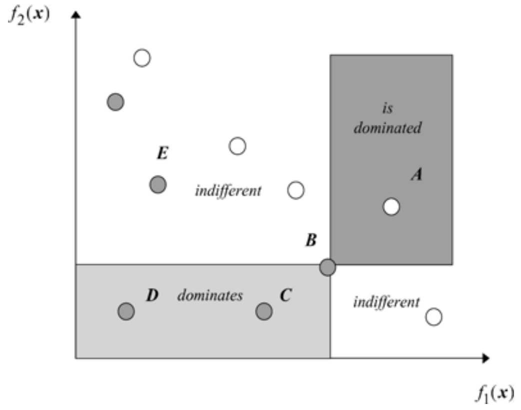


Fig. 13.1 Possible relations of solutions in the cost functions space

$f_2(B)$ Therefore, the solution represented by the point B is indifferent to the solution represented by the point E.

13.1.4 Replenishment as a Pareto Optimization Problem

In the case of the warehouse replenishment problem various objectives can be included in the objective vector $f(x)$. These can be:

$f_1(x)$: Distance of the products from the delivery points.
$f_2(x)$: Distance of the products from the collection points.
$f_3(x)$: Distance of the products from the picking locations.
$f_4(x)$: Cost to keep the products with close expiration date far from the collection points.
$f_5(x)$: Cost to keep the products with increased seasonal demand far from the delivery points.

More objectives can be added. Vector x denotes the permutation of the products in the storage positions (Fig. 13.2). Each one of its components $x_i,\ i = 1,2,\cdots,N$ corresponds to a storage position, and takes its value from the discrete set $C = c_1, c_2, \ldots, c_M$ of products codes that exist in the warehouse. A product code c_i is assigned to an item according to its content (e.g. food,beverages, chemicals etc.), its size (small, medium, large,etc.) and its package type (e.g. box, pallete, bulk, etc.).

The replenishment task is also subject to several constraints $g_i(x)$ such as:

$g_1(x)$: Products of specific types are not allowed to be placed in adjacent locations (e.g. foods or beverages with chemicals).
$g_2(x)$: Products of large size cannot be moved to storage positions for small sizes.
$g_3(x)$: Products of a specific package type (e.g. pallets) cannot be moved to storage positions for a different package type (e.g. boxes) etc.

Thus in the case of the warehouse replenishment the constraints definition set S contains the prohibited allocations of products to the storage positions.

13.1.5 Approaches to Obtain Pareto-optimal Solutions

A MOP may have multiple Pareto-optimal solutions. Different decision makers with different preferences may select different Pareto-optimal solutions. It is desirable to find all the Pareto-optimal solutions so that the decision maker (in our case the warehouse administrator)can select the best one based on his preference.

Assume for instance the 2-D MOP described by the cost functions $[f_1(x), f_2(x)]$. The Pareto-optimal solutions can be denoted as points in the cost functions space defined by the axes $f_1(x)$ and $f_2(x)$. The line that connects the Pareto-optimal solutions in the cost function space defines the Pareto frontier. This is shown in Fig. 13.3.

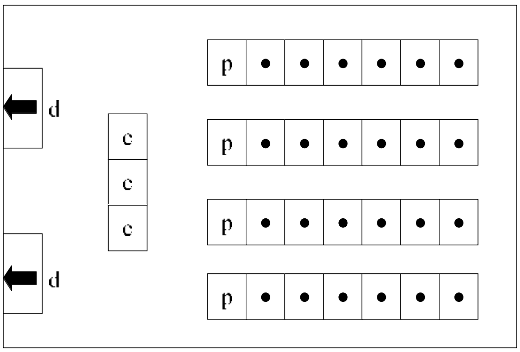


Fig. 13.2 Sketchy map of a warehouse. d: delivery points, c: collection points, p: picking locations, storage position

The most common strategy to generate Pareto-optimal solutions is to describe them in terms of some appropriate SOPs. Common classes of SOPs used for this purpose are the Lagrangian and the weighting approach [57], [219]. The weighting approach was introduced in [468], while the Lagrangian approach was elaborated in [27]. Formally, the weight characterization of the MOP in (1) is

$$J(x) : \min_{x \in X} W^T f(x) = \sum_{j=1}^{n} w_j f_j(x), \tag{13.3}$$

where $W = \{w | w \in R^N\}, w_j \geq 0, j = 1, 2, \ldots, n$ and $\sum_{j=1}^{n} w_j = 1$. The Lagrangian formulation is derived from the above relation. If for a specific $1 \leq k \leq n, w_k > 0$, letting $u_j = w_j / w_k$ for all $j \neq k, j = 1, \ldots, n$ yields a Lagrangian formulation:

$$J_k(x) = \min_{x \in X} f_k(x) + \sum_{j \neq k} u_j f_j(x) \tag{13.4}$$

where $u \in U_k = \{u | u = (u_1, \ldots, u_{(k-1)}, u_{(k+1)}, \ldots, u_n)^T \geq 0\}$.

It can be proven that:

Lemma. If x_0 *is a global optimum of* $J(x)$ or $J_k(x)$ *means that* x_0 *is also a Pareto optimum.*

Proof. Assume that x is a global optimum of $J(x)$ but not a Pareto optimum. Then there exists a solution x' that dominates x. Thus, without loss of generality, it can be assumed that $f_1(x') > f_1(x)$, and for $i = 2, \ldots, n$ holds $f_i(x') \geq f_i(x)$. Assuming also that the weights vector $[w = w_1, w_2, \ldots, w_n]$ is strictly positive, one gets $J(x') > J(x)$,i.e. x' is a global optimum, which contradicts the initial assumption.

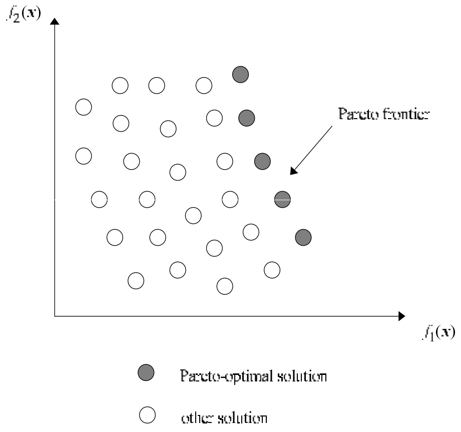


Fig. 13.3 Pareto-optimal solutions and Pareto frontier over a 2-D cost function space

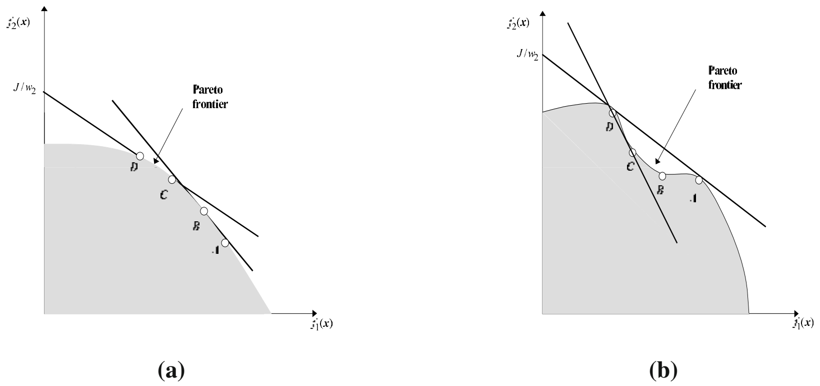


Fig. 13.4 Interpretation of the weighting method for (a) convex and (b) non-convex cost function.

The converse does not hold always, i.e. there may be Pareto optimums that do not coincide with any of the optimums of $J(x)$. This happens when $J(x)$ is not convex. If $J(x)$ is not convex then for some Pareto-optimal solutions x_0 it may be impossible to find a weights vector w,such that x_0 is also a global optimum of $J(x)$.

A graphical interpretation of the relation between global optimality and Pareto optimality is given in subsection 13.1.6.

13.1.6 Graphical Representation of Pareto Optimal Solution

As stated in Section 2.4 if $J(x)$ is convex then all the solutions of $J(x)$ are also Pareto-optimal solutions. However in the case that $J(x)$ is not convex there maybe some Pareto-optimal solutions that are not contained in the set of optimal solutions of $J(x)$. This is depicted in Figs. 13.4a and b.

Assume that the Pareto frontier is the border of the shaded area and that the points A,B,C and D represent different Pareto-optimal solutions. Assume also that x is a global optimum (maximum) of $J = w_1 f_1(x) + w_2 f_2(x)$ for certain weights w_1 and w_2. Then the tangent $f_2(x) = -(w_1/w_2) f_1(x) + (J/w_2)$ at x should be above all points of the shaded area. If $J(x)$ is convex then for every point of the Pareto frontier there is a combination of weights w_1 and w_2 such that the tangent line leaves all other solutions below it. In that case, all Pareto-optimal solutions can be identified because they coincide with a global maximum (see Fig. 13.4a). On the contrary if $J(x)$ is not convex then the tangent at some points of the Pareto frontier intersects the shaded area (see Fig. 13.4b, points B and C). For a Pareto optimum x_0 it is not always possible to find a weights combination (w_1, w_2) such that the tangent at the point $(f_1(x_0), f_2(x_0))$ leaves all other points $(f_1(x), f_2(x))$ below it. Thus, these points x_0 will not be found through the optimization of $J(x)$.

According to the above, the solutions found through the optimization of $J(x) = \sum_{(i=1)}^{n} w_i f_i(x)$ are also Pareto optimums. To acquire many Pareto-optimal solutions one should try to optimize $J(x)$ with a variety of weight vectors $[w_1, w_2, \ldots, w_n]$ This problem is addressed in Section 3.3.

13.2 Genetic Algorithms in the Search of Pareto-optimal Solutions

13.2.1 Basic Principles of Evolutionary Algorithms

An evolutionary algorithm (EA) consists of 3 elements: (i) There exists a set of candidate solutions (population), (ii) The population undergoes a selection process, (iii) New candidate solutions are created through *genetic operators*, usually *crossover and mutation*. An initial population is generated at random. An individual can be either a bit vector or a real valued vector and encodes a possible solution. A loop of the steps (ii) and (iii) is executed for a certain number of times. Each iteration of the loop is called *generation*.

The loop terminates after a number of generations, or if a sufficient solution has been produced. The selection process can be either deterministic or stochastic. Low-quality solutions (*individuals*) are removed from the population, while high-quality individuals are reproduced. The quality of an individual is represented by a scalar cost function (*fitness*).

Crossover and mutation produce new solutions in the search space by the variation of the existing ones. Recombining a number of parents the crossover operator generates a number of children. A crossover probability can be assigned to this operator. The mutation operator in turn, modifies a small sample of individuals that is selected according to a mutation probability.

13.2.2 Evolutionary Algorithms for Multi-objective Optimization

In the following, the basic structure of a GA for a MOP is formalized [369],[479]. The population P at a certain generation t is represented by the symbol P_t and the symbol $+$ denotes the multi-set union in conjunction of populations.

Input: N (population size)
T (maximum number of generations)
p_c (crossover probability)
p_m(mutation rate)

Output: A (non-dominated set)

Step 1: *Initialization*: Set $P_0 = \varnothing$ and $t = 0$. For $i = 1,\dots,N$ do
(a) Choose a candidate solution x according to some probability distribution.
(b) Set $P_0 = P_0 + x$

Step 2: Fitness assignment: For each individual $x \in P_t$ calculate the objective vector $[f_1(x), f_2(x),\dots, f_n(x)]$ and calculate the scalar fitness value $J(x)$.

Step 3: Selection: Set $P' = \varnothing$. For $i = 1,\dots,N$ do

(a) Select an individual $x \in P_t$ according to its fitness function $J(x)$.
(b) Set $P' = P' + x$

Step 4: Recombination: Set $P'' = \varnothing$: For $i = 1,\dots,N/2$ do

(a) Choose two individuals $x', x'' \in P'$ and remove them from P'.
(b) Crossover x' and x'' The resulting children are c' and c''
(c) Add c' and c'' to P'' with probability p_c Otherwise add x' and x'' to P''.

Step 5: Mutation: Set $P''' = \varnothing$ For each individual $x \in P''$ do

(a) Mutate x with mutation rate p_m The resulting individual is m.
(b) $P''' = P''' + m$.

Step 6: Termination: Set $P_{t+1} = P'''$ and $t = t + 1$. If $t \geq T$ or another stopping criterion is satisfied then set $A = P_t$.

A large number of selection, crossover and mutation operators have been proposed for different applications. The suitable operators for the warehouse replenishment problem are analyzed in Section 4.

13.2.3 Control of Diversity of the Pareto-optimal Solutions

It is desirable to find as many Pareto-optimal solutions as possible in order to provide more choices to the decision maker. To succeed this the weight vector w of the objective function $J(x)$ has to be successfully tuned. The choice of the weights w_i's can be done either in a stochastic or a deterministic way. If the aim is to have Pareto-optimal solutions that are uniformly scattered in the cost functions space, one has to perform a stochastic tuning of the weights w_i's, based for example, on probability distributions. On the other hand if the warehouse administrator knows approximatively the acceptable values of the objectives $f_i(x)$'s, a fuzzy rule-base can be used to choose the different values for the w_i's.

Assume that the weighting approach is used and the following cost function is defined:

$$J = w_1 f_1(x) + w_2 f_2(x) + \ldots + w_n f_n(x) \tag{13.5}$$

where $w_1, w_2, \ldots, w_n$ are non-negative weights such that $\sum_{i=1}^{n} w_i = 1$.The objectives $f_i(x)$ are assumed to be normalized with respect to their maximum value (see Section 3.4). If a GA uses one weight vector to compose one fitness function, there is one search direction.

For example, if $w' = (0.5, 0.5)$ is used in a 2-D MOP, the associated search direction is shown in Fig. 13.5. Since $w'_1 = w'_2 = 0.5$ the resulting Pareto-optimal solution is expected to satisfy $f_1(x) \approx f_2(x)$and is described by the points B and C.

However, there may be other Pareto-optimal solutions such as A and D (Fig. 13.5). Solution A corresponds to $f_1(x) > f_2(x)$ and can be approached if $w_1 < w_2$ i.e. moving along the direction of the weights vector w''.On the other hand, solution D corresponds to $f_1(x) < f_2(x)$and can be approached if $w_1 > w_2$ i.e. moving along the direction of the weights vector w'''.

To extend the search towards other Pareto-optimal solutions various combinations of the weight vectors have to be tried. To this end, the following methods have been proposed in the relevant bibliography:

(i) The division of the population into M sub-populations is proposed in [369] and M independent fitness functions are used where the first fitness function is $f_1(x)$ the second fitness function is $f_2(x)$ etc. Each sub-population is guided by one fitness function, and hence there are M fixed search directions.
(ii) A weighted sum of the objective functions $f_i(x)$ is proposed in [162] for the construction of the overall cost function, while the weight vector is randomly generated.

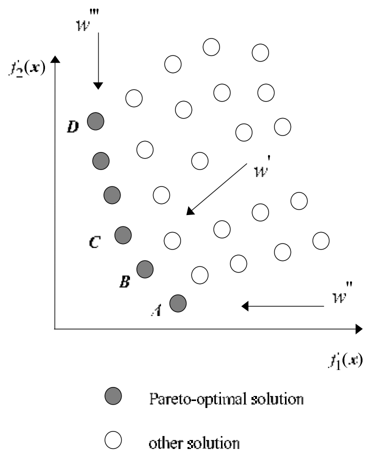


Fig. 13.5 Influence of the weight vector on the search direction of Pareto-optimal solutions.

(iii) The assignment of a probability to each objective $f_i(x)$ is proposed in [207]. This probability denotes how likely it is for each objective to become the sorting criterion of the chromosomes x at the next generation of the genetic algorithm.
(iv) A weighting-based Genetic Algorithm is proposed in [135] in order to maintain a diverse population and to achieve a well distributed non-dominated set. To search for multiple solutions in parallel, the weights are not fixed but instead attached to the candidate solution vector x.

The fitness assignment in the weighting-based GA described above is:

```
Input: $P_t$ (generation of the initial population).
Output:$J$ (fitness value).
Step 1: For each individual $x \in P_t$ do

(a) Create the augmented vector $x' = [x, w]$.
(b) Extract weights $w_j$ $(j = 1, 2, \ldots, k)$ from $x'$.
(c) Set $J(x') = w_1 f_1(x) + \ldots + w_n f_n(x)$
```

Thus, the chromosome in this weighting-based GA is the augmented vector $x' = [x, w]$. The diversity of the weight combinations is promoted by the GA search in the objective space. As a consequence this GA evolves solutions and weight combinations simultaneously.

A fuzzy rule-base is proposed in [14] for the tuning of the aggregate cost function. This approach takes advantage of the experts knowledge in order to restrict the search for the Pareto-optimal solutions only to specific regions of the cost function space. For a 2-D MOP the antecedent part of the fuzzy rules will contain the

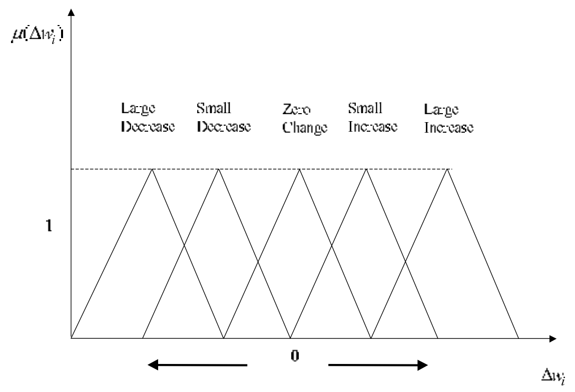


Fig. 13.6 Fuzzy incremental changes of the weights w_i.

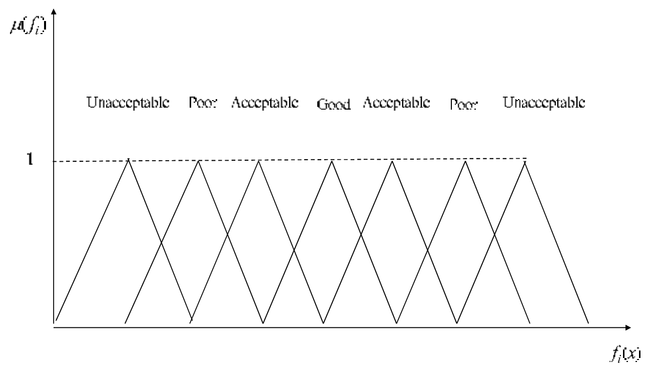


Fig. 13.7 Fuzzy values of the objective functions $f_i(x)$.

objective functions $f_1(x)$ and $f_2(x)$, while in the conclusion part the fuzzy increment of the value of the weight w_1 or w_2 will be included (see Table 13.1).

The incremental changes of the weights w_i are realized through fuzzy inference. The fuzzy rules are of the form:

$$R^{(l)}: \text{ IF } f_1(x) \text{ is Poor AND } f_2(x)\text{is Poor THEN}\Delta_{w_1}(\Delta_{w_2}) \text{ is Small,etc.} \tag{13.6}$$

The fuzzy sets in the antecedent and the consequent part of the rules are shown in Figs. 13.6 and 13.7. For the warehouse replenishment, a similar fuzzy rule base can be provided by the warehouses administrator.

It should be noted that since methods (i)-(v) contain heuristics, their relative advantages and drawbacks cannot be easily assessed. The stochastic search performed by the first four methods is computationally more demanding than the fuzzy rules approach but on the other hand results in a more detailed search of the cost functions space. The fuzzy rules approach bares resemblance to a gradient algorithm thus it is more likely to behave well when the Pareto frontier is convex (see Fig. 13.4).

13.2.4 Genetic Algorithm Convergence to Pareto-optimal Solutions

The factors that influence the convergence of GA optimization to global optimums (and thus Pareto optimums) are:

(i) *Generation of the initial population*: An initial population is generated in which the population members are scattered uniformly over the feasible solution space, so that the genetic algorithm can explore the whole solution space evenly.
(ii) *Normalization of the objective function*: The values of different objective functions may have different order of magnitude. If the fitness function $J(x)$ is equal to the weighted sum of objective functions, it may be dominated by the objective functions with large values.

For example, if $f_1(x)$ represents the aggregate distance from the picking positions of the warehouse with $10^4 \leq f_1(x) \leq 10^5$ and $f_2(x)$ represents the expiration date cost of the stored products with $10^2 \leq f_1(x) \leq 10^3$, then $w_1 f_1(x) + w_2 f_2(x)$ may be dominated by $f_1(x)$ To overcome this problem, each objective function is normalized as follows:

$$h_1(x) = \frac{f_i(x)}{\max_{y \in \Psi}\{|f_i(y)|\}} \tag{13.7}$$

where Ψ is the set of points in the current population and $h_i(x)$ is the normalized i-th objective function.

13.3 A Genetic Algorithm for the Warehouse Replenishment Task

13.3.1 Constraints of Genetic Algorithms in Ordering Problems

Regarding the warehouse replenishment problem the status of all storage places in the warehouse can be encoded into a linear chromosome. For instance, the chromosome $\{1-3-6-5-2-4\}$ describes that at the first storage position the product with code number 1 is placed while in the second storage position one finds the product with code number 2, etc. The value of the objective function depends on the relative order of these objects. Every new configuration of product placement in the warehouse corresponds to a permutation of the original chromosome.

It should also be stressed out that in such ordering problems the classic genetic operators, (crossover,mutation) cannot be applied the way they are usually defined in most applications [265],[434]. In a chromosome there must be a 1 to 1 mapping between storage position and product. Therefore, when crossover takes place, some parts of the chromosomes of the two parents cannot be freely copied, otherwise the offspring will carry instances of the same object. For the same reason mutation cannot swap arbitrarily two objects within the same chromosome.

In Fig. 13.8 are demonstrated illegal chromosomes which are generated when a conventional crossover operator (e.g. 1-point crossover) is applied. It is also shown how a specialized crossover operator works fine instead.

In some order-based problems there are also other techniques that permit the application of normal genetic operators, but they are in general inefficient and the case where permutations of elements are considered they would not work at all. In these methods there are two ways that prevent the production of illegal offspring.

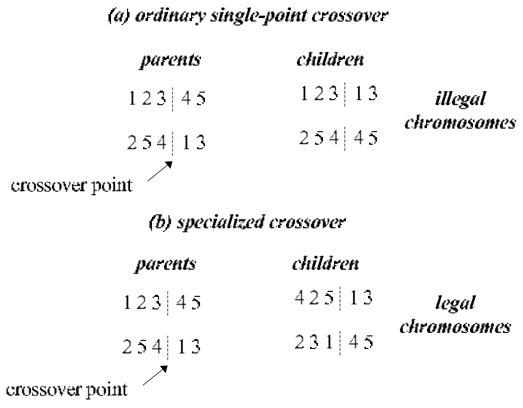


Fig. 13.8 Crossover in order-based problems: (a) ordinary crossover leads to illegal offspring, (b) specialized crossover produces legal offspring.

The first is to allow the generation of illegal chromosomes that are penalized in the sequel according to some criteria. The second is to try to reestablish the uniqueness of the permutation after the crossover has taken place and degrade the fitness of the new configuration by an amount proportional to the damage that has been repaired [434].

In Fig. 13.8, the crossover operator is applied at a randomly chosen site of the chromosome and divides every parent into two pieces: left side and right side. The first child is generated by copying the leftmost of the first parent and the rightmost of the second parent, while the other child is generated by copying the rightmost of the first parent and the leftmost of the second parent. It is obvious, that illegal configurations are generated in this way. It is also shown how a specialized crossover operator performs some additional swaps between genes in order to produce an admissible offspring. In the case described in this example both swaps (4 with 1) and (5 with 3), take place in the parents and generate the children. A mutation is simply implemented by swapping with some small pre-defined probability two elements of the same chromosome randomly chosen.

Similar problems with illegal chromosomes generated by crossover and mutation appear in the GA for the warehouse replenishment. The operators developed to resolve them are described in the following paragraph.

13.3.2 The Genetic Algorithm for Replenishment Optimization

Since the solution needed is the best warehouse configuration possible, it is reasonable to define the chromosomes of our genetic population of type warehouse. The algorithm produces a population of warehouses of random configurations but retaining the same contents as the initial state and then lets the population evolve into better warehouses. At each generation, the best half of the population mates to produce two children per couple, while the other half dies. This reproduction keeps steady the population of the warehousechromosomes.

However, during this process of evolution, certain constraints must be taken into account; warehouses in general-as well as the warehouse used in our simulation (see Section 13.4)-have many different types of storage space. As already explained in Section 13.2 products belonging to one type may not migrate to a location of a different type. Moreover, even locations of the same type are categorized by type of product (flammable, fragile, food, poison, etc). Thus, a tall palette may not move into the area of short palettes even if they are both in the back-to-back storage area, just as poison substances may not move into the area where food is stored. These distinct categories are assumed to be pre-defined in the warehouse.

To produce a valid solution of a MOP one has to find a reasonable way of mating warehouse chromosomes while at the same time complying with these constraints. This requires that the operators of the genetic algorithm be applied to each category separately; therefore in the discussion which follows, the term ”configuration” is used to refer to each chromosome category of the warehouse wherein products may be legally relocated.

13.3.3 Mating Procedure

In the work presented, warehouse configurations are defined as the string of product locations. Each product, i.e. each box, palette, etc, is given its own unique code.In the following example products are represented by letters, whereas locations by numbers. Locations not associated with a letter represent empty (unused) locations. Numbers are chosen to be consecutive in the interest of simplicity and better visualization of the encoding of the chromosomes; no specific connection between locations of consecutive numbers (such as physical proximity) is implied.

During reproduction, a random crossover point is picked, for instance just before location 6.

Combining the two parent configurations in a crosswise way as dictated by the classic approach in genetic algorithms would produce more than one instances of the same product (or missing products) in the resultant offspring and hence invalid configurations. To overcome this problem a more complex method of combining

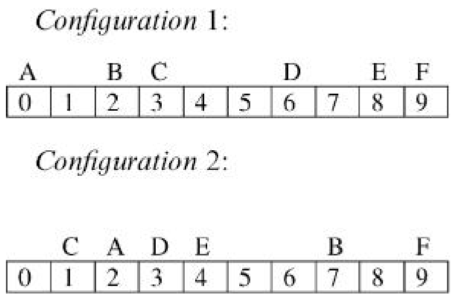


Fig. 13.9 Typical chromosomes

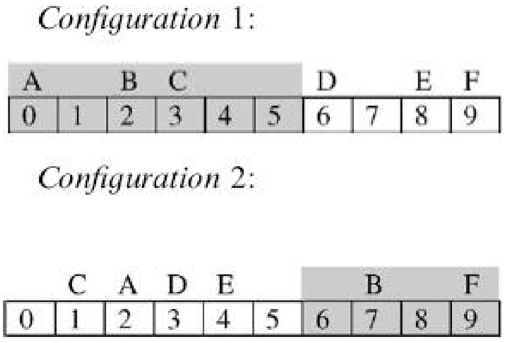


Fig. 13.10 Typical crossover

warehouse configurations is used. The basic properties of mating are maintained, namely that the offsprings bare some resemblance to their parents and also that the extent to which an offspring resembles its parent is proportional to the amount of chromosome it has inherited from that parent.

The first step is to align the two configurations with respect to products. Assuming we transform only one,say configuration 2, we end up with configuration 2′, as shown in Fig. 13.11.

The next step is to produce moulds for the offspring by combining the locations of each parent configuration in a crosswise fashion, thus producing configurations 3′ and 4′, shown again in Fig. 13.11.

The final step is to assign valid locations to our products so that there are no discrepancies and so that the basic rules of mating mentioned above are observed. This is done by considering products in the location order dictated by our moulds, ascending for configuration 3′ and descending for configuration 4′. Thus, our final offspring configurations would look like Configuration 3, shown in Fig. 13.12.

It should be noticed how products are shifted into the next available location (in configuration 3) when their designated location in the respective mould is occupied.

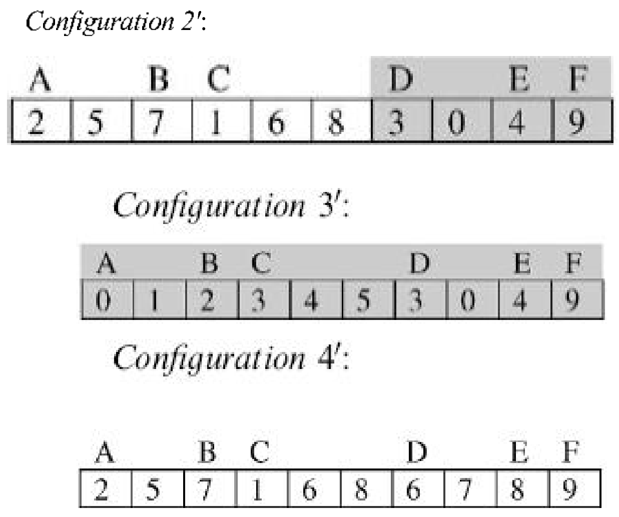


Fig. 13.11 Configuration 2': Aligned chromosome configuration with respect to products, Configuration 3' and 4': Offspring generated by combining the locations of each aligned parent configuration in a crosswise fashion

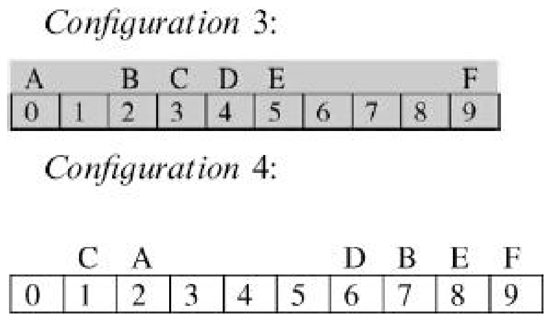


Fig. 13.12 Final offsprings configuration

Recall also that one of the parent configurations was altered in order to align with the other. Using ascending order with the offspring containing the first chromosome part of the unaltered parent and descending order with the offspring containing the last chromosome part ensures that this shifting of products does not extend past the limits of the warehouse configuration.

13.3.4 Mutation Procedure

Mutation is carried out by transforming a specified percentage of the population, the mutation level [423]. The chromosomes chosen for transformation undergo a predefined number of arbitrary swaps of their genes locations in the chromosome, i.e. the products locations in their configuration. Thus the resulting chromosomes are essentially permutations of the originals under the constraint that genes can only be permuted in their corresponding categories.

13.3.5 Definition and Tuning of the Cost Function

The evaluation of a specific configuration x is achieved by means of a cost function. This is defined as a weighted sum of several factors:

- $f_1(x)$ *Distance from delivery point.* This is the accumulated distance of all products in the configuration from the point of the warehouse where products are delivered, typically an entrance to the warehouse facility.
- $f_2(x)$: *Distance from collection point.* This is the accumulated distance of all products in the configuration from the point of the warehouse where products are shipped, typically an entrance to the warehouse facility.
- $f_3(x)$: *Distance from picking locations.* Every product receives a penalty for not being near picking locations. Products residing in picking locations receive no penalty. Products residing in pro-picking locations receive a penalty of 1 and products residing in stock locations receive a penalty of 3.
- $f_4(x)$: *Expiration date.* Each product receives a penalty P if its placed too far away from the collection point and its expiration date is near or it is placed too near the collection point and its expiration date is far. The penalty is defined as $P = |dL - dT|$ where dL is the normalized distance of the product from the collection area and dT is the normalized time remaining before the product expires. The cost of this factor is the sum of all products penalties.
- $f_5(x)$: *Seasonal demand.* If the product is not out of season, it is penalized with its distance from the delivery point. The cost of this factor is the sum of all these penalties.

All these factors are normalized, then weighted by user-supplied weights and finally summed to form the total cost of the configuration considered:

$$J(x) = \sum_{i=1}^{5} w_i f_i(x), \tag{13.8}$$

where the $f_i(x)$'s are the individual costs and the w_i's their corresponding weights.

Obviously, the cost function uses much more information than appears in the chromosome encoding. As far as implementation is concerned, this data is assumed to reside in a database indexed by location numbers and product letters. It is not incorporated into the chromosome encoding itself in order to avoid duplications of the same data and therefore save resources.

The lower the cost the better the configuration. This is the only mean by which one can evaluate a specific configuration (chromosome). No conclusions can be derived by visually examining the chromosome's encoding because there is simply no correspondence between the number assigned to a location and any of the locations properties, just as there is no correspondence between the letter assigned to a product and any of the products properties.

A Pareto-optimal solution x is found if no further decrease of an individual objective $f_i(x),\ i = 1,2,\ldots,5$ is possible without causing the degradation of another objective $f_j(x)$ (e.g. after some generations further decrease of the cost related to the distance from the picking locations will cause an increase of the cost associated with the seasonal demand).

Remarks

(1) $f_4(x)$ and $f_5(x)$ can be considered as dependent objective functions since they depend, respectively,on the distance from the collection point $f_2(x)$and the distance from the delivery point $f_1(x)$:
(2) The weights of the objective function (7) are derived by the experience of the warehouse operators and are organized in a fuzzy protocol as the one given in Table I. They depend upon the individual characteristics,goals and demands of the warehouse. Small, fast-moving warehouses will require quite different weights compared to vast, slow-moving ones.

13.4 Results on Genetic Algorithm-Based Warehouse Optimization

13.4.1 Cost Function Tuning through Weights Selection

For the evaluation of the performance of the proposed GA, real data from a working industrial warehouse were gathered and input into software implementing the algorithm. Performance was investigated in two different ways: (i) how it is affected by different weights (dependence of the diversity of the Pareto-optimal solutions on the weight vector w) and (ii) how it is affected by different mutation levels (dependence of the overall cost function $J(x)$ on the mutation rate).

It is important to note that it is not possible for the costs to decrease indefinitely since they depend on constrained factors, which impose a definite lower limit. In the spatial representation of the warehouse in Fig. 13.8, different shapes represent different types of storage,while crosses represent the beginning of subcategories within the same storage type. These subcategories, may correspond to further discrimination, either in type of storage or in product categories. The map is read from left to right and represents a continuous row of warehouse locations (much like our chromosome encoding, only products are not distinguished) fitted into a square. Products, designated by dots, may only move within their category (the area defined by two consecutive crosses).

Fig. 13.13 pictures the optimal replenishment suggested by the algorithm when run with a weight configuration bent heavily on bringing products to picking locations and to a lesser extent closer to the collection area.Within each category there are three shaded areas: (a) the picking area, (b) the pro-picking area and (c) the

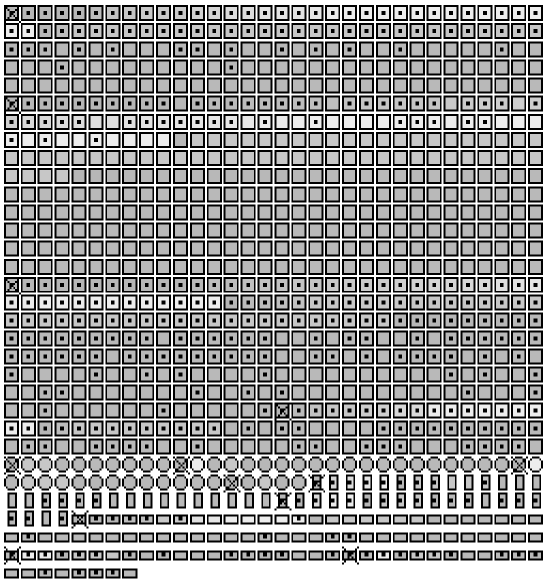


Fig. 13.13 Spatial representation of the warehouse after the GA-based replenishment. □,○,⊏⊐ storage positions of different geometry and capacity, $\times$: beginning of subcategories within the same storage type, •: products.

stock area. Darker hues designate small distances from the collection area. The algorithm has filled the picking areas entirely, as required. It then proceeds to fill the pro-picking areas and the stock areas. Finally, because it has been asked to slightly take into account distance, the algorithm makes a small attempt to bring products to darker locations.

Various runs with weights coming from the fuzzy protocol of Table 13.1 and with data coming from the daily operation of the warehouse were carried out. Two indicative of these runs correspond to the normalized weight configurations w_1 and w_2, and are shown in Table 13.2.

Distance represents the distance cost $f_2(x)$ associated with the collection point. Only one distance cost was taken into account (the other been set to zero) since the two are similar in nature, and therefore, when investigating the behavior of the algorithm using just one, would result in the same conclusions in a clearer way.

Both runs used populations of 1000 chromosomes,with 100 generations and a mutation level of 7%. The total cost decreases considerably (around 40%) in both cases, as shown in Fig. 13.10.

However, despite the apparently seamless reduction of the total cost, the detailed behavior of the algorithm is more complex. Useful insights into the inner workings of the algorithm can be gained by looking at the variation of each individual cost $f_i(x)$ with the number of generations.

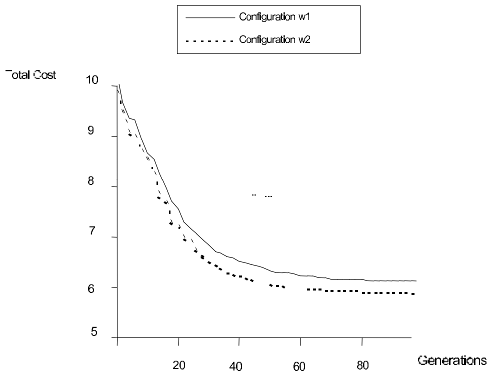


Fig. 13.14 Evaluation of the total cost $J(x)$ vs. generations.

Table 13.1 A protocol with $f_1(x)$ and $f_2(x)$ as inputs and w_1 (or) w_2 as output

$f_1(x)$	$f_2(x)$			
	good	*acceptable*	*poor*	*unacceptable*
good	zero change	zero change	small increase	large increase
acceptable	small decrease	small decrease	small increase	small increase
poor	small decrease	small decrease	small decrease	small increase
unacceptable	large decrease	large decrease	small decrease	small decrease

Table 13.2

Weight coefficients configuration	w_1	w_2
Distance cost function	0.091	0.6
Picking cost function	0.454	0.2
Expire cost function	0.136	0.08
Season cost function	0.318	0.12

The distance cost $f_2(x)$ displays the expected behavior; it decreases as the algorithm progresses much like the total cost. In configuration w_1 its contribution to the total cost is less than it is in configuration w_2 which is reflected in the fact that it reaches a lower (better) value in configuration w_2. Moreover, the cost decreases more steadily in configuration w_2 with fewer and smaller fluctuations (see Fig. 13.15).

A similar situation is seen in the case of the *Picking* cost $f_3(x)$. Its contribution is significant in both configurations and thus the performance of the algorithm with

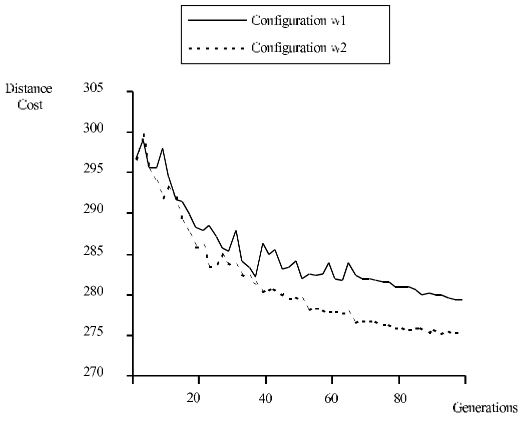


Fig. 13.15 Evaluation of the distance cost $f_2(x)$ vs. generations.

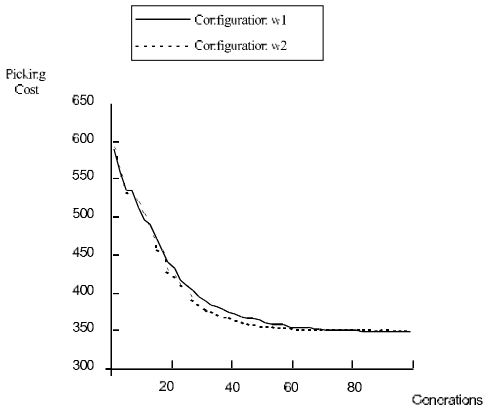


Fig. 13.16 Evaluation of the picking cost $f_3(x)$ vs. generations.

respect to this cost is almost entirely similar in both cases. Configuration w_1 is slightly smoother than w_2 and results in marginally lower cost, as expected (see Fig. 13.16).

A different aspect of the behavior of the algorithm is revealed by the variation of the *Season* cost $f_5(x)$; in configuration w_1 where the contribution of this cost is large, it decreases steadily as expected. However, in configuration w_2, where the corresponding weight is relatively low, the algorithm results in an early increase of $f_5(x)$ (see Fig. 13.17).

This practice is repeated in the case of the Expire cost $f_4(x)$, which, having small contributions in both configurations,does not affect significantly the total cost which keeps on decreasing. Since its weight in configuration w_1 is relatively larger than its weight in configuration w_2 (13% of total versus 8%), the cost in configuration w_1

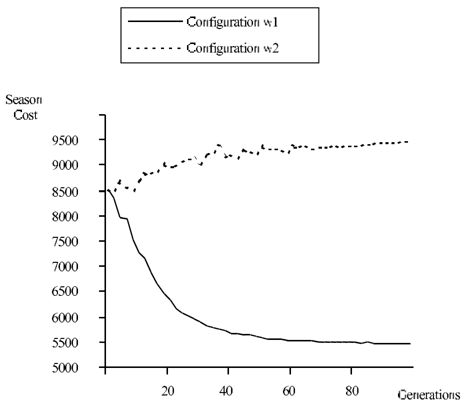


Fig. 13.17 Evaluation of the season cost $f_5(x)$ vs. generations.

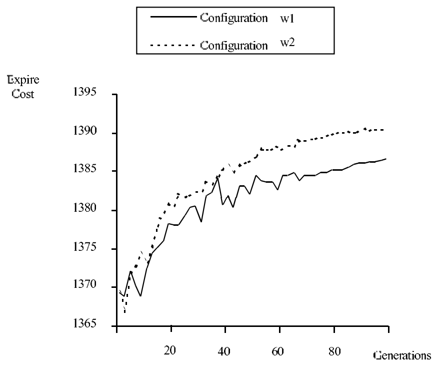


Fig. 13.18 Evaluation of the expiration cost $f_4(x)$. generations.

increases less than it does in w_2 and shows greater fluctuations as the algorithm is more reluctant to increase it (see Fig. 13.18).

It is evident that the algorithm does not exhibit the same behavior for all objective functions $f_i(x)$, given the same weights. For example, the distance cost $f_2(x)$ has a contribution around 9% in configuration w_1, while the expire cost $f_4(x)$ has a contribution of 8% in configuration w_2. Nevertheless, the former decreases satisfactorily while the latter increases. Even in configuration w_1, where the expire cost has a contribution of more than 13% it increases while the distance cost decreases. This behavior is due to the fact that the season and expire costs depend implicitly on the distance cost (Remark 1, in subsection 13.3.5); for them to decrease the distance cost needs to decrease too, so there is a greater momentum for reducing the distance cost. On the other hand, the picking cost not only is independent, but also has no

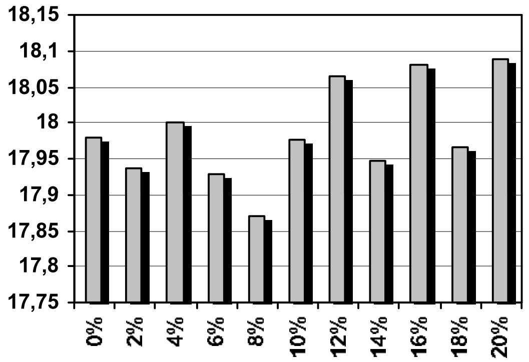


Fig. 13.19 Influence of the mutation rate on the overall cost function $J(x)$

objective functions which are influenced by it. Thus it is reduced effectively in both configurations,despite its contribution being more than double in the aggregate cost function.

The optimal level of mutation is an important consideration in genetic algorithms. Obviously, if one allows the population size to become extremely large, mutation becomes meaningless, as large amounts of genetic information will be readily available within the population itself without the need to mutate. Therefore mutation is essentially a mean by which population can be reduced without significantly affecting the quality of generated solutions. Reducing population size is very desirable, as it is what mainly devours computational resources. The number of generations is less important since the quality of solutions depends more heavily on the richness of genetic information present within the population.

To find the optimal mutation level, several runs were carried out, with the level of mutation varying from 0% to 20%. A population of 1000 individuals was left to evolve for 100 generations in all cases. The weight configuration used was a compromise between configurations w_1 and w_2. The diagram below shows how the total cost varies with mutation level. It is clear that a level of around 8% is optimal, thus the chosen level of 7% in the runs described in the previous section (see Fig. 13.19).

13.4.2 Evaluation of the Genetic Algorithm Performance

The warehouse replenishment is a multi-objective optimization problem (MOP) where various contradictory performance objectives have to be fulfilled. Since the combinatorial search of optimal solutions is impractical the GA-based search is deemed to be a good alternative. The individual costs with respect to which the warehouse replenishment was optimized were: $f_1(x)$: distance from the delivery points, $f_2(x)$: distance from the collection points,$f_3(x)$: distance from the picking locations, $f_4(x)$: cost due to expiration date, $f_5(x)$: cost due to seasonal demand.

It is desirable for the decision maker to have at his disposal as much as Pareto-optimal solutions as possible. To succeed that, a fuzzy rule base enables to tune the weight vector of the objective function $J = \sum_{i=1}^{n} w_i f_i(x)$. The fuzzy rules can be derived using the experience of the warehouse operators. The trial of various weight vectors steers the search of Pareto-optimal solutions to different directions and enriches the non-dominated solutions set.

The special crossover operator introduced in the proposed genetic algorithms, prevents the appearance of illegal permutations, i.e. of chromosomes where a product would appear in more than one storage positions.The efficiency of the algorithm was tested using data from the daily operation of a real warehouse. The results were satisfactory. An improvement of the aggregate cost function $J(x)$ of about 80% was recorded. More Pareto-optimal permutations x can be found if the weight vector w is better tuned. A drawback of the proposed methodology is the appearance of unnecessary small displacements of products. For example, a product could end up in the location next to its original one. These movements are part of the stochastic nature of the algorithm and are obviously a waste of resources as they rarely make any difference to the effectiveness of the replenishment. Human intervention could be employed in these cases to make real-time decisions regarding the usefulness of such displacements. Alternatively, the algorithm could be run once to establish products that appear to be well placed,i.e. products that do not move far from their original position after replenishment. These products could be marked as unmoveable and the algorithm would then run again to assign the remaining locations to the remaining products.

The convergence of the optimization algorithm can be evaluated by launching it multiple times starting from different initial populations. The convergence speed depends on the size of the population and the processing speed of the machine. For a population of 1000 individuals over 100 generations, the computation time was small enough to permit in-line decision making. The optimization algorithm can be executed when the warehouse remains idle or when the inventory at the picking locations falls bellow acceptable levels.

A final remark is that the presented genetic algorithm performs well in considerably complicated warehouse environments where many different constraints are in effect and compromises are unavoidable. On the other hand, when applied to warehouse environments with reduced complexity, the algorithm would perhaps be bettered by a more deterministic approach.

Chapter 14
Machine Learning Methods for Industrial Systems Control

Abstract. Machine learning methods are of particular interest in the design of intelligent industrial systems since they can provide efficient control despite model uncertainties and imprecisions. The chapter proposes neural networks with Gauss-Hermite polynomial basis functions for the control of flexible-link manipulators. This neural model employs basis functions which are localized both in space and frequency thus allowing better approximation of the multi-frequency characteristics of vibrating structures. Gauss-Hermite basis functions have also some interesting properties: (i) they remain almost unchanged by the Fourier transform, which means that the weights of the associated neural network demonstrate the energy which is distributed to the various eigenmodes of the vibrating structure, (ii) unlike wavelet basis functions the Gauss-Hermite basis functions have a clear physical meaning since they represent the solutions of differential equations of stochastic oscillators and each neuron can be regarded as the frequency filter of the respective vibration eigenfrequency.

14.1 Model-Free Control of Flexible-Link Robots

14.1.1 Approaches for Model-Based Control of Flexible-Link Robots

Machine learning methods, such as neural networks have been proven to be very efficient for the control of nonlinear dynamical systems, and particularly for robot control [422],[424],[427]. Control of flexible-link robots has already been discussed in Chapter 1, following a model-based approach. In this chapter model-free control of flexible-link robots will be analyzed based on machine learning methods. In general, the design of model-based flexible-link robot controllers can be performed either using a description in the s-frequency domain or using a description in the time domain through a state-space model (e.g. [8],[25],[289]). Regarding controller design in the s-frequency domain, some indicative results are as follows: In [24],[305] a description of the dynamic model of the flexible-link robot is given in the s-frequency domain, however this is constrained to a linear 1-DOF manipulator model [317]. In [317] a proportional derivative adaptive control scheme is proposed. This requires measurements of the motor's angle and of the coupling torque at the basis of the flexible beam. The associated description in the s-frequency domain for a 1-DOF flexible-link robot is a MIMO transfer function matrix. In [63] again a description of the flexible-link robot dynamics in the s-frequency domain is derived. The robotic model is separated in two parts with fast and slow dynamics. For the slow subsystem, which is associated to the rigid link dynamics included in the dynamic model of Eq. (1.23) a PD controller is used. For the fast subsystem another PID controller is used which performs modal feedback. The vibration modes are measured using a pair of strain gauges at each link.

Regarding controller design in the time domain through state-space description some representative results are summarized in the following: In [92], [366] and [141] a sliding mode control is used in the flexible-link robot control loop. The robot's model is decomposed into a slow subsystem which is due to the rigid-link part appearing in Eq. (1.23) and into a fast subsystem which is due to the vibrations of the flexible link. The sliding mode controller is used for the rigid-link part aiming also at the compensation of external disturbances and parametric uncertainties. For the compensation of the fast dynamics (vibrations of the flexible link) an H_∞ or LQR controller is proposed. This however requires linearization of the flexible robot's model at the final desirable configuration. A similar methodology is adopted in [223] where again the complete state vector of the flexible-link robot is assumed to be measurable. Finally, optimal control of the flexible-link robot model is proposed in [441], after previously linearizing the robot's dynamics through the calculation of nonlinearities with the use of a neural model.

Design of controllers for flexible-link robots using state-space description of the system and full state feedback is based on the assumption that the complete state vector of the robot is measurable [188]. However, the vibration modes of the flexible link are elements of the state vector which are difficult to measure. In attempts to measure indirectly the first vibration modes of the flexible link, higher order vibration modes are still neglected [64]. In general, vibration modes of the flexible links are hard to obtain. To overcome this, the state vector of the flexible-link robot can be reconstructed from joint position measurements using state observers of filters. Following this concept, results on observer-based control of flexible-link robots have been obtained. Estimation of the robot's vibration modes is performed either with the use of nonlinear state observers [10],[262],[278],[285], or with the use of Kalman Filtering [15].

The above analysis shows that model-based controller design for flexible-link robots comes against specific difficulties. In the design in the s-frequency domain a linear dynamic model has to be assumed which is directly applicable only to single-link flexible robots, while the unknown vibration modes have to be found through an identification procedure. In the design in state space, the main difficulty is that the state vector of the flexible-link robot is not completely measurable, therefore part of the state vector has to be estimated with the use of a state observer and a joint controller-observer scheme has to be implemented. Therefore, model-free techniques for flexible-link robot control appear to have significant advantages, such as simple structure, smaller number of parameters to be tuned and ability to compensate for the multi-frequency characteristics of the flexible-link vibrations [67], [481].

14.1.2 Approaches for Model-Free Control of Flexible-Link Robots

In parallel to model-based control for flexible-link robots, model-free control methods have been studied [26],[275],[328],[397]. A number of research papers employ model-free approaches for the control of flexible-link robots based on fuzzy logic and neural networks. In [410] control of a flexible manipulator with the use of a neuro-fuzzy method is described, where the weighting factor of the fuzzy logic controller is adjusted by the dynamic recurrent identification network. The controller works without any prior knowledge about the manipulator's dynamics. Control of the end-effector's position of a flexible-link manipulator with the use of a neural and a fuzzy controller has been presented in [134],[231],[398],[402],[441],[462]. In [441] an intelligent optimal control for a nonlinear flexible robot arm driven by a permanent-magnet synchronous servo motor has been designed using a fuzzy neural network control approach. This consists of an optimal controller which minimizes a quadratic performance index and a fuzzy neural-network controller that learns the uncertain dynamics of the flexible manipulator. In [402] a fuzzy controller has been developed for a three-link robot with two rigid links and one flexible fore-arm. This controller design is based on fuzzy Lyapunov synthesis where a Lyapunov candidate function has been chosen to derive the fuzzy rules. In [398] a neuro-fuzzy scheme has been proposed for position control of the end effector of a single-link flexible robot manipulator. The scale factors of the neuro-fuzzy controller are adapted on-line using a neural network which is trained with an improved back-propagation algorithm. In [54] two different neuro-fuzzy feed-forward controllers have been proposed to compensate for the nonlinearities of a flexible manipulator. In [323] the dynamics of a flexible link has been modeled using modal analysis and then an inverse dynamics fuzzy controller has been employed to obtain tracking and deflection control. In [378] a fuzzy logic controller has been applied to a flexible-link manipulator. In this distributed fuzzy logic controller the two velocity variables which have higher importance have been grouped together as the inputs to a velocity fuzzy controller while the two displacement variables which have lower importance degrees have been used as inputs to a displacement fuzzy logic controller. In [152] adaptive control for a flexible-link manipulator has been achieved using a neuro-fuzzy time-delay controller. In [286] a genetic algorithm has been used to improve the performance of a fuzzy controller designed to compensate for the links' flexibility and the joints' flexibility of a robotic manipulator. Research results on neural-adaptive control and PD control with feedforward control elements for the case of flexible-link manipulators have been also given in [269],[377].

14.1.3 Neural Control Using Multi-frequency Basis Functions

In this chapter, a neural controller using wavelet basis functions is first proposed for the control of the flexible-link robot. The neural controller operates in parallel to a PD controller the gains of which are calculated assuming rigid link dynamics. The general structure of neural control for flexible-link robots was discussed in Chapter 1. Neural networks with wavelet basis functions, also known as 'wavelet networks', were first introduced in [471] aiming at giving to feed-forward neural networks with multi-resolution analysis features and at providing neural models having good approximation features while using a small number of tunable parameters. The wavelet neural networks can be classified into orthogonal and non-orthogonal. Orthogonal wavelet networks depend on generating orthonormal basis using the wavelet function. However, in order to create the orthonormal basis the wavelet function has to satisfy restrictions. The training of the orthonormal wavelet network is fast and its expansion is easy. On the other hand, the non-orthogonal wavelet network uses the so-called wavelet frame. The family of the wavelet functions that constitute a frame are such that the energy of the resulting wavelet coefficients lies within a certain bounded range of the energy of the original signal [1]. Controllers based on orthogonal wavelets have been used in vibration control problems [178]. In this paper the neural controller that is designed for the suppression of the vibrations of the flexible links, contains orthogonal wavelet basis functions.

Next, a neural network with Gauss-Hermite polynomial basis functions is considered for the control of flexible-link manipulators. This neural model follows the concept of wavelet networks [471] and employs basis functions which are localized both in space and frequency thus allowing better approximation of the multi-frequency characteristics of vibrating structures [31],[58],[202],[232],[400]. Gauss-Hermite basis functions have also some interesting properties [319],[348]: (i) they remain almost unchanged by the Fourier transform, which means that the weights of the associated neural network demonstrate the energy which is distributed to the various eigenmodes of the vibrating structure, (ii) unlike wavelet basis functions the Gauss-Hermite basis functions have a clear physical meaning since they represent the solutions of differential equations describing stochastic oscillators and each neuron can be regarded as the frequency filter of the respective vibration eigenfrequency.

It is noted that unlike neural networks with sigmoidal or Gaussian basis functions, Hermite polynomial-based FNN remain closer to Fourier series expansions by employing activation functions which satisfy the property of orthogonality [480]. Other basis functions with the property of orthogonality are Hermite, Legendre, Chebyshev, and Volterra polynomials [319],[348],[461].

14.2 Neural Control Using Wavelet Basis Functions

14.2.1 Wavelet Frames

The continuous time wavelet is defined at scale a and b as

$$\psi_{a,b}(x) = \frac{1}{\sqrt{a}}\psi(\frac{x-b}{\alpha}) \tag{14.1}$$

It will be shown that a continuous time signal $f(x)$ can be expressed as a series expansion of discrete wavelet basis functions. The discrete wavelet has the form [1], [413]

$$\psi_{m,n}(x) = \frac{1}{\sqrt{\alpha_0^m}}\psi(\frac{x-nb_0\alpha_0^m}{\alpha_0^m}) \tag{14.2}$$

The wavelet transform of a continuous signal $f(x)$ using discrete wavelets of the form of Eq. (14.2) is given by

$$T_{m,n} = \int_{-\infty}^{+\infty} f(x)\frac{1}{\sqrt{\alpha_0^m}}\psi(\frac{x-nb_0\alpha_0^m}{\alpha_0^m})dx \tag{14.3}$$

which can be also expressed as the inner product $T_{m,n} =< f(x), \psi_{m,n} >$. For the discrete wavelet transform, the values $T_{m,n}$ are known as wavelet coefficients. To determine how good the representation of a signal is in the wavelet space one can use the theory of wavelet frames. The family of wavelet functions that constitute a frame are such that the energy of the resulting wavelet coefficients lies within a certain bounded range of the energy of the original signal

$$AE \leq \sum_{m=-\infty}^{+\infty}\sum_{n=-\infty}^{+\infty}|T_{m,n}|^2 \leq BE \tag{14.4}$$

where $T_{m,n}$ are the discrete wavelet coefficients, A and B are the frame bounds, and E is the energy of the signal given by $E = \int_{-\infty}^{+\infty}|f(x)|^2dt = ||f(x)||^2$. The values of the frame bounds depend on the parameters α_0 and b_0 chosen for the analysis and the wavelet function used. If $A = B$ the frame is known as tight and has a simple reconstruction formula given by the finite series

$$f(x) = \frac{1}{A}\sum_{m=-\infty}^{+\infty}\sum_{n=-\infty}^{+\infty}T_{m,n}\psi_{m,n}(x) \tag{14.5}$$

A tight frame with $A = B > 1$ is redundant, with A being a measure of the redundancy. When $A = B = 1$ the wavelet family defined by the frame forms an orthonormal basis. Even if $A \neq B$ a reconstruction formula of $f(x)$ can be obtained in the form:

$$f^{'}(x) = \frac{2}{A+B}\sum_{m=-\infty}^{+\infty}\sum_{n=-\infty}^{+\infty}T_{m,n}\psi_{m,n}(x) \tag{14.6}$$

where $f^{'}(x)$ is the reconstruction which differs from the original signal $f(x)$ by an error which depends on the values of the frame bounds. The error becomes acceptably small for practical purposes when the ratio B/A is near unity. The closer this ratio is to unity, the tighter the frame.

14.2.2 Dyadic Grid Scaling and Orthonormal Wavelet Transforms

The dyadic grid is perhaps the simplest and most efficient discretization for practical purposes and lends itself to the construction of an orthonormal wavelet basis. Substituting $\alpha_0 = 2$ and $b_0 = 1$ into Eq. (14.2) the dyadic grid wavelet can be written as

$$\psi_{m,n} = \frac{1}{\sqrt{2^m}} \psi(\frac{x - n2^m}{2^m}) \tag{14.7}$$

or more compactly

$$\psi_{m,n}(t) = 2^{-\frac{m}{2}} \psi(2^{-m}x - n) \tag{14.8}$$

Discrete dyadic grid wavelets are commonly chosen to be orthonormal. These wavelets are both orthogonal to each other and normalized to have unit energy. This is expressed as

$$\int_{-\infty}^{+\infty} \psi_{m,n}(x) \psi_{m^{'},n^{'}}(x) dx = \begin{cases} 1 \text{ if } m = m^{'}, \text{ and } n = n^{'} \\ 0 \text{ otherwise} \end{cases} \tag{14.9}$$

Thus, the products of each wavelet with all others in the same dyadic system are zero. This also means that the information stored in a wavelet coefficient $T_{m,n}$ is not repeated elsewhere and allows for the complete regeneration of the original signal without redundancy. In addition to being orthogonal, orthonormal wavelets are normalized to have unit energy. This can be seen from Eq. (14.9), as using $m = m^{'}$ and $n = n^{'}$ the integral gives the energy of the wavelet function equal to one. Orthonormal wavelets have frame bounds $A = B = 1$ and the corresponding wavelet family is an orthonormal basis. An orthonormal basis has components which, in addition to being able to completely define the signal, are perpendicular to each other.

Using the dyadic grid wavelet of Eq. (14.7) the discrete wavelet transform is defined as

$$T_{m,n} = \int_{-\infty}^{+\infty} x(t) \psi_{m,n}(x) dt \tag{14.10}$$

By choosing an orthonormal wavelet basis $\psi_{m,n}(x)$ one can reconstruct the original signal $f(x)$ in terms of the wavelet coefficients $T_{m,n}$ using the inverse discrete wavelet transform:

$$f(x) = \sum_{m=-\infty}^{+\infty} \sum_{n=-\infty}^{+\infty} T_{m,n} \psi_{m,n}(x) \tag{14.11}$$

requiring the summation over all integers m and n. In addition, the energy of the signal can be expressed as

$$\int_{-\infty}^{+\infty} |f(x)|^2 dx = \sum_{m=-\infty}^{+\infty} \sum_{n=-\infty}^{+\infty} |T_{m,n}|^2 \tag{14.12}$$

14.2.3 The Scaling Function and the Multi-resolution Representation

Orthonormal dyadic discrete wavelets are associated with scaling functions and their dilation equations. The scaling function is associated with the smoothing of the signal and has the same form as the wavelet

$$\phi_{m,n}(x) = 2^{-m/2}\phi(2^{-m}x - n) \tag{14.13}$$

The scaling functions have the property

$$\int_{-\infty}^{+\infty} \phi_{0,0}(x) dx = 1 \tag{14.14}$$

where $\phi_{0,0}(x) = \phi(x)$ is sometimes referred as the father scaling function or father (mother) wavelet. The scaling function is orthogonal to translations of itself, but not to dilations of itself. The scaling function can be convolved with the signal to produce approximation coefficients as follows:

$$S_{m,n} = \int_{-\infty}^{+\infty} f(x)\phi_{m,n}(x) dx \tag{14.15}$$

One can represent a signal $f(x)$ using a combined series expansion using both the approximation coefficients and the wavelet (detail) coefficients as follows:

$$f(x) = \sum_{n=-\infty}^{+\infty} S_{m_0,n}\phi_{m_0,n} + \sum_{m=-\infty}^{m_0} \sum_{n=-\infty}^{+\infty} T_{m,n}\psi_{m,n}(x) \tag{14.16}$$

It can be seen from this equation that the original continuous signal is expressed as a combination of an approximation of itself, at arbitrary scale index m_0 added to a succession of signal details form scales m_0 down to negative infinity. The signal detail at scale m is defined as

$$d_m(x) = \sum_{n=-\infty}^{+\infty} T_{m,n}\psi_{m,n}(x) \tag{14.17}$$

and hence one can write Eq. (14.16)

$$f(x) = f_{m_0}(t) + \sum_{m=-\infty}^{m_0} d_m(x) \tag{14.18}$$

From this equation it can be shown that

$$f_{m-1}(x) = f_m(x) + d_m(x) \tag{14.19}$$

which shows that if one adds the signal detail at an arbitrary scale (index m) to the approximation at that scale he gets the signal approximation at an increased resolution (at a smaller scale index $m-1$). This is the so-called multi-resolution representation.

14.2.4 Examples of Orthonormal Wavelets

The scaling equation (or dilation equation) describes the scaling function $\phi(x)$ in terms of contracted and shifted versions of itself as follows [1],[251]:

$$\phi(x) = \sum_k c_k \phi(2x-k) \tag{14.20}$$

where $\phi(2x-k)$ is a contracted version of $\phi(t)$ shifted along the time axis by an integer step k and factored by an associated scaling coefficient c_k. The coefficient of the scaling equation should satisfy the condition

$$\sum_k c_k = 2 \tag{14.21}$$

$$\sum_k c_k c_{k+2k'} = \begin{cases} 2 \text{ if } k'=0 \\ 0 \text{ otherwise} \end{cases} \tag{14.22}$$

This also shows that the sum of the squares of the scaling coefficients is equal to 2. The same coefficients are used in reverse with alternate signs to produce the associated wavelet equation

$$\psi(x) = \sum_k (-1)^k c_{1-k} \phi(2x-k) \tag{14.23}$$

This construction ensures that the wavelets and their corresponding scaling functions are orthogonal . For wavelets of compact support, which have a finite number of scaling coefficients N_k the following wavelet function is defined

$$\psi(x) = \sum_k (-1)^k c_{N_k-1-k} \phi(2x-k) \tag{14.24}$$

This ordering of scaling coefficients used in the wavelet equation allows for our wavelets and their corresponding scaling equations to have support over the same interval $[0, N_{k-1}]$. Often the reconfigured coefficients used for the wavelet function are written more compactly as

$$b_k = (-1)^k c_{N_k-1-k} \tag{14.25}$$

where the sum of all coefficients b_k is zero. Using this reordering of the coefficients Eq. (14.24) can be written as

$$\psi(x) = \sum_{k=0}^{N_k-1} b_k \phi(2x-k) \tag{14.26}$$

From the previous equations and examining the wavelet at scale index $m+1$ one can see that for arbitrary integer values of m the following holds

$$2^{-(m+1)/2}\phi(\frac{1}{2^{m+1}} - n) = 2^{-m/2}2^{-1/2}\sum_k c_k \phi(\frac{2t}{2\times 2^m} - 2n - k) \tag{14.27}$$

which may be written more compactly as

$$\phi_{m+1,n}(x) = \frac{1}{\sqrt{2}}\sum_k c_k \phi_{m,2n+k}(x) \tag{14.28}$$

That is the scaling function at an arbitrary scale is composed of a sequence of shifted functions at the next smaller scale each factored by their respective scaling coefficients. Similarly, for the wavelet function one obtains

$$\psi_{m+1,n}(x) = \frac{1}{\sqrt{2}}\sum_k b_k \phi_{m,2n+k}(x) \tag{14.29}$$

14.2.5 The Haar Wavelet

The Haar wavelet is the simplest example of an orthonormal wavelet. Its scaling equation contains only two nonzero scaling coefficients and is given by

$$\phi(x) = \phi(2x) + \phi(2x-1) \tag{14.30}$$

that is, its scaling coefficients are $c_0 = c_1 = 1$. These values can be obtained from Eq. (14.21) and Eq. (14.22). The solution of the Haar scaling equation is the single block pulse defined as

$$\phi(x) = \begin{cases} 1 & 0 \le x < 1 \\ 0 & \text{elsewhere} \end{cases} \tag{14.31}$$

Using this scaling function, the Haar wavelet equation is

$$\psi(x) = \phi(2x) - \phi(2x-1) \tag{14.32}$$

The Haar wavelet is finally found to be

$$\psi(x) = \begin{cases} 1 & 0 \le x < \frac{1}{2} \\ -1 & \frac{1}{2} \le x < 1 \\ 0 & \text{elsewhere} \end{cases} \tag{14.33}$$

The mother wavelet for the Haar wavelet system $\psi(x) = \psi_{0,0}(x)$ is formed from two dilated unit block pulses sitting next to each other on the time axis, with one of them inverted. From the mother wavelet one can construct the Haar system of wavelets on a dyadic grid $\psi_{m,n}(x)$.

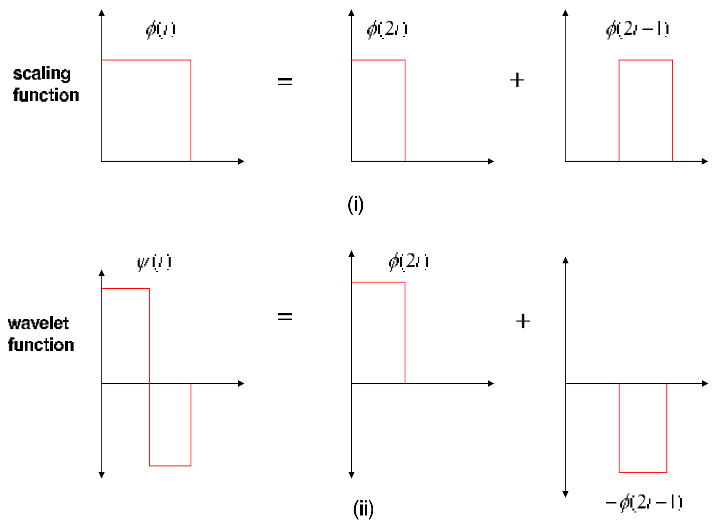


Fig. 14.1 i) The Haar scaling function in terms of shifted and dilated versions of itself, ii) The Haar wavelet in terms of shifted and dilated versions of its scaling function.

14.3 Neural Networks Using Hermite Activation Functions

14.3.1 Identification with Feed-Forward Neural Networks

The concept of function approximation with the use of feed-forward neural networks (FNN) comes from generalized Fourier series. It is known that any function $\psi(x)$ in a L^2 space can be expanded in a generalized Fourier series in a given orthonormal basis, i.e. $\psi(x) = \sum_{k=1}^{\infty} c_k \psi_k(x),\ a \leq x \leq b$. Truncation of the series yields in the sum $S_M(x) = \sum_{k=1}^{M} a_k \psi_k(x)$. If the coefficients a_k are taken to be equal to the generalized Fourier coefficients, i.e. when $a_k = c_k = \int_a^b \psi(x)\psi_k(x)dx$, then $S_M(x)$ is a mean square optimal approximation of $\psi(x)$ [348].

Unlike generalized Fourier series, in FNN the basis functions are not necessarily orthogonal. The hidden units in a FNN usually have the same activation functions and are often selected as sigmoidal functions or Gaussians. A typical feed-forward neural network consists of n inputs $x_i, \imath = 1, 2, \cdots, n$, a hidden layer of m neurons with activation function $h:\ R \rightarrow R$ and a single output unit (see Fig. 14.3). The FNN's output is given by

$$\psi(x) = \sum_{j=1}^{n} c_j h(\sum_{i=1}^{n} w_{ji} x_i + b_j) \tag{14.34}$$

The root mean square error in the approximation of function $\psi(x)$ by the FNN is given by $E_{RMS} = \sqrt{\frac{1}{N}\sum_{k=1}^{N}(\psi(x^k) - \hat{\psi}(x^k))^2}$, where $x^k = [x_1^k, x_2^k, \cdots, x_n^k]$ is the k-th

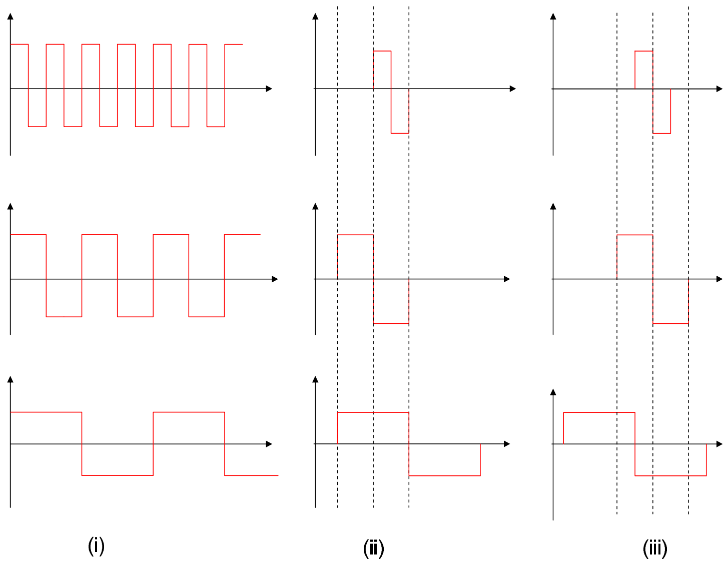


Fig. 14.2 i) Three consecutive scales shown from the Haar wavelet family specified on a dyadic grid, e.g. from the bottom $\psi_{m,n}(x)$, $\psi_{m+1,n}(x)$, $\psi_{m+2,n}(x)$, ii) Three Haar wavelets at three consecutive scales on a dyadic grid, iii) Three Haar wavelets at different scales. This time the Haar wavelets are not defined on a dyadic grid and are hence not orthogonal to each other.

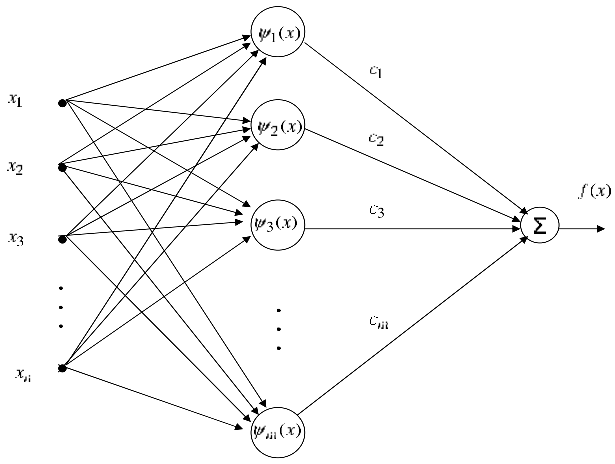


Fig. 14.3 Feed-forward neural network with Hermite polynomial basis functions

input vector of the neural network. The activation function is usually a sigmoidal function $h(x)=\frac{1}{1+e^{-x}}$ while in the case of radial basis functions networks it is a Gaussian [138]. Several learning algorithms for neural networks have been studied.

The objective of all these algorithms is to find numerical values for the network's weights so as to minimize the mean square error E_{RMS}. The algorithms are usually based on first and second order gradient techniques. These algorithms belong to: (i) batch-mode learning, where to perform parameters update the outputs of a large training set are accumulated and the mean square error is calculated (back-propagation algorithm, Gauss-Newton method, Levenberg-Marquardt method, etc.), (ii) pattern-mode learning, in which training examples are run in cycles and the parameters update is carried out each time a new datum appears (Extended Kalman Filter algorithm) [341].

14.3.2 The Gauss-Hermite Series Expansion

Next, as orthogonal basis functions of the feed-forward neural network Hermite polynomials are considered. These are the spatial components $X_k(x)$ of the solution of Schrödinger's differential equation and describe a stochastic oscillation:

$$X_k(x) = H_k(x)e^{\frac{-x^2}{2}},\ k = 0,1,2,\cdots \tag{14.35}$$

where $H_k(x)$ are the Hermite orthogonal functions (Fig. 14.4). The Hermite functions $H_k(x)$ are the eigenstates of the quantum harmonic oscillator. The general relation for the Hermite polynomials is

$$H_k(x) = (-1)^k e^{x^2} \frac{d^{(k)}}{dx^{(k)}} e^{-x^2} \tag{14.36}$$

According to Eq. (14.36) the first five Hermite polynomials are:

$H_0(x) = 1$, $H_1(x) = 2x$, $H_2(x) = 4x^2 - 2$, $H_3(x) = 8x^3 - 12x$, $H_4(x) = 16x^4 - 48x^2 + 12$

It is known that Hermite polynomials are orthogonal, i.e. it holds

$$\int_{-\infty}^{+\infty} e^{-x^2} H_m(x)H_k(x)dx = \begin{cases} 2^k k!\sqrt{\pi}\ if\ m = k \\ 0\ if\ m \neq k \end{cases} \tag{14.37}$$

Using now, Eq. (14.37), the following basis functions can be defined [319]:

$$\psi_k(x) = [2^k \pi^{\frac{1}{2}} k!]^{-\frac{1}{2}} H_k(x) e^{-\frac{x^2}{2}} \tag{14.38}$$

where $H_k(x)$ is the associated Hermite polynomial. From Eq. (14.37), the orthogonality of basis functions of Eq. (14.38) can be deduced, which means

$$\int_{-\infty}^{+\infty} \psi_m(x)\psi_k(x)dx = \begin{cases} 1\ if\ m = k \\ 0\ if\ m \neq k \end{cases} \tag{14.39}$$

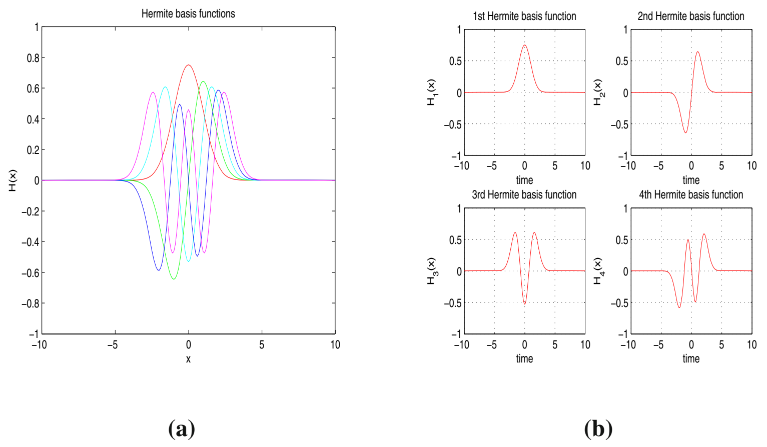


Fig. 14.4 (a) First five one-dimensional Hermite basis functions (b) Analytical represenation of the 1D Hermite basis function

Moreover, to succeed multi-resolution analysis Hermite basis functions of Eq. (14.38) are multiplied with the scale coefficient α. Thus the following basis functions are derived

$$\beta_k(x,\alpha) = \alpha^{-\frac{1}{2}}\psi_k(\alpha^{-1}x) \tag{14.40}$$

which also satisfy orthogonality condition

$$\int_{-\infty}^{+\infty}\beta_m(x,\alpha)\beta_k(x,\alpha)dx = \begin{cases} 1\ if\ m=k \\ 0\ if\ m\neq k \end{cases} \tag{14.41}$$

Any function $f(x)$, $x\in R$ can be written as a weighted sum of the above orthogonal basis functions, i.e.

$$f(x) = \sum_{k=0}^{\infty} c_k\beta_k(x,\alpha) \tag{14.42}$$

where coefficients c_k are calculated using the orthogonality condition

$$c_k = \int_{-\infty}^{+\infty} f(x)\beta_k(x,\alpha)dx \tag{14.43}$$

Assuming now that instead of infinite terms in the expansion of Eq. (14.42), M terms are maintained, then an approximation of $f(x)$ is succeeded. The expansion of $f(x)$ using Eq. (14.42) is a Gauss-Hermite series. Eq. (14.42) is a form of Fourier expansion for $f(x)$. Eq. (14.42) can be considered as the Fourier transform of $f(x)$ subject only to a scale change. Indeed, the Fourier transform of $f(x)$ is given by

$$F(s) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} f(x)e^{-jsx}dx \Rightarrow f(x) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} F(s)e^{jsx}ds \tag{14.44}$$

The Fourier transform of the basis function $\psi_k(x)$ of Eq. (14.38) satisfies [319]

$$\Psi_k(s) = j^n \psi_k(s) \tag{14.45}$$

while for the basis functions $\beta_k(x,\alpha)$ using scale coefficient α it holds that

$$B_k(s,\alpha) = j^n \beta_k(s,\alpha^{-1}) \tag{14.46}$$

Therefore, it holds

$$f(x) = \sum_{k=0}^{\infty} c_k \beta_k(x,\alpha) \overset{F}{\rightarrow} F(s) = \sum_{k=0}^{\infty} c_k j^n \beta_k(s,\alpha^{-1}) \tag{14.47}$$

which means that the Fourier transform of Eq. (14.42) is the same as the initial function , subject only to a change of scale. The structure of a a feed-forward neural network with Hermite basis functions has been given in Fig. 14.3(b).

14.3.3 Neural Networks Using 2D Hermite Activation Functions

Two-dimensional Hermite polynomial-based neural networks can be constructed by taking products of the one dimensional basis functions $B_k(x,\alpha)$ [319]. Thus, setting $x = [x_1, x_2]^T$ one can define the following basis functions

$$B_k(x,\alpha) = \frac{1}{\alpha} B_{k_1}(x_1,\alpha) B_{k_2}(x_2,\alpha) \tag{14.48}$$

These two dimensional basis functions are again orthonormal, i.e. it holds

$$\int d^2x B_n(x,\alpha) B_m(x,\alpha) = \delta_{n_1 m_1} \delta_{n_2 m_2} \tag{14.49}$$

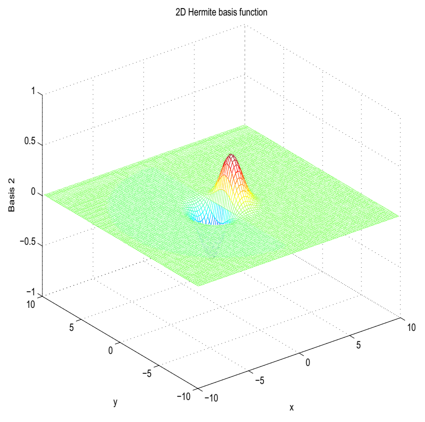
The basis functions $B_k(x)$ are the eigenstates of the two dimensional harmonic oscillator and form a complete basis for integrable functions of two variables. A two dimensional function $f(x)$ can thus be written in the series expansion:

$$f(x) = \sum_{k_1,k_2}^{\infty} c_k B_k(x,\alpha) \tag{14.50}$$

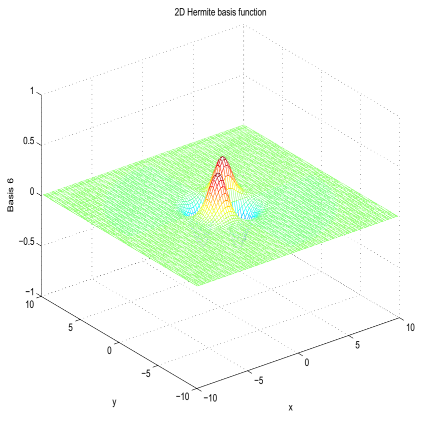
The choice of an appropriate scale coefficient α and maximum order k_{max} is of practical interest. The coefficients c_k are given by

$$c_k = \int dx^2 f(x) B_k(x,\alpha) \tag{14.51}$$

Indicative basis functions $B_2(x,\alpha)$, $B_6(x,\alpha)$, $B_9(x,\alpha)$, $B_{11}(x,\alpha)$ and $B_{13}(x,\alpha)$, $B_{15}(x,\alpha)$ of a 2D feed-forward neural network with Hermite basis functions are depicted in Fig. 14.5, Fig. 14.6, and Fig. 14.7.

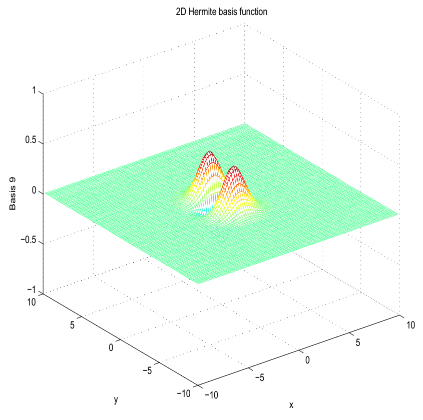


(a)

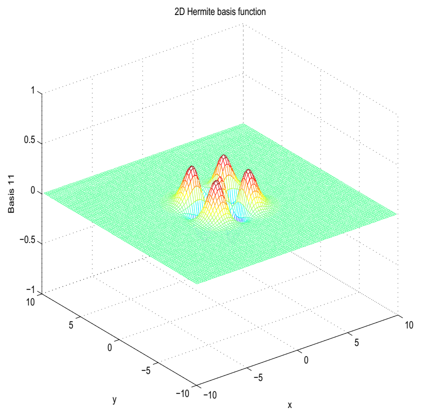


(b)

Fig. 14.5 2D Hermite polynomial activation functions: (a) basis function $B_2(x,\alpha)$ (b) basis function $B_6(x,\alpha)$.

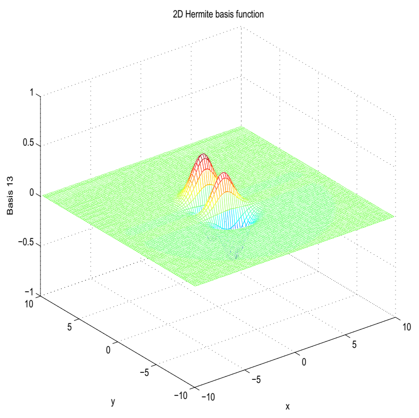


(a)

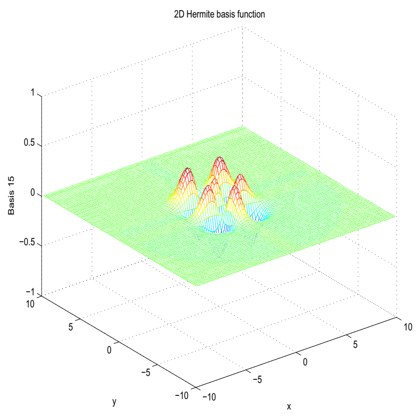


(b)

Fig. 14.6 2D Hermite polynomial activation functions: (a) basis function $B_9(x,\alpha)$ (b) basis function $B_{11}(x,\alpha)$.



(a)



(b)

Fig. 14.7 2D Hermite polynomial activation functions: (a) basis function $B_{13}(x,\alpha)$ (b) basis function $B_{15}(x,\alpha)$.

14.4 Results on Flexible-Link Control and Vibrations Suppression

14.4.1 The Flexible-Link Robot Model

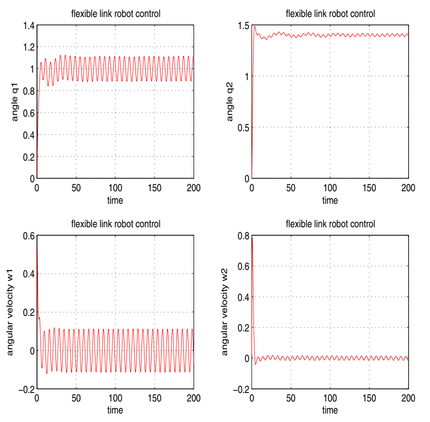
The 2-DOF flexible link robot of Fig. 1.3 is considered again. The robot is planar and consists of two flexible links of length $L_1 = 0.45m$ and $L_2 = 0.45m$, respectively. The dynamic model of the robot is given by Eq. (1.23). The elements of the inertia matrix M are:

$$M_{11} = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix},\ M_{12} = M_{21}^T = \begin{pmatrix} 1 & 1 & 0.2 & 0.3 \\ 0.5 & 0.1 & 2 & 0.7 \end{pmatrix},\ M_{22} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

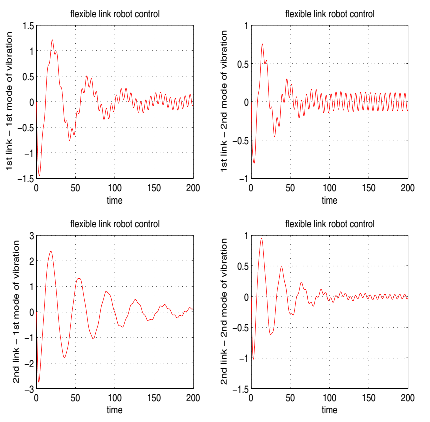
The damping matrix was taken to be $D = diag\{0.04, 0.08, 0.03, 0.06\}$ while the stiffness matrices was selected as $K = diag\{0.02, 0.04, 0.03, 0.06\}$. The inverse dynamics control law given in Eq. (1.52) and Eq. (1.53) is employed. The selection of the gain matrices K_p and K_d determines the transient response of the closed loop system. The following controller gains have been considered: $K_p = diag\{0.2, 0.2\}$ and $K_d = diag\{0.1, 0.1\}$. The desirable joints positions are $\theta_{d_1} = 1\ rad$ and $\theta_{d_2} = 1.4\ rad$.

14.4.2 Control Using Hermite Polynomial-Based Neural Networks

The neural controller operates in parallel to a PD controller, as shown in Fig. 1.5. An additive disturbance torque equal to $d_i(t) = 0.3cos(t)$ was considered to appear



(a)



(b)

Fig. 14.8 (a) Diagrams of the angular positions θ_1 and θ_2, and of the angular velocities ω_1 and ω_2 of the joints of the flexible link manipulator when only a PD controller is used, (b) Diagrams of the first two vibration modes v_{11} and v_{12} of the first joint, and of the vibration modes v_{21} and v_{22} of the second joint of the flexible link manipulator when only a PD controller is used.

at each joint. The simulation diagram of Fig. 14.8(a) shows the evolution in time of the angles of the robot's joints θ_1 and θ_2, respectively, when only a PD controller is used in the loop and the flexibility of the link is not taken into account int the controller's design. In Fig. 14.8(b) the evolution in time of the vibration modes of the first link v_{11}, v_{12} and of the second link v_{21} and v_{22}, respectively, is presented. It can be seen that vibrations around the desirable joint positions cannot be eliminated.

Next, a control loop with a neural controller which used the Haar orthogonal wavelet functions of subsection 14.2.5 has been considered. The neural controller is a single layer NN with wavelet basis functions, as shown in Fig. 14.3, and it is linear with respect to the output weights. Fig. 14.9(a) presents the evolution in time of the joint angles of the robot when NN with wavelet basis functions are used for suppressing the vibrations of the flexible links. Fig. 14.9(b) shows the variation in time of the joint angles of the robot θ_1 and θ_2, respectively, when the neural controller uses Gauss-Hermite basis functions.

Finally, simulation diagrams are presented showing how the proposed neural controllers succeed the suppression of the vibration modes of the flexible links. Fig. 14.10(a) shows the evolution in time of the vibration modes v_{11} and v_{12} of the first link, as well as of the vibration modes v_{21} and v_{22} of the second link, when the neural controller uses wavelet basis functions. Similarly, Fig. 14.10(b) shows the variation in time of the vibration modes v_{11}, v_{12} of the first link and v_{21}, v_{22} of the second link, respectively, when the neural controller uses Gauss-Hermite basis functions.

From the simulation experiments it can be observed, that using a neural controller with basis functions which are localized both in space and frequency allows better

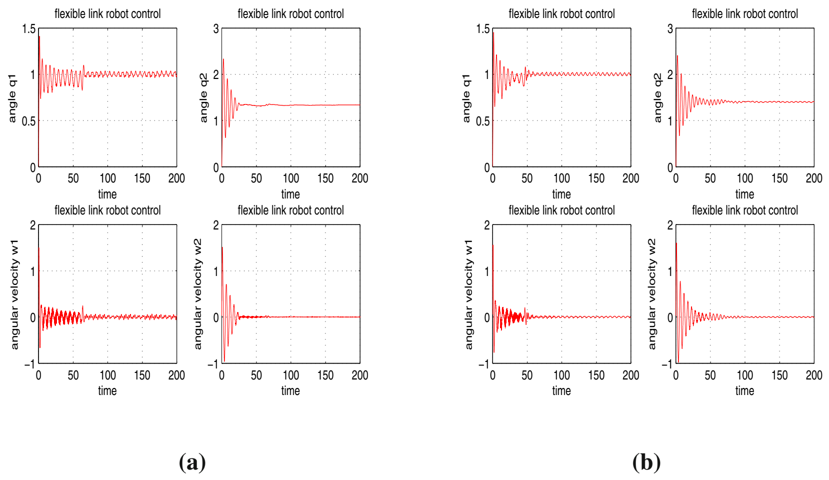
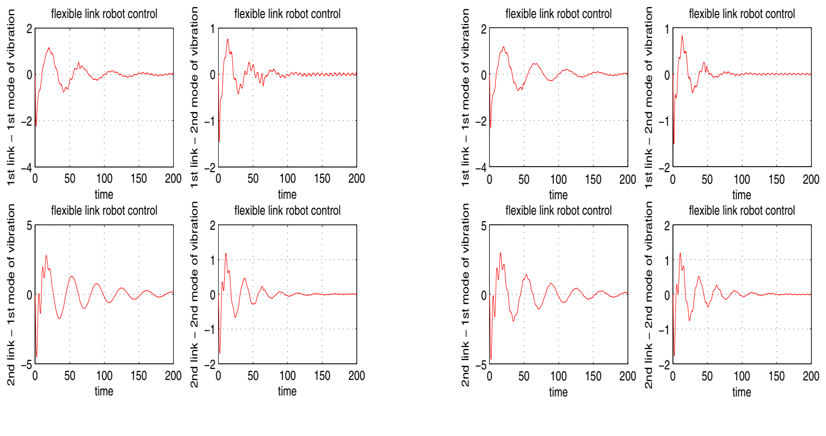


Fig. 14.9 Diagrams of the robot's joint angles θ_1 and θ_2 (a) when a wavelet neural network is included in the control loop to suppress vibrations, (b) a Gauss-Hermite neural network is included in the control loop to suppress vibrations

approximation of the multi-frequency characteristics of the vibrating robot links. The angles of the robot's joints converge to the desirable set-points θ_1^d and θ_2^d, while fast and efficient suppression of the vibration modes v_{11}, v_{12} and v_{21}, v_{22} is also succeeded.

Comparing the Gauss-Hermite basis functions to the Haar wavelet basis functions one can note the following: (i) both are basis functions which are local in space and spatial frequency. This allows better approximation of the multi-frequency characteristics of vibrating structures, such as the flexible-link robot, (ii) both satisfy the orthogonality property which helps to locally improve the accuracy of approximation of the unknown system dynamics. This means that the neural controller can be dynamically expanded by adding new basis functions which are orthogonal to the existing ones and the coefficients of the new basis functions can be computed independently of the existing coefficients (iii) unlike wavelet basis functions, the Gauss-Hermite basis functions have a clear physical meaning, since they represent the solutions of differential equations of stochastic oscillators and each neuron can be regarded as the frequency filter of the respective vibration eigenfrequency, (iv) unlike the Haar wavelet basis functions, the Gauss-Hermite basis functions remain almost unchanged by the Fourier transform, which means that the weights of the associated neural network demonstrate the energy which is distributed in the various eigenmodes of the vibrating structure. This in turn allows to define thresholds for truncating the expansion and using neural controller with a small number of nodes and weight coefficients, (v) in the above simulation experiments it was observed that the control signal (torque) generated by the neural controller with Hermite basis functions is smoother than the control signal of the neural controller which employs wavelet basis functions.



(a) **(b)**

Fig. 14.10 Diagrams of the first two vibration modes v_{11} and v_{12} of the first joint, and of the vibration modes v_{21} and v_{22} of the second joint of the flexible link manipulator (a) when a wavelet neural network is included in the control loop to suppress vibrations, (b) a Gauss-Hermite neural network is included in the control loop to suppress vibrations

Chapter 15
Machine Learning Methods for Industrial Systems Fault Diagnosis

Abstract. Machine learning methods can be of particular interest for fault diagnosis of systems that exhibit event-driven dynamics. For this type of systems fault diagnosis based on automata and finite state machine models has to be performed. In this chapter the application of fuzzy automata for fault diagnosis is analyzed. The output of the monitored system is partitioned into linear segments which in turn are assigned to pattern classes (templates) with the use of membership functions. A sequence of templates is generated and becomes input to fuzzy automata which have transitions that correspond to the templates of the properly functioning system. If the automata reach their final states, i.e. the input sequence is accepted by the automata with a membership degree that exceeds a certain threshold, then normal operation is deduced, otherwise, a failure is diagnosed. Fault diagnosis of a DC motor is used as a case study.

15.1 Automata in Fault Diagnosis Tasks

15.1.1 Fault Diagnosis of Systems with Event-Driven Dynamics

As shown in several research studies machine learning methods can be particularly useful for fault diagnosis [418],[419],[420]. To some level, industrial systems can be described by continuous-time variables and for such models the fault detection and isolation methodologies presented in Chapters 9 and 10 have been proven to be quite effective [16],[161]. On the other hand there are industrial systems which can be better described using event driven dynamics [116],[163]. For this type of systems fault diagnosis based on automata has to be performed [7],[36],[316],[364].

The use of automata in fault diagnosis tasks has gained particular attention in the case of discrete event dynamical systems and hybrid systems [38],[70],[71],[129] [243],[456]. Fault diagnosis based on syntactic analysis considers that the output of a dynamic system is a sequence of linear segments of variable length and slope which leads from an initial state to a final one. This sequence of segments is a regular expression and according to Kleene's theorem is equivalent to a finite automaton

M [200],[271]. Thus the output of the system can be described by the five-tuple $M = \{\Phi, B, \delta, s, F\}$ where: i) Φ is the set of states, ii) B is the set of *input* strings b, iii) $\delta : \Phi \times B \rightarrow \Phi$ is the transition function, iv) $s \in \Phi$ is the start state, and v) $F \subseteq \Phi$ is the set of final states. The automaton is said to accept the input string b if starting from s and following the permitted transitions a final state is reached. A string of segments leading to a final state of M is a regular expression and is called *pattern*. The language of M is denoted by $L(M)$ and consists of all regular expressions.

In fault diagnosis based on syntactic analysis, the system's failures can be detected if the following two strings are compared: i) pattern a which is the segmented output of the properly functioning system, ii) string b which is the segmented output of the monitored system. If b matches a, i.e. b is accepted by M, then the monitored system operates properly and no fault is detected. If b does not match a, i.e. b is rejected by M, then a fault is deduced. To isolate that fault, pattern matching between b and a set of fault patterns A_f can be attempted. Each fault pattern A_{f_i} is in turn equivalent to an automaton M_{f_i}.

The detection of a fault is based on distance or similarity measures [425]. If the distance between the input string b and pattern a exceeds a certain threshold then a fault is reported, otherwise the system is assumed to work properly. The distance between two strings is related to the sequence of edit operations (substitution, insertion and deletion) required to transform one string into another.

In this chapter, to compare strings a and b, similarity measures are used and the concept of fuzzy automata is employed. Update of the membership value of the state ϕ_j of M, which is connected to state ϕ_i via the transition a_i takes place using the fuzzy inference rule [393],[432],[433]. After having applied input b, the membership function of the final state of a provides a measure of the matching between strings b and a. If this measure goes below a certain threshold then fault is deduced.

15.1.2 System Modelling with the Use of Finite Automata

Automata have been used to model the dynamics of Discrete Event Systems (DES) [279],[297],[298],[303],[304],[315]. In that case automata describe dynamical systems the behavior of which cannot be completely represented by differential equations, because of the existence of asynchronous events that affect the system's state. Apart from industry and manufacturing, such systems are also met in database concurrency control, telecommunication or computer networks, and in biomedical applications such as clinical monitoring. Usually a DES is described by a finite automaton, which is defined by the five-tuple [230],[306],[310],[311],[324],[424]

$$M = (\Phi, B, \delta, S, q_0) \tag{15.1}$$

where Φ is a set of discrete states, B is the set of events that enable the transition between states, $\delta : \Phi \times B \rightarrow \Phi$ is the transitions mapping, and q_0 is the initial state. An example of a DES is depicted in Fig. 15.1(a) where $\Phi = \{E, F, P\}$, $B = \{\alpha_1, \alpha_2, \alpha_3, \beta_1, \beta_2, \beta_3\}$, S contains the arcs that appear in Fig. 15.1(a). At time

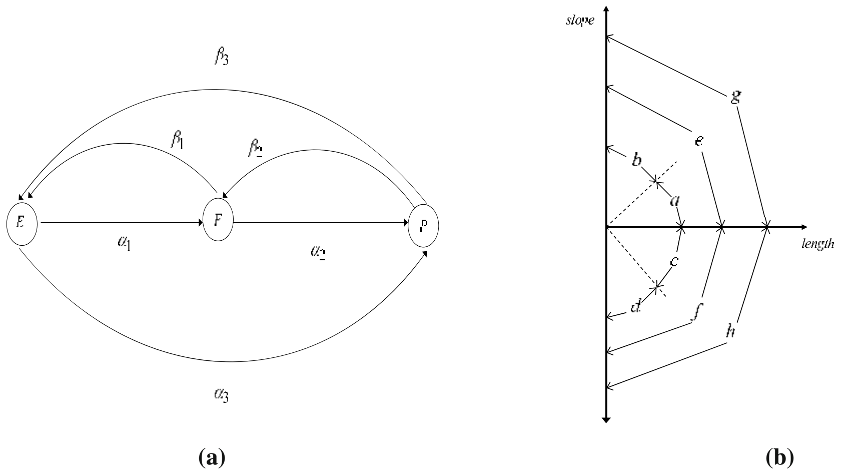


Fig. 15.1 (a) Modelling of a discrete event system with the use of an automaton (b) Class labels for segments of variable slope (vertical axis) and of variable length (horizontal axis)

instant k the system can be in one of the nodes depicted in the above graph, for instance node i, thus the state of the system is the vector $\phi^k = [0,\cdots,1,\cdots,0]$.

The transition between the various states of the automaton are enabled through the events $a_i,\ i=1,2,3$, and $\beta_i,\ i=1,2$. Each event is associated with a transition matrix which in the case of the automaton shown in Fig. 15.1(a) becomes [230]

$$\begin{array}{c}\alpha_1 = \begin{pmatrix}0&1&0\\0&0&0\\0&0&0\end{pmatrix},\ \beta_1 = \begin{pmatrix}0&0&0\\1&0&0\\0&0&0\end{pmatrix},\ \alpha_2 = \begin{pmatrix}0&0&0\\0&0&1\\0&0&0\end{pmatrix}\\ \beta_2 = \begin{pmatrix}0&0&0\\0&0&0\\0&1&0\end{pmatrix},\ \alpha_3 = \begin{pmatrix}0&0&1\\0&0&0\\0&0&0\end{pmatrix},\ \beta_3 = \begin{pmatrix}0&0&0\\0&0&0\\1&0&0\end{pmatrix}\end{array} \tag{15.2}$$

Thus, if the system is initially in state $F=[0,1,0]$ and event β_1 appears then the next state of the DES is state $E=[1,0,0]$ as shown in the following calculation

$$\begin{pmatrix}0&1&0\end{pmatrix}\cdot\begin{pmatrix}0&0&0\\1&0&0\\0&0&0\end{pmatrix} = \begin{pmatrix}1&0&0\end{pmatrix} \tag{15.3}$$

In DES the following properties are of importance (i) observability, i.e. measure of the uncertainty that a supervisor can have about the state of the automaton, (ii) diagnosability, i.e. the possibility to deduce a fault (occurrence of an event that results in a faulty state) in the DES by monitoring a sequence of observable events. The diagnoser is an automaton that accepts the string of the events which follow the fault. Several problems of diagnosability and diagnosers' design have been studied in the recent bibliography of discrete event dynamical systems [38],[70],[106],[191],[236],[296],[363]. These properties have been also extended to fuzzy DES [230],[311]. Recently, the results of [363] have been generalized to the framework of stochastic DESs in [235],[312].

15.1.3 System Modelling with the Use of Fuzzy Automata

Crisp DES are not adequate for the cases in which the states and the transitions of a system are uncertain. Subjective human observation, judgement and interpretation invariably play a significant role in describing the status of a state, usually not crisp. Thus, classical automata theory was extended in [309] where a framework of automata based on completed residuated lattice-valued logic was established and then [225] dealt with automata theory based on ordered lattice monoids. To overcome the limitations of crisp DES, fuzzy discrete event systems (FDES) have been proposed [230],[311]. In FDES the possibility of being at a state of the automaton depicted in Fig. 15.1(a), at a time instant k, is denoted by a membership function [301],[302]. Assume that the states of the automaton are $\phi_1, \phi_2, \cdots, \phi_n$, and the associated membership functions at time instant k are $[\mu_1^k, \mu_2^k, \cdots, \mu_n^k]$. Then at time instant k the FDES is at a state defined by the vector $g^k = [\mu_1^k, \cdots, \mu_n^k]$. The transition matrix between the state of the FDES at time instant k and the state of the FDES at time instant $k+1$ is given by

$$\alpha = [\alpha_{ij}]_{n\times n} = \begin{pmatrix} \alpha_{11} & \cdots & \alpha_{1n} \\ \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \\ \alpha_{n1} & \cdots & \alpha_{nn} \end{pmatrix} \tag{15.4}$$

A fuzzy discrete event system is represented by a fuzzy automaton where [230],[424]: (i) the possibility to find the automaton at a certain state at time instant k is given by a membership function, (ii) the possibility of a transition between states to take place is also given by a membership function. It is noted that the equivalence between probabilistic and possibilistic (fuzzy) representations of uncertainty has been studied in [84].

Next, it is assumed that the states of the automaton depicted in Fig. 15.1(a) are fuzzy and the associated membership functions are:

$$\tilde{\phi} = [0.4, 0.8, 0] \tag{15.5}$$

It is also assumed that the fuzzy event $\tilde{\alpha}_1$ is given by the transition matrix

$$\tilde{\alpha}_1 = \begin{pmatrix} 0.1 & 0.9 & 0.1 \\ 0.2 & 0.1 & 0.2 \\ 0 & 0.1 & 0.1 \end{pmatrix} \tag{15.6}$$

i.e. again it is most likely to have a transition from state E to state F. Then using the max-product inference one has

$$\tilde{\phi} \circ \tilde{\alpha}_1 = [0.4, 0.8, 0] \circ \begin{pmatrix} 0.1 & 0.9 & 0.1 \\ 0.1 & 0.1 & 0.1 \\ 0 & 0.1 & 0.1 \end{pmatrix} = \begin{pmatrix} 0.08 & 0.36 & 0.08 \end{pmatrix} \tag{15.7}$$

A typical definition of a fuzzy automaton is the five-tuple $\tilde{M} = (\tilde{\Phi}, \tilde{B}, \tilde{\delta}, \tilde{s}, \tilde{F})$, where

- $\tilde{\Phi}$ is the finite set of fuzzy states. A membership value $\mu(\phi_i) \in [0,1]$ is assigned to each state.
- $\tilde{B}$ is the set of inputs where each input has a membership function $\mu(b_i) \in [0,1]$, provided by the classification procedure.
- $\tilde{\delta} : \Phi \times B \to \Phi$ is the set of fuzzy transitions, where a membership function $\mu(\tilde{\delta}) \in [0,1]$ is associated with each transition from state ϕ_i to state ϕ_j.
- $\tilde{s}$ is the fuzzy start state.
- $\tilde{F} \subset \tilde{\Phi}$ is the set of fuzzy final states.

15.1.4 Monitoring Signals with the Use of Fuzzy Automata

It was shown that fuzzy automata can be used to describe discrete event systems with uncertain states or uncertain transitions. Fuzzy automata can be also used to describe the degree of similarity between the output of a dynamical system, and a reference signal [433]. If each linear segment of the output is considered as a state, then the monitored system can be viewed as a discrete-state system. The knowledge of the system states and of the transitions between different states is subject to uncertainty. This uncertainty can be described by a possibilistic model such as a fuzzy automaton. In this case fuzzy states and fuzzy transitions are assumed for the description of the system's condition.

The advantages of fuzzy automata for describing the matching of dynamical system's output to a reference signal are summarized as follows:

- Fuzzy automata give a measure of similarity between patterns that is tolerant to measurement noise. Fuzzy automata can be used instead of Markov models to represent discrete-state systems subject to uncertainty. Unlike Markov models where transition probabilities have to be approximated, fuzzy automata have transition membership functions which can be provided by experts.
- Unlike correlation, syntactic analysis based on fuzzy automata permits to associate changes of certain parts of the output signal with parametric changes of the monitored system. Correlation provides a similarity measure between signals but does not identify the uneven segments in case of mismatch.
- In fuzzy automata, fault thresholds are defined by experts, thus human knowledge about the monitored system can be exploited.

15.2 A Fault Diagnosis Approach Based on Fuzzy Automata

15.2.1 Generation of the Templates String

The main concept in Fault Detection and Isolation (FDI) based on fuzzy automata, is to divide the output signal into consecutive linear segments and to classify each one of them in pattern classes according to a membership function. A candidate segment of n points is selected and the line that connects the first to the last point is calculated. If the distances of all points from this line are below a certain threshold ε then it is considered that all n points belong to the same segment. Otherwise, the first point $(x_j,y_j),\ j \in \{1,2,\cdots,n\}$ which exceeds ε is found, the candidate segment $(x_1,y_1),\cdots,(x_j,y_j)$ is defined and a new check is performed to see if points 1 to j can be assigned to the same segment.

To decompose the output signal into segments, a sliding window is used. The size of the sliding window determines the number of segments. A classification algorithm, assigns each segment to a template (pattern class) and provides also the corresponding membership function. The steps of the segmentation procedure are [198]:

Step 1: Preprocessing of the output signal. First the output signal is filtered with a low pass filter to remove high frequency noise. The preprocessed signal is a set of points $\{(x_1,y_1),\cdots,(x_N,y_N)\}$ where x_i is measured in time units and $y_i (1 \leq i \leq N)$ is the associated output sample.

Step 2: Segmentation of the output signal. A subset of points $X_i^{(0)}$: $(x_1,y_1)\cdots(x_n,y_n)$ is collected, and the equation of the line that connects the first to the last element of $X_i^{(0)}$ is calculated, $L: Ax+By+C=0$. The segment's end is the last point in $X_i^{(0)}$ which has a distance from L less than ε. Thus, if $\exists (x_j,y_j), j=1,\cdots,n$ such that the distance D of (x_j,y_j) from L exceeds a threshold ε, i.e.

$$D\{(x_j,y_j), Ax+By+C=0\} > \varepsilon \tag{15.8}$$

then n is set to $n = min\{j\}$ satisfying Eq. (15.8) and the calculation of L is repeated for the subset of data $C_i^{(1)} = \{(x_1,y_1),\cdots,(x_j,y_j)\}$. The segmentation algorithm can be summarized as follows:

• Inputs: threshold ε, $X_i^{(0)} = \{(x_1,y_1),\cdots,(x_n,y_n)\}$
• Output: segments $x_h,\ h=1,2,\cdots$.

1. Set $j=n$, $h=1$.
2. Examine the last point (x_j,y_j) and calculate $A_h = (y_j - y_1)$, $B_h = (x_j - x_1)$ and $C_h = x_1 y_j - x_j y_1$.
3. For $i = 1,2,\cdots,j$ calculate the distance of (x_i,y_i) from $L: A_h x + B_h y + C_h = 0$
4. If Eq. (15.8) is true then set $j = j-1$ and go to 2.
5. If Eq. (15.8) is false $\forall i = 1,\cdots,j$ then $X_i^{(1)} = \{(x_1,y_1),\cdots,(x_j,y_j)\}$
6. Set $h = h+1$, $j = n$, go to 2 and repeat for another candidate segment.

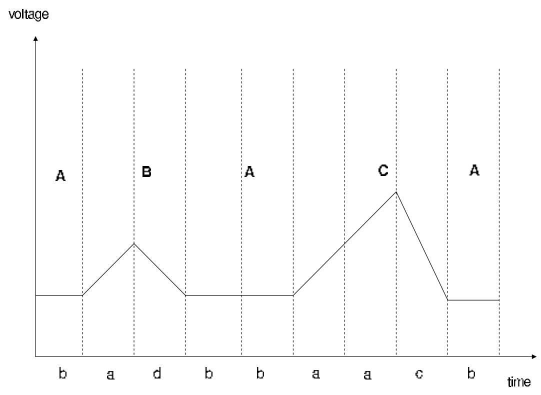


Fig. 15.2 Generation of a string of templates from a monitored signal

Step 3: To organise segments in pattern classes (templates), each segment x_i is taken to be a line described by an equation $A_hX + B_hY + C_h = 0, \quad h = 1, \cdots, M$ where M is the number of templates in which the output is segmented. The slope $-\frac{A_h}{B_h}$ of each segment is calculated. The segments are organised in classes according to their slope and length, using algorithms of statistical pattern recognition (e.g. C-Means). An example of class labels is given in Fig. 15.1(b).

Once the pattern classes have been found, a neural network can be used to memorize the mapping of input segments to classes. By considering membership in multiple pattern classes, more information is provided to the syntactic analyzer. In that case there will be multiple paths that connect the start to the end state of the automaton.

15.2.2 Syntactic Analysis Using Fuzzy Automata

As already mentioned, the sequence of segments of the output signal leads from an initial to a final state and can be considered as a formal language. Thus it can be recognized by an automaton. Usually, this automaton can be decomposed into several sub-automata. Each sub-automaton is capable to recognize a specific part of the output signal and operates as a transition between the states of the main automaton [415],[416], [433].

The syntactic analyzer consists of the main automaton and a number of sub-automata and is shown in Fig. 15.3. Syntactic analysis is accomplished through the following steps:

• *Step 1*: The main automaton is designed. This is also a finite state machine where the transitions correspond to the sub-automata associated with the patterns of the output signal. The degree of activation of a transition of the main automaton depends on the degree of acceptance of a pattern of the output signal. If the main automaton

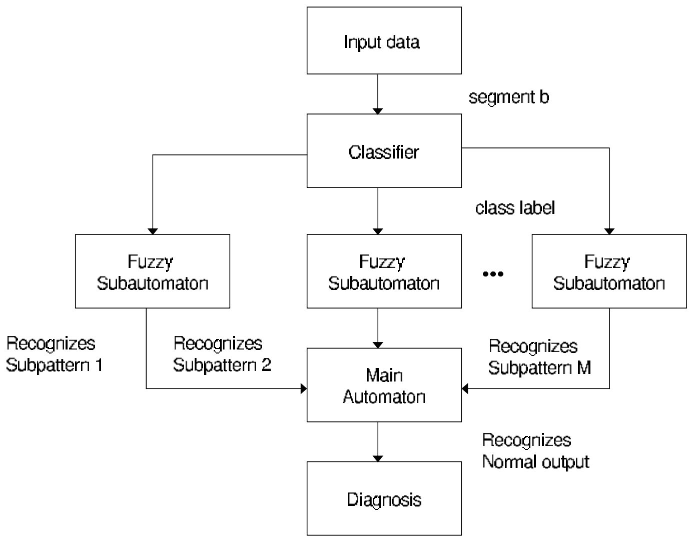


Fig. 15.3 Automata-based system for fault diagnosis

recognizes the input string which comes from the properly functioning system then no fault exists.

To design a sub-automaton for a certain pattern, all occurrences of the pattern are recorded. Each occurrence is a sequence of alphabet symbols. Next, a path of states is inserted to the sub-automaton and is used to recognize this sequence of symbols. Human experts provide the transition possibility (transition threshold) for each sub-automaton.

Initialization with memberships reflecting the maximum ambiguity $\mu(\delta_{ij}) = 0.5$ is used to allow any possible transition that occurs in the sub-automaton. Prunning of redundant transitions may result in simpler fuzzy sub-automata.

• *Step 2*: Synchronization of the input. The objective is to find a unique state in the main automaton that is activated by the incoming string of templates. Initially all states of the main automaton have a non-zero membership function and are examined as possible start states.

The first template is applied to all sub-automata and is examined if the sub-automata reach a final state. The procedure continues until only one state of the main automaton remains active. At this point synchronization of the input has been succeeded.

• *Step 3*: The sequence of templates $a^n b^n c^n d^n \cdots$ continues and becomes input to each sub-automaton and if a final state is reached with a membership degree above a certain threshold then a specific pattern is recognized. When a sub-automaton terminates, then transition to another state of the main automaton takes place, and a

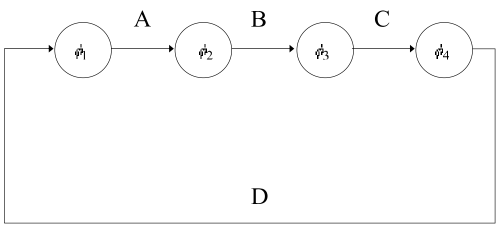


Fig. 15.4 Main automaton

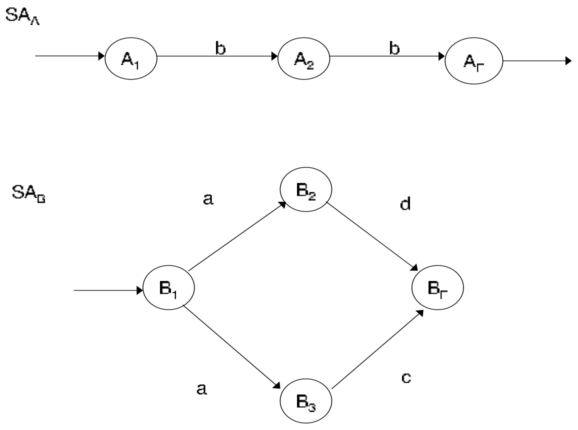


Fig. 15.5 Automata corresponding to subpatterns A and B

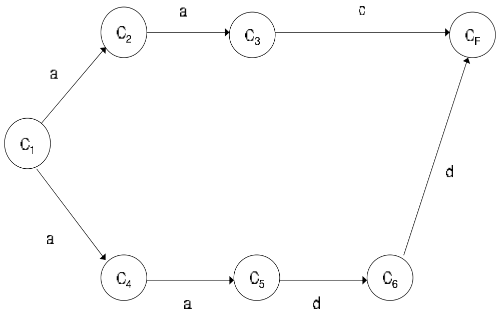


Fig. 15.6 Automaton corresponding to subpattern C

fuzzy membership degree is assigned to this transition. Thus these automata can be considered as transitions of a higher level automaton which is the main automaton. The sub-automata which are associated with the transitions A, B, C of the main automaton of Fig. 15.4, are shown in Fig. 15.5 and Fig. 15.6.

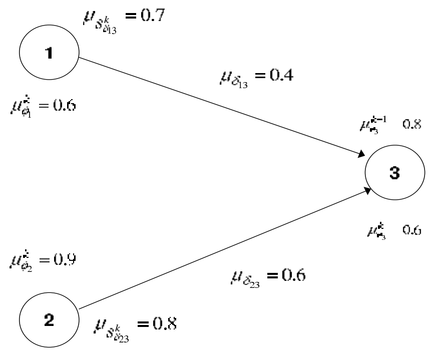

$\mu_{s^k_{\delta_{13}}} = 0.7$
1
$\mu_{\delta_{13}} = 0.4$
$\mu_{\phi_1}^{\cdot} = 0.6$
$\mu_{\phi_3}^{k-1}$ 0.8
3
$\mu_{\phi_3}^{k}$ 0.6
$\mu_{\phi_2}^{\cdot} = 0.9$
$\mu_{\delta_{23}} = 0.6$
2
$\mu_{s^k_{\delta_{23}}} = 0.8$

Fig. 15.7 Update rule of fuzzy subautomaton

It should be noted that the fault diagnosis scheme shown in Fig. 15.3 is a centralized one. It considers that all information contained in the fuzzy sub-automata is collected centrally by the main automaton which gives diagnosis about the system's condition. However, centralized fault diagnosis may not be recommended for some spatially distributed systems (e.g. communication networks or the power generation and transmission grid). For the latter category of systems it may be preferable to have a set of local diagnosers (sub-automata) that will be running at different sites and will be processing local observations (sub-patterns), without explicit communication with a central unit, or with local communication subject to time delays. In this case, the concept of co-detectability (co-diagnosability) is of importance, i.e a fault in the overall system is detected if at least one diagnoser (sub-automaton) recognizes a faulty situation in a locally recorded sequence of events [235]. Decentralized fault diagnosis, as well as necessary and sufficient conditions for diagnosability in distributed systems has been a significant research direction in the area of distributed discrete event dynamical systems [30],[69],[93],[169],[312],[394].

15.2.3 Detection of Fault Patterns by the Fuzzy Automata

The segments sequence x is said to be recognized by the fuzzy automaton $M = \{\tilde{Q}, \tilde{\Sigma}, \tilde{\delta}, \tilde{s}, \tilde{F}\}$ if there is an initial state $\tilde{s}$, such that when applying a sequence of inputs x^* and following the transition rule given in Eq. (15.9), a final state $\tilde{f}$ is reached, with $\tilde{f}$ having a nonzero fuzzy membership value, above a certain threshold. The output signal can be decomposed into patterns, such as the one depicted in Fig. 15.2. Patterns are denoted by A which represents the 1st stage, B which represents the 2nd stage and C which is associated to the $3rd$ complex. In Fig. 15.4, the states of the main automaton are denoted by ϕ_1, ϕ_2, ϕ_3 and ϕ_4. The templates become inputs to the sub-automata which are associated to the transitions of the main automaton. The sub-automata update rule is given by Eq. (15.9):

$$\mu(\phi_j^{(k+1)}) = \begin{cases} max_{I_j^k} \{\mu(\phi_j^k), min(\mu(\phi_i^k), \mu(x_i^k))\} \ if \ \mu(\delta_{ij}^{(k)}) \leq \mu(x_i^{(k)}) \\ \mu\phi_i^{(k)} \ otherwise \end{cases} \tag{15.9}$$

where

- $\mu(\phi_j)^{(k)}$ is the fuzzy membership value of state ϕ_j at instant k
- $\mu(\phi_j)^{(k+1)}$ is the fuzzy membership value of the state ϕ_j at instant $k+1$
- $\mu(\delta_{ij})$ is the fuzzy membership value (threshold of activation) of the transition δ_{ij}. This value depends on how close is the input template associated with the transition from i to j.
- $I_j^{(k)}$ is the set of states ϕ_i which are connected to state ϕ^j through a transition.

For a transition from one state of the main automaton to another to take place, a sub-automaton must recognize the incoming templates and reach its final state. To design a sub-automaton several experiments are carried out and several instances of the same pattern are collected. These patterns are segmented and thus several strings that connect the start to the end state are obtained. These strings are the paths of the graph that represents the fuzzy sub-automaton. The simplest form of a sub-automaton is a single state-transitions chain between the start and the end state. Inside the sub-automata, for a transition to occur from a state ϕ_i to a state ϕ_j, the membership of the current input template x_i must be greater than or equal to the membership of the transition. If this condition is fulfilled, the membership of the resulting state ϕ_j is the maximum between the membership of the destination state ϕ_j and the minimum between the membership of the input template x_i and the membership of the source state ϕ_i. This is depicted in Fig. 15.7.

Example

It is assumed that the discrete states system is described by the main automaton given in Fig. 15.4. The subautomata associated with the transitions of the main automaton are given in Fig. 15.5 and Fig. 15.6. Transitions from states ϕ_1 or ϕ_3 can be enabled if the sequence of templates forming pattern A is recognized. Transition from state ϕ_2 is enabled if the sequence of templates forming pattern B is recognized. Finally, transition from state ϕ_4 can be enabled if the sequence of templates that form pattern C is recognized.

Initially, the start states of the sub-automata A, B, C are assigned a membership of 0.5 (it is equipossible for a state of a sub-automaton to be active or not). Once the pattern associated with a transition of the main automaton has been recognized, the main automaton moves from state ϕ_i to state ϕ_j. Without loss of generality it can be assumed that the main automaton shown in Fig. 15.4 is in state ϕ_3 and that the automaton which substantiates the transition from ϕ_3 to ϕ_4 is C shown in Fig. 15.6. Assume also that the incoming template is a. Then subautomaton C moves from state C_1 to state C_2 via a. If the next two input templates are a and c respectively then the subautomaton will move from state C_2 to state C_3 and finally to state C_F. The membership value of the final state C_F depends on the degree of matching of the input templates a, a, c to the templates that have been defined in the classification procedure. This process continues until all templates are presented to the system.

15.3 Simulation Tests of Fault Diagnosis with Fuzzy Automata

The performance of fuzzy automata for fault diagnosis is tested in the case of a DC-motor. An example of modelling and fault detection and isolation (FDI) for electromechanical systems with the use of formal languages theory, has been also given in [258]. A sinusoidal output of amplitude A and frequency f is assumed, as shown in Fig. 15.8 and Fig. 15.9. Faults should cause a change to the output's amplitude or frequency. In this case, syntactic analysis is expected to identify a change of the sinusoidal output pattern. The sinusoid is a single pattern and thus it can be recognized by one single fuzzy automaton. The events sequence that corresponds to the fault-free operation of the motor is given in Fig. 15.9.

In Section 4.3.1 the DC motor model controlled by the armature current was analyzed. Equivalently, one can derive a DC motor model controlled by the field (stator) voltage. This is explained in the following, using the DC motor circuit depicted in Fig. 4.4.

The produced (electromagnetic) torque T is proportional to the field current i_f, i.e. $T = K_f i_f$, where K_f is a constant. This happens because $T = K\Phi i_a$ where the magnetic flux Φ is proportional to the current i_f and the armature current i_a is assumed to be constant. The relation between i_f and v_f is $v_f = R_f i_f + L_f \dot{i_f}$ and the torque T which is developed is used to move a load with moment of inertia J and friction coefficient β i.e.

$$T = J \cdot \ddot{\theta}_m + \beta \dot{\theta}_m \tag{15.10}$$

Introducing T and i_f from the previous relations and applying the Laplace transform yields

$$\frac{\theta_m(s)}{v_f(s)} = \frac{K}{s(1+s\cdot\tau_f)(1+s\cdot\tau_m)} \tag{15.11}$$

where, $K = \frac{K_f}{\beta R_f}$, $\tau_f = \frac{L_f}{R_f}$ and $\tau_m = \frac{J}{\beta}$ are time constants. The corresponding state space equations are

$$\begin{pmatrix} \dot{\theta}_m \\ \dot{\omega}_m \\ \dot{\gamma}_m \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & \frac{-1}{\tau_f\tau_m} & \frac{-(\tau_f+\tau_m)}{\tau_f\tau_m} \end{pmatrix} \begin{pmatrix} \theta_m \\ \omega_m \\ \gamma_m \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \frac{K}{\tau_f\tau_m} \end{pmatrix} \cdot v_f \tag{15.12}$$

where θ_m is the angular position , ω_m is the angular speed and γ_m is the angular acceleration. In practice some of the parameters of the motor model, namely the moment of inertia or the friction coefficient β are not known or can be time varying.

When the parameters of the motor are subject to a fault/change then the motor's output deviates from the output of the nominal model, as shown in Fig. 15.8. The templates string that corresponds to the normal output ($R_f = 100$) is a simple chain

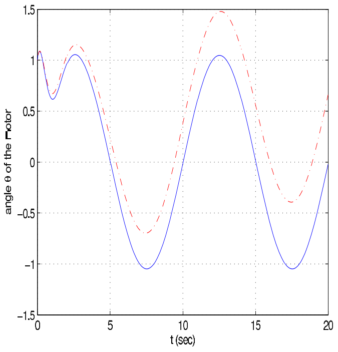


Fig. 15.8 Angle θ of the motor when operating in fault free conditions (continuous line), and when a fault takes place at the stator's resistance R_f (dashed line)

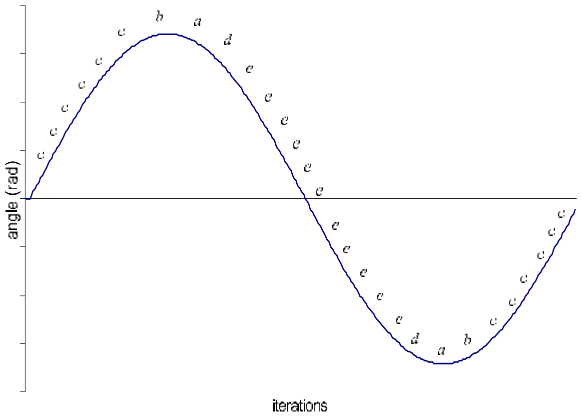


Fig. 15.9 Language of the normally functioning motor

$a_1 a_2 \cdots a_n$ (the start point of the automaton is connected to the end point through one single path).

The segmentation procedure of the reference output resulted into a symmetric chain a of $M = 27$ templates, namely $c^6 bade^{11} dabc^6$. The threshold for each transition was taken to be $\mu(\delta_{ij}) = 0.5$. The initial state memberships were set to $\mu(\phi_1^0) = 1$ and $\mu(\phi_i^0) = 0,\ i = 2, \cdots, M+1$. Then, a change in resistance R_f was introduced, the output was monitored and the associated string of templates b was generated. The elements b_i of b were classified to fuzzy sets and fuzzy memberships $\mu(b_i)$ were obtained. For $\mu(b_i) \geq \mu(\delta_{ij})$, $i = 1, \cdots, 27$ the application of Eq. (1) gave the membership of the final state $\mu(\phi_{M+1}) = min\{\mu(b_1), \mu(b_2), \cdots, \mu(b_M)\}$.

Changes of R_f caused a drop of $\mu(\phi_{M+1})$. For large changes of R_f mismatch between the template string of the reference and the monitored output appeared. For

$\mu(\phi_{M+1})$ greater than the fault threshold the monitored signal was considered to be normal. The fuzziness in the automaton, enabled the processing of an imperfect signal and allowed for toleration of measurement noise and other ambiguities.

In conclusion, fuzzy automata and the syntactic analysis approach can be used for fault diagnosis. The main concept is to segment the output of the monitored system and to classify each one of its segments into pattern classes according to a fuzzy membership value.

The string of templates a which corresponds to the properly operating system is represented by a fuzzy automaton M. The string of templates b which corresponds to the monitored output, becomes input to the fuzzy automaton M. Update of the membership value of the state ϕ_j of a, which is connected to state ϕ_i via the transition a_i takes place using the fuzzy inference rule of Eq. (15.9). If the automaton ends at a final state with membership degree that exceeds a certain threshold then normal operation can be deduced. Transition fuzziness provides the automaton with the flexibility to make multiple transitions simultaneously. The state fuzziness provides the automaton with the capability of being at multiple states at the same time. Syntactic analysis based on fuzzy automata is an easily interpretable method for model validation.

Chapter 16
Applications of Machine Vision to Industrial Systems

Abstract. Applications of vision-based industrial robotic systems are rapidly expanding due to the increase in computer processing power and low prices in machine vision hardware. Visual servoing over a network of synchronized cameras is an example where the significance of machine vision and distributed filtering for industrial robotic systems can be seen. A robotic manipulator is considered and a cameras network consisting of multiple vision nodes is assumed to provide the visual information to be used in the control loop. A derivative-free implementation of the Extended Information Filter is used to produce the aggregate state vector of the robot by processing local state estimates coming from the distributed vision nodes. The performance of the considered vision-based control scheme is evaluated through simulation experiments.

16.1 Machine Vision and Imaging Transformations

16.1.1 Some Basic Transformations

In Chapter 1 the kinematics of a multi-link robot were analyzed. In the following, several important transformations used in machine vision for robotic applications will be discussed. It will also be explained how a camera model can be derived and how one can treat the stereo imaging problem. The basic imaging transformations are [107]:

(i) *Translation*: The objective is to translate a point with coordinates (X,Y,Z) to a new location by using displacements (X_0,Y_0,Z_0). The translation is easily accomplished by using the following equation:

$$\begin{pmatrix} X^* \\ Y^* \\ Z^* \end{pmatrix} = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} + \begin{pmatrix} X_0 \\ Y_0 \\ Z_0 \end{pmatrix} \tag{16.1}$$

where (X^*, Y^*, Z^*) are the coordinates of the new point. Eq. (16.1) can be also written in the form

$$\begin{pmatrix} X^* \\ Y^* \\ Z^* \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & X_0 \\ 0 & 1 & 0 & Y_0 \\ 0 & 0 & 1 & Z_0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{16.2}$$

It is often useful to concatenate several transformations to produce a composite result, such as translation, followed by scaling, and then rotation. Eq. (16.2) can be written as

$$v^* = Tv \tag{16.3}$$

where T is a 4×4 transformation matrix, v is a column vector containing the original coordinates and v^* is a column vector whose components are the transformed coordinates:

$$v = \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}, \quad v^* = \begin{pmatrix} X^* \\ Y^* \\ Z^* \\ 1 \end{pmatrix} \tag{16.4}$$

while the matrix used for translation is given by

$$T = \begin{pmatrix} 1 & 0 & 0 & X_0 \\ 0 & 1 & 0 & Y_0 \\ 0 & 0 & 1 & Z_0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{16.5}$$

(ii) *Scaling*: Scaling by factors S_x, S_y and S_z along the X, Y and Z axes is given by the transformation matrix

$$S = \begin{pmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{16.6}$$

(iii) *Rotation*: Rotation of a point about the Z coordinate axis by an angle θ is achieved by using the transformation:

$$R_\theta = \begin{pmatrix} cos(\theta) & sin(\theta) & 0 & 0 \\ -sin(\theta) & cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{16.7}$$

The rotation angle θ is measured clockwise. Similarly, rotation of a point about the X axis by an angle α is performed by using the transformation

$$R_\alpha = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\alpha) & sin(\alpha) & 0 \\ 0 & -sin(\alpha) & cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{16.8}$$

while rotation of a point about the Y axis by an angle β is achieved by using the transformation

$$\begin{pmatrix} cos(\beta) & 0 & -sin(\beta) & 0 \\ 0 & 1 & 0 & 0 \\ sin(\beta) & 0 & cos(\beta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{16.9}$$

(iv) *Concatenation and inverse transformations*: The application of several transformations can be represented by a single 4×4 transformation matrix. For example, translation, scaling and rotation about the Z axis of a point v is given by

$$v^* = R_\theta[S(Tv)] = Av \tag{16.10}$$

where A is the 4×4 matrix $A = R_\theta ST$.

(v) *inverse translation*: The inverse translation matrix is given by

$$T^{-1} = \begin{pmatrix} 1 & 0 & 0 & -X_0 \\ 0 & 1 & 0 & -Y_0 \\ 0 & 0 & 1 & -Z_0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{16.11}$$

(vi) *Inverse rotation*: The inverse rotation matrix R_θ^{-1} is given by

$$R_\theta^{-1} = \begin{pmatrix} cos(-\theta) & sin(-\theta) & 0 & 0 \\ -sin(-\theta) & cos(-\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{16.12}$$

16.1.2 Perspective Transformation

A perspective transformation (also called imaging transformation) projects 3D points onto a plane. Perspective transformations play a central role in image processing because they provide an approximation to the manner in which an image is formed by viewing the three-dimensional world. These transformations are different from the ones described in the previous section because they are nonlinear in the sense that they involve division by coordinate values [107].

A model of the image formation process is shown in Fig. 16.2. The camera coordinate system (x, y, z) is defined as having the image plane coincident with the xy plane, and optical axis (established by the center of the lens) along the z axis. Thus,

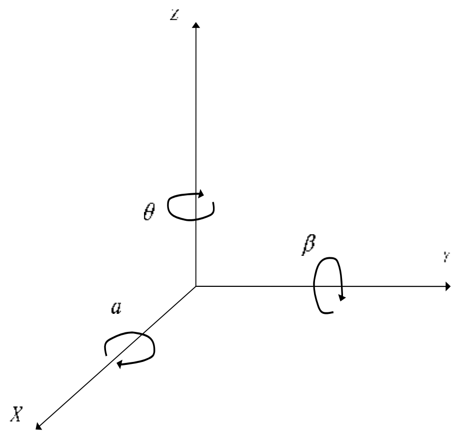


Fig. 16.1 Rotation of a point about each of the coordinate axes. Angles are measured clockwise when looking along the rotation axis towards the origin

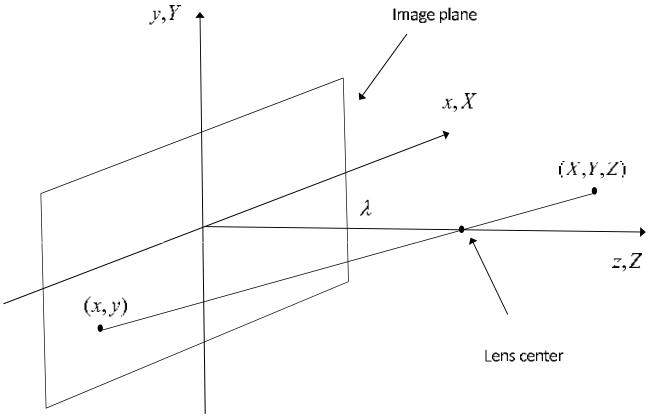


Fig. 16.2 Basic model of the imaging process. The camera coordinate system (x,y,z) is aligned with the world coordinate system (X,Y,Z)

the center of image plane is at the origin, and the center of the lens is at coordinates $(0,0,\lambda)$. If the camera is in focus for distant objects, λ is the focal length of the lens. It is assumed that the camera coordinate system (X,Y,Z). This restriction will be removed in the following section.

Let (X,Y,Z) be the world coordinates of any point in a 3D scene, as shown in Fig. 16.2. It will be assumed throughout the following discussion that $Z > \lambda$, that is all points of interest lie in the front of the lens. What one wishes to do first is obtain a relationship that gives the coordinates (x,y) of the projection of the point (X,Y,Z)

onto the image plane. This is easily accomplished by the use of similar triangles. With reference to Fig. 16.2, it follows that

$$\begin{aligned} \tfrac{x}{\lambda} &= -\tfrac{X}{Z-\lambda} = \tfrac{X}{\lambda-Z} \\ \tfrac{y}{\lambda} &= -\tfrac{Y}{Z-\lambda} = \tfrac{Y}{\lambda-Z} \end{aligned} \tag{16.13}$$

where the negative signs in front of X and Y indicate the image points are actually inverted, as it can be seen from the geometry of Fig. 16.2. The image-plane coordinates of the projected 3D point follow directly from

$$\begin{aligned} x &= \tfrac{\lambda X}{\lambda-Z} \\ y &= \tfrac{\lambda Y}{\lambda-Z} \end{aligned} \tag{16.14}$$

where the negative signs in front of X and Y indicate that image points are actually inverted, as can be seen from Fig. 16.2. The homogeneous coordinates of a point with cartesian coordinates (X,Y,Z) are defined as (kX,kY,kZ,k), where k is an arbitrary, nonzero constant. Conversion of homogeneous coordinates back to cartesian coordinates is accomplished by dividing the first three homogeneous coordinates by the fourth. A point in the cartesian coordinates system is expressed in vector form as

$$w = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \tag{16.15}$$

and its homogeneous equivalent is given by

$$w_h = \begin{pmatrix} kX \\ kY \\ kz \\ k \end{pmatrix} \tag{16.16}$$

If one defines the perspective transformation matrix

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{\lambda} & 1 \end{pmatrix} \tag{16.17}$$

Then, the product Pw_h yields a vector that is denoted by c_h and which provides the camera coordinates in homogeneous form:

$$c_h = Pw_h = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{\lambda} & 1 \end{pmatrix} \begin{pmatrix} kX \\ kY \\ kZ \\ k \end{pmatrix} = \begin{pmatrix} kX \\ kY \\ kZ \\ -\frac{kZ}{\lambda} + k \end{pmatrix} \tag{16.18}$$

These coordinates can be converted to cartesian form by dividing each of the first three components of c_h by the fourth. Thus, the cartesian coordinates of any point in the camera coordinate system are given in vector form by

$$c = \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \frac{\lambda X}{\lambda - Z} \\ \frac{\lambda Y}{\lambda - Z} \\ \frac{\lambda Z}{\lambda - Z} \end{pmatrix} \tag{16.19}$$

The first two components of c are the (x,y) coordinates in the image plane of a projected 3D point (X,Y,Z) as given in Eq. (16.14). In terms of the model of Fig. 16.2 the third variable is not of interest. This component acts as a free variable in the inverse perspective transformation. The inverse perspective transformation maps a point back into 3D. In terms of Eq. 16.18 the inverse perspective transformation is written as

$$w_h = P^{-1} c_h \tag{16.20}$$

where P^{-1} is easily found to be

$$P^{-1} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{\lambda} & 1 \end{pmatrix} \tag{16.21}$$

It is assumed that a given image point has coordinates $(x_0, y_0, 0)$ where the 0 in the z location simply indicates the fact that the image plane is located at $z = 0$. This point can be expressed in homogeneous vector form as

$$c_h = \begin{pmatrix} kx_0 \\ ky_0 \\ 0 \\ k \end{pmatrix} \tag{16.22}$$

The application of Eq. (16.20) provides the homogeneous coordinate vector

$$w_h = \begin{pmatrix} kx_0 \\ ky_0 \\ 0 \\ k \end{pmatrix} \tag{16.23}$$

or, in cartesian coordinates

$$w = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} x_0 \\ y_0 \\ 0 \end{pmatrix} \tag{16.24}$$

This is undesirable since it gives $Z = 0$ for any 3D point. The problem is caused by the fact that mapping a 3D scene onto the image plane is a many-to-one transformation. The image point (x_0, y_0) corresponds to the set of colinear 3D points which lie on the line that passes through $(x_0, y_0, 0)$ and $0, 0, \lambda$. The equations of this line in the world-coordinate system are obtained from Eq. (16.14), which gives

$$\begin{array}{c} X = \frac{x_0}{\lambda}(\lambda - Z) \\ Y = \frac{y_0}{\lambda}(\lambda - Z) \end{array} \tag{16.25}$$

According to the previous equations, unless one knows something about the 3D point which generated a given image point (for example, its Z coordinates), it is not possible to completely recover the 3D point from its 2D image. This observation can be used to formulate the inverse perspective transformation simply by using the z component of c_h as a free variable instead of 0. Thus, setting the homogeneous transform

$$c_h = \begin{pmatrix} kx_0 \\ ky_0 \\ kz \\ k \end{pmatrix} \tag{16.26}$$

one has about the inverse perspective transformation

$$w_h = \begin{pmatrix} kx_0 \\ ky_0 \\ kz \\ \frac{kz}{\lambda} \end{pmatrix} \tag{16.27}$$

which after transforming to cartesian coordinates is written as

$$w = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} \frac{\lambda x_0}{\lambda + z} \\ \frac{\lambda y_0}{\lambda + z} \\ \frac{\lambda z}{\lambda + z} \end{pmatrix} \tag{16.28}$$

Equivalently, treating z as a free variable results in the equations

$$X = \frac{\lambda x_0}{\lambda + z}, \; Y = \frac{\lambda y_0}{\lambda + z}, \; Z = \frac{\lambda z}{\lambda + z} \tag{16.29}$$

Finally, solving for z in terms of Z in the last equation and substituting in the first two expressions yields

$$\begin{array}{c} X = \frac{x_0}{\lambda}(\lambda - Z) \\ Y = \frac{y_0}{\lambda}(\lambda - Z) \end{array} \tag{16.30}$$

This shows that reconstructing a 3D point from its image by means of the inverse perspective transformation requires knowledge of at least of the world coordinates of the point.

16.1.3 Camera Model

The formation of an image via the projection of 3D points onto an image plane is described by Eq. (16.18) and Eq. (16.20). These two equations give the mathematical model of an imaging camera, which is based on the assumption that the camera and the world a coordinate system coincide. Here a more general problem is considered in which the two coordinate systems are allowed to separate. This is shown in Fig. 16.3, where the world coordinate system (X,Y,Z) is used to locate both the camera and the 3D points (denoted by w). The camera coordinate system is denoted by (x,y,z) and the image points are denoted by c. It is assumed that the camera is mounted on a gimbal which allows pan through an angle θ and tilt through an angle α. Pan is defined as the angle between the x and X axes, and tilt as the angle between the z and Z axes. The offset of the center of the gimbal from the origin of the world coordinate system is denoted by vector w_0 and the offset of the center of the imaging plan with respect to the gimbal center is denoted by a vector r with components (r_1,r_2,r_3).

The objective is to bring the camera and the world coordinate system into alignment by applying a set of transformations. Once this is done, one can apply the perspective transformation given in Eq. (16.17) to obtain the image-plane coordinates of any given world point [107]. Translation of the origin of the world coordinate

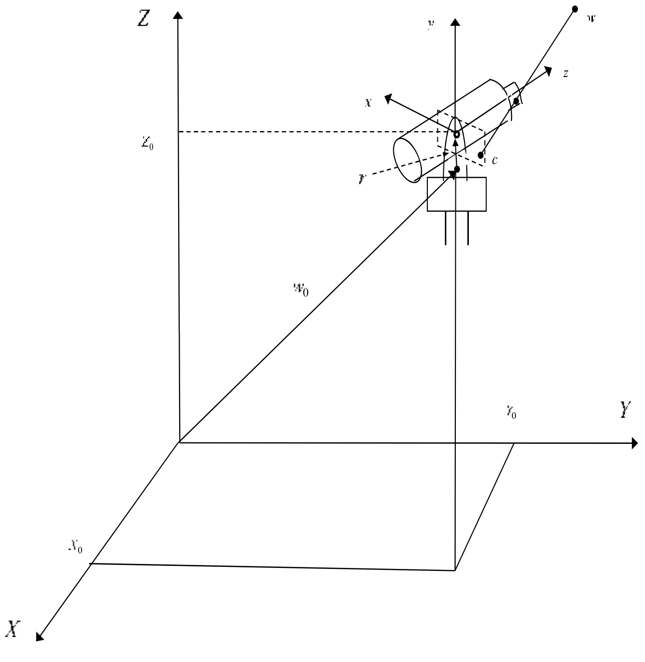


Fig. 16.3 Camera imaging and the two associated axes reference systems

system to the location of the gimbal center is accomplished by using the following transformation matrix

$$G = \begin{pmatrix} 1 & 0 & 0 & -X_0 \\ 0 & 1 & 0 & -Y_0 \\ 0 & 0 & 1 & -Z_0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{16.31}$$

This means that a homogeneous world point w_h that is at coordinates X_0, Y_0, Z_0 is at the origin of the new coordinate system after the transformation Gw_h. The pan angle is measured between the x and X axes. In normal position, these two axes are aligned. In order to pan the x axis through the desired angle, the angle is simply rotated by θ. The rotation is with respect to the z-axis and is accomplished by using the transformation matrix R_θ given in Eq. (16.6). Application of this matrix to all points, (including the point Gw_h) effectively rotates the x axis to the desired location. When using Eq. (16.6) it is important to use the notation that angles are considered positive when points are rotated clockwise, which implies a counterclockwise rotation of the camera about the z-axis. The unrotated position 0^o corresponds to the case when the x and X axes are aligned.

Moreover, since tilt is the angle between the z and Z axes, the camera is tilt an angle α by rotating the z axis by α. The rotation is with respect to the x axis and is accomplished by applying the transformation matrix R_α given in Eq. (16.8) to all points (including point $R_\theta Gw_h$). As above, a counterclockwise rotation of the camera implies positive angles, and the 0^o mark is where the z and Z axes are aligned.

The rotating matrices, which enable camera's alignment, can be concatenated into a single matrix $R = R_\alpha R_\theta$ which is given by

$$R = \begin{pmatrix} cos(\theta) & sin(\theta) & 0 & 0 \\ -sin(\theta)cos(\alpha) & cos(\theta)cos(\alpha) & sin(\alpha) & 0 \\ sin(\theta)sin(\alpha) & -cos(\theta)sin(\alpha) & cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{16.32}$$

Finally, displacement of the origin of the image plane by r is achieved by the transformation matrix

$$C = \begin{pmatrix} 1 & 0 & 0 & -r_1 \\ 0 & 1 & 0 & -r_2 \\ 0 & 0 & 1 & -r_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{16.33}$$

Consequently, the application of the series of transformations $CRGw_h$ to w_h brings the world and camera coordinate systems into coincidence. The image-plane coordinates of a point w_h are finally obtained by using Eq. (16.17). An homogeneous world point which is being viewed by a camera as derived in Fig. 16.3 has the following homogeneous representation in the camera coordinate system

$$c_h = PCRGw_h \tag{16.34}$$

The cartesian coordinates of the imaged points can be expressed with the reference to the camera coordinates

$$x = \lambda \frac{(X-X_0)cos(\theta)+(Y-Y_0)sin(\theta)-r_1}{-(X-X_0)sin(\theta)sin(\alpha)+(Y-Y_0)cos(\theta)sin(\alpha)-(Z-Z_0)cos(\alpha)+r_3+\lambda}$$
$$y = \lambda \frac{-(X-X_0)sin(\theta)cos(\alpha)+(Y-Y_0)cos(\theta)cos(\alpha)+(Z-Z_0)sin(\alpha)-r_2}{-(X-X_0)sin(\theta)sin(\alpha)+(Y-Y_0)cos(\theta)sin(\alpha)-(Z-Z_0)cos(\alpha)+r_3+\lambda} \qquad (16.35)$$

16.1.4 Camera Calibration

In the previous section explicit equations for the image coordinates (x,y) of a world point have been obtained. In Eq. (16.35) it has been shown that implementation of these equations requires knowledge of the focal length, camera offsets, and angles of pan and tilt. While these parameters could be measured directly, it is often more convenient (e.g. when the camera moves frequently) to determine one or more of the parameters by using the camera itself as a measuring device. This requires a set of image points with known world coordinates, and the computational procedure is used to obtain the camera parameters using these points, which is a procedure known as *camera calibration*.

Starting from Eq. (16.34) one can set $A = PCRG$. The elements of A contain all the camera parameters, and it holds that $c_h = Aw_h$. Thus the relation between c_h and w_h is given by

$$\begin{pmatrix} c_{h1} \\ c_{h2} \\ c_{h3} \\ c_{h4} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \qquad (16.36)$$

The camera coordinates in cartesian form are given by

$$\begin{pmatrix} x = \frac{c_{h1}}{c_{h4}} \\ y = \frac{c_{h2}}{c_{h4}} \end{pmatrix} \qquad (16.37)$$

Then, performing the substitutions $c_{h1} = xc_{h4}$ and $c_{h2} = xc_{h4}$ in Eq. (16.36) and expanding the matrix product yields

$$\begin{aligned} xc_{h4} &= \alpha_{11}X + \alpha_{12}Y + \alpha_{13}Z + \alpha_{14} \\ yc_{h4} &= \alpha_{21}X + \alpha_{22}Y + \alpha_{23}Z + \alpha_{24} \\ c_{h4} &= \alpha_{41}X + \alpha_{42}Y + \alpha_{43}Z + \alpha_{44} \end{aligned} \qquad (16.38)$$

where the expansion of c_{h3} has been ignored because it is related to z. Substitution of c_{h4} in the first two equations of Eq. (16.38) yields two equations with two twelve unknown coefficients:

$$\begin{aligned} \alpha_{11}X + \alpha_{12}Y + \alpha_{13}Z - \alpha_{41}xX - \alpha_{42}xY - \alpha_{43}xZ - \alpha_{44}x + \alpha_{14} = 0 \\ \alpha_{21}X + \alpha_{22}Y + \alpha_{23}Z - \alpha_{41}xX - \alpha_{42}xY - \alpha_{43}xZ - \alpha_{44}x + \alpha_{24} = 0 \end{aligned} \qquad (16.39)$$

The calibration procedure is analyzed in the following steps: (i) obtaining $m \geq 6$ world points with known coordinates $(X_i, Y_i, Z_i),\ i = 1, 2, \cdots, m$ (there are two equations involving the coordinates of these points, so at least six points are needed), (ii) imaging these points with the camera in a given position to obtain the corresponding image points $(x_i, y_i),\ i = 1, 2, \cdots, m$, using these results in Eq. (16.39) to solve for the unknown coefficients.

16.1.5 Stereo Imaging

Mapping a 3D scene onto an image plane is a many-to-one transformation, therefore an image point does not uniquely determine the location of a corresponding world point. However, the missing *depth* information can be obtained by using stereoscopic imaging techniques. Stereo imaging as shown in fig. 16.4 involves obtaining two separate image views of an object of interest (e.g. a world point w). The distance between the centers of the two lenses is called the *baseline* and is denoted by B, and the objective is to find the coordinates (X, Y, Z) of a point w given its image points (x_1, y_1) and (x_2, y_2). It is assumed that the cameras are identical and that the coordinate systems of both cameras are aligned, differing only in the location of their origins. After the camera and world coordinate system have been brought into coincidence, the xy plane of the image is aligned with the XY plane of the world coordinate system. Then, the Z coordinate of w is exactly the same for both camera coordinate systems [107].

It is assumed that the first camera is brought into coincidence with the world coordinate system, as shown in Fig. 16.5. Then, from Eq. (16.25) point w lies on the line with (partial) coordinates

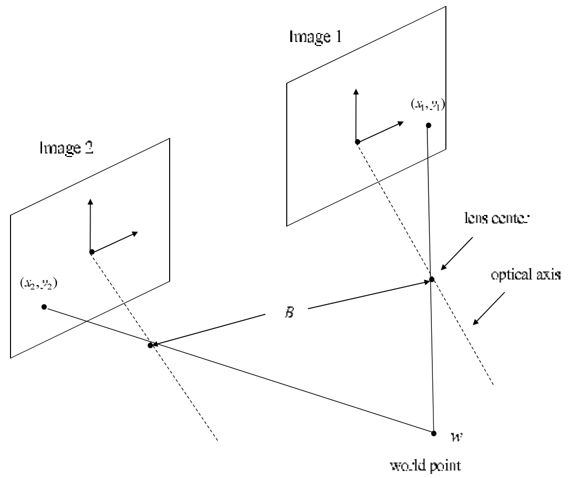


Fig. 16.4 Model of the stereo imaging process

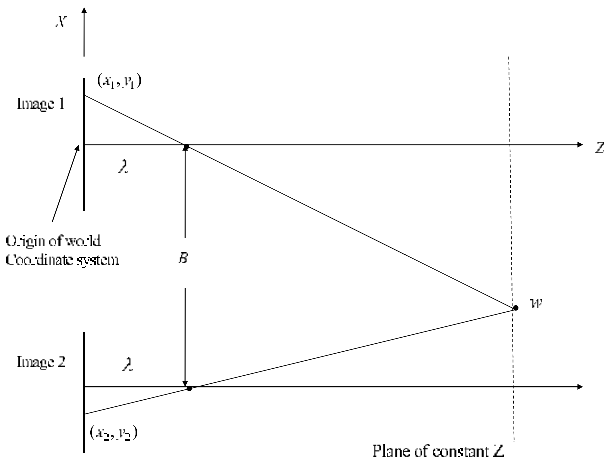


Fig. 16.5 Top of the model of the stereo imaging process with the first camera brought into coincidence with the world coordinate system

$$X_1 = \frac{x_1}{\lambda}(\lambda - Z_1) \tag{16.40}$$

where the subscripts on X and Z indicate that the first camera was moved to the origin of the world coordinate system, while the second camera keeps its relative arrangement shown in Fig. 16.4. If instead, the second camera had been brought to the origin of the world coordinate system, then one would have that point w lies on the line with coordinates

$$X_2 = \frac{x_2}{\lambda}(\lambda - Z_2) \tag{16.41}$$

However, due to the separation between cameras and the fact that the Z coordinate of w is the same for both camera coordinate systems, it follows that

$$\begin{aligned} X_2 &= X_1 + B \\ Z_2 &= Z_1 = Z \end{aligned} \tag{16.42}$$

Substituting Eq. (16.42) into Eq. (16.40) and Eq. (16.41) one obtains the following equations

$$\begin{aligned} X_1 + B &= \tfrac{x_2}{\lambda}(\lambda - Z) \\ X_1 &= \tfrac{x_1}{\lambda}(\lambda - Z) \end{aligned} \tag{16.43}$$

Using the above two equations and solving for Z one obtains

$$Z = \lambda - \frac{\lambda B}{x_2 - x_1} \tag{16.44}$$

which indicates that if the difference between the corresponding image coordinates x_2 and x_1 can be determined, and the baseline and focal length are known, then the calculation of the measured point's depth is also possible. Knowing Z one can also calculate coordinates X and Y from Eq. (16.40) and Eq. (16.41). The most difficult task in using Eq. (16.44) is to find two corresponding points in different images of the same scene. Since these points are in the same vicinity, a frequently used approach is to select a point within a small region in one of the image views and then attempt to find the best matching region in the other view by using correlation techniques. When the scene contains distinct features, such as prominent corners, a feature matching-matching approach is easier to provide a solution for establishing correspondence.

16.2 Multi Cameras-Based Visual Servoing for Industrial Robots

Going beyond the previous analysis on machine vision and the basic imaging transformations, nonlinear dynamical systems control will be examined and its implementation will be attempted based on the fusion of state estimates which are derived from local filters processing position measurements from distributed cameras. A possible approach for fusing the state estimates from distributed filters, under the assumption of Gaussian noises and nonlinear dynamics, is the Extended Information Filter (EIF). As analyzed in Chapter 8, the Extended Information Filter is a distributed Extended Kalman Filter, which performs fusion of state estimates provided by local Extended Kalman Filters. In the Extended Information Filter the local filters do not exchange raw measurements but send to an aggregation filter their local information matrices (local inverse covariance matrices which can be also associated to Fisher Information Matrices) and their associated local information state vectors (products of the local information matrices with the local state vectors) [218],[435].

Since the operation of the local Extended Kalman Filters is based on a linearization assumption around the local state estimates and on the truncation of the higher order terms that result from the associated Taylor series expansion, cumulative state estimation errors are introduced and the optimality property of the standard Kalman Filter is no longer maintained. These cumulative local state estimation errors are also transferred to the Extended Information Filter, and consequently when the aggregate state estimation is used in a control loop, the stability of the control loop may be affected [335],[336]. To overcome the aforementioned drawbacks, it is proposed to use a derivative-free Kalman Filter in place of the local Extended Kalman Filters. The nonlinear system is first subject to a linearization transformation and next state estimation is performed by applying the Kalman Filter to the linearized model. Unlike EKF, the proposed method provides state estimates without the need to calculate partial derivatives and Jacobians. Then, fusion of the state estimates provided by the local derivative-free Kalman Filters is possible through the stages of the standard Information Filter.

The aggregate state vector produced by the derivative-free Extended Information Filter can be used for sensoless control and robotic visual servoing. The problem of visual servoing over a network of synchronised cameras has been previously studied in [371]. In this chapter, visual servoing over a cameras network is considered for the nonlinear dynamic model of a planar single-link robotic manipulator. The position of the robot's end effector in the cartesian space (and equivalently the angle of the robotic link) is measured through m cameras (see Fig. 16.6). In turn m distributed derivative-free Kalman Filters are used to estimate the robotic link's state vectors. Next, the local state estimates are fused with the use of the standard Information Filter. Finally, the aggregate estimation of the state vector is used in a control loop which enables the robotic link to perform trajectory tracking.

16.3 Distributed Filtering for Sensorless Control

16.3.1 Visual Servoing over a Network of Synchronized Cameras

Visual servoing over a network of synchronized cameras is an example where the significance of the proposed distributed filtering approach can be seen. Applications of vision-based robotic systems are rapidly expanding due to the increase in computer processing power and low prices of cameras, image grabbers, CPUs and computer memory [80],[154],[250],[261],[465]. In order to satisfy strict accuracy constraints imposed by demanding manufacturing specifications visual servoing systems must be fault tolerant. This means that in the presence of temporary or permanent failures of the robotic system components, the system must continue to provide valid control outputs which will allow the robot to complete its assigned tasks. By parallelizing the execution of modules in the vision-based control loop, not only are the delays minimized such that the vision constraints can be satisfied, but also the system can be made fault tolerant.

Next, a position-based visual servoing system will be analyzed. This visual servoing system is characterized by the use of multiple fixed cameras to control a planar robot, as shown in Fig. 16.6 and Fig. 16.7. The presented approach relies on neither position nor velocity sensors, and directly sets the motor control current using only visual feedback. Direct visual servoing is implemented using a distributed filtering scheme which permits to fuse the estimates of the robot's state vector computed by local filters, each one associated to a camera in the cameras network. The cameras' network can be based on multiple cameras connected to a computer with a frame grabber to form a vision node. Each vision node has the ability to control the camera's synchronization and to capture and process images. Each vision node consists of the camera, the frame grabber and the filter which estimates motion characteristics of the monitored robot joint. The vision nodes are connected by a network to form a distributed vision system controlled by a master computer. The master computer is in turn connected to a planar 1-DOF robot joint and uses the vision feedback to perform direct visual servoing.

The master computer communicates video synchronization information over the network to each vision node. Typical sources of measurement noise include charge-coupled device (CCD) noise, analog-to-digital (A/D) noise and finite word-length

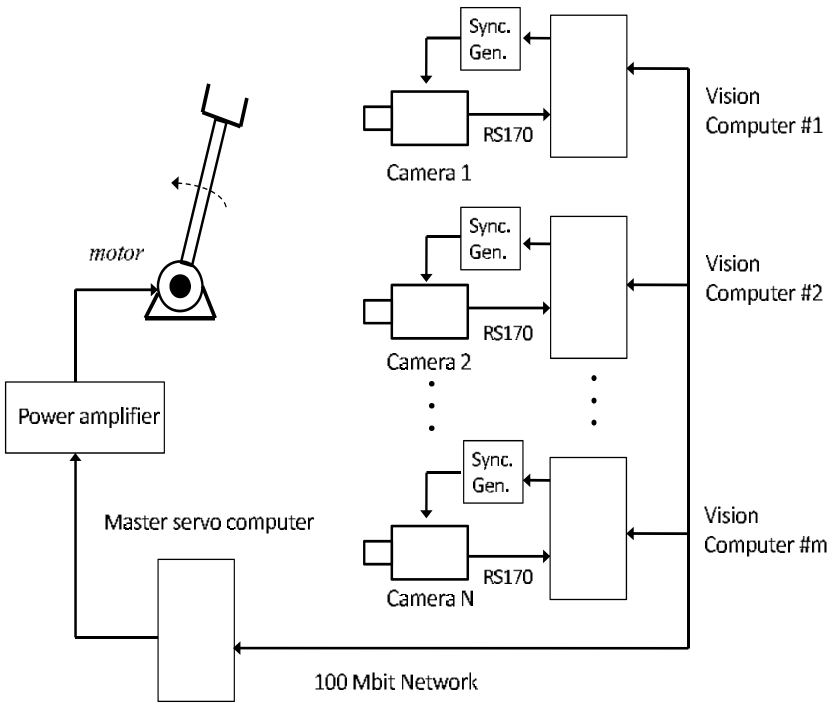


Fig. 16.6 Distributed cameras network and distributed information processing units for visual servoing

effects. Under ideal conditions, the effective noise variance from these sources should remain relatively constant. Occlusions can be also considered as a noise source.

16.3.2 Robot's State Estimation through Distributed Filtering

Fusion of the local state estimates which are provided by filters running on the vision nodes can improve the accuracy and robustness of the performed state estimation, thus also improving the performance of the robot's control loop. Under the assumption of Gaussian noise, a possible approach for fusing the state estimates from the distributed local filters is the Extended Information Filter (EIF). The Extended Information Filter provides an aggregate state estimate by weighting the state vectors produced by the local Extended Kalman Filters with the inverse of the associated estimation error covariance matrices.

Visual servoing over the previously described cameras network is considered for the nonlinear dynamic model of a single-link robotic manipulator (see Fig. 16.7). The robot can be programmed to execute a manufacturing task, such as disassembly or welding. The position of the robot's end effector in the cartesian space (and

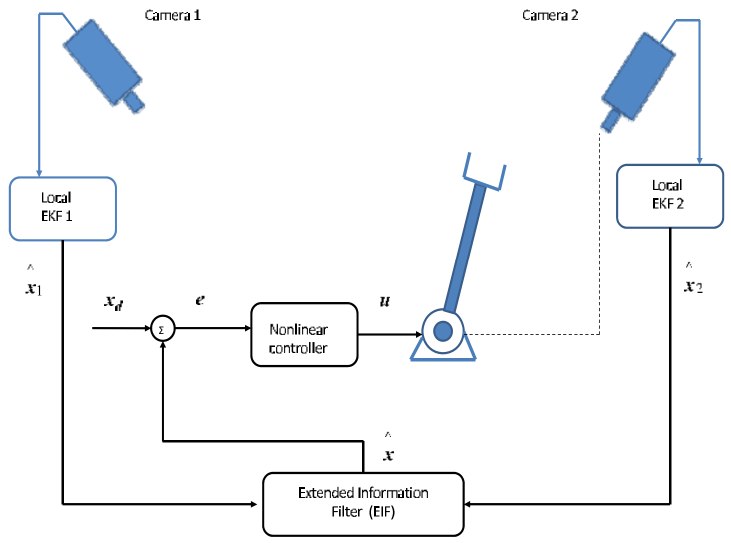


Fig. 16.7 Visual servoing based on fusion of distributed EKF state estimates

consequently the angle for the robotic link) is measured by the aforementioned m distributed cameras. The proposed multi-camera based robotic control loop can be useful in several vision-based industrial robotic applications where the vision is occluded or heavily disturbed by noise sources, e.g. arc welding and plasma cutting [428],[429]. In such applications there is need to fuse measurements from multiple cameras so as to obtain redundancy in the visual information and permit the robot to complete safely and within the specified accuracy constraints its assigned tasks [270]. The nonlinear control loop of the robotic manipulation which is based on the fusion of position measurements from distributed cameras with the use of the distributed filtering is depicted in Fig. 16.7.

16.4 Distributed State Estimation Using the EIF

16.4.1 Local State Estimation with Extended Kalman Filtering

As analyzed in Chapter 6, in the discrete-time case a dynamical system can be expressed in the form of a discrete-time state model:

$$\begin{aligned} x(k+1) &= \Phi(k)x(k)+L(k)u(k)+w(k) \\ z(k) &= Cx(k)+v(k) \end{aligned} \qquad (16.45)$$

where the state $x(k)$ is a m-vector, $w(k)$ is a m-element process noise vector and Φ is a $m\times m$ real matrix. Moreover the output measurement $z(k)$ is a p-vector, C is an $p{\times}m$-matrix of real numbers, and $v(k)$ is the measurement noise. It is assumed that the process noise $w(k)$ and the measurement noise $v(k)$ are uncorrelated. The process and measurement noise covariance matrices are denoted as $Q(k)$ and $R(k)$, respectively. Now the problem is to estimate the state $x(k)$ based on the measurements $z(1), z(2), \cdots, z(k)$. This can be done with the use of Kalman Filtering. As explained in Chapter 6, the discrete-time Kalman filter can be decomposed into two parts: i) time update (prediction stage), and ii) measurement update (correction stage).

measurement update:

$$\begin{aligned} K(k) &= P^-(k)C^T[C{\cdot}P^-(k)C^T+R]^{-1} \\ \hat{x}(k) &= \hat{x}^-(k)+K(k)[z(k)-C\hat{x}^-(k)] \\ P(k) &= P^-(k)-K(k)CP^-(k) \end{aligned} \tag{16.46}$$

time update:

$$\begin{aligned} P^-(k+1) &= \Phi(k)P(k)\Phi^T(k)+Q(k) \\ \hat{x}^-(k+1) &= \Phi(k)\hat{x}(k)+L(k)u(k) \end{aligned} \tag{16.47}$$

Again as explained in Chapter 6, in a more generic case, the following nonlinear state-space model can be considered:

$$\begin{aligned} x(k+1) &= \phi(x(k))+L(k)u(k)+w(k) \\ z(k) &= \gamma(x(k))+v(k) \end{aligned} \tag{16.48}$$

The operators $\phi(x)$ and $\gamma(x)$ are $\phi(x) = [\phi_1(x), \phi_2(x), \cdots, \phi_m(x)]^T$, and $\gamma(x) = [\gamma_1(x), \gamma_2(x), \cdots, \gamma_p(x)]^T$, respectively. It is assumed that ϕ and γ are sufficiently smooth in x so that each one has a valid series Taylor expansion. Following a linearization procedure, about the current state vector estimate $\hat{x}(k)$ the the linearized version of the system is obtained: $x(k+1) = \phi(\hat{x}(k)) + J_\phi(\hat{x}(k))[x(k)-\hat{x}(k)] + w(k)$, $z(k) = \gamma(\hat{x}^-(k)) + J_\gamma(\hat{x}^-(k))[x(k)-\hat{x}^-(k)] + v(k)$, where $J_\phi(\hat{x}(k))$ and $J_\gamma(\hat{x}(k))$ are the associated Jacobian matrices of ϕ and γ respectively. Now, the EKF recursion is as follows:

Measurement update. Acquire $z(k)$ and compute:

$$\begin{aligned} K(k) &= P^-(k)J_\gamma^T(\hat{x}^-(k)){\cdot}[J_\gamma(\hat{x}^-(k))P^-(k)J_\gamma^T(\hat{x}^-(k))+R(k)]^{-1} \\ \hat{x}(k) &= \hat{x}^-(k)+K(k)[z(k)-\gamma(\hat{x}^-(k))] \\ P(k) &= P^-(k)-K(k)J_\gamma(\hat{x}^-(k))P^-(k) \end{aligned} \tag{16.49}$$

Time update. Compute:

$$\begin{aligned} P^-(k+1) &= J_\phi(\hat{x}(k))P(k)J_\phi^T(\hat{x}(k))+Q(k) \\ \hat{x}^-(k+1) &= \phi(\hat{x}(k))+L(k)u(k) \end{aligned} \tag{16.50}$$

16.4.2 State Estimation through a Nonlinear Transformation

It will be shown that through a nonlinear transformation it is possible to design a state estimator (observer) for a class of nonlinear systems, including models of industrial robotic manipulators. This result will be generalized towards derivative-free Kalman Filtering for nonlinear systems. The following continuous-time nonlinear single-output system is examined [256]

$$\dot{x} = f(x) + q_0(x,u) + \sum_{i=1}^{p} \theta_i q_i(x,u), \;\; z = h(x), \; y \in R \tag{16.51}$$

with $q_i: R^n \times R^m \rightarrow R^n, 0 \leq i \leq p, f: R^n \rightarrow R^n, h: R^n \rightarrow R$, smooth functions, $h(x_0) = 0$, $q_0(x,0) = 0$ for every $x \in R^n$; x is the state vector, $u(x,t): R^+ \rightarrow R^m$ is the control which is assumed to be known, θ is the parameter vector which is supposed to be constant and z is the scalar output. A main assumption is that the systems of Eq.(16.51) are transformable by a parameter independent state-space change of coordinates in R^n , given by $\zeta = T(x), \;\; T(x_0) = 0$, into the system

$$\dot{\zeta} = A_c \zeta + \psi_0(z,u) + \sum_{i=1}^{p} \theta_i \psi_i(z,u), \;\; z = C_c \zeta \tag{16.52}$$

$$A_c = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{pmatrix} \;\; C_c^T = \begin{pmatrix} 1 \\ 0 \\ 0 \\ \cdots \\ 0 \end{pmatrix} \tag{16.53}$$

and $\psi_i: \;\; R \times R^m \rightarrow R^n$ smooth functions for $i = 0, \cdots, p$. The necessary and sufficient conditions for the initial nonlinear system to be transformable into the form of Eq.(16.52) have been given in [255],[256]. Then for every parameter vector θ, the system

$$\begin{array}{l} \dot{\hat{\zeta}} = A_c \hat{\zeta} + \psi_0(z,u) + \sum_{i=1}^{p} \theta_i \psi_i(z,u) + K(z - C_c \hat{\zeta}) \\ \hat{x} = T^{-1}(\hat{\zeta}) \end{array} \tag{16.54}$$

is an asymptotic observer for a suitable choice of K provided that the state $x(t)$ is bounded, with estimation error dynamics

$$\dot{e} = (A_c - KC_c)e = \begin{pmatrix} -k_1 & 1 & 0 & \cdots & 0 \\ -k_2 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ -k_{n-1} & 0 & 0 & \cdots & 1 \\ -k_n & 0 & 0 & \cdots & 1 \end{pmatrix} e \tag{16.55}$$

The eigenvalues of $A_c - KC_c$ can be arbitrarily placed by choosing the vector K, since they coincide with the roots of the polynomial $s^n + k_1 s^{n-1} + \cdots + k_n$.

16.4.3 Derivative-Free Kalman Filtering for Nonlinear Systems

Since Eq. (16.54) provides an asympotic observer for the initial nonlinear system of Eq. (16.51) one can consider a case in which the observation error gain matrix K can be provided by the Kalman Filter equations given initially in the continuous-time KF formulation, or in their discrete-time equivalent of Eq. (16.46) and Eq. (16.47) [42]. The following single-input single-output nonlinear dynamical system is considered

$$x^{(n)} = f(x,t) + g(x,t)u(x,t) \tag{16.56}$$

where $y = x$ is the system's output, and $f(x,t)$, $g(x,t)$ are nonlinear functions. It can be noticed that the system of Eq. (16.56) belongs to the general class of systems of Eq. (16.51). Assuming the transformation $\zeta_i = x^{(i-1)}$, $i = 1,\cdots,n$, and $x^{(n)} = f(x,t) + g(x,t)u(x,t) = v(\zeta,t)$, i.e. $\dot{\zeta}_n = v(\zeta,t)$, one obtains the linearized system of the form: $\dot{\zeta}_1 = \zeta_2$, $\dot{\zeta}_2 = \zeta_3$, $\cdots\ \cdots$, $\dot{\zeta}_{n-1} = \zeta_n$, $\dot{\zeta}_n = v(\zeta,t)$, which in turn can be written in state-space equations as

$$\begin{pmatrix} \dot{\zeta}_1 \\ \dot{\zeta}_2 \\ \cdots \\ \dot{\zeta}_{n-1} \\ \dot{\zeta}_n \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & 0 \end{pmatrix} \begin{pmatrix} \zeta_1 \\ \zeta_2 \\ \cdots \\ \zeta_{n-1} \\ \zeta_n \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \cdots \\ 0 \\ 1 \end{pmatrix} v(\zeta,t) \tag{16.57}$$

$$z = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \end{pmatrix} \zeta \tag{16.58}$$

The system of Eq. (16.57) and Eq. (16.58) has been written in the form of Eq. (16.52), which means that Eq. (16.54) is the associated asymptotic observer. Therefore, the observation gain K appearing in Eq. (16.54) can be found using either linear observer design methods (in that case the elements of the observation error gain matrix K have fixed values), or the recursive calculation of the Kalman Filter gain described Eq. (16.46) and Eq. (16.47) [337]. The latter approach will be followed in this chapter.

16.4.4 Fusing Estimations from Local Distributed Filters

Again the discrete-time nonlinear system of Eq. (16.48) is considered. The Extended Information Filter (EIF) performs fusion of the local state vector estimates which are provided by the local Extended Kalman Filters, (running at vision nodes shown in Fig. 16.6) using the *Information matrix* and the *Information state vector* [218]. As noted in Chapter 8, the Information Matrix is the inverse of the state vector covariance matrix, and can be also associated to the Fisher Information matrix [344]. The Information state vector is the product between the Information matrix and the local state vector estimate

$$\begin{aligned} Y(k) &= P^{-1}(k) = I(k) \\ \hat{y}(k) &= P^{-}(k)^{-1}\hat{x}(k) = Y(k)\hat{x}(k) \end{aligned} \tag{16.59}$$

Following the analysis of Chapter 8, the update equation for the Information Matrix and the Information state vector are given by $Y(k)=P^-(k)^{-1}+J_\gamma^T(k)R^{-1}(k)J_\gamma(k)=Y^-(k)+I(k)$ and $\hat{y}(k)=\hat{y}^-(k)+J_\gamma^T R(k)^{-1}[z(k)-\gamma(x(k))+J_\gamma\hat{x}^-(k)]=\hat{y}^-(k)+i(k)$, where $I(k)=J_\gamma^T(k)R^(k)^{-1}J_\gamma(k)$ is the associated information matrix and, $i(k)=J_\gamma^T R^(k)^{-1}[(z(k)-\gamma(x(k)))+J_\gamma\hat{x}^-(k)]$ is the information state contribution. The predicted information state vector and Information matrix are obtained from $\hat{y}^-(k)=P^-(k)^{-1}\hat{x}^-(k)$, $Y^-(k)=P^-(k)^{-1}=[J_\phi(k)P^-(k)J_\phi(k)^T+Q(k)]^{-1}$. It is assumed that an observation vector $z^i(k)$ is available for the N different sensor sites (vision nodes) $i=1,2,\cdots,N$ and each vision node observes the robot according to the local observation model, expressed by $z^i(k)=\gamma(x(k))+v^i(k),\ i=1,2,\cdots,N$, where the local noise vector $v^i(k)\sim N(0,R^i)$ is assumed to be white Gaussian and uncorrelated between sensors. The variance of a composite observation noise vector v_k is expressed in terms of the block diagonal matrix $R(k)=diag[R^1(k),\cdots,R^N(k)]^T$. The information contribution can be expressed by a linear combination of each local information state contribution i^i and the associated information matrix I^i at the i-th sensor site $i(k)=\sum_{i=1}^N J_\gamma^{i^T}(k)R^i(k)^{-1}[z^i(k)-\gamma^i(x(k))+J_\gamma^i(k)\hat{x}^-(k)]$, $I(k)=\sum_{i=1}^N J_\gamma^{i^T}(k)R^i(k)^{-1}J_\gamma^i(k)$. Thus, the update equations for fusing the local state estimates become $\hat{y}(k)=\hat{y}^-(k)+\sum_{i=1}^N J_\gamma^{i^T}(k)R^i(k)^{-1}[z^i(k)-\gamma^i(x(k))+J_\gamma^i(k)\hat{x}^-(k)]$, and $Y(k)=Y^-(k)+\sum_{i=1}^N J_\gamma^{i^T}(k)R^i(k)^{-1}J_\gamma^i(k)$. It is noted that in the Extended Information Filter an aggregation (master) fusion filter produces a global estimate by using the local sensor information provided by each local filter. As in the case of the Extended Kalman Filter the local filters which constitute the Extended Information Filter can be written in terms of *time update* and a *measurement update* equation.

Measurement update: Acquire $z(k)$ and compute

$$\begin{gathered} Y(k)=P^-(k)^{-1}+J_\gamma^T(k)R(k)^{-1}J_\gamma(k),\ \text{or}\ Y(k)=Y^-(k)+I(k) \\ \text{where}\ I(k)=J_\gamma^T(k)R^{-1}(k)J_\gamma(k),\text{and} \\ \hat{y}(k)=\hat{y}^-(k)+J_\gamma^T(k)R(k)^{-1}[z(k)-\gamma(\hat{x}(k))+J_\gamma\hat{x}^-(k)] \\ \text{or}\ \hat{y}(k)=\hat{y}^-(k)+i(k) \end{gathered} \tag{16.60}$$

Time update: Compute

$$\begin{gathered} Y^-(k+1)=P^-(k+1)^{-1}=[J_\phi(k)P(k)J_\phi(k)^T+Q(k)]^{-1} \\ \text{and}\ y^-(k+1)=P^-(k+1)^{-1}\hat{x}^-(k+1) \end{gathered} \tag{16.61}$$

If the derivative-free Kalman Filter is used in place of the Extended Kalman Filter then in the EIF equations the following matrix substitutions should be performed: $J_\phi(k)\rightarrow A_c$, $J_\gamma(k)\rightarrow C_c$, where matrices A_c and C_c have been defined in Eq. (16.53). Moreover, the covariance matrices $P(k)$ and $P^-(k)$ are the ones obtained from the linear Kalman Filter update equations given in Eq. (16.46) and Eq. (16.47).

16.4.5 Calculation of the Aggregate State Estimation

The outputs of the local filters are treated as measurements which are fed into the aggregation fusion filter [218]. Then each local filter is expressed by its respective error covariance and estimate in terms of information contributions and is described by $P_i^{-1}(k) = P_i^{-}(k)^{-1} + J_\gamma^T R^(k)^{-1} J_\gamma(k)\hat{x}_i(k) = P_i(k)(P_i^{-}(k)^{-1}\hat{x}_i^{-}(k)) + J_\gamma^T R(k)^{-1}[z^i(k) - \gamma^i(x(k)) + J_\gamma^i(k)\hat{x}_i^{-}(k)]$. As shown in Chapter 8, the global estimate and the associated error covariance for the aggregate fusion filter can be rewritten in terms of the computed estimates and covariances from the local filters using the relations

$$\begin{gathered} J_\gamma^T(k)R(k)^{-1}J_\gamma(k) = P_i(k)^{-1} - P_i^{-}(k)^{-1} \\ J_\gamma^T(k)R(k)^{-1}[z^i(k) - \gamma^i(x(k)) + J_\gamma^i(k)\hat{x}^{-}(k)] = P_i(k)^{-1}\hat{x}_i(k) - P_i(k)^{-1}\hat{x}_i(k-1). \end{gathered} \tag{16.62}$$

For the general case of N local filters $i = 1, \cdots, N$, the distributed filtering architecture is described by the following equations

$$\begin{aligned} &P(k)^{-1} = P^{-}(k)^{-1} + \sum_{i=1}^{N}[P_i(k)^{-1} - P_i^{-}(k)^{-1}] \\ &\hat{x}(k) = P(k)[P^{-}(k)^{-1}\hat{x}^{-}(k) + \sum_{i=1}^{N}(P_i(k)^{-1}\hat{x}_i(k) - P_i^{-}(k)^{-1}\hat{x}_i^{-}(k))] \end{aligned} \tag{16.63}$$

The global state update equation in the above distributed filter can be written in terms of the information state vector and of the information matrix, i.e. $\hat{y}(k) = \hat{y}^{-}(k) + \sum_{i=1}^{N}(\hat{y}_i(k) - \hat{y}_i^{-}(k))$, and $\hat{Y}(k) = \hat{Y}^{-}(k) + \sum_{i=1}^{N}(\hat{Y}_i(k) - \hat{Y}_i^{-}(k))$. From Eq. (16.63) it can be seen that if a local filter (processing station) fails, then the local covariance matrices and the local state estimates provided by the rest of the filters will enable an accurate computation of the target's state vector.

16.5 Simulation Tests of the Vision-Based Control System

16.5.1 Dynamics and Control of the Robotic Manipulator

The 1-DOF robotic model to be controlled with the use of visual servoing over a distributed cameras network consists of a rigid link which is rotated by a DC motor, as shown in Fig. 16.7. The model of the DC motor was analyzed in subsection 4.3.1 and is described by the set of equations: $L\dot{I} = -k_e\omega - RI + V$, $J\dot{\omega} = k_e I - k_d\omega - \Gamma_d$, with the following notations L : armature inductance, I : armature current, k_e : motor electrical constant, R : armature resistance, V : input voltage, taken as control input, J : motor inertia, ω : rotor rotation speed, k_d : mechanical dumping constant, Γ_d : disturbance or external load torque. It is assumed that $\Gamma_d = mgl{\cdot}sin(\theta)$, i.e. that the DC motor rotates a rigid robotic link of length l with a mass m attached to its end. Then, denoting the state vector as $[x_1, x_2, x_3]^T = [\theta, \dot{\theta}, \ddot{\theta}]^T$, a nonlinear model of the DC motor is obtained $\dot{x} = f(x,t) + g(x,t)u$, where $f(x,t) = [f_1(x,t), f_2(x,t), f_3(x,t)]^T$ is a vector field function with elements: $f_1(x,t) = x_2$, $f_2(x,t) = x_3$, $f_3(x,t) = -\frac{k_e^2 + k_d R}{JL}x_2 - \frac{RJ + K_d L}{JL}x_3 -$, $-\frac{Rmgl}{JL}sin(x_1) - \frac{mgl}{J}cos(x_1)x_2$. Similarly $g(x,t) = [g_1(x,t), g_2(x,t), g_3(x,t)]^T$ is a vector field function with elements: $g_1(x,t)=0$, $g_2(x,t)=0$, $g_3(x,t)=\frac{k_e}{JL}$. Having chosen the joint's angle to be the system's output, the state space equation of the 1-DOF robot manipulator can be rewritten as

$$x^{(3)}=\bar{f}(x)+\bar{g}(x)u \tag{16.64}$$

where functions $\bar{f}(x)$ and $\bar{g}(x)$ are given by

$$\begin{gathered}\bar{f}(x)=-\frac{k_e^2+k_dR}{JL}x_2-\frac{RJ+K_dL}{JL}x_3-\frac{Rmgl}{JL}sin(x_1)-\frac{mgl}{J}cos(x_1)x_2\\ \text{and } \bar{g}(x)=\frac{k_e}{JL}.\end{gathered} \tag{16.65}$$

This is a system in the form of Eq. (16.56), therefore a state estimator can be designed according to the results on derivative-free Kalman Filtering.

The controller has to make the system's output (angle θ of the motor) follow a given reference signal x_d. For measurable state vector x and uncertain functions $f(x,t)$ and $g(x,t)$ an appropriate control law for the 1-DOF robotic model is $u=\frac{1}{g(x,t)}[x_d^{(n)}-f(x,t)-K^Te+u_c]$, with $e=x-x_d$, $e^T=[e,\dot{e},\ddot{e},\cdots,e^{(n-1)}]^T$, $K^T=[k_n,k_{n-1},\cdots,k_1]$, such that the polynomial $e^{(n)}+k_1e^{(n-1)}+k_2e^{(n-2)}+\cdots+k_ne$ is Hurwitz. The previously defined control law results into $e^{(n)}=-K^Te+u_c+\tilde{d}$, where the supervisory control term u_c aims at the compensation of modeling errors as well as of the additive disturbance $\tilde{d}$. Suitable selection of the feedback gain K assures that the error dynamics will result into $\lim_{t\to\infty}e(t)=0$. In case of state estimation-based (sensorless control), and denoting, $\hat{x}$ as the estimated state vector and $\hat{e}=\hat{x}-x_d$ as the estimated tracking error one has

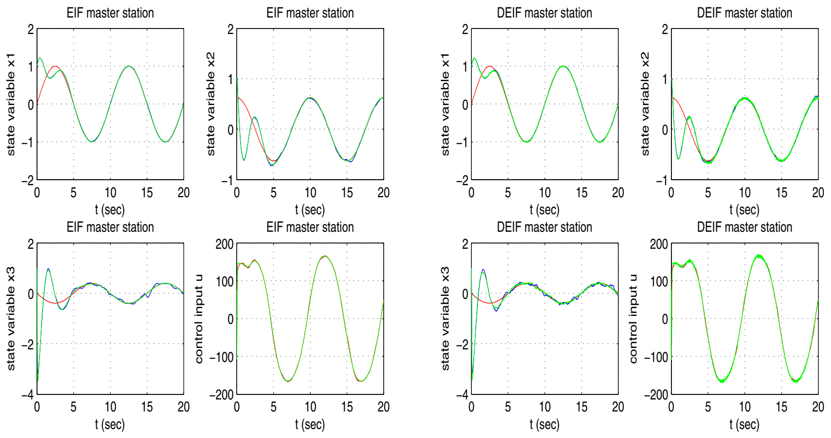
$$u=\frac{1}{g(\hat{x},t)}[x_d^{(n)}-f(\hat{x},t)-K^T\hat{e}+u_c] \tag{16.66}$$

As explained in the previous sections the estimated state vector $\hat{x}$ is obtained by fusing local state estimates from the vision nodes with the use of the Extended Information Filter, or with the use of the derivative-free Extended Information Filter.

16.5.2 Evaluation of Results on Vision-Based Control

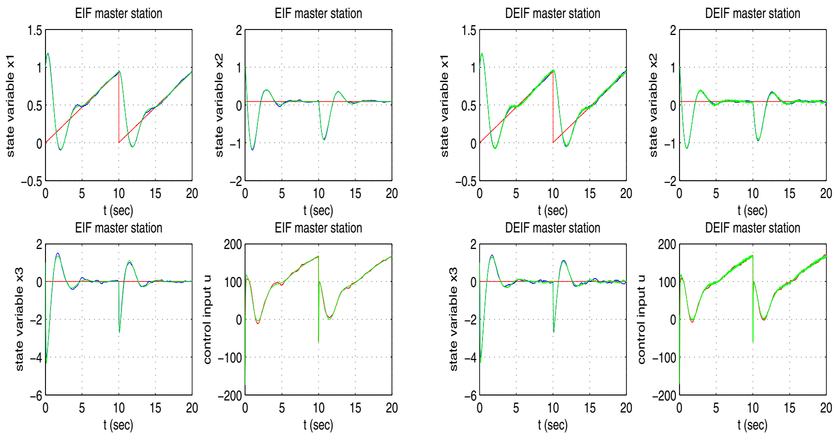
The simulation experiments show that for the considered class of nonlinear systems, and particularly for the multi-cameras visual servoing model, it is possible to apply the standard Information Filter for fusing distributed state estimates from derivative-free local Kalman Filters.

This approach enables to avoid use of the Extended Information Filter which is based on local Extended Kalman Filters, thus also avoiding to introduce in the estimation procedure linearization errors due to the truncation of higher order Taylor expansion terms. Consequently, this means that one can have a more robust state estimation, as in the case of the Unscented Information Filter. The performance of the control loop which is based on Extended Information Filtering and on derivative-free Extended Information Filtering is shown in Fig. 16.8 and Fig. 16.9.



(a) **(b)**

Fig. 16.8 Tracking of a sinusoidal setpoint using fusion of position measurements from distributed cameras with the use of (a) the Extended Information Filter (b) the derivative-free Extended Information Filter



(a) **(b)**

Fig. 16.9 Tracking of a seesaw setpoint using fusion of position measurements from distributed cameras with the use of (a) the Extended Information Filter (b) the derivative-free Extended Information Filter

References

1. Addison, P.S.: The illustrared wavelet transform handbook. Institute of Physics Publishing (2002)
2. Adibi, M.M., Kafka, R.J., Maram, S., Mili, L.M.: On power system controlled separation. IEEE Transactions on Power Systems 21(4), 1894–1902 (2006)
3. Ahrens, J.H., Khalil, H.K.: Closed-Loop Behavior of a Class of Nonlinear Systems Under EKF-Based Control. IEEE Transactions on Automatic Control 52, 536–540 (2007)
4. Al-Baiyat, S.A.: Power system transient stability enhancement by STATCOM with nonlinear H_∞ stabilizer. Electric Power Systems Research 73, 45–52 (2005)
5. Amari, S., Murata, N., Müller, K.-R., Finke, M., Yang, H.H.: Asymptotic Statistical Theory of Overtraining and Cross-Validation. IEEE Transactions on Neural Networks 8(5), 985–996 (1997)
6. Anderson, G., Donalek, P., Farmer, R., Hatziargyriou, N., Kamwa, I., Kundur, P., Martins, N., Paserba, J., Pourbeik, P., Sanchez-Gasca, J., Schulz, R., Stankovic, A., Taylor, C., Vittal, V.: Causes of the 2003 Major Grid Blackouts in North America and Europe, and Recommended Means to Improve System Dynamic Performance. IEEE Transactions on Power Systems 20(4), 1922–1928 (2005)
7. Antsaklis, P., Kohn, W., Nerode, A., Sastry, S.: HS 1994. LNCS, vol. 999. Springer, Heidelberg (1995)
8. Aoustin, Y., Fliess, M., Mounier, H., Rouchon, P., Rudoplh, J.: Theory and practice in the motion planning control of a flexible robot arm using Mikusiǹski operators, In: Proc. of 4th Symposium on Robotics and Control, pp. 287-293, Nantes (1997)
9. Archer, B.A., Annakkage, U.D.: Monitoring and predicting power system behavior by tracking dominant modes of oscillation. In: Proc. of the IEEE Power Engineering Society General Meeting, vol. 2, pp. 1475–1482 (2005)
10. Arteaga, M.A.: Tracking control of flexible robot arms with a nonlinear observer. Automatica 36, 1329–1337 (2000)
11. Arteaga, M.A., Siciliano, B.: On Tracking Control of Flexible Robot Arms. IEEE Transactions on Automatic Control 45(3), 520–527 (2000)
12. Arulampalam, S., Maskell, S.R., Gordon, N.J., Clapp, T.: A tutorial on particle filters for on-line nonlinear/non-Gaussian Bayesian tracking. IEEE Transactions on Signal Processing 50, 174–188 (2002)
13. Bahbah, A.G., Girgis, A.A.: New Method for Generators' Angles and Angular Velocities Prediction for Transient Stability Assessment of Multimachine Power Systems Using Recurrent Artificial Neural Network. IEEE Transactions Power Systems 19, 1015–1022 (2004)

14. Baptistella, L.B.F., Ollero, A.: Fuzzy methodologies for interactive multicriteria optimization. IEEE Transactions on Systems, Man and Cybernetics 10, 355–365 (1980)
15. Bascetta, L., Rocco, P.: End-point vibration sensing of planar flexible manipulators through visual servoing. Mechatronics 16, 221–232 (2006)
16. Basseville, M., Nikiforov, I.: Detection of abrupt changes: Theory and Applications. Prentice-Hall, Englewood Cliffs (1993)
17. Basseville, M.: Information Criteria for Residual Generation and Fault Detection and Isolation. Automatica 33(5), 783–803 (1997)
18. Basseville, M.: On-board component fault detection and isolation using the statistical local approach. Automatica 34(11), 1391–1415 (1998)
19. Basseville, M., Benveniste, A., Zhang, Q.: Surveilliance d' installations industrielles: démarche générale et conception de l' algorithmique, IRISA Publication Interne No 1010 (1996)
20. Basseville, M., Abdelghani, M., Benveniste, A.: Sub-space-based fault detection and isolation methods - application to vibration monitoring, Publication Interne IRISA no 1143 (1997)
21. Basseville, M., Benveniste, A., Mathis, G., Zhang, Q.: Monitoring the combustion set of a gas turbine. In: Proc. Safeprocess 1994, Helsinki, Finland (1994)
22. Basseville, M., Benveniste, A., Gach-Devauchelle, B., Goursat, M., Bonnecase, D., Dorey, P., Prevosto, M., Olagnon, M.: Damage monitoring in vibration mechanics: issues in diagnostics and predictive maintenance. Mechanical Systems and Signal Processing 7(5), 401–423 (1993)
23. Beard, R.W., McLain, T.W., Goodrich, M., Anderson, E.P.: Coordinated target assignment and intercept for unmanned air vehicles. IEEE Transactions on Robotics and Automation 18, 911–922 (2002)
24. Becedas, J., Ramon Trapero, J., Felia, V., Sira-Ramirez, H.: Adaptive controller for single-link flexible manipulators based on algebraic identification and generalized proportional-integral control. IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics 39(3), 735–751 (2009)
25. Becker, J., Meurer, T.: Feedforward tracking control for non-uniform Timoshenko beam models: Combining differential flatness, modal analysis, and FEM. ZAMM Zeitschrift fur Angewandte Mathematik und Mechanik 87(1), 37–58 (2005)
26. Benosman, A., Le Vey, G.: Control of flexible manipulators: a survey. Robotics 22(5), 533–545 (2004)
27. Benson, H.P., Morin, T.L.: The vector maximization problem:proper efficiency and stability. SIAM Journal of Applied Mathematics 32, 64–72 (1977)
28. Benveniste, A., Basseville, M., Moustakides, G.: The asymptotic local approach to change detection and model validation. IEEE Transactions on Automatic Control 32(7), 583–592 (1987)
29. Benvensite, A., Metivier, P., Priouret, P.: Adaptive algorithms and stochastic approximations. Applications of Mathematics Series, vol. 22. Springer, Heidelberg (1990)
30. Benveniste, A., Fabre, E., Haar, S., Jard, C.: Diagnosis of asynchronous discrete event systems: a net unfolding approach. IEEE Transactions on Automatic Control 48(5), 714–727 (2003)
31. Bernard, C.P., Slotine, J.J.: Adaptive Control with Multiresolution Bases. In: IEEE CDC 1997, 36th IEEE Intl. Conference on Decision and Control, San Diego, California, USA (December 1997)
32. Bernsten, P.I.B., Aamo, O.M., Leira, B.J.: Ensuring mooring line integrity by dynamic positioning: Controller design and implementation tests. Automatica 45, 1285–1290 (2009)

33. Bertsekas, D.P.: Incremental least squares methods and the extended Kalman Filter. SIAM J. Optimization 6(3), 807–822 (1996)
34. Bertsekas, D.P.: Nonlinear Programming. Athena Scientific (1995)
35. Bhattacharyya, S.P., Chapelat, H., Keel, L.H.: Robust control: The Parametric approach. Prentice-Hall, Englewood Cliffs (1997)
36. Bhowal, P., Sarkar, D., Mukhopadhyay, S., Basu, A.: Fault diagnosis in discrete time hybrid systems - a case study. Information Sciences 177(5), 1290–1308 (2007)
37. Bishop, B.E.: On the use of redundant manipulator techniques for control of platoons of cooperative robotic vehicles. IEEE Transactions on systems, Man and Cybernetics - Part A 33(5), 608–615 (2003)
38. Biswas, S., Sarkar, D., Bhawal, P., Mukhopadhyay, S.: Diagnosis of delay-deadline failures in real-time discrete event models. ISA Transactions 46(4), 569–582 (2007)
39. Blauer, M., Belanger, P.R.: State and parameter estimation for robotic manipulators using force measurements. IEEE Transactions on Automatic Control 32, 1055–1066 (1987)
40. Bodson, M., Chiasson, J., Novotnak, R.: High-Performance Induction Motor Control via Input-Output Linearization. IEEE Control Systems Magazine, 24–33 (1994)
41. Bolić, M., Djurić, P.M., Hong, H.: Resampling Algorithms and Architectures for Distributed Particle Filters. IEEE Transactions on Signal Processing 53, 2242–2450 (2005)
42. Boutayeb, M., Rafaralahy, H., Darouach, M.: Convergence analysis of the Extended Kalman Filter used as an observer for nonlinear deterministic discrete-time systems. IEEE Transactions on Automatic Control 42(4), 581–586 (1997)
43. Brandwajn, V., Kumar, A.B.R., Ipakchi, A., Bose, A., Kuo, S.D.: Severity indices for contingency screening in dynamic security assessment. IEEE Transactions on Power Systems 12, 1136–1142 (1997)
44. Brault, P., Mounier, H., Petit, N., Rouchon, P.: Flatness based tracking control of a manoeuvrable vehicle: the π-Car. In: Proc. MTNS 2000, Mathematical Theory of Networks and Systems, Perpignan, France (2000)
45. Brémaud, P.: Markov-chains, Gibbs fields, Monte-Carlo simulation and queues. Texts in Applied Mathematics. Springer, Heidelberg (1999)
46. Bruno, S., de Benedictis, M., Delfanti, M., LaScala, M.: Preventing blackouts through reactive rescheduling under dynamical and protection system constraints. In: Proc: IEEE PowerTech 2005, St. Petersburg, Russia, pp. 1–8 (2005)
47. Caballero, F., Merino, L., Ferruz, J., Ollero, A.: A particle filtering method for Wireless Sensor Network localization with an aerial robot beacon. In: Proc: IEEE International Conference on Robotics and Automation 2006, pp. 2860–2865 (2008)
48. Calderaro, V., Galdi, V., Piccolo, A., Siano, P.: Adaptive relays for overhead line protection. Electrical Power Systems Research 77, 1552–1559 (2007)
49. Campillo, F., Cérou, F., LeGland, F., Rakotozafy, R.: Particle and cell approximations for nonlinear filtering, Rapport de Recherche INRIA, No. 2567 (June 1995)
50. Campillo, F.: Particulaire & Modèles de Markov Cachés, Master Course Notes “Filtrage et traitement des doneées”. Université de Sud-Toulon Var, France (2006)
51. Caron, F., Davy, M., Duflos, E., Vanheeghe, P.: Particle Filtering for Multi-Sensor Data Fusion with Switching Observation Models: Applications to Land Vehicle Positioning. IEEE Transactions on Signal Processing 55(6), 2703–2719 (2007)
52. Carreras, B.A., Lynch, V.E., Dobson, I.: Dynamical and probabilistic approaches to the study of blackout vulnerability of the power transmission grid. In: Proc. of the 37th Annual Hawaii International Conference on System Sciences, pp. 55–61 (January 2004)
53. Cassilo, C.L., Moreno, W., Valavanis, K.P.: Unmanned helicopter waypoint trajectory tracking using model predictive control. In: 2007 Mediterranean Conference on Control and Automation, Athens, Greece (July 2007)

54. Caswara, F.M., Unbenhauen, H.: A neuro-fuzzy approach to the control of a flexible-link manipulator. IEEE Transactions on Robotics and Automation 18(6), 932–944 (2002)
55. Cetinkunt, S., Yu, W.L.: Closed Loop Behavior of a Feedback Controlled Flexible Arm: A Comparative Study. International Journal of Robotics Research 10(3), 263–275 (1991)
56. Chak, C.K., Feng, G., Mia, J.: An adaptive fuzzy neural network for MIMO system model approximation in high dimensional spaces. IEEE Trans. on Syst. Man and Cybern. - Part B: Cybernetics 28(3), 436–446 (1998)
57. Chankong, V., Haimes, Y.Y.: Multi-objective Decision Making:Theory and Methodology. North-Holland, New York (1983)
58. Cannon, M., Slotine, J.J.E.: Space-frequency localized basis function networks for non-linear system estimation and control. Neurocomputing 9, 293–342 (1995)
59. Chen, M.-Y., Linkens, D.A.: A systematic neuro-fuzzy modeling framework with application to material property prediction. IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics 31(5), 781–790 (2001)
60. Chen, S., Malik, O.: H_∞ optimization-based power system stabilizer design. IET Proceedings on Generation, Transmission and Distribution 193, 253–257 (1996)
61. Chen, B.-S., Lee, C.-H., Chang, Y.-C.: H_∞ Tracking Design of Uncertain Nonlinear SISO Systems: Adaptive Fuzzy Approach. IEEE Transactions on Fuzzy Systems 4(1), 32–43 (1996)
62. Chen, Q., Zhu, K., McCalley, J.D.: Dynamic decision-event trees for rapid response to unfolding events in bulk transmission systems. In: Proc. IEEE 2001 Power Technology Conf. (September 2001)
63. Cheong, J., Lee, S.: Linear PID composite controller and its tuning for flexible-link robots. Journal of Vibration and Control 14(3), 291–318 (2008)
64. Cheong, J., Chung, W.K., Youn, Y.: Two-step controller for 3D flexible-link manipulators: Banbdwidth modulation and modal feedback approach. ASME Journal of Dynamic Systems, Measurement and Control 124, 566–574 (2002)
65. Clerk, M., Kennedy, J.: The Particle Swarm-Explosion, Stability, and Convergence in a Multidimensional Complex Space. IEEE Transactions on Evolutionary Computation 6(1), 58–73 (2002)
66. Colandairaj, J., Scanlon, W., Irwin, G.: Understanding wireless networked control systems through simulation. Journal of Computing and Control Engineering 16(2), 26–31 (2005)
67. Cole, M.O.T., Keogh, P.S., Burrows, C.R., Sahinkaya, M.M.: Adaptive control of robot vibration using compact wavelets. ASME Journal of Vibration and Acoustics 128, 653–665 (2006)
68. Comets, F., Meyre, T.: Calcul stochastique et modèles de diffusions. Dunod (2006)
69. Contant, O., Lafortune, S., Teneketzis, D.: Diagnosability of discrete event systems with modular structure. Discrete Event Dynamical Systems 16(1), 9–37 (2006)
70. Cordier, M.O., Dague, P., Lévy, F., Montmain, L., Staroswiecki, M., Travé-Massuyès, L.: Conflicts versus analytical redundancy relations: A comparative analysis of the model-based diagnosis approach from the artificial intelligence and the automatic control perspectives. IEEE Transactions on Systems-Man and Cybernetics-Part B: Cybernetics 34(5), 2163–2177 (2006)
71. Corrault, G., Cordier, M.O., Quiniou, R., Wang, F.: Temporal abstraction and inductive logic programming for arythmia recognition from ECG. Artificial Intelligence in Medicine 28(3), 231–263 (2003)

72. Costa, P.J., Dente, J.A.: An experiment in automatic modelling of an electrical drive system using fuzzy logic. IEEE Transactions on Systems, Man and Cybernetics - Part C: Applications and Reviews 28(2), 254–262 (1998)
73. Coué, C., Pradalier, C., Laugier, C., Fraichard, T., Bessiére, P.: Bayesian Occupancy Filtering for Multitarget Tracking: An Automotive Application. The International Journal of Robotics Research 25(1), 19–30 (2006)
74. Coué, C., Pradalier, C., Laugier, C.: Bayesian Programming Multi-Target Tracking: an Automotive Application. In: Int. Conf. on Field and Service Robotics, Lake Yamanaka, Japan (July 2003)
75. Crisan, D., Doucet, A.: A Survey of Convergence Results on Particle Filtering Methods for Practitioners. IEEE Transactions on Signal Processing 50(3), 736–746 (2002)
76. Dearden, R., Clancy, D.: Particle filters for real-time fault detection in planetary rovers. In: Proc. 13th Intl. Workshop on Principles Diagnosis, Semmering, Austria (May 2002)
77. Deming, R.W., Perlovsky, L.I.: Concurrent multi-target localization, data association, and navigation for a swarm of flying sensors. Information Fusion 8(3), 316–330 (2007)
78. De Luca, A., Siciliano, B.: Regualation of flexible arms under gravity. IEEE Transactions on Robotics and Automation 9, 463–467 (1993)
79. DeMarco, C.L.: A phase transition model for cascading network failure. IEEE Control Systems Magazine, 40–51 (2001)
80. DeSouza, G.N., Kak, A.C.: A subsumptive, hierarchical and distributed vision-based architecture for smart robotics. IEEE Transactions on Systems, Man and Cybernetics - Part B 34(5), 1988–2002 (2004)
81. Ding, S.X.: Model-based fault diagnosis techniques: Design schemes, algorithms and tools. Springer, Heidelberg (2008)
82. Doucet, A., De Freitas, N., Gordon, N. (eds.): Sequential Monte Carlo Methods in Practice. Springer, Heidelberg (2001)
83. Doyle, J.C., Glover, K., Khargonekar, P.P., Francis, B.A.: State-Space Solutions to Standard H_2 and H_∞ Control Problems. IEEE Transactions on Automatic Control 34(8), 831–847 (1989)
84. Dubois, D., Foulloy, L., Mauris, G., Prade, H.: Probability-possibility transformations, triangular fuzzy sets and probabilistic inequalities. Reliable Computing 10(4), 273–297 (2004)
85. Dubois, D., Prade, H., Ughetto, L.: Checking the coherence and redundancy of fuzzy knowledge bases. IEEE Transactions on Fuzzy Systems 5(3), 398–417 (1997)
86. Duflo, M.: Algorithmes stochastiques, Mathématiques et Applications, vol. 23. Springer, Heidelberg (1996)
87. Edwards, A.R., Chan, K.W., Dunn, R.W., Daniels, A.R.: Transient stability screening using artificial neural networks within a dynamic security assessment system. IET Proceedings on Power Generation, Transmission and Distribution 143, 129–134 (1996)
88. Elizondo, D.C., de La Ree, J., Phadke, A.G., Horowitz, S.: Hidden failures in protection systems and their impact on wide-area disturbances. In: Proc. of the IEEE 2001 PES Winter Meeting, vol. 2, pp. 710–714 (2001)
89. Elshafei, A.L., Metwally, K.A., Shaltout, A.A.: A variable-structure adaptive fuzzy-logic stabilizer for single and multi-machine power systems. Control Engineering Practice 13, 413–423 (2005)
90. El-Sharkawi, M.A., El-Samahy, A.A., Ek-Saayed, M.L.: High performance drive of DC brushless motors using neural networks. IEEE Transactions on Energy Conversion 9, 317–322 (1994)
91. Elston, J., Frew, E.: Net-centric cooperative tracking of moving targets. In: American Institute of Aeronautics and Astronautics (AIAA) 2007 Conference and Exhibit, California, USA (May 2007)

92. Etxebarria, V., Sanz, A., Lizarraga, I.: Control of a lightweight flexible robotic arm using sliding modes. Intl. Journal of Advanced Robotic Systems 2(2), 103–110 (2005)
93. Fabre, E., Benveniste, A.: Partial order techniques for distributed discrete event systems: why you cannot avoid using them. Discrete Event Dynamical Systems 17(3), 355–403 (2007)
94. Fang, Y., Zergeroglu, E., de Queiroz, M.S., Dawson, D.M.: Global output feedback control of dynamically positioned surface vessels: an adaptive control approach. Mechatronics 14, 341–356 (2004)
95. Fichou, J., LeGland, F., Mevel, L.: Particle-Based Methods for Parameter Estimation and Tracking: Numerical Experiments, Publication Interne IRISA, No 1604, Rennes, France (2004)
96. Filaretov, V.F., Vukobratovic, M.K., Zhirabok, A.N.: Observer-based fault diagnosis in manipulation robots. Mechatronics 9, 929–939 (1999)
97. Fliess, M., Mounier, H., Rouchon, P., Rudolph, J.: Systèmes lineaires sur les operateurs de Mikusiǹski et commande d' une poutre flexible, In: ESAIM Proc. Élasticité, viscoélasticité et contrôle optimal, huitièmes entretiens du Centre Jacques Cartier, Lyon (1996)
98. Fliess, M., Mounier, H.: Tracking control and π-freeness of infinite dimensional linear systems. In: Picci, G., Gilliam, D.S. (eds.) Dynamical Systems, Control, Coding and Computer Vision, vol. 258, pp. 41–68. Birkhaüser, Basel (1999)
99. Fliess, M., Mounier, H., Rouchon, P., Rudolph, J.: A distributed parameter approach to the control of a tubular reactor: a multi-variable case. In: Proc. of 37th Conference on Decision and Control, Tampa, FL, pp. 439–442 (1998)
100. Fonseca, C.M., Fleming, P.J.: An overview of evolutionary algorithms in multiobjective optimization. Evolutionary Computation 3, 1–16 (1995)
101. Fossen, T.I., Strand, J.P.: Passive nonlinear observer design for ships using Lyapunov methods: full scale experiments with a supply vessel. Automatica 35, 3–16 (1999)
102. Fossen, T.I., Perez, T.: Kalman Filtering heading control of ships and offshore rigs. IEEE Control Systems Magazine, 32–46 (2009)
103. Fossen, T.I., Grovlen, A.: Nonlinear output feedback control of dynamically positioned ships using vectorial observer backstepping. IEEE Trans. on Control Systems Technology 6(1), 121–128 (1998)
104. Fossen, T.I., Strand, J.P.: A tutorial on nonlinear backstepping application to ship control. Modeling, Identification and Control 20(2), 83–135 (1999)
105. Fossen, T.I., Strand, J.P.: Nonlinear Ship Control. In: IFAC Conference on Control Applications in Marine Systems (CAMS 1998), Fukuoka, Japan (October 1998)
106. Fromont, E., Cordier, M.O., Quiniou, R.: Extraction de connnaissances provenant de données multisources pour la characterisation d' arythmies cardiaques. In: Proceedings of EGC 2004, Workshop: Fouille de données complexes dans un processus d' extraction de connaissances, Clermont-Ferrand, France (2004)
107. Fu, K.S., Gonzalez, R.C., Lee, G.S.G.: Robotics: control, sensing, vision and intelligence. McGraw-Hill, New York (1987)
108. Fukuda, T., Kiguchi, K.: Intelligent Position/Force Control for Industrial Robot Manipulators: Application of Fuzzy-Neural Network. IEEE Transactions on Industrial Electronics 44(6) (1997)
109. Fukumizu, K.: A regularity condition of the information matrix of a multilayer perceptron network. Neural Networks 9(5), 871–879 (1996)
110. Fukumizu, K.: Statistical analysis of unidentiable models and its application to multilayer neural networks. In: Post-conference of Bernoulli-Riken BSI 2000 on Neural Networks and Learning (2000)

111. Fukumizu, K.: Statistical Active Learning in Multilayer Perceptrons. IEEE Transactions on Neural Networks 11(1), 17–26 (2000)
112. Gaing, Z.L.: Wavelet-based neural network for Power Disturbance Recognition and Classification. IEEE Transactions on Power Delivery 19(4), 1560–1568 (2004)
113. Gazi, V., Passino, K.: Stability analysis of social foraging swarms. IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics 34(1), 539–557 (2004)
114. Galdi, V., Piccolo, A., Siano, P.: Designing an adaptive fuzzy controller for maximum wind energy extraction. IEEE Transactions on Energy Conversion 28, 559–569 (2008)
115. Gan, Q., Harris, C.J.: Comparison of two measurement fusion methods for Kalman-filter-based multisensor data fusion. IEEE Transactions on Aerospace and Electronic Systems 37(1), 273–280 (2001)
116. Gbollinger, J., Duffie, N.A.: Computer Control of Machines and Processes. Addison-Wesley Publishing Company, Reading (1989)
117. Gao, S., Zhong, Y., Zhang, X., Shirinzadeh, B.: Multi-sensor optimal data fusion for INS/GPS/SAR integrated navigation system. Aerospace Science and Technology 13, 232–237 (2009)
118. Ge, S.S., Lee, T.H., Harris, C.J.: Adaptive Neural Network Control of Robotic Manipulators. World Scientific, Singapore (1998)
119. Ge, S.S., Lee, T.H., Zhu, G.: Energy-based robust controller design for multi-link flexible robots. Mechatronics 6(7) (1996)
120. Ge, S.S., Zhang, J.: Neural-Network Control of Nonaffine Nonlinear System With Zero Dynamics By State and Output Feedback. IEEE Transactionson Neural Networks 14(4), 900–918 (2003)
121. Ge, S.S., Hang, C.C., Zhang, T.: Adaptive Neural Network Control of Nonlinear Systems by State and Output Feedback. IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics 29(6), 818–828 (1999)
122. Ge, S.S., Wang, C.: Adaptive Neural Control of Uncertain MIMO Nonlinear Systems. IEEE Transactions on Neural Networks 15(3), 674–692 (2004)
123. Ghasemi, H., Canizares, C., Moshref, A.: Oscillatory stability limit prediction using stochastic subspace identification. IEEE Transactions on Power Systems 21(2), 736–745 (2006)
124. Ge, Q.-B., Wen, G.-L.: Relative ship positioning based on information fusion in the Marine Intelligent Transportation System (MITS). In: Proc. of the 7th Intl. Conference on Machine Learning and Cybernetics, Kunming, China (July 2008)
125. Glorennec, P.Y.: Algorithmes d' apprentissage pour systμemes d' inférence floue, Hermes Science Publications (1999)
126. Godhavn, J.M., Fossen, T.I., Berge, S.P.: Nonlinear and adaptive backstepping designs for tracking control of ships. Journal of Adaptive Control and Signal Processing, Special Issue on Marine Systems Control 12(8), 649–670 (1998)
127. Gomez-Bravo, F., Cuesta, F., Ollero, A.: Parallel and diagonal parking in nonholonomic autonomous vehicles. Engineering Applications of Artificial Intelligence 14, 419–434 (2001)
128. Gordon, N.J., Salmond, D.J., Smith, A.F.M.: A novel approach to nonlinear/non-Gaussian Bayesian state estimation. IEE Proceedings on Radar and Signal Processing 140, 107–113 (1993)
129. Grastein, A., Cordier, M.O., Largouët, C.: Automata slicing for diagnosing discrete-event systems with partially ordered observations. In: Proceedings of AI*IA, 9th Congress of the Italian Association for Artificial Intelligence, pp. 282–285 (2005)
130. Groß, R., Bonani, M., Mondada, F., Dorigo, M.: Autonomous self-assembly in swarm-bots. IEEE Transactions on Robotics 22(6), 1115–1130 (2006)

131. Guo, Y., Parker, L.E.: A distributed and optimal motion planning approach for multiple mobile robots. In: Proc. 2002 IEEE Intl. Conference on Robotics and Automation, Washington DC, pp. 2612–2619 (May 2002)
132. Guoqiang, H., Renmu, H., Huachun, Y., Peng, W., Rui, M.: Iterative Prony method based power system low frequency oscillation mode analysis and PSS design. In: Proc. of the Transmission and Distribution Conference and Exhibition, vol. 1, pp. 1–6 (2005)
133. Gustafsson, F., Gunnarsson, F., Bergman, N., Forssel, U.: Particle Filters for Positioning, Navigation and Tracking. IEEE Transactions on Signal Processing - Special Issue on Monte Carlo Methods for Statistical Signal Processing 50, 425–437 (2002)
134. Guterrez, L.B., Lewis, F.L., Lowe, J.A.: Implementation of neural network tracking for a single flexible link: comparison with PD and PID controllers. IEEE Transactions on Industrial Electronics 45(2), 307–318 (1998)
135. Hajela, P., Lin, C.Y.: Genetic search strategies in multi-riterion optimal design. Structural Optimization 4, 99–107 (1992)
136. Hardiansyah, S., Furuya, J., Irisawa, J.: A robust H_∞ power system stabilizer design using reduced-order models. Electrical Power and Energy Systems 28, 21–28 (2006)
137. Harris, C.J., Hang, X., Gon, X.: Adaptive modelling, estimation and fusion from data: a neurofuzzy approach. Springer, Heidelberg (2002)
138. Haykin, S.: Neural Networks: A Comprehensive Foundation. Macmillan, Basingstoke (1994)
139. Haykin, S.: Adaptive Filter Theory. Prentice-Hall, Englewood Cliffs (1991)
140. Hermosillo, J., Sekhavat, S.: Feedback Control of a Bi-steerable Car Using Flatness Application to Trajectory Tracking. In: Proc 2003, IEEE 2003 American Control Conference, Denver Colorado, pp. 3567–3572 (2003)
141. Hisseine, D., Lohmann, B.: Robust control for a flexible-link manipulator using sliding-mode techniques and nonlinear H_∞ control design methods. In: Proc. of the 2001 IEEE Intl. Conference on Robotics and Automation, Seoul, Korea (2001)
142. Hiyama, T., Suzuki, N., Kita, T., Funakoshi, T.: Development of real time stability monitoring system for power system operation. Proc. of the IEEE Power Engineering Society Winter Meeting 1, 525–530 (1999)
143. Hiyama, T., Suzuki, N., Funakoshi, T.: On-line coherency monitoring of power system oscillations. Proc. of the IEEE Power Engineering Society Summer Meeting 3, 839–1844 (2000)
144. Hojo, M., Ohnishi, T., Mitani, Y., Saeki, O., Ukai, H.: Observation of frequency oscillation in western Japan 60 Hz power system based on multiple synchronized phasor measurements. In: Proc. of the IEEE Power Tech. Conf., vol. 2, pp. 1–6 (2003)
145. Hong, Y., Gao, L., Cheng, D., Hu, J.: Luapunov-based approach to multi-agent systems with switching jointly connected interconnection. IEEE Transactions on Automatic Control 52(5), 943–948 (2007)
146. Horng, J.H.: Neural adaptive tracking control of a DC motor. Information Sciences 118, 1–13 (1999)
147. Hristache, M., Juditsky, A., Polzelh, J., Spokoiny, V.: Structure adaptive approach for dimension reduction. Annals of Statistics 29(6), 1537–1566 (2001)
148. Hua, Y., Yutian, L., Xinsheng, H.: Low frequency oscillation analysis and damping based on Prony method and sparse eigenvalue technique. In: Proc. of the IEEE International Conference on Networking, Sensing and Control, vol. 1, pp. 1006–1010 (2006)
149. Huang, Y.J., Wang, Y.J.: Robust PID tuning strategy for uncertain plants based on the Kharitonov theorem. ISA Transactions 39, 419–431 (2000)
150. Huang, S.N., Tan, K.K.: Fault detection, isolation and accomodation control in robotic systems. IEEE Transactions on Automation Science and Engineering 5(3), 480–489 (2008)

151. Hue, C., Le Cadre, J.P., Pérez, P.: Tracking Multiple Objects with Particle Filtering. IEEE Transactions on Aerospace and Electronic Systems 38(3), 791–812 (2002)
152. Hui, D., Fuchun, S., Zengqi, S.: Observer-based adaptive controller design of flexible manipulators using time-delay neuro-fuzzy networks. Journal of Intelligent and Robotic Systems 34(4), 453–466 (2002)
153. Hunt, K.J., Sbarbaro, D., Zbikowski, R., Gawthrop, P.J.: Neural networks for control systems-A survey. Automatica 28(6), 1083–1112 (1992)
154. Hwang, C.L., Shih, C.Y.: A distributed active-vision network-space approach for the navigation og a car-like wheeled robot. IEEE Transactions on Industrial Electronics 56(3), 846–855 (2002)
155. Ibbini, M.S., Zakaria, W.S.: Nonlinear control of DC machines. Electric Machines and Power Systems 24, 21–35 (1996)
156. IEEE PES CAMS Task Force on Undestanding, Prediction, Mitigation and Restoration of Cascading Failures: 'Initial review of methods for cascading failure analysis in electric power transmission systems. In: Proc. IEEE Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century, Pittsburg, USA, pp. 1–8 (2008)
157. Ing, J., Coates, M.G.: Parallel particle filters for tracking in wireless sensor networks. In: IEEE Workshop on Signal Processing Advances in Wireless Communications, SPAWC 2005, art. no. 1506277, pp. 935–939 (2005)
158. Ippolito, L., Siano, P.: Identification of Tagaki-Sugeno-Kang fuzzy model for power transformers predictive overload system. IET Proceedings- Generation, Transmission and Distribution 151(5), 582–589 (2004)
159. Isermann, R.: Local basis function networks for identification of a turbocharger. In: Proc. IEE Conference, vol. 427(1), pp. 7–12 (1996)
160. Isermann, R., Schreiber, A.: Identification methods for experimental modelling of nonlinear combustion processes. In: Collection of Technical Papers - 4th International Energy Conversion Engineering Conference, vol. 2, pp. 1468–1485 (2006)
161. Isermann, R.: Fault diagnosis systems: an introduction from fault detection to fault tolerance. Springer, Heidelberg (2006)
162. Ishibuchi, H., Murata, T.: A multi-objective genetic local search algorithm and its application to flowshop scheduling. IEEE Transactions on Systems, Man and Cybernetics-Part C 28, 392–403 (1998)
163. Jafari, M.: Introduction to Petri Nets in flexible and agile automation. In: Zhou, M. (ed.) Petri Nets in Flexible and Agile Automation. Kluwer Academic Publishers, Dordrecht (1995)
164. Jassemi-Zargani, R., Necsulescu, D.: Extended Kalman Filter-based sensor fusion for operational space control of a robot arm. IEEE Transactions on Instrumentation and Measurement 51, 1279–1282 (2002)
165. Jang, J.-S.R., Sun, C.-T., Mizutani, E.: Neurofuzzy and Soft-Computing, A computational approach to learning and machine intelligence. Prentice-Hall, Englewood Cliffs (1997)
166. Jang, J.O.: A deadzone compensator of a DC motor system using fuzzy logic control. IEEE Transactions on Systems Man and Cybernetics - Part C: Applications and Reviews 31(1), 42–48 (2001)
167. Jatmitko, W., Sekiyama, K., Fukuda, T.: A PSO-based mobile robot for odour source localization in dynamic advection-diffusion with obstacles environment. IEEE Computational Intelligence Magazine, 37–42 (2007)
168. Jetto, L., Longhi, S., Venturini, G.: Development and Experimental Validation of an Adaptive Extended Kalman Filter for the Localization of Mobile Robots. IEEE Transactions on Robotics and Automation 15(2) (1999)

169. Jiang, G., Chen, H., Urugueanu, C., Yoshihira, K.: Multiresolution abnormal trace detection using varied-length n-Grams and automata. IEEE Transactions on Systems, Man and Cybernetics - Part C: Applications and Reviews 37(1), 86–97 (2007)
170. Julier, S., Uhlmann, J., Durrant-Whyte, H.F.: A new method for the nonlinear transformations of means and covariances in filters and estimators. IEEE Transactions on Automatic Control 45(3), 477–482 (2000)
171. Julier, S.J., Uhlmann, J.K.: Unscented Filtering and Nonlinear Estimation. Proceedings of the IEEE 92, 401–422 (2004)
172. Kadirkamanathan, V., Jaward, M.H., Fabri, S.G., Kadirkamanathan, M.: Particle Filters for Recursive Model Selection in Linear and Nonlinear System Identification. In: Proc. 39th IEEE Conference on Decision and Control, Sydney, Australia, vol. 3, pp. 2391–2396 (2000)
173. Kakimoto, N., Sugumi, M., Makino, T., Tomiyama, K.: Monitoring of inter-area oscillation mode by synchronized phasor measurement. IEEE Transactions on Power Systems 21, 260–268 (2006)
174. Kalman, R.E., Falb, P.L., Arbib, M.A.: Topics in Mathematical System Theory. McGraw-Hill, New York (1969)
175. Kamen, E.W., Su, J.K.: Introduction to Optimal Estimation. Springer, Heidelberg (1999)
176. Kanoh, H., Tzafestas, S.G., Lee, H.G., Kalat, J.: Modelling and control of flexible robot arms. In: Proc 25th Conference on Decision and Control, Athens, Greece, pp. 1866–1870 (1986)
177. Kandepu, R., Foss, B., Imsland, L.: Applying the unscented Kalman filter for nonlinear state estimation. Journal of Process Control 18, 753–768 (2008)
178. Karimi, H.R., Lohmann, B.: A Computational Method to Robust Vibration Control of Vehicle Engine-Body System using Haar Wavelets. In: Proceedings of the 2006 IEEE International Conference on Control Applications, Munich, Germany, pp. 169–174 (October 2006)
179. Karlsson, R., Norrlöf, M.: Bayesian Position Estimation of an Industrial Robot Using Multiple Sensors. In: Proc. IEEE International Conference on Control Applications, Taipei, Taiwan, pp. 303–308 (2004)
180. Karlsson, R., Norrlöf, M.: Bayesian position estimation of an industrial robot using multi sensors, Technical Report-ISY-R-2613, University of Linköping (2004)
181. Karlsson, R., Schön, T.B., Törnquist, D., Conte, G., Gustafson, F.: Utilizing model structure for efficient simultaneous localization and mapping for a UAV application. In: IEEE 2008 Aerospace Conference, 10.1109/AERO.2008.4526442 (2008)
182. Katebi, M.R., Yamamoto, I., Matsuura, M., Grimble, M.J., Hiroyama, H., Okamoto, N.: Robust dynamic ship positioning control system design and application. Intl. Journal of Robust and Nonlinear Control 11, 1257–1284 (2001)
183. Kazerooni, H., Sheridan, T.B., Houpt, P.K.: Robust Compliance Motion for Manipulators: Part I-The Fundamental Concepts of Compliant Motion, Part II-Design Method. IEEE Transactions on Robotics and Automation 2(2), 83–105 (1986)
184. Khalil, H.: Nonlinear Systems. Prentice-Hall, Englewood Cliffs (1996)
185. Khatib, O.: Real-time obstacle avoidance for manipulators and mobile robots. International Journal of Robotic Research 5(1), 90–99 (1986)
186. Khatib, O.: A Unified Approach for Motion and Force Control of Robot Manipulators: The Operational Space Formulation. IEEE Journal of Robotics and Automation 3(1), 43–53 (1987)
187. Khatib, O., Burdik, J.: Motion and force control of robot manipulators. In: Proc. of the IEEE Conf. On Robotics and Automation, pp. 1381–1386 (1986)

188. Khorrami, F., Jain, S., Tzes, A.: Experimental results on adaptive nonlinear control and input preshaping for multi-link flexible manipulators. Automatica 31(1), 83–97 (1995)
189. Kiguchi, K., Fukuda, T.: Fuzzy Neural Controller for Robot Manipulator Force Control. In: Proc. Joint Conf. 4th IEEE Int. Conf. Fuzzy Systems and 2nd Ind. Fuzzy Engineering Symp., vol. 2, pp. 869–874 (1995)
190. Kiguchi, K., Fukuda, T.: Robot Manipulator Contact Force Control Application of Fuzzy-Neural Network. In: Proc. IEEE Int. Conf. Robotics and Automation, vol. 1, pp. 875–880 (1995)
191. Kilic, E.: Diagnosability of fuzzy discrete event systems. Information Sciences 178(3), 858–870 (2008)
192. Kim, K.H., Baik, I.C., Chung, S.K., Youn, M.J.: Robust speed control of brushless DC motor using adaptive input-output linearization technique. IEEE Proc. on Electric Power Applications 144, 469–475 (1997)
193. Kitagawa, G.: Monte-Carlo filter and smoother for non-Gaussian non-linear state-space models. Journal of Computat. Graph. Statist. 5(1), 1–25 (1996)
194. Klebaner, F.C.: Introduction to stochastic calculus with applications. Imperial College Press (2005)
195. Klein, M., Rogers, G.J., Murty, S., Kundur, P.: Analytical Investigation of Factors Influencing Power Systems Stabilizers Performance. IEEE Transactions on Energy Conversion 7(3), 382–390 (1992)
196. Kolla, S., Varatharasa, L.: Identifying three-phase induction motor faults using artificial neural networks. ISA Transactions 39, 433–439 (2000)
197. Korba, P., Larsson, M., Rehtanz, C.: Detection of oscillations in power systems using Kalman filtering techniques. In: Proc. of the IEEE Conf. on Control Applications, vol. 1, pp. 183–188 (2003)
198. Koski, A., Juhola, M., Meriste, M.: Syntactic Recognition of ECG signals by attributed finite automata. Pattern Recognition 28(12), 1927–1940 (1995)
199. Kovacic, Z., Balenovic, M., Bogdan, S.: Sensitivity-based self-learning fuzzy logic control for a servo system. IEEE Control Systems Magazine 18(3), 41–51 (1998)
200. Kozen, D.C.: Automata and Computability. Springer, Heidelberg (1997)
201. Krstic, M., Kanellakopoulos, I., Kokotovic, P.V.: Nonlinear and Adaptive Control Design. J. Wiley, New York (1995)
202. Krzyzak, A., Sasiadek, J.: Flexible robot identification using nonparametric techniques. In: Proc. 30th IEEE Conference on Decision and Control, Brighton UK (December 1991)
203. Ku, C.-C., Lee, K.Y.: Diagonal Recurrent Neural Networks for Dynamic Systems Control. IEEE Transactions on Neural Networks 6(1), 144–156 (1995)
204. Kuan, C.P., Young, K.Y.: Reinforcement learning and robust control for compliance tasks. Journal of Intelligent and Robotic Systems 23(2-4), 165–182 (1998)
205. Kukolj, P., Levi, E.: Identification of complex systems based on neural and Takagi-Sugeno fuzzy model. IEEE Transactions on Systems Man and Cybernetics - Part B: Cybernetics 34(1), 272–282 (2004)
206. Kundur, P.: Power System Stability and Control. McGraw-Hill, New York (1994)
207. Kursawe, F.: A variant of evolution strategies for vector quantization. In: Schwefel, H.-P., Männer, R. (eds.) PPSN 1990. LNCS, vol. 496, pp. 193–197. Springer, Heidelberg (1991)
208. Kurylowicz, A., Jaworska, I., Tzafestas, S.G.: Robust Stabilizing Control: An Overview. In: Tzafestas, S.G. (ed.) Applied Control: Current Trends and Modern Methodologies, pp. 289–324. Marcel Dekker, New York (1993)

209. Laroche, B., Martin, P., Petit, N.: Commande par platitude: Equations différentielles ordinaires et aux derivées partielles. Ecole Nationale Supérieure des Techniques Avancées, Paris (2007)
210. Lask, T.A., Hsia, T.C.: Applied Control - Current Methods and Modern Methodologies. In: Tzafestas, S.G. (ed.), pp. 639–631. Marcel-Dekker, New York (1993)
211. Laumont, J.P., Jacobs, P.E., Taix, M., Murray, M.: A motion planner for nonholonomic mobile robots. IEEE Transactions on Robotics and Automation 10(5), 577–593 (1994)
212. Layne, J.R., Passino, K.M.: Fuzzy Model Reference Learning Control for Ship Steering. IEEE Control Systems Magazine 13(6), 23–34 (1993)
213. Lee, K.R., Kim, J.H., Jeung, E.T., Park, H.B.: Output Feedback Robust H_∞ Control of Uncertain Fuzzy Dynamic Systems with Time-Varying Delay. IEEE Trans. on Fuzzy Systems 8(6), 657–664 (2000)
214. Leisch, F., Jain, L.C., Hornik, K.: Cross-Validation with Active Pattern Selection for Neural-Network Classifiers. IEEE Transactions on Neural Networks 9(1), 35–41 (1998)
215. Leu, Y.-G., Lee, T.-T., Wang, W.-Y.: Observer-Based Adaptive Fuzzy-Neural Control for Unknown Nonlinear Dynamical Systems. IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics 29(5), 583–591 (1999)
216. Léchevin, N., Rabbath, C.A.: Sampled-data control of a class of nonlinear flat systems with application to unicycle trajectory tracking. ASME Journal of Dynamical Systems, Measurement and Control 128, 722–728 (2006)
217. Lee, D.J.: Unscented Information Filtering for Distributed Estimation and multiple sensor fusion. In: AIAA Guidance, Navigation and Control Conference and Exhibit, Hawai, USA (August 2008)
218. Lee, D.J.: Nonlinear estimation and multiple sensor fusion using unscented information filtering. IEEE Signal Processing Letters 15, 861–864 (2008)
219. Leung, Y.W., Wang, Y.: Multi-objective programming using uniform design and genetic algorithm. IEEE Transactions on Systems, Man and Cybernetics-Part C 30(3), 293–304 (2000)
220. Levine, H., Rappel, W.J.: Self-organization in systems of self-propelled particles. Physical Review E 63 (2000)
221. Lévine, J., Nguyen, D.V.: Flat output characterization for linear systems using polynomial matrices. Systems & Control Letters 48, 69–75 (2003)
222. Lewis, F.L., Jagannathan, S., Yesildirek, A.: Neural Network Control of Robot Manipulators and Nonlinear Systems. Taylor & Francis, Abington (1999)
223. Li, Y., Tang, B., Shi, Z., Lu, Y.: Experimental study for trajectory tracking of a two-link manipulator. International Journal of Systems Science 31(1), 3–9 (2000)
224. Li, G., Lie, T.T., Soh, C.B., Yang, G.H.: Design of state-feedback decentralized H_∞ controllers in power systems. Electrical Power & Energy Systems 24, 601–610 (2002)
225. Li, M., Pedrycz, W.: Fuzzy finite automata and fuzzy regular expressions with membership values in lattice ordered monoids. Fuzzy Sets and Systems 156(1), 68–92 (2005)
226. Li, P., Kadirkamanathan, V.: Particle Filtering Based Likelihood Ratio Approach to Fault Diagnosis in Nonlinear Stochastic Systems. IEEE Transactions on Systems, Man and Cybernetics - Part C: Applications and Reviews 31, 337–343 (2001)
227. Li, Y., Qiang, S., Zhuang, X., Kaynak, O.: Robust and Adaptive Backstepping Control for Nonlinear Systems Using RBF Neural Networks. IEEE Transactions on Neural Networks 15(3), 693–701 (2004)
228. Li, W., Leung, H.: Simultaneous registration and fusion of multiple dissimilar sensors for cooperative driving. IEEE Transactions on Intelligent Transportation Systems 5(2), 84–98 (2004)

229. Lin, F.J., Wai, R.J., Lin, C.H., Liu, D.C.: Decoupled stator-flux-oriented induction motor drive with fuzzy neural network uncertainty observer. IEEE Transactions on Industrial Electronics 47(2), 356–367 (2000)
230. Lin, F., Ying, H.: Modelling and control of fuzzy discrete event systems. IEEE Transactions on Systems, Man and Cybernetics- Part B: Cybernetics 32(4), 408–415 (2002)
231. Lin, J., Lewis, F.L.: Fuzzy controller for flexible-link robot arm by reduced-order techniques. IET Proceedings on Control Theory and Applications 149(3), 177–187 (2002)
232. Lin, C.K.: Nonsingular terminal sliding mode control of robot manipulators using fuzzy wavelet networks. IEEE Transactions on Fuzzy Systems 14, 849–859 (2006)
233. Lin, S.C., Kung, C.C.: Linguistic fuzzy sliding mode controller. In: Proc. 1992 Am. Contr. Conf, Chicago, IL, pp. 1904–1905 (1992)
234. Lindhé, M., Johansson, K.H., Bicchi, A.: An experimental study of exploiting multipath fading for robot communications. In: Robotics: Science and Systems, Atlanta, GA, USA (2007)
235. Liu, F., Qiu, D., Xing, H., Fan, Z.: Decentralized diagnosis of stochastic discrete event systems. IEEE Transactions on Automatic Control 53(2), 535–546 (2008)
236. Liu, F., Qiu, D.: Safe diagosability of stochastic discrete event systems. IEEE Transactions on Automatic Control 53(5), 1291–1296 (2008)
237. Liu, C.C., Jung, J., Heydt, G.T., Vittal, V., Phadke, A.G.: The strategic power infrastructure defense (SPID) system: a conceptual design. IEEE Control Systems Magazine 20(4), 40–52 (2000)
238. Liu, S., Asada, H.: Transfer of Human Skills to Robots: Learning from Human Demonstrations for Building an Adaptive Control System. In: Proc. 1993 American Control Conf., pp. 1414–1417 (1993)
239. Lo, J.C., Lin, M.L.: Observer-Based Robust H_∞ Control for Fuzzy Systems Using Two-Step Procedure. IEEE Transactions on Fuzzy Systems 12(3), 350–359 (2004)
240. Loria, A., Fossen, T.I., Panteley, E.: A separation principle for dynamic positioning of ships: Theoretical and Experimental Resutls. IEEE Transactions on Control Systems Technology 8(2), 332–343 (2000)
241. Lublin, L., Athans, M.: An experimental comparison of and designs for interferometer testbed. In: Francis, B., Tannenbaum, A. (eds.) Feedback Control, Nonlinear Systems and Complexity. LNCIS, pp. 150–172. Springer, Heidelberg (1995)
242. Luh, J.Y.S., Walker, W.M., Paul, R.P.C.: Resolved Acceleration Control of Manipulators. IEEE Transactions on Automatic Control 25, 468–474 (1980)
243. Lunze, J.: Fault diagnosis of discretely controlled continuous systems by means of discrete event models. Discrete Event Dynamical Systems 18(2), 181–210 (2008)
244. Luo, R.C., Kay, M.G.: Multisensor Integration and Fusion in Intelligent Systems. IEEE Transactions on Systems, Man and Cybernetics 19, 901–931 (1989)
245. Makarenko, A., Durrany-Whyte, H.: Decentralized Bayesian algorithms for active sensor networks. Information Fusion 7, 418–433 (2006)
246. Mackey, M.C., Glass, L.: Oscillation and Chaos in Physiological Control Systems. Science, New Series 197(4300), 287–289 (1997)
247. Mahler, R.P.S.: Statistical Multisource-Multitarget Information Fusion. Artech House Inc., Norwood (2007)
248. Majumber, R., Chaudhuri, B., Pal, B.C., Zhong, Q.C.: A unified Smith predictor approach for power system damping control design using remote signals. IEEE Transactions on Control Systems Technology 13, 1063–1068 (2005)
249. Makarov, Y.V., Voropai, N.I., Efimov, D.N.: Complex emergency control system against blackout in Russia. In: Proceedings of IEEE Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century, Pittsburg, USA, pp. 1–8 (2008)

250. Malis, E., Chaumette, F., Boudet, S.: Multi-cameras visual servoing. In: IEEE Intrl. Conf. on Robotics and Automation, ICRA 2000, pp. 3183–3188 (2000)
251. Mallat, S.: A wavelet tour of signal processing, 2nd edn. Acedemic Press, New York (1999)
252. Manyika, J., Durrant-Whyte, H.: Data fusion and sensor management: a decentralized information theoretic approach. Prentice-Hall, Englewood Cliffs (1994)
253. Mansour, Y., Vaahedi, E., EL-Sharkawi, M.A.: Large scale dynamic security screening and ranking using neural networks. IEEE Transactions on Power Systems 12, 954–960 (1997)
254. Marino, R., Peresada, S., Valigi, P.: Adaptive Input-Output Linearizing Control of Induction Motors. IEEE Transactions on Automatic Control 38(2), 208–221 (1991)
255. Marino, R.: Adaptive observers for single output nonlinear systems. IEEE Transactions on Automatic Control 35(9), 1054–1058 (1990)
256. Marino, R., Tomei, P.: Global asymptotic observers for nonlinear systems via filtered transformations. IEEE Transactions on Automatic Control 37(8), 1239–1245 (1992)
257. Martin, P., Rouchon, P.: Systèmes plats: planification et suivi des trajectoires. Journées X-UPS, École des Mines de Paris, Centre Automatique et Systèmes (Mai 1999)
258. Martins, J.F., Pires, A.J., Vilela Mendes, R., Dente, J.A.: Modelling Electromechanical Drive Systems: A Formal Language Approach. In: Proceedings of 35th IEEE Industry Applications Society Annual Meeting, IAS 2000, Rome, Italy (2000)
259. Masoud, S.A., Masoud, A.A.: Motion planning in the presence of directional and regional avoidance constraints using nonlinear, anisotropic, harmonic potential fields: a physical metaphor. IEEE Transactions on Systems, Man and Cybernetics - Part A 32(6), 705–723 (2002)
260. Mauris, G., Berrah, L., Foulloy, L., Haurat, A.: Fuzzy handling of measurement errors in instrumentation. IEEE Transactions on Measurement and Instrumentation 49, 89–93 (2000)
261. Medeiros, H., Park, J., Kak, A.C.: Distributed object tracking using a cluster-based Kalman Filter in wireless camera networks. IEEE Journal of Selected Topics in Signal Processing 2(4), 448–463 (2008)
262. Medina Martinez, J.I., Sawat, U., Nakano, K.: Application of non-linear observer with simultaneous perturbation stochastic approximation method to single flexible-link SMC. In: SICE Annual Conference 2008, The University of Electro-Communications, Japan (2008)
263. van der Merwe, R., Wan, E.A., Julier, S.I.: Sigma-Point Kalman Filters for nonlinear estimation and sensor-fusion applications to intergrated navigation. In: Proc. of the AIAA Guidance, Navigation and Control Conference, Providence, RI, USA (August 2004)
264. Meurer, T., Zeitz, M.: A modal approach to flatness-based control of flexible structures. PAMM: Proceedings on Applied Mathematics and Mechanics 4, 133–134 (2004)
265. Michalewicz, Z.: Genetic Algorithms+Data Structures=Evolution Programs. Springer, Berlin (1996)
266. Míguez, J.: Analysis of parallelizable resampling algorithms for particle filtering. Signal Processing 87, 3155–3174 (2007)
267. Milano, F.: An open-source power system analysis toolbox. IEEE Transactions on Power Systems 20(3), 1199–1206 (2006)
268. Minkova, M.D., Minkov, D., Rodgerson, J.L., Harley, R.G.: Adaptive neural speed controller of a DC motor. Electric Power Systems Research 47(2), 123–132 (1998)
269. Mohamed, Z., Tokhi, M.O.: Hybrid control schemes input tracking and vibration suppression of flexible manipulators. Proc. IMechE - Part I: Journal of Systems and Control Engineering 217(1), 23–34 (2003)

270. Moon, H.S., Kim, Y.B., Beattie, R.J.: Multi sensor data fusion for improving performance and reliability of fully automatic welding system. International Journal of Advanced Manufacturing Technology 28, 286–293 (2006)
271. Morderson, J.N., Malik, D.S.: Fuzzy Automata and Languages. Chapman & Hall, Boca Raton (2002)
272. Morelande, M.R., Mušicki, D.: Fast multiple target tracking using particle filters. In: Proc. of the 44th IEEE Conference on Decision and Control, and the European Control Conference 2005, Seville, Spain, December 12-15 (2005)
273. Mounier, H., Rudolph, J.: Trajectory tracking for π-flat nonlinear dealy systems with a motor example. In: Isidori, A., Lamnabhi-Lagarrigue, F., Respondek, W. (eds.) Nonlinear Control in the year 2000, vol. 1. LNCIS, vol. 258, pp. 339–352. Springer, Heidelberg (2001)
274. Moseler, O., Isermann, R.: Application of model-based fault detection to a brushless DC motor. IEEE Transactions on Industrial Electronics 47(5), 1015–1020 (2000)
275. Mudgal, V., Kwong, W.A., Passino, K.M., Yurkovich, S.: Fuzzy learning control for a flexible-link robot. IEEE Transactions on Fuzzy Systems 3(2), 199–210 (1995)
276. Muenchhof, M., Beck, M., Isermann, R.: Fault-tolerant actuators and drives - Structures, fault detection principles and applications. Annual Reviews in Control 33, 136–148 (2009)
277. Musso, C., Oudjane, N., Le Gland, F.: Imrpoving regularized particle filters. In: Doucet, A., de Freitas, N., Gordon, N. (eds.) Sequential Monte Carlo Methods in Practice, pp. 247–272. Springer, Heidelberg (2001)
278. Nagarkatti, S.P., Rahn, C.D., Dawson, D.M., Zergeroglou, E.: Observer-based modal control of flexible systems using distributed sensing. In: Proceedings of the 40th IEEE Conference on Decision and Control, Orlando, Florida, USA (December 2001)
279. Narendra, K.S., Parthasarathy, K.: Identification and control of dynamical systems using neural networks. IEEE Transactions on Neural Networks 1(1), 4–27 (1990)
280. Narendra, K.S., Thathachar, M.A.: Learning Automata: an Introduction. Prentice Hall, Englewood Cliffs (1989)
281. Nelles, O., Fischer, M.: Local linear model trees (LOLIMOT) for linear system identification of a cooling blast. In: Proc: EUFIT 1996 Conference, Aachen, Germany (1996)
282. Nethi, S., Pohjola, M., Eriksson, L., Jänti, R.: Simulation case studies of Wireless Networked Control Systems. In: Proc. 2nd ACM Intl. Conference, International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems, Chania, Crete, Greece (October 2007)
283. Nettleton, E., Durrant-Whyte, H., Sukkarieh, S.: A robust architecture for decentralized data fusion. In: 11th International Conference on Advanced Robotics, ICAR 2003, Coimbra, Portugal (2003)
284. Ng, G.W.: Intelligent Systems - Fusion, Tracking and Control. Research Studies Press Ltd. (2003)
285. Nguyen, T.D., Egeland, O.: Observer design for a flexible robot arm with a tip load. In: 2005 American Control Conference, Portland, Oregon, USA (2005)
286. Nguyen, V.B., Morris, A.S.: Genetic algorithm tuned fuzzy logic controller for a robot arm with two-link flexibility and two-joint elasticity. Journal of Intelligent and Robotic Systems 49(1), 3–18 (2007)
287. Nounou, H.N., Rehman, H.: Application of adaptive fuzzy control to AC machines. Applied Soft Computing 7(3), 899–907 (2007)
288. Novosel, D., Begovic, M.M., Madani, V.: Shedding light on blackouts. IEEE Power & Energy Magazine 2(1), 32–43 (2006)

289. O' Connor, W.J.: Wave-based analysis and Control of Lump-Modeled Flexible Robots. IEEE Transactions on Robotics 23(1), 342–352 (2007)
290. Olfati-Saber, R.: Distributed Kalman Filtering and Sensor Fusion in Sensor Networks, LNCIS, vol. 331, pp. 157–167. Springer, Heidelberg (2006)
291. Olfati-Saber, R.: Distributed Kalman Filter with Embedded Consensus Filters. In: Proc. 44th IEEE Conference on Decision and Control, Seville, Spain, pp. 8179–8184 (2005)
292. Oliveira, J.V.: Towards Neuro-linguistic modelling: Constraints for Optimization of Membership functions. Fuzzy Sets and Systems 106, 357–380 (1999)
293. Ong, L.L., Bailey, T., Durrant-Whyte, H., Upcroft, B.: Decentralized Particle Filtering for Multiple Target Tracking in Wireless Sensor Networks. In: Fusion 2008, The 11th International Conference on Information Fusion, Cologne, Germany (July 2008)
294. Ong, L.L., Upcroft, B., Bailey, T., Ridley, M., Sukkarieh, S., Durrant-Whyte, H.: A decentralized particle filtering algorithm for multi-target tracking across multiple flight vehicles. In: 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing, China (October 2006)
295. Oriolo, G., De Luca, A., Vendittelli, M.: WMR Control Via Dynamic Feedback Linearization: Design, Implementation and Experimental Validation. IEEE Transactions on Control Systems Technology 10(6), 835–852 (2002)
296. Paoli, A., Lafortune, S.: Diagnosability analysis of a class of hierarchical state machines. Discrete Event Dynamical Systems 18, 385–413 (2008)
297. Passino, K., Burgess, K.: Stability Analysis of Discrete Event Systems. J. Wiley, New York (1998)
298. Passino, K., Mitchel, A., Antsaklis, P.: Lyapunov stability of a class of discrete event systems. IEEE Transactions on Automatic Control 37, 269–279 (1994)
299. Pagello, E., D' Angelo, A., Menegatti, E.: Cooperation issues and distributed sensing for multi-robot systems. Proceedings of the IEEE 94(7), 1370–1383 (2006)
300. Palm, R., Driankov, D., Hellendoorn, H.: Model Based Fuzzy Control, pp. 75–113. Springer, Berlin (1996)
301. Pedrycz, W.: Fuzzy Control and Fuzzy Systems, 2nd edn. J. Wiley, Chichester (1993)
302. Pedrycz, W.: Why triangular membership functions? Fuzzy Sets and Systems 64, 21–30 (1994)
303. Perez, P., Vermaak, J., Blake, A.: Data Fusion for Visual Tracking with Particles. Proceedings of the IEEE 92(3), 495–513 (2004)
304. Peterson, J.L.: Petri Net Theory and The Modeling of Systems. Prentice-Hall, Englewood Cliffs (1981)
305. Poyo, I., Feliu, V., Cortezar, O.D.: Force control of a very lightweight single-link flexible arm based on coupling torque feedback. Mechatronics 19, 334–347 (2007)
306. Proth, J.-M., Xie, X.: Petri Nets: A Tool for Design and Management of Manufacturing Systems. J. Wiley, Chichester (1996)
307. Proud, A.W., Pachter, M., D' Azzo, J.J.: Close formation flight control. In: AIAA Conference on Guidance, Navigation and Control, AIAA-1999-4207 (1999)
308. Psichogios, D.C., Ungar, L.H.: SVD-NET: An algorithm that automatically selects network structure. IEEE Transactions on Neural Networks 5(3), 513–515 (1994)
309. Qiu, D.W.: Automata theory based on completed residuated lattice-valued logic (II). Science in China (F) 45(6), 442–452 (2002)
310. Qiu, D.: Characterizations of fuzzy finite automata. Fuzzy Sets and Systems 141(3), 391–414 (2004)
311. Qiu, D.: Supervisory control of fuzzy discrete event systems: A formal approach. IEEE Transactions on Systems, Man and Cybernetics- Part B: Cybernetics 35(1), 72–88 (2005)

312. Qiu, W., Kumar, R.: Distributed diagnosis under bounded-delay communication of immediately forwarded local observations. IEEE Transactions on Systems, Man and Cybernetics - Part A: Systems and Humans 38(3), 628–643 (2008)
313. Rahma, M.A., Hoque, M.A.: On-line adaptive neural network based vector control of permanent magnet synchronous motors. IEEE Transactions on Energy Conversion 13, 311–318 (1998)
314. Raibert, M.H., Craig, J.J.: Hybrid Position/Force Control of Manipulators. ASME Journal of Dynamical Systems, Measurement and Control 102, 126–133 (1981)
315. Ramadge, P.J., Wohnam, W.M.: Supervisory control of a class of discrete event processes. SIAM J. Contr. Optimization 25(1), 206–230 (1987)
316. Ramirez Trevino, A., Ruiz-Beltran, E., Rivera-Rangel, I., Lopez Mellado, E.: On-line fault diagnosis of discrete event systems - a Petri-Net-based approach. IEEE Transactions on Automation Science and Engineering 4(1), 31–39 (2007)
317. Ramos, F., Feliu, V.: New online payload identification for flexible robots: Applications of adaptive control. Journal of Sound and Vibration 315, 34–57 (2008)
318. Rao, B.S., Durrant-Whyte, H.F.: Fully decentralized algorithm for multisensor Kalman Filtering. IET Proceedings: Control Theory and Applications 138(5), 413–451 (1991)
319. Refregier, A.: Shapelets - I, A method for image analysis. Mon. Not. R. Astron. Soc. 338, 35–47 (2003)
320. Reif, J.H., Wang, H.: Social potential fields: A distributed behavioral control for autonomous robots. Robotics and Autonomous Systems 27, 171–194 (1999)
321. Ren, H., Dobson, I.: Using transmission line outage data to estimate cascading failure propagation in an electric power system. IEEE Transactions on Circuits and Systems-II 55(9), 927–931 (2008)
322. Ren, W., Beard, R.W.: Trajectory tracking for unmanned air vehicles with velocity and heading rate constraints. IEEE Transactions on Control Systems Technology 12(5), 706–716 (2004)
323. Renno, J.M.: Inverse dynamics-based tuning of a fuzzy logic controller for a single-link flexible manipulator. Journal of Vibrations Control 13(12), 1741–1759 (2007)
324. Reutenauer, C.: Non commuting variables: finite automata. In: Singh, M.G. (ed.) Systems and Control Encyclopedia, pp. 3268–3272. Pergamon Press, Oxford (1987)
325. Rigatos, G.G.: Distributed gradient for multi-robot motion planning. In: ICINCO 2005, 2nd Intl. Conference on Informatics in Control, Automation and Robotics, Barcelona, Spain (September 2005)
326. Rigatos, G.G.: Distributed gradient and particle swarm optimization for multi-robot motion planning. Robotica 26(3), 357–370 (2008)
327. Rigatos, G.G.: Model-free control of flexible-link robots. In: IC-SCCE 2006, 2nd Intl. Conference "From Scientific Computing to Computational Engineering", Athens, Greece (July 2006)
328. Rigatos, G.G.: Model-based and model-free control of flexible-link robots: a comparison between representative methods. Applied Mathematical Modelling 33, 3906–3925 (2009)
329. Rigatos, G.G., Tzafestas, S.G., Evangelidis, G.A.: Reactive parking control of a nonholonomic vehicle via a Fuzzy Learning Automaton. IET Proceedings: Control Theory and Applications, 169–179 (2001)
330. Rigatos, G.G.: Fuzzy Stochastic Automata for Intelligent Vehicle Control. IEEE Transactions on Industrial Electronics 50(1), 76–79 (2003)
331. Rigatos, G.G., Tzafestas, S.G.: Extended Kalman Filtering for Fuzzy Modelling and Multi-Sensor Fusion. Mathematical and Computer Modelling of Dynamical Systems 13, 251–266 (2007)

332. Rigatos, G.G.: Distributed particle filtering over sensor networks for autonomous navigation of UAVs. In: Lazinica, A. (ed.) Robot Manipulators. InTech Publications (2010)
333. Rigatos, G.G.: Particle Filtering for state estimation in industrial robotic systems. IMeche Journal of Systems and Control Engineering 222, 437–455 (2008)
334. Rigatos, G.G.: Particle Filtering for State Estimation in Nonlinear Industrial Systems. IEEE Transactions on Instrumentation and Measurement 58, 3885–3900 (2009)
335. Rigatos, G.G.: Particle and Kalman filtering for state estimation and control of DC motors. ISA Transactions 48(1), 62–72 (2009)
336. Rigatos, G.G., Siano, P., Merola, E.: Sensorless control of DC and induction motors using Kalman Filtering. In: MASCOT 2009, IMACS Workshop on Scientific Computation, Italian Institute for Calculus Applications, Roma, Italy (October 2009)
337. Rigatos, G.G.: A derivative-free Kalman Filtering approach for sensorless control of nonlinear systems. In: IEEE ISIE 2010, IEEE International Symposium on Industrial Electronics, Bari, Italy (July 2010)
338. Rigatos, G.G.: Adaptive fuzzy control with output feedback for H_∞ tracking of SISO nonlinear systems. International Journal of Neural Systems 18(4), 1–16 (2008)
339. Rigatos, G.G., Tzafestas, S.G.: Adaptive Fuzzy Control for the Ship Steering Problem. Journal of Mechatronics 16(6), 479–489 (2006)
340. Rigatos, G.G.: Adaptive fuzzy control of DC motors using state and output feedback. Electric Power Systems Research 79(11), 1579–1592 (2009)
341. Rigatos, G.G., Tzafestas, S.G.: H_∞ tracking for uncertain SISO nonlinear systems: an observer-based adaptive fuzzy approach. International Journal of Systems Science 38(6), 459–472 (2007)
342. Rigatos, G.G.: Extended Kalman and Particle Filtering for sensor fusion in motion control of mobile robots. Mathematics and Computers in Simulation (2010), doi:10.1016/j.matcom.2010.05.003
343. Rigatos, G.G.: Autonomous robots navigation using flatness-based control and multisensor fusion. In: Pecherkova, P., Fliidr, M., Dunik, J. (eds.) Robotics, Automation and Control, pp. 394–416. InTech Publications (2008)
344. Rigatos, G., Zhang, Q.: Fuzzy model validation using the local statistical approach. Fuzzy Sets and Systems 60(7), 882–904 (2009)
345. Rigatos, G.G., Tzafestas, C.S., Tzafestas, S.G.: Mobile Robot Motion Control in Partially Unknown Environments Using a Sliding-Mode Fuzzy-Logic Controller. Robotics and Autonomous Systems 33(1), 1–11 (2000)
346. Rigatos, G.G., Siano, P., Piccolo, A.: A neural network-based approach for early detection of cascading events in electric power systems. IET Journal on Generation Transmission and Distribution 3, 650–665 (2009)
347. Rigatos, G.G., Tzafestas, S.G.: Neural structures using the eigenstates of the Quantum Harmonic Oscillator. Open Systems and Information Dynamics 13, 27–41 (2006)
348. Rigatos, G.G., Tzafestas, S.G.: Feed-forward neural networks using Hermite polynomial activation functions. In: Valmari, A. (ed.) SPIN 2006. LNCS (LNAI), vol. 3925, pp. 323–333. Springer, Heidelberg (2006)
349. Rigatos, G.G., Tzafestas, S.G.: Fuzzy automata for fault diagnosis: A syntactic analysis approach. In: Vouros, G.A., Panayiotopoulos, T. (eds.) SETN 2004. LNCS (LNAI), vol. 3025, pp. 301–310. Springer, Heidelberg (2004)
350. Rigatos, G.G.: Sigma-point Kalman Filters and Particle Filters for integrated navigation of unmanned aerial vehicles. In: Intl. Workshop on Robotics for Risky Interventions and Environmental Surveillance, RISE 2009, Brussels, Belgium (January 2009)
351. Rimon, E., Koditscheck, D.E.: Exact robot navigation using artifical potential functions. IEEE Transactions on Robotics and Automation 8, 501–518 (1991)

352. Ristic, B., Arulampalam, M.S., Gordon, N.J.: Beyond the Kalman Filter: Particle Filters for Tracking Applications. Artech House, Norwood (2004)
353. Robak, S.: Robust SVC controller design and analysis for uncertain power systems. Control Engineering Practice 17, 1280–1290 (2009)
354. Rosencrantz, M., Gordon, G., Thrun, S.: Decentralized data fusion with distributed particle filtering. In: Proceedings of the Conference of Uncertainty in AI (UAI), Acapulco, Mexico (2003)
355. Rouchon, P.: Flatness-based control of oscillators. ZAMM Zeitschrift fur Angewandte Mathematik und Mechanik 85(6), 411–421 (2005)
356. Rouchon, P., Laroche, B., Martin, P.: Motion planning of the heat equation. Int. Journal of Robust and Nonlinear Control 10, 629–643 (2000)
357. Rubaai, A., Kotaru, R.: Online identification and control of a DC motor using learning adaptation of neural networks. IEEE Transactions on Industry Applications 36(3), 935–942 (2000)
358. Rubuai, A., Kotaru, R.: Neural-net based robust controller design for brushless DC motor drives. IEEE Transactions on Systems, Man and Cybernetics - Part C: Applications and Reviews 29(3), 460–474 (1999)
359. Rubaai, A., Ricketts, D., Kankam, M.D.: Development and implementatio of an adaptive fuzzy-neural-network controller for brushless drives. IEEE Transactions on Industry Applications 38(2), 441–447 (2002)
360. Rubaai, A., Kotaru, R., Kankam, M.D.: A continually online-trained neural network for brushless DC motor drives. IEEE Transactions on Industry Applications 36(2), 475–483 (2000)
361. Rudolph, J.: Flatness Based Control of Distributed Parameter Systems, Steuerungs- und Regelungstechnik. Shaker Verlag, Aachen (2003)
362. Ruiz, A.R.J., Granja, F.S.: A short-range ship navigation system based on Ladar imaging and target tracking for improved safety and efficiency. IEEE Transactions on Intelligent Transportation Systems 10(1), 186–197 (2009)
363. Sampath, M., Sengupta, R., Lafortune, S., Sinnamohiddenen, K., Teneketzis, D.: Diagnosability of Discrete-Event Systems. IEEE Transactions on Automatic Control 40(9), 1555–1575 (1995)
364. Sala, A.: Encoding fuzzy possibilistic diagnostics as a constrained optimization problem. Information Sciences 178(22), 4246–4263 (2008)
365. Sanchez-Gasca, J.J., Chow, J.H.: Performance comparison of three identification methods for the analysis of electromechanical oscillation. IEEE Transactions on Power Systems 14, 995–1002 (1999)
366. Sanz, A., Extebarria, V.: Experimental control of a single-link flexible robot arm using energy shaping. International Journal of Systems Science 38(1), 61–71 (2007)
367. Saporta, G.: Probabilités, analyse des données et statistique, Editions Technip (1990)
368. Särrkä, S.: On Unscented Kalman Filtering for state estimation of continuous-time nonlinear systems. IEEE Transactions on Automatic Control 52(9), 1631–1641 (2007)
369. Schaffer, J.D.: Multi-objective optimization with vector evaluated genetic algorithms. In: Proceedings of the First International Conference on Genetic Algorithms, pp. 93–100 (1985)
370. Schön, T.B., Törnqvist, D., Gustafsson, F.: Fast Particle Fitlers for Multi-rate Sensors. In: EUSIPCO 2007 Conference, Proceedings of the 15th European Signal Processing Conference, Poznan, Poland (September 2007)
371. Schuurman, D.C., Capson, D.W.: Robust direct visual servo using network-synchronized cameras. IEEE Transactions on Robotics and Automation 20(2), 319–334 (2004)

372. Sepulchre, R., Paley, D.A., Leonard, N.E.: Stabilization of planar collective motion: all to all communication. IEEE Transactions on Automatic Control 52(5), 811–824 (2007)
373. Setnes, M., Babuska, R.: Fuzzy modeling for predictive control. In: Farinwata, Filev, Langari (eds.) Fuzzy control - Synthesis and Analysis, pp. 23–46. J. Wiley, Chichester (2000)
374. Setnes, M., Babuska, R., Kaymak, U., van Nauta Lemke, H.R.: Similarity measures in fuzzy rule base simplification. IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics 28(3), 376–386 (1998)
375. Setnes, M., Kaymak, U.: Extended fuzzy c-means with volume prototypes and cluster merging. In: Proc. EUFIT 1998, Aachen, Germany (1998)
376. Setnes, M., Babuska, R.: Fuzzy modeling for predictive control. In: Farinwata, Filev, Langari (eds.) Fuzzy control - Synthesis and Analysis, pp. 23–46. J. Wiley, Chichester (2000)
377. Shahed, M.H., Poerwanto, H., Tokhi, M.O.: Adaptive inverse-dynamic and neuro-inverse-dynamic active vibration control of a single-link flexible manipulator. Proc. IMechE - Part I: Journal of Systems and Control Engineering 219(6), 431–448 (2005)
378. Shi, L.Z., Trabia, M.B.: Design and tuning of importance-based fuzzy logic controller for a flexible-link manipulator. Journal of Intelligent and Fuzzy Systems 17(3), 313–323 (2006)
379. Shima, T., Rasmussen, S.J., Chandler, P.: UAV team decision and control using efficient collaborative estimation. ASME Journal of Dynamical Systems, Measurement and Control 129(5), 609–619 (2007)
380. Siciliano, B., Villani, L.: An Inverse Kinematics Algorithm for Interaction Control of a Flexible Arm with a Compliant Surface. Control Engineering Practice 9, 191–198 (2001)
381. Simandl, M., Straka, O.: Sampling densities of Particle: A survey and comparison. In: Proc. Americal Control Conference 2007, New York, USA, pp. 4437–4442 (July 2007)
382. Simon, D.: Training fuzzy systems with the Extended Kalman Filter. Fuzzy Sets and Systems 132, 189–199 (2002)
383. Simon, D.: Sum normal optimization of fuzzy membership functions. International Journal of Uncertainty, Fuzziness and Knowledge-based Systems 10, 363–384 (2002)
384. Singh, G.K., Al Kazzaz, S.A.S.: Induction machine drive condition monitoring and diagnostic research - a survey. Electric Power Systems Research 64, 145–158 (2003)
385. Singh, L., Fuller, J.: Trajectory generation for a UAV in urban terrain using nonlinear MPC. In: Proc. American Control Conference 2001, pp. 2301–2308 (2001)
386. Sinha, A., Ghose, D.: Generalization of Linear Cyclic Pursuit with Application to Rendezvous of Multiple Autonomous Agents. IEEE Transactions on Automatic Control 51(11), 1819–1824 (2006)
387. Skjetne, R., Fossen, T.I.: Nonlinear maneuvering and control of ships. In: Proceedings of OCEANS 2001 MTS/IEEE Conference and Exhibition, pp. 1808–1815 (2001)
388. Slotine, J.E., Li, W.: Applied Nonlinear Control. Prentice Hall, Englewood Cliffs (1991)
389. Soliman, H.M., Elshafei, A.L., Shaltout, A.A., Morsi, M.F.: Robust power system stabilizer. IET Proceedings on Electric Power Applications 147, 285–291 (2000)
390. Song, S., Kezunovic, M.: A new analysis method for early detection and prevention of cascading events. Electric Power Systems Research 77, 1132–1142 (2007)
391. Spong, M.W., Vidyasagar, M.: Robot Dynamics and Control. J. Wiley, Chichester (1989)
392. Spooner, J.T., Passino, K.M.: Stable Adaptive Control Using Fuzzy Systems and Neural Networks. IEEE Transactions on Fuzzy Systems 4(3), 339–359 (1996)

393. Steimann, F., Adlassnig, K.P.: Clinical monitoring with fuzzy automata. Fuzzy Sets and Systems 61(1), 37–42 (2009)
394. Su, R., Wonham, W.M.: Global and local consistencies in distributed fault diagnosis for discrete event systems. IEEE Transactions on Automatic Control 50(12), 1923–1935 (2005)
395. Su, Z., Khorasani, K.: A neural-network-based controller for a single link flexible manipulator using the inverse dynamics approach. IEEE Transactions on Industrial Electronics 48, 1074–1086 (2001)
396. Su, C.Y., Stephanenko, Y.: Adaptive fuzzy control of a class of nonlinear systems. IEEE Transactions on Fuzzy Systems 2(4), 285–294 (1994)
397. Subudhi, B., Morris, A.S.: Soft computing methods applied to the control of a flexible robot manipulator. In: Applied Soft Computing. Elsevier, Amsterdam (2009), doi:10.1016/j.asoc.2008.02.2004
398. Subudhi, B., Morris, A.S.: Fuzzy and neuro-fuzzy approaches to control a flexible single-link manipulator. Proc. IMechE - Journal of Systems and Control Engineering 217, 387–399 (2003)
399. Summerer, L., Keller, J.Y., Darouach, M.: Robust fault diagnosis with a two-stage Kalman estimator. European Journal of Control 3, 247–252 (1997)
400. Sureshbabu, N., Farell, J.A.: Wavelet-based system identification for nonlinear control. IEEE Transactions on Automatic Control 44, 412–417 (1999)
401. Takagi, T., Sugeno, M.: Fuzzy identification of Systems and its applications to modeling and control. IEEE Transactions on Systems, Man and Cybernetics 15, 116–132 (1985)
402. Talebi, H.A., Khorasani, K., Patel, R.V.: Neural network based control schemes for flexible-link manipulators: simulations and experiments, Special Issue "Neural Control Robotics". Biol. Technol. 11, 1357–1377 (1998)
403. Tan, N., Kaya, I., Yeroglu, C., Atherton, D.P.: Computation of stabilizing PI and PID controllers using the stability boundary locus. Energy Conversion & Management 47, 3045–3058 (2006)
404. Tan, J.C., Crossley, P.A., McLaren, P.G.: Application of a wide area backup protection expert system to prevent cascading outages. IEEE Transactions on Power Delivery 17(2), 375–380 (2002)
405. Tannuri, E.A., Morishita, H.M.: Experimental and numerical evaluation of a typical dynamic positioning system. Applied Ocean Research 28, 133–146 (2006)
406. Teeuwsen, S.P., Erlich, I., Fischer, A., El-Sharkawi, M.A.: Assessment of the small signal stability of the European interconnected electric power system using neural networks. In: Proc. Large Engineering Systems Conference on Power Engineering, pp. 158–161 (2001)
407. Teeuwsen, S.P., Erlich, I., El Sharkawi, M.A.: Small-signal stability assessment based on advanced neural network methods. IEEE Power Engineering Society General Meeting 4, 13–17 (2003)
408. Tharmarasa, R., Kirubarajan, T., Peng, J., Lang, T.: Optimization-Based Dynamic Sensor Management for Distributed Multitarget Tracking. IEEE Transactions on Systems Man and Cybernetics - Part C 39(5), 534–546 (2009)
409. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. MIT Press, Cambridge (2005)
410. Tian, L., Collins, C.: Adaptive neuro-fuzzy control of a flexible manipulator. Mechatronics, 1305–1320 (2005)
411. Tian, L., Wang, J., Mao, Z.: Constrained motion control of flexible robot manipulators based on recurrent neural networks. IEEE Transactions on System, Man and Cybernetics - Part B: Cybernetics 34(3), 1541–1552 (2004)

412. Tong, S., Li, H.-X., Chen, G.: Adaptive Fuzzy Decentralized Control for a Class of Large-Scale Nonlinear Systems. IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics 34(1), 770–775 (2004)
413. Torrésani, B.: Analyse continue par ondelettes, CNRS Editions (1995)
414. Toscano, R., Lyonnet, P.: Robust static output feedback controller synthesis using Kharitonov's theorem and evolutionary algorithms. Information Sciences 180, 2023–2028 (2010)
415. Trahanias, P., Skordalakis, E.: Syntactic Pattern Recognition of the ECG. IEEE Transactions on Pattern Analysis and Machine Intelligence 12(7), 648–657 (1990)
416. Trahanias, P., Skordalakis, E., Papakonstantinou, G.: A syntactic method for the classification of the QRS patterns. Pattern Recognition Letters 9(1), 13–18 (1989)
417. Tsai, J.S.H., Lin, M.H., Zheng, C.H., Guo, S.M., Shieh, L.S.: Actuator fault detection and performance recovery with Kalman filter-based adaptive observer. International Journal of General Systems 36(4), 375–398 (2007)
418. Tzafestas, S.G.: Knowledge-Based System Diagnosis. In: Supervision and Control. Plenum Press, New York (1989)
419. Tzafestas, S.G., Verbruggen, H.B.: Artificial Intelligence in Industrial Decision Making. In: Control and Automation. Kluwer Academic Publishers, Dordrecht (1994)
420. Tzafestas, S.G.: Expert Systems in Engineering Applications. Springer, Berlin (1993)
421. Tzafestas, S.G., Rigatos, G.G.: Fuzzy Reinforcement Learning Control for Robot Compliance Tasks. IEEE Transactions on Systems, Man and Cybernetics - Part B 32(1), 107–113 (2002)
422. Tzafestas, S.G., Rigatos, G.G.: Neural and Neurofuzzy FELA Adaptive Robot Control Using Feedforward and Counterpropagation Networks. Journal of Intelligent and Robotic Systems 23, 291–330 (1999)
423. Tzafestas, S.G., Rigatos, G.G.: Self-tuning multivariable fuzzy and neural control using genetic algorithms. Journal of Information & Optimization Sciences 21(2), 257–287 (2000)
424. Tzafestas, S.G., Rigatos, G.G.: Stability analysis of an adaptive fuzzy control system using Petri Nets and learning automata. Mathematics and Computers in Simulation 51(3), 315–341 (2000)
425. Tzafestas, S.G., Singh, M.G., Schmidt, G.: System fault diagnosis, reliabilty and related Knowledge-based Approaches. Fault Diagnostics and Reliability. Knowledge-based and Fault Tolerant techniques, vol. 1. Reidel, Dordrecht (1989)
426. Tzafestas, S.G., Rigatos, G.G., Vagelatos, G.A.: Design of fuzzy-gain scheduled robust controllers using Kharitonov's extremal gain margin theorem. Journal of Intelligent and Fuzzy Systems 10, 39–56 (2001)
427. Tzafestas, S.G., Skoundrianos, E.N., Rigatos, G.G.: Nonlinear Neural Control of Discrete-Time Systems Using Local Model Networks. Knowledge-Based & Intelligent Engineering Systems 4 (2000)
428. Tzafestas, S.G., Kyriannakis, E., Rigatos, G.: Adaptive, model based predictive and neural control of the welding process. In: Tzafestas, S.G. (ed.) Methods and Applications of Intelligent Control, pp. 509–548. Kluwer Academic Publishers, Dordrecht (1997)
429. Tzafestas, S.G., Rigatos, G.G., Kyriannakis, E.J.: Geometry and Thermal Regulation of GMA Welding via Conventional and Neural Adaptive Control. Journal of Intelligent and Robotic Systems 19, 153–186 (1997)
430. Tzafestas, S.G., Poulos, P.N., Rigatos, G.G., Koukos, A.K.: A genetic algorithm for warehouse multi-objective optimization. In: Proceedingsof the MCPL 2000: Second

Conference on Management and Control of Production and Logistics, Grenoble, France (2000)
431. Tzeng, C.Y.: An Internal Model Control Approach to the Design of Yaw-Rate-Control Ship Steering Autopilot. IEEE Journal of Ocean Engineering 24(4), 507–513 (1999)
432. Tümer, M.B., Belfore, L.A., Ropella, K.M.: Applying hierarchical fuzzy automata to automatic diagnosis. In: Proceedings of Mtg. North America Fuzzy Information Process. Syst., Pensacola, FL (1998)
433. Tümer, M., Belfore, L., Ropella, K.: A Syntactic Methodology for Automatic Diagnosis by Analysis of Continuous Time Measurements Using Hierarchical Signal Representations. IEEE Trans. on Systems, Man and Cybernetics - Part B: Cybernetics 33(6), 951–965 (2005)
434. Turini, S.: Optimizations and placements with the genetic workbench. Research Report 96/4, Western Research Laboratory (1996)
435. Vercauteren, T., Wang, X.: Decentralized Sigma-Point Information Filters for Target Tracking in Collaborative Sensor Networks. IEEE Transactions on Signal Processing 53(8), 2997–3009 (2005)
436. Venayagamoorthy, G.K., Harley, R.G.: Implementation of an adaptive neural network identifier for effective control of turbogenerators. In: Proc. of the IEEE Budapest PowerTech Conference, paper BPT99-431-23 (1999)
437. Venayagamoorthy, G.K., Harley, R.G.: A Continually Online Trained Neurocontroller for Excitation and Turbine Control of a Turbogenerator. IEEE Transactions on Energy Conversion 16(3), 261–269 (2001)
438. Villagra, J., d'Andrea-Novel, B., Mounier, H., Pengov, M.: Flatness-based vehicle steering control strategy with SDRE feedback gains tuned via a sensitivity approach. IEEE Transactions on Control Systems Technology 15, 554–565 (2007)
439. Vissière, D., Chang, D.E., Petit, N.: Experiments of trajectory generation and obstacle avoidance for a UGV. In: Proc. ACC 2007, American Control Conference 2007, New York, pp. 2828–2835 (July 2007)
440. Vissière, D., Bristeau, P.-J., Martin, A.P., Petit, N.: Experimental autonomous flight of a small-scaled helicopter using accurate dynamics model and low-cost sensors. In: Proceedings of the 17th World Congress The International Federation of Automatic Control Seoul, Korea (July 2008)
441. Wai, R.J., Lee, M.C.: Intelligent optimal control of a single-link, flexible robot arm. IEEE Transactions on Industrial Electronics 51(1), 201–220 (2004)
442. Wai, R.J., Chang, J.M.: Intelligent control of induction servo motor drive via wavelet neural network. Electric Power Systems Research 61(1), 67–76 (2001)
443. Wai, R.J., Chang, H.H.: Backstepping Wavelet Neural Network Control for Indirect Field-Oriented Induction Motor Drive. IEEE Transactions on Neural Networks 15(2), 367–382 (2004)
444. Wai, R.J., Chang, J.M.: Implementation of Robust Wavelet-Neural-Network Sliding-Mode Control for Induction Servo Motor Drive. IEEE Transactions on Industrial Electronics 50(6), 1317–1334 (2003)
445. Wang, F.-Y., Gao, Y.: Advanced Studies of Flexible Robotic Manipulators. World Scientific, Singapore (2004)
446. Wang, L.X.: Adaptive Fuzzy Systems and Control: Design and Stability Analysis. Prentice Hall, Englewood Cliffs (1994)
447. Wang, L.X.: A course in fuzzy systems and control. Prentice-Hall, Englewood Cliffs (1998)
448. Watanabe, K., Tzafestas, S.G.: Learning algorithms for neural networks with the Kalman Filters. Journal of Intelligent and Robotic Systems 3, 305–319 (1990)

449. Watanabe, K., Tzafestas, S.G.: Filtering, Smoothing and Control in Discrete-Time Stochastic Distributed-Sensor Networks. In: Tzafestas, S.G., Watanabe, K. (eds.) Stochastic Large-Scale Engineering Systems, pp. 229–252. Marcel Dekker, New York (1992)
450. Weerasooriya, S., El-Sharkawi, M.A.: Identification and control of a DC motor using back-propagation neural networks. IEEE Transactions on Energy Conversion 6, 663–669 (1991)
451. Weerasooriya, S., El-Sharkawi, M.A.: Laboratory implementation of a neural network trajectory controller for a DC motor. IEEE Transactions on Energy Conversion 8, 107–113 (1993)
452. Wilson, D.H.: Managing oscillatory stability using online dynamics measurements. In: Proc. of the IEEE Power Systems Conference and Exposition, vol. 2, pp. 1101–1108 (2004)
453. Wu, J.K.: Neural Networks and Simulation Methods. Marcel Dekker, Inc., New York (1994)
454. Wu, J.C., Liu, T.S.: A sliding-mode approach to fuzzy control design. IEEE Transactions on Control Systems Technoogy 4(2), 141–151 (1996)
455. Yamashita, K., Joo, S.-K., Li, J., Zhang, P., Liu, C.C.: Analysis, control and economic impact assessment of major blackout events. European Transactions on Electrical Power 18, 854–871 (2008)
456. Yan, K.Q., Wang, S.C.: Reaching fault diagnosis agreement on an unreliable general network. Information Sciences 170(2-4), 397–407 (2005)
457. Yang, B., Asada, H.: Hybrid Linguistic/Numeric Control of Deburring Robots Based on Human Skills. In: Proc. 1992 Int. Conf. Robotics and Automation, pp. 1467–1474 (1992)
458. Yang, N., Tian, W.F., Jin, Z.H., Zhang, C.B.: Particle Filter for sensor fusion in a land vehicle navigation system. Measurement Science and Technology 16, 677–681 (2005)
459. Yang, Y., Zhou, C.: Adaptive Fuzzy H_∞ Stabilization for Strict-Feedback Canonical Nonlinear Systems Via Backstepping and Small-Gain Approach. IEEE Transactions on Fuzzy Systems 13(1) (2005)
460. Yang, Y., Ren, J.: Adaptive Fuzzy Robust Tracking Controller Design via Small Gain Approach and its Application. IEEE Transactons on Fuzzy Systems 11(6), 783–795 (2003)
461. Yang, S.S., Cheng, C.S.: An orthogonal neural network for function approximation. IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics 26, 779–784 (1996)
462. Yazdizadeh, A., Khorasani, K., Patel, R.V.: Identification of a two-link flexible manipulator using adaptive time delay neural networks. IEEE Transactions on Systems, Man and Cybernetics, Part B 30, 165–172 (2000)
463. Yen, J., Wang, L.: Simplifying Fuzzy Rule-Based Models Using Orthogonal Transformation Methods. IEEE Transactions on Systems, Man and Cybernetics - Part B 29, 13–24 (1999)
464. Ying, H., Zheng, X., Wulue, P.: A practical analysis method of low frequency oscillation for large power systems. In: Proc. of the Power Engineering Society General Meeting, vol. 2, pp. 1623–1629 (2005)
465. Yoshimoto, Y., Watanabe, K., Iwatani, Y., Hashimoto, K.: Multi-camera visual servoing of a micro helicopter under occlusions. In: Fung, R.-F. (ed.) Visual Servoing. InTech Publications (2010)
466. Yosihkawa, T., Sugie, T., Tanaka, M.: Dynamic Hybrid Position/Force Control of Robot Manipulators-Controller Design and Experiment. IEEE Journal of Robotics and Automation 4(6), 699–705 (1988)

467. Yutian, L., Chi, X., Yuanyuan, S.: Real-time phasor measurement for low frequency oscillation in power system. In: Proc. of the IEEE International Symposium on Circuits and Systems, vol. 4, pp. 3889–3893 (2005)
468. Zadeh, L.A.: Optimality and nonscalar-valued performance criteria. IEEE Transactions on Automatic Control 8, 59–60 (1963)
469. Zhang, Q.: Adaptive observer for Multiple-Input-Multiple-Output (MIMO) linear time-varying systems. IEEE Transactions on Automatic Control 47(3), 525–529 (2002)
470. Zhang, Q.: Fault detection and isolation with nonlinear black-box models. In: Proc. SYSID 1997, Kitakyushu, Japan (1997)
471. Zhang, Q., Benveniste, A.: Wavelet Networks. IEEE Transactions on Neural Networks 3(6), 869–898 (1993)
472. Zhang, Q., Campillo, F., Cérou, F., Legland, F.: Nonlinear fault detection and isolation based on bootstrap particle filters. In: Proc. of the 44th IEEE Conference on Decision and Control, and European Control Conference, Seville Spain (December 2005)
473. Zhang, Q., Basseville, M., Benveniste, A.: Fault Detection and Isolation in Nonlinear Dynamic Systems: A Combined Input-Output and Local Approach. Automatica 34(11), 1359–1373 (1998)
474. Zhang, Q., Basseville, M., Benveniste, A.: Early warning of slight changes in systems. Special Issue on Statistical Methods in Signal Processing and Control, Automatica 30(1), 95–113 (1994)
475. Zhang, Y., Hearn, G.E., Sen, P.: A Neural Network Approach to Ship Track-Keeping Control. IEEE Journal of Ocean Engineering 21(4), 513–527 (1996)
476. Zhang, S., Xie, X., Wu, J.: WAMS-based detection and early-warning of low-frequency oscillations in large-scale power systems'. Electric Power Systems Research 78(5), 897–906 (2008)
477. Zhao, Y., Bai, L., Gordon, B.W.: Distributed simulation and virtual reality visualization of multi-robot distributed receding horizon control systems. In: Proc. of the 2007 IEEE Intl. Conference on Robotics and Biomimetics, Sanya, China (Decemeber 2007)
478. Zhu, Q., Guo, L.: Stable Adaptive Neurocontrol for Nonlinear Discrete-Time Systems. IEEE Transactions on Neural Networks 15(3), 653–662 (2004)
479. Zitzler, E.: Evolutionary algorithms for multi-objective optimization:methods and applications. Ph.D. Thesis, Swiss Federal Institut of Technology, Zurich (1999)
480. Zuo, W., Zhu, Y., Cai, L.: Fourier-Neural-Network-Based Learning control for a class of nonlinear systems with flexible components. IEEE Transactions on Neural Networks 20, 139–151 (2009)
481. Zuo, W., Cai, L.: A new iterative learning controller using variable structure Fourier neural network. IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics 40(2), 458–468 (2010)

Index