



Hassan Yousefi

**On Modelling, System Identification and Control
of Servo-Systems with a Flexible Load**

*Thesis for the degree of Doctor of Science (Technology) to be
presented with due permission for public examination and
criticism in Auditorium 1382 at Lappeenranta University
of Technology, Lappeenranta, Finland, on 16th June, 2007,
at noon.*

Acta Universitatis
Lappeenrantaensis
269

Supervisor	Professor Heikki Handroos Department of Mechanical Engineering Lappeenranta University of Technology Finland
Reviewers	Professor Takao Nishiumi National Defense Academy Department of Mechanical Systems Eng. Japan
	Professor Saied Habibi Department of Mechanical Engineering McMaster University Canada
Opponents	Professor Saied Habibi Department of Mechanical Engineering McMaster University Canada
	Professor Takao Nishiumi National Defense Academy Department of Mechanical Systems Eng. Japan

ISBN 978-952-214-397-6
ISSN 1456-4491
Lappeenrannan teknillinen yliopisto
Digipaino 2007

To my loving wife, Azita, the most beautiful flower in the garden of my heart.

To my son, Daniel who is aged one. I dedicated this work to you to ensure a bright future for you.

Preface

The work presented in the thesis has been carried out at the Institute of Mechatronics and Virtual Engineering in the Department of Mechanical Engineering of Lappeenranta University of Technology, during the years 2004-2007.

I would like to express my deepest thanks to all the people who have influenced my work. First, I would like to express my appreciation for my supervisor, Professor Heikki Handroos, who did support me step by step during my study, research, and made this interesting research possible. I would like to express my gratitude to the personnel of Institute of Mechatronics and Virtual Engineering for their support in my research especially, Juha Koivisto, and Dr. Markus Hirvonen for his help in driving the govern equations of the linear motor, and designing the Kalman filter. I am also grateful of Dr. Samantha Kiljunen for her valuable work in proof-reading of the final manuscript.

I would like to thank the reviewers and opponents of the thesis, Professor Saied Habibi and Professor Takao Nishiumi for their valuable comments.

I wish to thank my parents for building the foundations of my education and my life and for all their devotion, and encouragement throughout my life. I would like to thank my sister Roya, and my brothers in law, Mansour and Amir. I would like to especially thank my loving wife, Azita for her continuous support. My son, Daniel and she are the source of my inspiration. Indeed, without their support in this endeavour, I may never have accomplished it.

Lappeenranta, May 2007

Hassan Yousefi

Abstract

Hassan Yousefi

On Modelling, System Identification and Control of Servo-Systems with a Flexible Load

Lappeenranta 2007

169 p.

Acta Universitatis Lappeenantaensis 269

Diss. Lappeenranta University of Technology

ISBN 978-952-214-397-6, ISSN 1456-4491

The present study was done with two different servo-systems. In the first system, a servo-hydraulic system was identified and then controlled by a fuzzy gain-scheduling controller. The second servo-system, an electro-magnetic linear motor in suppressing the mechanical vibration and position tracking of a reference model are studied by using a neural network and an adaptive backstepping controller respectively. Followings are some descriptions of research methods.

Electro Hydraulic Servo Systems (EHSS) are commonly used in industry. These kinds of systems are nonlinear in nature and their dynamic equations have several unknown parameters. System identification is a prerequisite to analysis of a dynamic system. One of the most promising novel evolutionary algorithms is the Differential Evolution (DE) for solving global optimization problems. In the study, the DE algorithm is proposed for handling nonlinear constraint functions with boundary limits of variables to find the best parameters of a servo-hydraulic system with flexible load. The DE guarantees fast speed convergence and accurate solutions regardless the initial conditions of parameters.

The control of hydraulic servo-systems has been the focus of intense research over the past decades. These kinds of systems are nonlinear in nature and generally difficult to control. Since changing system parameters using the same gains will cause overshoot or even loss of system stability. The highly non-linear behaviour of these devices makes them ideal subjects for applying different types of sophisticated controllers. The study is concerned with a second order model reference to positioning control of a flexible load servo-hydraulic system using fuzzy gain-scheduling. In the present research, to compensate the lack of damping in a

hydraulic system, an acceleration feedback was used. To compare the results, a p-controller with feed-forward acceleration and different gains in extension and retraction is used. The design procedure for the controller and experimental results are discussed. The results suggest that using the fuzzy gain-scheduling controller decrease the error of position reference tracking.

The second part of research was done on a Permanent Magnet Linear Synchronous Motor (PMLSM). In this study, a recurrent neural network compensator for suppressing mechanical vibration in PMLSM with a flexible load is studied. The linear motor is controlled by a conventional PI velocity controller, and the vibration of the flexible mechanism is suppressed by using a hybrid recurrent neural network. The differential evolution strategy and Kalman filter method are used to avoid the local minimum problem, and estimate the states of system respectively. The proposed control method is firstly designed by using non-linear simulation model built in Matlab Simulink and then implemented in practical test rig. The proposed method works satisfactorily and suppresses the vibration successfully.

In the last part of research, a nonlinear load control method is developed and implemented for a PMLSM with a flexible load. The purpose of the controller is to track a flexible load to the desired position reference as fast as possible and without awkward oscillation. The control method is based on an adaptive backstepping algorithm whose stability is ensured by the Lyapunov stability theorem. The states of the system needed in the controller are estimated by using the Kalman filter. The proposed controller is implemented and tested in a linear motor test drive and responses are presented.

Keywords: *Differential Evolution; System Identification; Servo-hydraulic System; Flexible load; fuzzy gain-scheduling, Linear Motor, Load Control, Neural Network, Adaptive Backstepping.*

UDC 681.5.033 : 681.511.4 : 004.032.26 : 681.58

Nomenclature

Roman letters

a_1	bulk modulus constant
a_2	bulk modulus constant
a_3	bulk modulus constant
A_1	piston area at chamber one
A_2	piston area at chamber two
a_{load}	acceleration of load
A_N	nominal current
A_{r1}	amplitude of 1st harmonic
A_{r2}	Amplitude of 2nd harmonic
b	damping coefficient
CR	crossover rate
c_v	flow coefficient
D	number of unknown parameters
e	error
E	total associated error of network output
E_i	associated error of network output
$F()$	cost function
$F_{coulomb}$	coulomb friction
F_{dist}	disturbance force
F_{dx}	electromagnetic thrust in the direction x
F_N	nominal force
F_{ripple}	force ripple
F_s	scaling factor
$F_{viscous}$	viscous force
F_μ	motor friction force
$f(v_k)$	activation function
G_{max}	maximum number of generation
H_j	input to the hidden layer unit j
i_{ad}	d-axis armature current
i_{aq}	q-axis armature current
i_D	d-axis damper winding current
i_Q	q-axis damper winding current
I_n	input to the unit n
k	spring constant
k_c	modification constant
K_i	I gain of vel. controller
K_p	P gain of vel. controller

K_s	scaling factor
L	stroke
L_{ad}	d-axis armature self-inductance
L_D	d-axis damper winding inductance
L_{md}	d-axis magnetizing inductance
L_{mq}	q-axis magnetizing inductance
L_Q	q-axis damper winding inductance
m_i	inlet mass
m_{ji}	mutation vector
m_L	load mass
m_M	motor mass
NP	population size
P	pressure
\bar{P}	a vector in population size
p_1	pressure in chamber one
p_2	pressure in chamber two
p_s	pressure supply
$P()$	penalty function
$P_{i(\max)}$	maximum range limit
$P_{i(\min)}$	minimum range limit
P_{\max}	maximum pressure
\bar{P}_{\max}	maximum range limit
\bar{P}_{\min}	minimum range limit
p_τ	tank pressure
Q	flow rate
Q_o	outlet flow rate
Q_1	flow rate at chamber one
Q_2	flow rate at chamber two
Q_i	inlet flow rate
Q_{Le1}	internal leakage at port one
Q_{Le2}	internal leakage at port two
Q_{Li}	internal leakage
R	armature winding resistance
R_D	d-axis damper winding resistance
R_P	Electric resistance
R_Q	q-axis damper winding resistance
T	temperature
T_s	sampling time
u	control effort signal

u_d	d-axis terminal voltage
u_q	q-axis terminal voltage
$u_{j,i,G+1}$	trail vector
U_n	output of the neural compensator
v	velocity
V_1	compressed volume
V_2	compressed volume
v_{ji}	hidden layer weight
v_k	summation of net
v_{load}	velocity of load
v_{ls}	synchronous velocity
v_{ref}	velocity reference
$V(x)$	Lyapunov function
w_{ji}	weight from neuron i to j
$W(x)$	positive definite function
$x_{i,G}$	parent vector
$x_{i,G+1}$	next generation
$x_{j,i}$	a member vector from population size
$x_{j,i}^{hi}$	upper bound
$x_{j,i}^{lo}$	lower bound
$x_{j,i,G}$	initial population member
$x_{j,r,G}$	randomly vector chosen from population size
X_l	load position
X_p	piston position
X_d	desire position
X_r	reference position
\dot{X}_p	piston velocity
X_s	spool position

Greek letters

α	momentum coefficient
β	bulk modulus
β_e	effective bulk modulus
ε	air percentage in fluid
φ_1	1st wavenumber
φ_2	2nd wavenumber
$\mu(x)$	membership parameter

θ	unknown parameter
$\hat{\theta}$	estimation of unknown parameter
v_0	dead volume
ρ	mass density
τ	pole pitch
τ_1	valve gain
τ_2^{-1}	valve time constant
ω_n	natural frequency
ω_s	electrical angular velocity
ψ_D	d-axis flux linkage of damping windings
ψ_{PM}	flux linkage of permanent magnet
ψ_q	q-axis flux linkage
ψ_Q	q-axis flux linkage of damping windings

Acronyms

ACC	American Control Conference
ANNs	Artificial Neural Network
SAAE	Summation of the Absolute Amount of Error
CLF	Control Lyapunov Function
DE	Differential Evolution
EHSS	Electro Hydraulic Servo System
FL	Fuzzy Control
FLC	Fuzzy Logic Controller
GA	Genetic Algorithm
MLP	Multi-Layer Perceptron
NNs	Neural Networks
PI	Proportional-Integral
PMLSM	Permanent Magnet Linear Synchronous Motor

Contents

Chapter 1

INRODUCTION

1.1 Problem definition.....	17
1.2 Previous Work.....	17
1.3 Contribution of the work.....	22
1.4 Organization of Thesis.....	24

Chapter 2

HYDRAULIC SYSTEMS

2.1 BACKGROUND.....	25
2.2 Servo-Hydraulic System.....	26
2.2.1 MASS DENSITY.....	26
2.2.2 BULK MODULUS.....	26
2.2.3 FLOW CONTINUITY EQUATION.....	27
2.2.4 FLOW THROUGH AN ORIFICE.....	28
2.2.5 PRESSURE RATES.....	28
2.2.6 SERVO-VALVE.....	29
2.2.7 System Dynamics.....	30
2.3 Model of PMLSM.....	32
2.3.1 Non-Idealities.....	34

Chapter 3

Differential Evolution Strategy

3.1 Background.....	37
3.2 Differential Evolution ALGORITHM.....	38
3.2.1: Initialization.....	39
3.2.2: Mutation.....	39
3.2.3: Crossover.....	40
3.2.4: Selection.....	41
3.2.5: THE OBJECTIVE FUNCTION.....	41

3.2.5.1: THE MEAN SQUARE ERROR (MSE).....	41
3.2.5.2: Constraints.....	41
3.2.5.3: PENALTY FUNCTION.....	41

Chapter 4

Fuzzy Control

4.1 Background.....	44
4.2 Conventional Sets.....	44
4.3 Fuzzy Sets.....	45
4.4 Fuzzy Controllers.....	45
4.4.1 Fuzzification.....	45
4.4.2 Rule Base.....	46
4.4.3 Inference Engine.....	46
4.4.4 Defuzzification.....	47

Chapter 5

Recurrent Neural Network

5.1 Background.....	48
5.2 Feedforward And Recurrent Networks.....	50
5.3 Backpropagation With Momentum Term.....	51
5.4 Modified Backpropagation.....	52

Chapter 6

Adaptive Backstepping

6.1 Background.....	58
6.2 Lyapunov Stability method.....	58
6.3 Adaptive Backstepping.....	59

Chapter 7

Results and Conclusion

7.1 DE-Identification.....	62
7.1.1 Results.....	62

7.1.2 Conclusions.....	66
7.2 Fuzzy Gain-Scheduling.....	67
7.2.1 Results.....	67
7.2.2 Conclusions	68
7.3 Neural network in PMLM.....	69
7.3.1 Results.....	69
7.3.2 Conclusions	70
7.4 Adaptive Backstepping in PMLM.....	71
7.4.1 Results.....	71
7.4.2 Conclusions	73
Bibliography.....	74
Publications 1-4.....	83

Chapter 1

INTRODUCTION

1.1 Problem Definition

The thesis covers several control aspects for two different electro-mechanical servo systems namely, a servo hydraulic system with a flexible load, and a permanent magnet linear motor with a flexible load.

The main aims of the present study are the system identification and control position tracking for the former, and to suppress mechanical vibration for velocity tracking, and control position tracking for the later.

There are varieties of methods to identify the unknown parameters of a system. Here, the differential evolution algorithm is applied to find the best values for unknown parameters of the servo hydraulic system. For position tracking in the servo hydraulic system, a fuzzy gain scheduling is used.

For the linear motor system, the PI-controller has poor performance in velocity tracking of a reference model, so the neural network compensator is proposed to suppress the mechanical vibration. A nonlinear load control method is developed and implemented for the linear motor system. The purpose of the controller is to track a flexible load to the desired position reference as fast as possible and without awkward oscillation. The control method is based on an adaptive backstepping algorithm whose stability is ensured by the Lyapunov stability theorem.

1.2 Previous Work

Servo-hydraulic systems have under-gone great development in the past decades. Standard textbooks such as those by [Merritt, 1967], McCloy and Martin [McCloy & Martin, 1973], and Viersma [Viersma, 1980] provide a thorough analysis of the basics of the servo-hydraulic technique. Servo-hydraulic system modelling can also be found in various other references like Blackburn [Blackburn et al., 1960] and Lewis and Stern [Lewis & Stern, 1962].

The main control component of a hydraulic system is the servo valve. Modelling of servo-valves can be found in the literature. Merritt's [Merritt, 1967] work on spool valves and flapper nozzle valves are mostly design oriented. Lin and Akers [Lin & Akers, 1991] developed a nonlinear model of the first stage of a servo valve. Shukla [Shukla, 2002] proposed a reduced order model of a servo valve that is decomposed into static nonlinear part and a dynamic linear part.

Goodson and Leonard [Goodson & Leonard, 1972] presented a clear overview of different representations of the transmission line models. Krus [Krus et al., 1994] developed simple methods that capture all essential behaviour of the line dynamic such as the time delay and the distributed frequency-dependent friction. Work on theoretical modelling as been covered by many researchers. The main contributions are by [Merritt, 1967] and Viersma [Viersma, 1980]. Watton and Braton [Watton & Braton, 1985] examined the hydraulic actuation with unequal areas. This completes the overview of the research done on the modelling of a servo-hydraulic system.

A wide range of control design techniques ranging from linear control to nonlinear control techniques have been used in the past. In general, hydraulic control problems are classified as position control [Viersma, 1980], rotary velocity control [Merritt, 1967] and linear velocity control [Shukla, 2002] and as a force control problem [Alleyne, 1996]. Linear control analysis such as classical feedback control [Viersma, 1980] is widely applied. Nonlinear techniques such as adaptive control [Yao et al., 1997], self tuning regulators [Yao et al., 1997] and the feedback linearised technique [Vossoughi & Donath, 1995] are also used for control of hydraulic systems.

Feedback linearisation [Slotine, & Li, 1990] involves a nonlinear control strategy that cancels out the nonlinearities of the plant in open loop. Feedback linearisation was applied to hydraulic systems by Axelson and Kumar [Axelson & Kumar, 1988]. Their approach accounted only for the nonlinear pressure-gain characteristic. Donath and Vossoughi [Vossoughi & Donath, 1995] in their work on feedback linearisation as applied to hydraulic systems included asymmetric actuation and variations in the trapped fluid volume. Their work required the use of velocity feedback which requires costly sensors. Kugi [Kugi & Schlacher, 2002] developed a feedback linearised controller based on state transformation that does not require velocity information.

Kremer and Thompson [Kremer & Thompson, 1998] developed a bifurcation based procedure for analysing stability of nonlinear hydraulic systems. Their work uses the concept of the shortest distance to instability to investigate the robust stability of hydraulic models. Shukla [Shukla, 2002] performed stability analysis of control induced separation in the extended parameter space.

One of the most promising novel evolutionary algorithms is the Differential Evolution (DE) algorithm for solving global optimisation problems with continuous parameters. The DE was first introduced by Storn [Storn & Price, 1995] and Schwefel [Schwefel, 1995].

The extensive application areas of DE are testimony to the simplicity and robustness that have fostered their widespread acceptance and rapid growth in the research community. In 1998, DE was mostly applied to scientific applications involving curve fitting, for example fitting a non-linear function to photoemissions data [Cafolla, 1998]. DE enthusiasts then hybridised it with Neural Networks and Fuzzy Logic [Schmitz & Aldrich, 1998] to enhance or extend its performance. In 1999 DE was applied to problems involving multiple criteria as a spreadsheet solver application [Bergey, 1999]. New areas of interest also emerged, such as: heat transfer [Babu & Sastry, 1999], and constraint satisfaction problems [Storn, 1999]. In 2000, the popularity of DE continued to grow in areas of electrical power distribution [Chang & Chang, 2000], and magnetics [Stumberger et al., 2000]. 2001 furthered extensions of DE in areas of environmental science [Booty et al., 2000], and linear system models [Cheng, 2001]. By the year 2002, DE penetrated the field of medical science [Abbass, 2002]. In 2003, there has been a resurgence of interest in applying DE to problems involving multiple criteria [Babu, 2003]. It was in 2004, DE was applied in noisy optimisation problems [Krink et al., 2004]. In 2005, DE as an easy and efficient evolutionary algorithm for model optimisation was introduced [Mayer et al., 2005]. Most recently in 2006, DE was applied to efficient parameter estimation in bio-filter modelling [Babu & Angira, 2006].

The field of fuzzy control developed by Lotfi Zadeh [Zadeh, 1965]. He believed that control theory was becoming increasingly complex and mathematical, while neglecting the importance of practicality and applicability to real-world control problems. He suggested that his idea of fuzzy set theory, developed in 1965 [Zadeh,

1965], showed that what he called 'fuzzy sets' were the foundation of any logic, regardless of the number of truth levels. The first implementation in control systems was accomplished by Mamdani in 1974 who demonstrated the viability of Fuzzy Logic (FL) for a small model steam engine [Mamdani, 1974]. The mathematical foundation of the FL control was provided by Wang who showed that fuzzy systems are universal approximators [Wang, 1994]. One of the most complex systems in which fuzzy logic has been successfully applied is cement kilns [Jamshidi et al., 1993].

In recent years, FL control has become an important approach in designing nonlinear controllers because of its simplicity, ease of design and ease of implementation using commercially available programming tools [Bartos, 1996]. The control of knowledge based systems using linguistic variables that do not have precise values is of concern, and this allows the use of traditional human heuristic and experience in designing the systems [Ahmed et al., 2001]. The method provides a man-machine interface that facilitates the acceptance, validation and transparency of the control algorithm, and it has some advantages over many other approaches when the system description requires certain human experience [Ahmed et al., 2001], [Chen & Chang, 1998], [Akkizidis et al., 2000], [Ghiaus, 2000] and [Lin & Yang, 2003]. Compared with conventional control approaches, FL control utilises more information from experts and relies less on mathematical modelling about a physical system. It is also preferable, especially when low cost and easy operations are involved. The advantages of FL control over other applicable techniques have been given by many researchers [Verbruggen & Bruijn, 1997], [Akkizidis, et al., 2000], [Antonio & Pacifico, 2000] and [Wai et al., 2004].

Many industrial applications using fuzzy technology have been developed for a furnace temperature control problem [Radakovic et al., 2002], a wind energy problem [Mohamed et al., 2001], power system stability [El-Sherbiny et al., 2002], biological processes [Horiuchi & Kishimoto, 2002], a jet engine fuel system [Zilouchian et al., 2000], flatness control of hot strip mill [Zhu et al., 2003], electro-hydraulic fin actuator [Lee & Cho, 2003], traffic junction signals [Chou & Teng, 2002] and robot control [Akkizidis, et al., 2000], [Kuo & Lin, 2002]. More and more industrial applications and commercial products of FL control are appearing every year, and typical applications

already in use are process control, automobiles, battery chargers, subway systems, motorway tunnel air conditioning, washing machines and many more [Zhang & Chang, 2003] and [Fraichard & Garnier, 2001].

Applications of the neural network to adaptive control of nonlinear systems have been intensively conducted in recent years. For example, Narendra and Parthasarathy [Narendra & Parthasarathy, 1990] introduced multilayer neural networks for identification and adaptive control of nonlinear systems. A number of studies such as [Chen & Khalil, 1992 and Chen, & Khalil, 1995] for adaptive control of unknown feedback linearisation systems and [Lewis et al., 1995 and Lewis et al., 1996] for achieving guaranteed performance of the neural net controller have been reported. This is due to the fact that the neural network has excellent capabilities of nonlinear mapping, learning ability, and parallel computations. Yabuta and Yamada [Yabuta & Yamada, 1992] proposed the adaptive neural control that replaces a conventional feedback controller with a neural network. Also, they discussed the stability of the linear discrete-time single-input-single-output (SISO) plant [Yamada & Yabuta, 1991]. Although their method is quite simple and can be applied to various feedback control systems, the uncertainty of the controlled plant cannot be identified and some parameters included in the neural network is quite difficult to be set. Their idea is to compensate the control input computed from the conventional feedback controller for canceling the effect of the plant uncertainties. Also, Carelli et al. [Carelli et al., 1995] proposed an adaptive controller using feedback error learning. Moreover, Akhyar and Omatsu [Akhyar & Omatsu, 1992] and Khalid et al. [Khalid & Omatu, 1995] presented a self-tuning controller that uses a set of neural networks for regulating the gains of the conventional feedback controller in order to improve the performance of the control system. Their methods could maintain stability of the adaptive control system through the function of the conventional feedback controller. The neural network can identify the uncertainties included in the controlled plant and can adaptively modify the control input computed from a pre-designed conventional feedback controller at the same time. Another approach to the neural adaptive control is to utilize multiple neural networks [Iiguni et al., 1991, Ku & Lee, 1995, Levin & Narendra, 1996, Ploycarpou & Helmicki, 1995, Rovithakis & Christodoulou, 1994 and Rovithakis & Christodoulou, 1995]. In this approach, one neural network is

dedicated to the forward model for identifying the uncertainties of the controlled plant and the other neural networks may compensate for the effect of the uncertainties based on the trained forward model. However, multiple neural networks must be trained and stability of this approach is quite difficult to assure. Different structures of neural controllers for control of nonlinear plants, especially induction motor drives have been presented [Branštetter & Skotnica, 2000, Burton et al., 1999 and Wishart & Harley, 1995].

Permanent magnet synchronous motors (PMSM) are receiving increased attention for electric drive applications due to their high power density, large torque to inertia ratio and high efficiency over other kinds of motors such as DC motors (Leonhard, 1995). But the dynamic model of a PMSM is highly nonlinear because of the coupling between the motor speed and the electrical quantities.

In order to deal with this problem, sliding-mode variable structure control approach has been used due to its favourable advantages such as insensitive to parameter uncertainties and external disturbances and only the bounds of the uncertainties are needed in design procedure [Slotine & Li, 1991, Hung et al., 1993 and Zhang & Panda, 1999].

Backstepping control is a newly developed technique to the control of uncertain nonlinear systems, particularly those systems that do not satisfy matching conditions [Krstic et al., 1995]. The most appealing point of it is to use the virtual control variable to make the original high order system to be simple enough thus the final control outputs can be derived step by step through suitable Lyapunov functions. A nonlinear torque controller for an induction motor was designed based on adaptive backstepping approach [Shieh & Shyu, 1999]. Adaptive nonlinear velocity controller for a flexible mechanism of a linear motor implemented by [Hirvonen, 2006].

1.3 Contribution of the work

The work presented in this thesis is mainly based on the following articles in journals and conference proceedings:

- Yousefi, H., Handroos, H.,” Application of Differential Evolution in System Identification of a Servo-Hydraulic System with a Flexible Load”, submitted to the Journal of Mechatronics.
- Yousefi, H., Handroos, H., Mattila, J.K., Application of Fuzzy Gain-Scheduling in Position Control of a Servo Hydraulic System with a Flexible Load, *Accepted at International Journal of Fluid Power.*
- Yousefi, H., Hirvonen, M. and Handroos, H., “Application of Neural Network in Suppressing Mechanical Vibrations of a Permanent Magnet Linear Motor”, Accepted at Journal of Control Engineering practice.
- Yousefi, H., Hirvonen, M. and Handroos, H. Adaptive backstepping method in Position Control of a Permanent Magnet Linear Motor with a Flexible Load, *submitted to the IET Control Theory & Applications.*
- Yousefi, H., Handroos, H., “Experimental and simulation study on control of a flexible servo-hydraulic system using adaptive neural network and differential evolution strategy”, 8th Biennial ASME Conference on Engineering Systems Design and Analysis, ESDA 2006, Torino, Italy, July, 2006.
- Yousefi, H., Handroos, H., ”Adaptive neural network in position control of a flexible servo-hydraulic system with fuzzy gain scheduling”, sixth SIAM Conference on Control and its Applications, New Orleans, USA, 2005.
- Yousefi, H., Handroos, H., “Adaptive Neural Network in Compensation the Dynamics and Position Control of a Servo-hydraulic System with a Flexible Load”, ASME International Mechanical Engineering Congress & Exposition, Orlando, Florida, USA, November, 2005.

The above articles were produced under the supervision of Prof. Heikki Handroos. In the third and fourth articles, Dr. Markus Hirvonen, acted as a co-author and specialist

in the electrical part of the Permanent Magnet Linear Synchronous Motor (PMLSM) system, driving the govern equations, and Kalman filtering.

The following contributions made by the thesis can be highlighted:

- System identification of a servo-hydraulic system with flexible load.
- Design and implementation of adaptive backstepping control for position tracking of a flexible mechanism attached to a PMLSM.
- Design, modification and Implementation of a neural network hybrid controller for a flexible mechanism attached to a PMLSM.
- Applying differential evolution algorithm to find global minima for the initial weights in a neural network-based controller.
- Design and implementation of fuzzy gain-scheduling in a servo-hydraulic system.
- Design and implementation of a neural network for a servo hydraulic system.

1.4 Organisation of the Thesis

This thesis is divided into 7 chapters. Chapter 2 covers the govern equations for the two servo systems. Chapter 3 concentrates on the *Differential Evolution* (DE) strategy. It covers the original method to minimise the cost function and define the penalty function. In Chapter 4, a brief description of the fuzzy controller is presented. Chapter 5 describes the standard backpropagation algorithm with a momentum term. Modification of the algorithm, that makes it suitable for any compensation application when there is no target value for the output of the neural network, is presented. In Chapter 6, the adaptive backstepping algorithm is described. Chapter 7 summarises the results and conclusions of the four attached journal papers.

Chapter 2

SERVO SYSTEMS

2.1 BACKGROUND

Use of pressurised fluid to transmit and control energy is the basis of fluid power systems. Its' roots can be traced back to Greek and Roman civilisations. The Industrial Revolution saw new inventions that are the basics of the present day servomechanisms. The second half of the twentieth century saw major developments in the design of servo valves. Owing to the evolution in hardware, automatic control theory and signal transmission, the application of servo hydraulic systems has grown rapidly.

Though the hydraulic power systems have notable drawbacks, they still find use in a wide range of industrial applications. Electro-hydraulic systems are used in the machine tool industry, milling machines, automobiles, punching presses, etc.

Electro Hydraulic Servo Systems (EHSS) have been used in industry in a wide number of applications due to their small size to power ratio, the ability to apply a very large force and control accuracy. However, the dynamics of hydraulic systems are highly nonlinear; the system may be subjected to non-smooth nonlinearities due to control input saturation, directional change of valve opening, friction, and valve overlap. Aside from the nonlinear nature of hydraulic dynamics, EHSS also have large extent of model uncertainties, such as the external disturbances and leakages that cannot be modelled exactly; and the nonlinear functions that describe them may not be known. While accurate modelling involving all the nonlinearities is helpful in identifying the complex dynamic behaviour, it complicates the hydraulic system model for any nonlinear control strategy.

Industry's growing need for faster development times and more accurate manufacturing is placing new demands on the technology used in automation. Systems are becoming more and more complex, consisting of mechanisms, actuators, controllers and sensors. Even simple components will contain sensors which are used as a feedback for their controllers.

Conventional have been superseding by new forms of motor and drive technology. In particular, permanent magnet direct drive motors are becoming more and more popular in machine automation nowadays.

The advantages of permanent magnet motor drives are their gearless structure, better control characteristics and better efficiency. Permanent magnet motors are used in lifts, paper machines, propulsion units of ships, windmills *etc.*

The chapter covers the govern equations for two different electro-mechanical servo systems namely, a servo hydraulic system with a flexible load, and a permanent magnet linear motor with a flexible load. In section 2.2, an overview of the fluid properties and govern equations of the servo-hydraulic system are provided. By using the basic laws of fluid flow, and the Newton's second law the equations describing the servo-hydraulic system are obtained. Section 2.3 demonstrates the dynamics of a permanent magnet linear motor.

2.2 SERVO-HYDRAULIC SYSTEM

Some of the important physical properties of fluids and govern equations of the servo-hydraulic system are presented here.

2.2.1 MASS DENSITY

Mass density, defined as the mass per unit of volume is an important property of fluids. It is represented by (ρ). In the *SI* system it has units of kg/m^3 . The density of a liquid varies with temperature and pressure. Since the variation of density with temperature and pressure is small, Taylor's series approximation can be used.

$$\rho(P,T) = \rho_o + \left(\frac{\partial\rho}{\partial P}\right)_T (P - P_o) + \left(\frac{\partial\rho}{\partial T}\right)_P (T - T_o) \quad (2.1)$$

where ρ , P and T are the mass density, pressure and temperature respectively of the liquid about initial values of ρ_o , P_o , T_o .

2.2.2 BULK MODULUS

The quantity β is called the *isothermal bulk modulus*. It is defined as the ratio of change in pressure to the fractional change in volume at constant temperature. It has the units of pressure (N/m^2 , in *SI* system). The effective bulk modulus varies with pressure. It is an important property in determining the dynamic properties of the

hydraulic system. The bulk modulus varies with fluid pressure, temperature, air content, and rigidity of the container and hoses.

It is reasonable to assume the container to be rigid and the effect of temperature on bulk modulus negligible. A theoretical equation for effective bulk modulus as given by [Merritt, 1967] is

$$\frac{1}{\beta_e} = \frac{1}{\beta_{hose}} + \frac{1}{\beta_{air\ free}} + \frac{\varepsilon}{1.4(P_{chamber} + P_{atmosphere})} \quad (2.2)$$

where ε is the percentage of air in the fluid. A small amount of entrapped air can drastically reduce the bulk modulus. Experimentally it is possible to determine the effective bulk modulus as a function of pressure and entrained air percentage.

There are lots of empirical formulas for the effective bulk modulus, including the effects of entrained air and mechanical compliance, based on direct measurements. The Commonly used equation for effective bulk modulus, β_e in hydraulic cylinder is [Jalali and Kroll, 2004],

$$\beta_e = a_1 E_{max} \log \left(a_2 \frac{p}{p_{max}} + a_3 \right) \quad (2.3)$$

where, the constant parameters are $E_{max} = 1.8e9$ (Pa) and $p_{max} = 2.8$ (MPa). The other parameters, a_1 , a_2 and a_3 , must be identified.

The fundamental laws and equations which govern the flow of fluids are described in the section. Only the equations relevant for study of hydraulic systems are presented.

2.2.3 FLOW CONTINUITY EQUATION

Consider the fluid volume V_o as shown in Figure 2.1, with inlet mass flow rate of m_i and outlet mass flow rate of m_o . Using the law of conservation of mass, the rate at which the mass of fluid is accumulated is equal to the input flow rate minus the output flow rate.

$$\rho_i Q_i - \rho_o Q_o = \frac{d}{dt}(\rho V) \quad (2.4)$$

where Q_i and Q_o are the inlet and outlet flows.

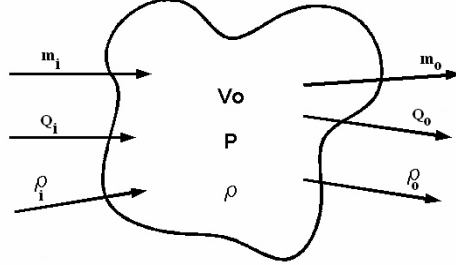


Fig. 2.1: Generalized flow volume

If the density of the fluid is assumed to be constant, the above equation can be rewritten as

$$Q_i - Q_o = \frac{dV}{dt} + \frac{V}{\rho} \frac{d\rho}{dt} \quad (2.5)$$

Using the following equation of bulk modulus [Jalali and Kroll, 2004]

$$\frac{d\rho}{\rho} = \frac{dP}{\beta_e} \quad (2.6)$$

we obtain the flow continuity equation.

$$Q_i - Q_o = \frac{dV}{dt} + \frac{V}{\beta_e} \frac{dP}{dt} \quad (2.7)$$

The first term on the right hand side is due to the boundary deformation and the second term is due to the fluid compressibility term.

2.2.4 FLOW THROUGH AN ORIFICE

Due to high flow rates in the hydraulic systems, most of the orifice flows are turbulent in nature. Using Bernoulli's equation and continuity equation, the flow through an orifice is obtained as [Munson et al., 1996]

$$Q = C_d A_o \sqrt{\frac{2}{\rho} (P_1 - P_2)} \quad (2.8)$$

where C_d is called the discharge coefficient. It has been assumed that the orifices used in hydraulic system are sharp edged.

2.2.5 PRESSURE RATES

Applying the continuity equation (Eq. 2.7) for the servo valve chambers and to the volume between the nozzles and outlet orifice yields [Jalali and Kroll, 2004],

$$\begin{cases} \dot{p}_1 = \frac{\beta_{e1}}{V_1}(Q_1 - A_1\dot{X}_p) \\ \dot{p}_2 = \frac{\beta_{e2}}{V_2}(-Q_2 + A_2\dot{X}_p) \end{cases} \quad (2.9)$$

where \dot{X}_p is the piston velocity, A_1 and A_2 are the piston areas. The nonlinear equations of valve flows are usually described by the orifice equations with a linear relationship between the valve spool position, X_s and pressures as follows [Merritt, 1967],

$$Q_1 = \begin{cases} c_v X_s \sqrt{p_s - p_1} & u \geq 0 \\ c_v X_s \sqrt{p_1 - p_t} & u < 0 \end{cases} \quad (2.10)$$

$$Q_2 = \begin{cases} c_v X_s \sqrt{p_2 - p_t} & u \geq 0 \\ c_v X_s \sqrt{p_s - p_2} & u < 0 \end{cases} \quad (2.11)$$

The above equations of flows are nonlinear.

2.2.6 SERVO-VALVE

The servo-valve is an important control element in a hydraulic system. The schematic diagram of an under study servo-valve is shown in Figure 2.2. The servo valve consists of a torque motor, flapper nozzle, and a valve spool.

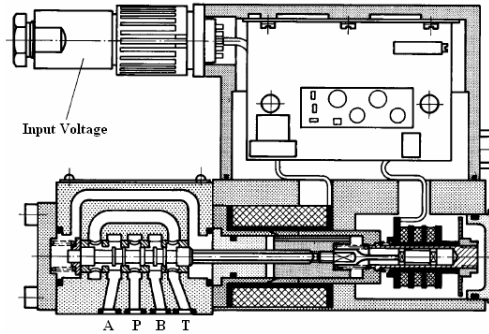


Fig. 2.2: Schematic of a servo valve with onboard electronics

In order to represent servo-valve approximation dynamics through a wider frequency range, a first or second order transfer function is used. For low frequencies (up to about 50 Hz), a first order approximation may be sufficient [Zeb, 2003, Avilaet al.,

2004]. The relation between servo valve spool position X_s and the input voltage u can be written as,

$$G_v(s) = \frac{X_s}{u} = \frac{\tau_1}{s + \tau_2} \quad (2.12)$$

where, τ_1 is the valve gain and τ_2^{-1} is the valve time constant.

2.2.7 SYSTEM DYNAMICS

The servo hydraulic system with a flexible load is comprised of a servo-valve, a hydraulic cylinder and two masses that are connected by a parallel combination of a spring and damper. The schematic diagram of the system is illustrated in Figure 2.3.

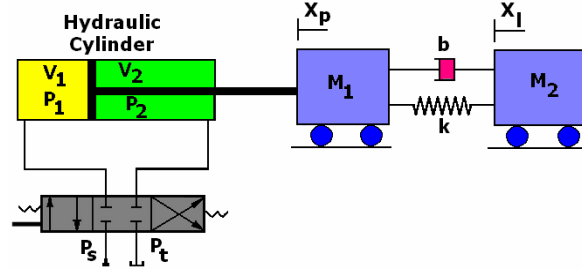


Fig. 2.3: Schematic diagram of system.

Applying Newton's second law to each mass, taking friction into account yields,

$$m_1 \ddot{X}_p = -b(\dot{X}_p - \dot{X}_L) - k(X_p - X_L) + p_1 A_1 - p_2 A_2 - F_f, \quad (2.13)$$

$$m_2 \ddot{X}_L = -b(\dot{X}_L - \dot{X}_p) - k(X_L - X_p), \quad (2.14)$$

where, m_1 and m_2 are the first and second masses respectively, b is the damping factor, k is the spring constant and X_p (X_L) is the position of piston (mass load). Friction in the hydraulic cylinder is taken into account as an external disturbance (F_f). Contact between two surfaces occurs at surface asperities (microscopic roughness) [Owen et al, 2003]. Due to the tight sealing, hydraulic cylinders feature a strong dry friction effect. The behaviour of this friction force is rather complex [Lischinsky et al, 1999; Olsson et al, 1998]. Friction is usually modelled as a discontinuous static mapping between the velocity and the friction force that depends on the velocity's sign. It is often restricted to the Coulomb and viscous friction components. However, there are several important properties observed in systems with friction which cannot

be explained by static models only. This is basically due to the fact that the friction does not have an instantaneous response to a change in velocity, i.e. it possesses internal dynamics. Examples of these complex properties are: (1) stick-slip motion, characterised by large friction at rest and on low velocities, and small friction during rapid motion; (2) pre-sliding displacement, which shows that the friction behaves like a spring when the applied force is less than the static friction break-away force; (3) friction lag, which means that there is a hysteresis that characterises the relationship between the friction and velocity. All these static and dynamic characteristics of the friction are captured by the analytic model of friction dynamics proposed in [Wit et al, 1995], which is called the LuGre model and defined by,

$$\frac{dz}{dt} = \dot{X}_p - \frac{|\dot{X}_p|}{g(\dot{X}_p)} z, \quad (2.15)$$

$$g(\dot{X}_p) = \frac{1}{\sigma_0} \left(F_c + (F_s - F_c) e^{-\left(\frac{\dot{X}_p}{v_s}\right)^2} \right), \quad (2.16)$$

$$F_f = \sigma_0 z + \sigma_1 \frac{dz}{dt} + k_v \dot{X}_p, \quad (2.17)$$

where \dot{X}_p is the piston velocity, and F_f is the friction force described by a linear combination of z , dz/dt and viscous friction. Equation (6) represents the dynamics of the friction where the internal state z , is not measurable. The function $g(\dot{X}_p)$ describes part of the "steady state" characteristics of the model for constant velocity motions: v_s is the Stribeck velocity, F_s is the static friction, F_c is the Coloumb friction, k_v is the viscous friction. Thus, the complete friction model is characterised by four static parameters and two dynamic parameters, a stiffness coefficient and a damping coefficient.

The friction parameters are difficult to estimate since they appear nonlinearly in the model and the average deflection of the bristles cannot be measured [Wit et al, 1995].

The nonlinear equations of the system are used for designing the controllers in simulation mode. Table 1 shows the known parameters of the system, so the other parameters, 16 parameters, are unknown. To estimate the unknown parameters

differential evolution is used [more details in publication I].

Table 1- known Parameters.

$m_1 = 210 \text{ Kg}$	$m_2 = 80 \text{ Kg}$
$A_1 = 8.04 \times 10^{-4} \text{ m}^2$	$A_2 = 4.24 \times 10^{-4} \text{ m}^2$
$v_{01} = 2.13 \times 10^{-4} \text{ m}^3$	$v_{02} = 1.07 \times 10^{-4} \text{ m}^3$
$k = 42950 \text{ N/m}$	$L = 1 \text{ m}$
$P_t = 0.9 \text{ MPa}$	$P_s = 14 \text{ MPa}$

2.3 MODEL OF PMLSM

This section introduces a nonlinear model for a permanent magnet linear synchronous motor (PMLSM). Figure 2.4 shows the schematic diagram of PMLSM. The moving part (the mover) consists of a slotted armature, while the surface permanent magnets are mounted along the whole length of the path (the stator). The moving part is set up on an aluminium base with four recirculating roller bearing blocks on steel rails. The position of the linear motor was measured using an optical linear encoder with a resolution of approximately one micrometer.

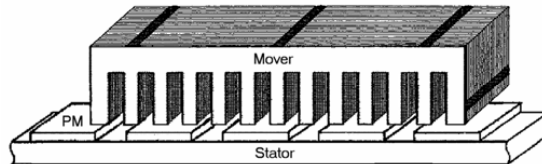


Fig. 2.4: The principle of PMLSM.

The modelling of the dynamics of the linear synchronous motor is presented [Hirvonen, 2006]. The time-varying parameters are eliminated and all the variables are expressed on orthogonal or mutually decoupled direct and quadrature axes, which move at a synchronous speed of ω_s . The d- and q-axes equivalent to the circuit of the PMLSM are shown in Figure 2.5, and the corresponding equations are (2.18) and (2.19), respectively.

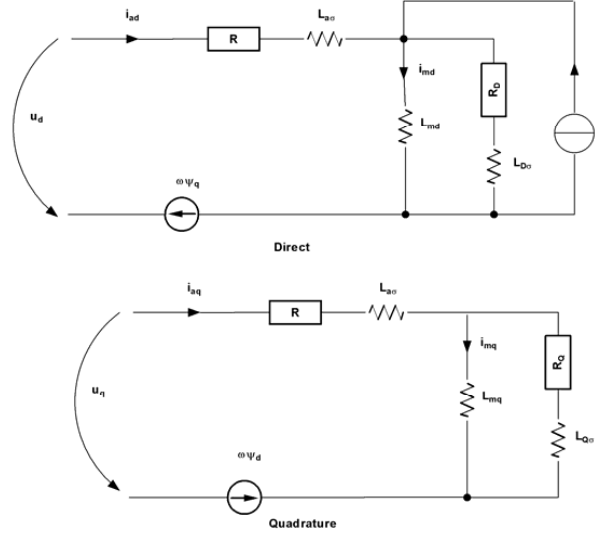


Fig. 2.5: Two-axial model of the linear synchronous motor.

The voltage equations for the synchronous machines are

$$u_d = Ri_{ad} + \frac{d\psi_d}{dt} - \omega_s \psi_q, \quad (2.18)$$

$$u_q = Ri_{aq} + \frac{d\psi_q}{dt} + \omega_s \psi_d, \quad (2.19)$$

where u_d and u_q are the d- and q-axis components of the terminal voltage, i_{ad} and i_{aq} the d- and q-axis components of the armature current, R is the armature winding resistance and ψ_d , ψ_q are the d- and q-axis flux linkage components of the armature windings. The synchronous speed can be expressed as $\omega_s = \pi v_{ls} / \tau$, where v_{ls} is the linear synchronous velocity, i.e. the velocity in which the motor moves, and τ the pole pitch, i.e. the length of the pole pair of permanent magnets. Although the physical system does not contain a damper, which in PMLSM usually takes the form of an aluminium cover on the PMs, virtual damping must be included in the model due to eddy currents. The voltage equations of the short-circuited damper winding are

$$0 = R_D i_D + \frac{d\psi_D}{dt}, \quad (2.20)$$

$$0 = R_Q i_Q + \frac{d\psi_Q}{dt}, \quad (2.21)$$

where R_D and R_Q are the d- and q-axis components of the damper winding resistance and i_D and i_Q the d- and q-axis components of the damper winding current. The armature and damper winding flux linkages in the above equations are

$$\psi_d = L_{ad}i_{ad} + L_{md}i_D + \psi_{pm}, \quad (2.22)$$

$$\psi_q = L_{aq}i_{aq} + L_{mq}i_Q, \quad (2.23)$$

$$\psi_D = L_{md}i_{ad} + L_D i_D + \psi_{pm}, \quad (2.24)$$

$$\psi_Q = L_{mq}i_{aq} + L_Q i_Q, \quad (2.25)$$

where L_{ad} and L_{aq} are the d- and q-axis components of the armature self-inductance, L_D and L_Q the d- and q-axis components of the damper winding inductance, L_{md} and L_{mq} the d- and q-axis components of the magnetising inductance and ψ_{pm} the flux linkage per phase of the permanent magnet. By solving the flux linkage differential equations from (2.18) to (2.21) and substituting the current equations from (2.22) to (2.25) into these equations, the equations for the simulation model of the linear motor can be derived. The electromagnetic thrust of a PMLSM is

$$F_{dx} = \frac{p_e}{v_{ls}} = \frac{3p}{2} \frac{\pi}{\tau} (\psi_d i_{aq} - \psi_q i_{ad}), \quad (2.26)$$

where p_e is the electromechanical power and p represents the number of pole-pairs.

2.3.1 NON-IDEALITIES

The force ripple of the PMLSM is larger than that of rotary motors because of the finite length of the stator or mover and the wide slot opening. In the PMLSM, the thrust ripple is caused mainly by the detent force generated between the PMs and the armature. This type of force can be divided into two components: tooth and core-type detent force. The core-type detent force can be efficiently reduced by optimising the length of the moving part or smooth-forming the edges of the mover and the tooth-type detent force can be reduced by skewing the magnets and chamfering the edges of the teeth [Hyun et al., 1999, and Jung and Jung, 2002].

The ripple of the detent force produces both vibration and noise and reduces controllability [Chun et al., 2000]. The force ripple is dominant at low velocities and accelerations. At higher velocities, the cogging force is relatively small and the

influence of dynamic effects (acceleration and deceleration) is more dominant [Ottensmeyer et al., 1997].

The force ripple can be described by sinusoidal functions of the load position, x , with a period of φ and an amplitude of A_r , i.e.

$$F_{ripple} = K_s A_{r1} \sin(\varphi_1 x) [A_{r1} + A_{r2} \sin(\varphi_2 x)], \quad (2.27)$$

where K_s is a scaling factor.

The model also takes into account the effect of friction. Friction is highly nonlinear and may result in steady-state errors, limit cycles and poor performance [Olsson et al., 1998]. The friction model took into account the Coulomb (static) and viscous (dynamic) components

$$F_{\mu} = \text{sign}(v) [F_{coulomb} + \text{abs}(v) F_{viscous}], \quad (2.28)$$

where v is the velocity of the motor.

The main disadvantage when using highly nonlinear models for simulations or control purposes is high discontinuity at near-zero speed, which gives rise to problems such as numerical chatter. The total disturbance force equation can be described using the equations of detent force and friction force; i.e. the disturbance force F_{dist} is

$$F_{dist} = F_{ripple} + F_{\mu}, \quad (2.29)$$

This resultant disturbance force component is added to the electromotive force to influence the dynamical behaviour of the linear motor system.

The effect of load variation was also taken into consideration. The mechanism in this study is made according to the ACC Benchmark Problem [Wie, and Bernstein, 1990], i.e. a two-mass model, Figure 2.6. The ACC Benchmark Problem consists of two masses attached by a single spring, moving on a friction-free horizontal surface.

The control input is the force applied to the first mass, and the output is the position or velocity of the second mass.

The motion equations of this system are:

$$\begin{aligned} m_M \ddot{x}_M &= k\Delta x + b\Delta \dot{x} + F_T \\ m_L \ddot{x}_L &= -k\Delta x - b\Delta \dot{x} \end{aligned}, \quad (2.30)$$

where k is the spring constant, b is the damping factor, m_M and m_L are motor and load masses, F_T is the summation of the controller force, F , and the disturbance

force, F_{dist} , x_M and x_L are motor and load positions and Δx and $\Delta \dot{x}$ are compression of the spring and the velocity difference between masses, respectively. In Table 2, the system parameters are presented.

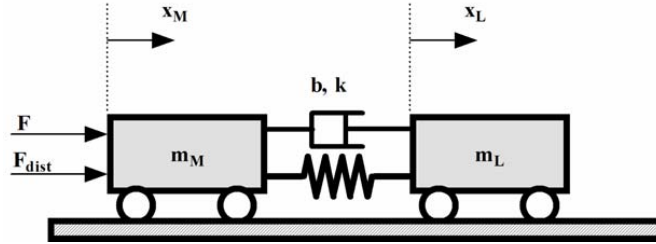


Fig. 2.6: Two-mass model of the system.

Table 2. The system parameters

Symbol	PARAMETER	VALUE
m_M	Motor Mass	20 kg
m_L	Load Mass	4 kg
k	Spring Constant	13700 N/m
b	Damping coefficient	6 Ns/m
F_N	Nominal force	675 N
A_N	Nominal current	7.2 A
K_m	Motor constant	94 N/A
L_P	Winding inductance	20 mH
τ_M	Pole pitch (180°)	15 mm
P_N	Nominal power	3910 W
L	Stroke	1000 mm
K_p	P gain of vel. controller	10000 Ns/m
K_i	I gain of vel. controller	0.01 N/m
φ_1	1 st wavenumber	67.2 1/m
φ_2	2 nd wavenumber	8.5 1/m
$A_{r,1}$	Amplitude of 1 st harmonic	35 N
$A_{r,2}$	Amplitude of 2 nd harmonic	15 N
K_s	Scaling factor	-0.7

Chapter 3

Differential Evolution Strategy

3.1 Background

Problems, which involve global optimisation over continuous spaces, are ever-present throughout the scientific community. In general, the task is to optimise certain properties of a system by pertinently choosing the system parameters. For convenience, a system's parameters are usually represented as a vector. The standard approach to an optimisation problem begins by designing an objective function that can model the problem's objectives while incorporating any constraints. Consequently, we will only concern ourselves with optimisation methods that use an objective function. In most cases, the objective function defines the optimisation problem as a minimisation task. To this end, the following investigation is restricted to minimisation problems. For such problems, the objective function is more accurately called a "cost" function.

When the cost function is nonlinear and non-differentiable, central to every direct search method is an algorithm that generates variations in the parameter vectors. Once a variation is generated, a decision must then be made whether or not to accept the newly derived parameters. Most stand and direct search methods use the greedy criterion to make this decision. Under the greedy criterion, a new parameter vector is accepted if and only if it reduces the value of the cost function.

A genetic algorithm is a search technique used in computer science to find approximate solutions to optimization and search problems. Specifically it falls into the category of local search techniques and is therefore generally an incomplete search. Genetic algorithms are a particular class of evolutionary algorithms that use techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover (also called recombination).

Genetic Algorithms (GA) are typically implemented as a computer simulation in which a population of abstract representations (called chromosomes) of candidate solutions (called *individuals*) to an optimization problem evolves toward better

solutions. Traditionally, solutions are represented in binary as strings of 0s and 1s, but different encodings are also possible. The evolution starts from a population of completely random individuals and happens in generations. In each generation, the fitness of the whole population is evaluated, multiple individuals are stochastically selected from the current population (based on their fitness), and modified (mutated or recombined) to form a new population. The new population is then used in the next iteration of the algorithm.

The first aim of applying the DE algorithm in the following study is identification of a nonlinear servo-hydraulic system with a flexible load and the second aim is tuning the weights and biases of neural network systems.

3.2 Differential Evolution ALGORITHM

The Differential Evolution (DE) algorithm is a stochastic direct-search optimisation method that is fast and reasonably robust. Differential evolution is capable of handling nondifferentiable, nonlinear, and multimodal objective functions.

In a population of potential solutions within an n-dimensional search space, a fixed number of vectors are randomly initialised, then evolved over time to explore the search space and to locate the minima of the objective function.

At every iteration, called a generation, new vectors are generated by the combination of vectors randomly chosen from the current population (mutation). The out-coming vectors are then mixed with a predetermined target vector. This operation is called recombination and produces the trial vector. Finally, the trial vector is accepted for the next generation if and only if it yields a reduction in the value of the cost function. This last operator is referred to as a selection [Weisstein and Vassilis, 2006]. There are four essential versions for DE that differ only by how new solutions are generated. In the study, a classic version (DE/rand/1/bin) is presented. In this shorthand notation, the first term after “DE” specifies how the base vector is chosen and it means that the base vectors are randomly chosen. The number that follows indicates how many vector differences are used. The DE version that uses uniform crossover is appended with the additional term “bin” for “binomial” or uniform distribution [Price et al., 2005].

The Differential Evolution optimisation process is conducted by means of the following operations:

3.2.1: Initialization

In order to establish a starting point for the optimisation process, an initial population must be created. Typically, each decision parameter in every vector of the initial population is assigned a randomly chosen value from within its corresponding feasible bounds:

$$x_{j,G=0} = x_j^{(lo)} + rand[0,1] \cdot (x_j^{(hi)} - x_j^{(lo)}) \quad (3.1)$$

where, $x_j^{(hi)}$ and $x_j^{(lo)}$ are the upper and lower bound to the j^{th} decision parameter, respectively. After that the initial population is created, it evolves through the operation of mutation, crossover and selection.

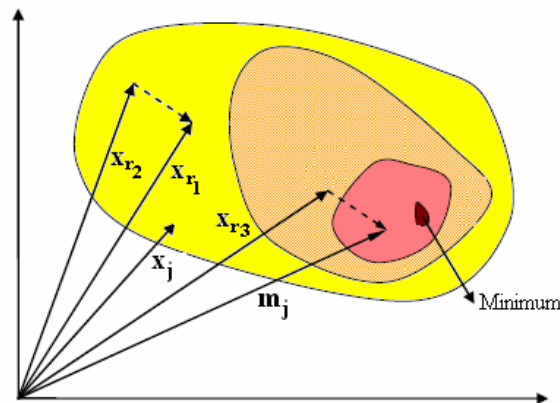


Fig. 3.1: The process of generating a mutant vector of a two- dimensional cost function.

3.2.2: Mutation

Mutation generally refers to an operation that adds a uniform random variable to one or more vector parameters. The DE relies upon the population itself to supply increments of the appropriate magnitude and orientation. At every generation G , each vector in the population has to serve once a target vector. For each target vector $x_{j,G}$, a mutant vector m_j is generated according to:

$$m_j = x_{j,G} + F_1 \cdot (x_{r_1,G} - x_{r_2,G}) \quad (3.2)$$

where $x_{r_1,G}$, $x_{r_2,G}$, and $x_{r_3,G}$ are randomly chosen vectors from the set $\{1, \dots, N_p\}$, N_p is the population size, F_1 is a user-defined constant (also known as the mutation scaling factor), which is typically chosen from the range (0,1] and G is the generation number. Figure (3.1) shows a two dimensional example which plays a part in the generation of the mutant vector m_j . It is clear that the mutation is the summation of third vector, x_{r_3} plus scaled of the difference of two other vectors, $F_1 \cdot (x_{r_1} - x_{r_2})$.

3.2.3: Crossover

In order to increase the diversity of the generated vectors, the crossover operation is introduced. Finally, the trial vector $u_{j,G+1}$ is created by mixing the parameter of the parent vector $x_{j,G}$ and the mutant vector m_j in the following form.

$$u_{j,G+1} = \begin{cases} m_j & \text{If } (\text{rand}_j[0,1) < \text{CR} \vee j = j_r) \\ x_{j,G} & \text{otherwise} \end{cases} \quad (3.3)$$

where $\text{rand}_j[0,1)$ generates a new uniformly distributed random number for each value of the j within the range of $[0, 1)$. CR is known as a crossover rate constant and is a user-defined parameter within the range $[0, 1]$. The index j_r is randomly chosen from the set $\{1, \dots, D\}$, which is used to ensure that $u_{j,G+1}$ gets at least one parameter from m_j . The index D shows the number of parameters. Figure 3.2 gives an example of the crossover mechanism for a 7-dimensional vector.

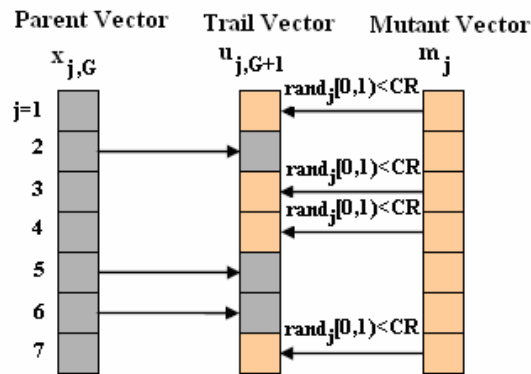


Fig. 3.2: The crossover process for a 7-dimensional vector

3.2.4: Selection

To decide whether or not it should become a member of the next generation, the trial vector $u_{j,G+1}$ is compared to the target vector $x_{j,G}$ using a cost function value criterion. An individual (vector) with a minimum value of cost function is allowed to advance to the next generation. That is,

$$x_{j,G+1} = \begin{cases} u_{j,G+1} & \text{if } F(u_{j,G+1}) \leq F(x_{j,G}) \\ x_{j,G} & \text{otherwise} \end{cases} \quad (3.4)$$

where, $F()$ indicates the cost function and $x_{j,G+1}$ is the next generation of vector $x_{j,G}$. By using this selection procedure, all individuals of the next generation are as good as or better than the individuals of the current population.

3.2.5: THE OBJECTIVE FUNCTION

3.2.5.1: THE MEAN SQUARE ERROR (MSE)

The objective (cost) function used for this study is the summation of the square of the normalised error for a period of time. By using (1ms) the sampling time, the error between the measured and simulation points can be calculated.

3.2.5.2: Constraints

One of the most important subjects to work with DE is constraints. Constraints typically make optimisation harder for DE because they can create forbidden regions on the objective function landscape that restrict the free movement of the vectors. It often happens, depending on how they are handled, that constraints divide the search space into several disjoint islands. On the other hand, constraints can eliminate local minima that might otherwise trap vectors, thereby reducing the chance that DE will prematurely converge.

3.2.5.3: PENALTY FUNCTION

To obtain the parameters of vector, \bar{p} which minimises the cost function, the parameters are subjected to the following constraints,

$$\bar{p}_{\min} \leq \bar{p} \leq \bar{p}_{\max} \quad (3.5)$$

where, \bar{p}_{\min} and \bar{p}_{\max} are the minimum and maximum range limits for the parameter of \bar{p} . The direct search optimisation methods do not usually provide a mechanism to restrict the parameters in the range defined by inequality 3.5; neither does the differential evolution. However, an unconstrained optimisation method can be transformed to a constrained one using the concept of the penalty function. This function determines a penalty to be added to the value of the cost function whenever any parameter exceeds the range limits. The penalty function selected in this case is given by equation (3.6). Figure 3.3 shows the effect of the penalty function in trying to constrain the parameter in the region of 10-30.

Notice the rapid increase in the penalty as the parameter diverges from the bounds of the range and the difference in the rate of change at either end. At the lower end this is higher because the coefficient of the square difference is higher.

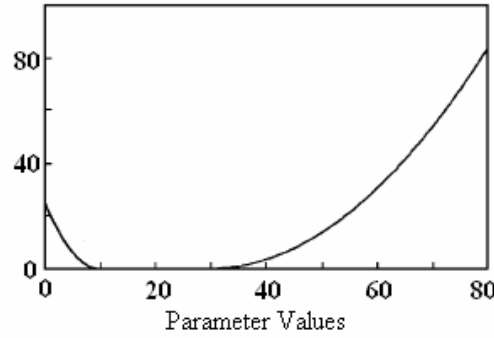


Fig. 3.3: The penalty function

$$P(p_i) = \begin{cases} 20([p_{i(\min)} - p_i] / p_{i(\min)})^2 & \text{for } p_i < p_{i(\min)} \\ 0 & \text{for } p_i = p_{i(\min)} \\ 20([p_{i(\max)} - p_i] / p_{i(\max)})^2 & \text{for } p_i > p_{i(\max)} \end{cases} \quad (3.6)$$

Differential Evolution (DE) is a simple and extremely powerful evolutionary computation technique that solves real-valued problems based on the principles of natural evolution. The following is the suitable form of DE for programming,

Inputs:

$$D, G_{\max}, NP \geq 4, F_1 \in (0,1+) , CR \in [0,1] , x^{(lo)}, x^{(hi)} \quad (3.7)$$

G_{\max} is the maximum number of generation.

Initialize:

$$\begin{cases} \forall i \leq NP \wedge \forall j \leq D: \\ x_{j,G=0} = x_{j,i}^{(lo)} + rand[0,1] \cdot (x_j^{(hi)} - x_j^{(lo)}) \end{cases} \quad (3.8)$$

While $G < G_{\max}$

$\forall i \leq NP$:

Mutation:

$$\begin{aligned} r_1, r_2, r_3 &\in \{1, 2, \dots, NP\}, \text{ randomly selected except: } r_1 \neq r_2 \neq r_3 \neq i \\ \forall j \leq D \text{ (} j_r \text{ random selected once each } i\text{):} & \quad (3.9) \\ m_j &= x_{r_3,G} + F_1 \cdot (x_{r_1,G} - x_{r_2,G}) \end{aligned}$$

Recombination:

$$u_{j,G+1} = \begin{cases} m_j & \text{If } (rand_j[0,1] < CR \vee j = j_r) \\ x_{j,G} & \text{otherwise} \end{cases} \quad (3.10)$$

Selection:

$$x_{j,G+1} = \begin{cases} u_{j,G+1} & \text{if } F(u_{j,G+1}) \leq F(x_{j,G}) \\ x_{j,G} & \text{otherwise} \end{cases} \quad (3.11)$$

$$G = G + 1 \quad (3.12)$$

Chapter 4

Fuzzy Control

4.1 Background

The term *fuzzy logic* emerged in the development of the theory of fuzzy sets by Lotfi Zadeh in 1965. He believed that the control theory was becoming increasingly complex and mathematical, while neglecting the importance of practicality and applicability to real-world control problems. He suggested that his idea of fuzzy set theory, developed in 1965 [Zadeh, 1965], might be useful in addressing his concerns about conventional control theory. The central idea of fuzzy set theory is that elements can have partial membership in a given set. In contrast, traditional set theory only allows elements to be either complete members of a set or complete non-members of a set. Fuzzy sets therefore allow for the expression of vagueness with respect to set membership, which reflects everyday experience better than traditional set theory.

Given the advantage of being able to model the vagueness of real world variables, researchers began to investigate the use of fuzzy logic in control systems. The first practical investigations of fuzzy control began with Mamdani et al. In 1975, they applied the first fuzzy logic controller (FLC) to a steam engine [Mamdani and Assilian, 1975]. Since that time, substantial research into fuzzy control systems has been performed. As such, fuzzy control is now recognised as a standard, established method of controlling feedback systems, especially those that are nonlinear, complex or ill-defined. In this Chapter, FLCs will be discussed starting with their basis in conventional and fuzzy set theory.

4.2 Conventional Sets

In traditional set theory, a set is defined as a collection of "crisp", or discrete, values taken from a universal set [Passino and Yukovich, 1998, Zimmermann, 2001]. For example, if a universal set X is defined as including all positive integers, or $X = \{1, 2, 3, \dots\}$, it is possible to define a set $A = \{1, 3, 6\}$. In a conventional set, each

element x of X is either an element of A or not an element of A . This leads to the definition of a characteristic function $\mu_A(x)$ that expresses the membership of a given element x in the set A . If x is a member of A , then the characteristic function evaluated at x is exactly one; conversely, if x is not a member of A then the characteristic function evaluated at x is exactly zero. In mathematical terms, the characteristic function is expressed as:

$$\mu(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{if } x \notin A \end{cases} \quad (4.1)$$

4.3 Fuzzy Sets

The fundamental difference between conventional set theory and fuzzy set theory is that instead of defining the elements x in the universal set X is absolutely a member or not a member of set A , fuzzy sets allow the elements to have a degree of membership in fuzzy set A . This is achieved by replacing the characteristic function $\mu_A(x)$ by a membership function, which is allowed to have values in the continuous range $[0, 1]$ inclusive. This membership function can be represented mathematically as follows [Passino and Yukovich, 1998, Zimmermann, 2001]:

$$\mu(x) = \begin{cases} 1 & \text{if } x \in A \\ \{0,1\} & \text{if } x \text{ has partial membership in } A \\ 0 & \text{if } x \notin A \end{cases} \quad (4.2)$$

4.4 Fuzzy Controllers

The fuzzy sets described in Section 4.3 are essential to the operation of fuzzy controllers as imagined by Zadeh [Zadeh, 1965] and implemented by Mamdani [Mamdani and Assilian, 1975] and others. Though essential, fuzzy sets alone do not make a fuzzy controller. The controllers consist of four main elements [Harris, J., 2006]: a fuzzifier, an inference engine, a rule base, and a defuzzifier. These processes are illustrated in Fig. 4.1 and described in the following sections. The following sections also describe the main elements.

4.4.1 Fuzzification

Fuzzification is the process of translating crisp input values into fuzzy linguistic values through the use of membership functions. In other words, determining how

much of each discrete input value belongs to each input fuzzy set using the corresponding membership function.

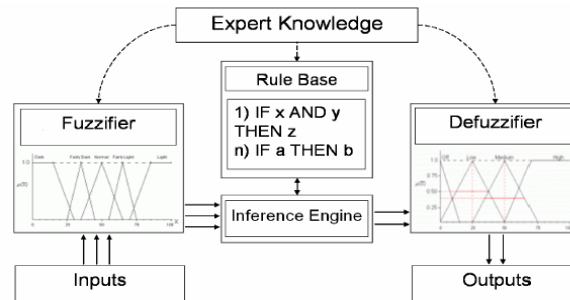


Fig. 4.1: Overview of a fuzzy controller.

4.4.2 Rule Base

The fuzzy rule base is simply a database of the desired control rules for the system. It is most equivalent to the controller of a traditional control system, and formalises the designer's "expert knowledge" of what control output should result from a given combination of system states, expressed in a linguistic manner. It often takes the form of a truth table consisting of rules constructed in the following form:

$$IF < condition1 \text{ AND/Or } condition2 > THEN < consequence > \quad (4.3)$$

In fuzzy logic terminology, the statement following the IF condition is known as the "premise," "antecedent," or "condition." The corresponding statement following THEN is known as the "conclusion" or "consequent." The actual calculation of the consequent using the premises calculated from the fuzzified inputs is reserved for the inference engine. This aspect of FLC is discussed in the following section.

4.4.3 Inference Engine

The inference engine is the heart of a FLC. It acts as the bridge between the fuzzification input stage and the defuzzification output stage of the controller, translating the designer's desired control rules from a linguistic representation to a numeric computation. The inference engine can be divided into three elements: aggregation, composition, and accumulation. The first step of the inference process is known as "aggregation." During this step the premise (IF statement) of each rule in the rule base is calculated using the fuzzified controller inputs. In the aggregation process the conditions are aggregated according to the logical statement connecting

them, such as AND/OR. It should be noted that the traditional formulation of logical statements such as AND/OR has been modified to accommodate the use of membership functions. In particular, the result of an AND operation is often defined as either the minimum (min) of the two fuzzy values compared, or the product (prod) of the two values.

Similarly, an OR operation is often defined as either the maximum (max) of the two fuzzy values compared, or the probabilistic sum (sum). The probabilistic sum is defined as the sum of the two values compared, minus their product. Any premise with a value greater than zero means that its corresponding rule is active, or has "fired" in FL terminology.

The second step of the inference process is known as "composition" or "implication." In this step the consequent (THEN statement) of each rule is created using the premises calculated in the first step. The output of the composition step is not a single value for each rule in the rule base.

The third and final step of the inference process is known as "accumulation," or "results aggregation." In this step the implied fuzzy sets that are the output of the composition process are combined into an accumulated fuzzy set, which is the input to the defuzzification process. The sets are combined by calculating the union of the implied membership functions.

4.4.4 Defuzzification

In general, the system to be controlled using a FLC requires a crisp or discrete input, rather than a membership function such as is produced by the inference engine. Defuzzification is the process of converting the fuzzy output set which is a result of the inference process into a discrete number suitable for input to the plant. There are many different methods of defuzzification described in the literature, with varying levels of complexity. Two fundamental methods are known as the Mean of Maxima (MoM) method and the Center of Gravity (CoG) method. The CoG method, also known as the Center of Area (CoA) method, can be interpreted as a weighted average of the elements in the set.

Chapter 5

Recurrent Neural Network

5.1 Background

Artificial neural networks (ANNs) or simply neural networks (NNs) are made of interconnected devices called neurons. NNs have been used in a wide variety of signal processing and pattern recognition applications and have been successfully applied in such diverse fields as speech processing, handwritten character recognition, time series prediction, data compression, feature extraction, and pattern recognition in general. Their attractiveness lies in the relative simplicity with which the networks can be designed for a specific problem, along with their ability to perform nonlinear data processing.

As the neuron is the building block of a brain, a neural unit is the building block of a neural network. Although the two are far from being the same or from performing the same functions, they still possess similarities that are remarkably important. NNs consist of a large number of interconnected units that give them the ability to process information in a highly parallel way. The brain, as well, is a massively parallel machine as has long been recognized. As each of the 10^{11} neurons of the human brain integrates incoming information from all other neurons directly or indirectly connected to it, an artificial neuron sums all inputs to it and creates an output that is carrying information to other neurons. The connection from one neuron's cell body to another neuron's processes is called a *synapse*. The strength by which two neurons influence each other is called a *synaptic weight*. In a NN all neurons are connected to all other neurons by synaptic weights that can have seemingly arbitrary values, but in reality, these weights show the effect of a stimulus on the neural network and the ability or lack of it to recognize that stimulus neuron on other changes. All NNs have certain architectures, and all consist of several layers of neuronal arrangements. The most widely used architecture is that of the perceptron first described in 1958 by Rosenblatt [Rosenblatt, 1958]. A simplified biological neural system is compared with the block diagram of an artificial neuron in Figure 5.1.

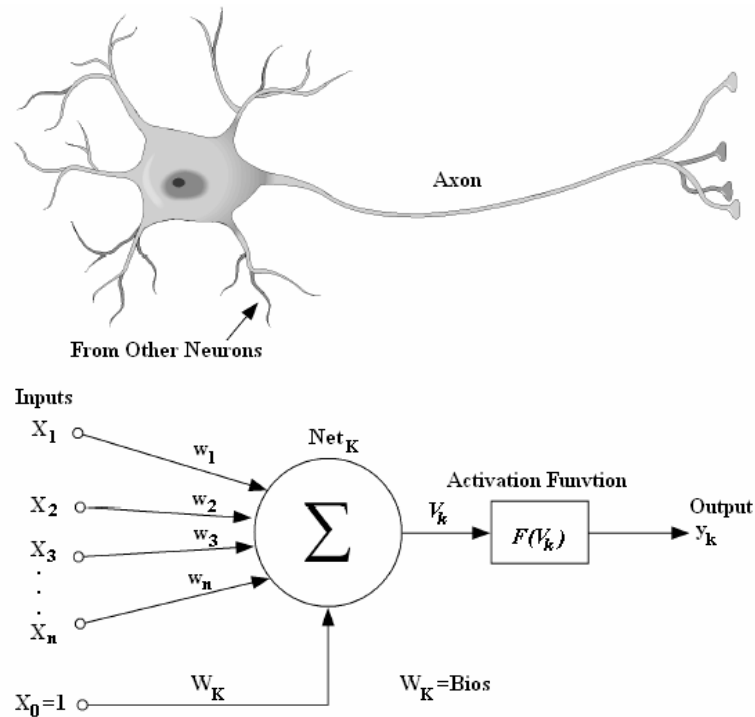


Fig. 5.1: Biological (above) and artificial (below) neurons.

Artificial neural networks usually operate in one of two modes. Initially there exists a training phase where the interconnection strengths are adjusted until the network has a desired output. Only after training does the network become operational, i.e., capable of performing the task it was designed and trained to do. The training phase can be either supervised or unsupervised. In supervised learning, there exists information about the correct or desired output for each input training pattern presented. The original perceptron and backpropagation are examples of supervised learning. In this type of learning the NN is trained on a training set consisting of vector pairs. One of these vectors is used as input to the network; the other is used as the desired or target output. During training the weights of the NN are adjusted in such a way as to minimise the error between the target and the computed output of the network. This process might take a large number of iterations to converge, especially because some training algorithms (such as backpropagation) might converge to local minima instead of the global one. If the training process is successful, the network is capable of performing the desired mapping.

5.2 Feedforward and Recurrent Networks

Neural networks can be classified as feedforward and recurrent networks. In feedforward networks, the nodes are connected in such a way that all signals flow in one direction from the input units to the output units. In recurrent networks, there are both inputs feed-forward, and output(s) feedback as shown in Fig. 5.2. In the study a recurrent network is used and therefore, it inherently has a capacity for modelling dynamic systems [Pham and OH, 1993].

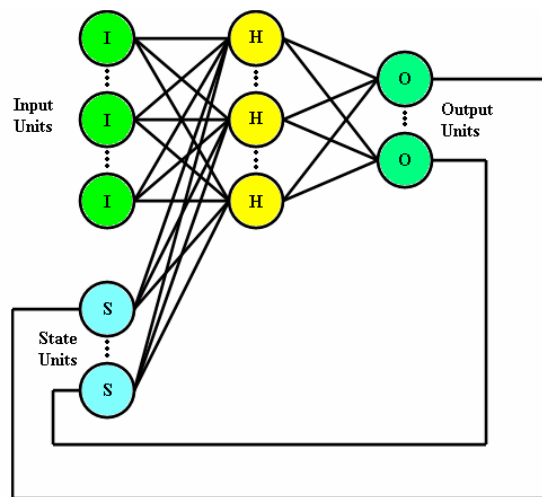


Fig.5.2: The Jordan network.

One of the earliest recurrent neural networks was the Jordan network [Jordan, 1986a, Jordan, 1986b]. The Jordan Networks use input of the past output like a memory for a neural network. An exemplar network is shown in Figure (5.2). In the Jordan network, the activation values of the output units are feedback into the input layer through a set of extra input units called the *state units*. The numbers of state units are equal to the number of network outputs. The connections between the output and state units have a fixed weight of +1; learning takes place only in the connections between input and hidden units as well as hidden and output units. Thus all the Learning rules derived for the multilayer perceptron (Backpropagation) can be used to train this network.

Applications in forecasting signal, processing and control require explicit treatment of dynamics. Feedforward networks can accommodate dynamics by including past input and target values in an augmented set of inputs. A much richer dynamic representation results from also allowing for internal network feedbacks. These types of network models are called recurrent network models and are used by Jordan [Jordan and Rumelhart, 1992, Jordan and Jacobs, 1990] for controlling and learning smooth robot movements and by Elman [Elman, 1990] for learning and representing temporal structure in linguistics. In Jordan's network, past values of the network output, feed back into input units; in Elman's network, past values of hidden units feed back into themselves or into input units.

Jordan networks have been applied successfully in many industrial applications since 1986. They use input of the past output like a memory for a neural network, so they can accommodate dynamics. In general, stability of feedforward and recurrent networks are hard task.

Minsky and Papert [Minsky and Papert, 1969.] showed in 1969 that a two layer feed-forward network can overcome many restrictions, but did not present a solution to the problem of how to adjust the weights from input to hidden units. An answer to this question was presented by Rumelhart et al in 1986 [Rumelhart et al., 1986], and similar solutions appeared to have been published earlier [Werbos, 1974, Parker, 1985].

The central idea behind this solution is that the errors for the units of the hidden layer are determined by backpropagating the errors of the units of the output layer. For this reason the method is often called the backpropagation learning rule. Back-propagation can also be considered as a generalisation of the delta rule for non-linear activation functions and multilayer networks.

5.3 Backpropagation with Momentum Term

The backpropagation algorithm is a learning scheme in which the error is backpropagated layer by layer and used to update the weights. The algorithm is a gradient descent method that minimizes the error between the desired outputs and the actual outputs calculated by the Multiple Layer Perceptron (MLP).

The backpropagation training process requires that the activation functions be bounded, differentiable functions. One of the most commonly used functions satisfying these requirements is the hyperbolic tangent function as follow;

$$f(v_k) = \frac{e^{v_k} - e^{-v_k}}{e^{v_k} + e^{-v_k}} \quad (5.1)$$

where v_k is the summation of net and $f(v_k)$ calculates the output for neuron p .

Let E_i be the error associated with template i ;

$$E_i = \frac{1}{2} \sum_{p=1}^N (t_p - o_p)^2 \quad (5.2)$$

where N is the number of output neurons in the MLP, t_p is the target or desired output and o_p is the output of network calculated by the MLP.

Let $E = \sum E_i$ be the total measure of error. The learning procedure requires only that the change in weights and biases are proportional to the $\partial E / \partial w$. True gradient descent requires that infinitesimal steps be taken. The constant of proportionality is the learning rate in the procedure. The higher amount of the learning rate makes the greater value change in the weights of neural network. One way to increase the learning rate without leading to oscillation is to modify the general delta rule to include a momentum term [Rumelhart et al., 1986]. This can be accomplished by the following rule:

$$\Delta w_{ji}(n+1) = \eta(\delta_{pj} o_{pi}) + \alpha \Delta w_{ji}(n) \quad (5.3)$$

where n indicates the presentation number. The parameter of η is the learning rate, and α is a constant which determines the effect of past changes on the current direction of movement in weight space, and $\Delta w(n)$ is the change in the weight in step n .

5.4 Modified Backpropagation

The standard backpropagation algorithm is modified to be useful for the dynamics compensation. The nonlinear plant is a Permanent Magnet Linear Motor. The input of the system is the reference velocity (v_{ref}) and the out of the nonlinear plan is the velocity of load (v_{load}).

The training procedure can be deduced from Fig. 5.3. For calculating the error, there is no target for the output of the neural network, so the error of the neural network

can not be updated from its output directly. To solve the problem, the error was defined based on the control target and it was defined as the difference between the reference velocity and the load velocity. The updating of the neural network is based on the *modified backpropagation* method. The structure of updating is the same as the Backpropagation, but it has only one extra term (k_c).

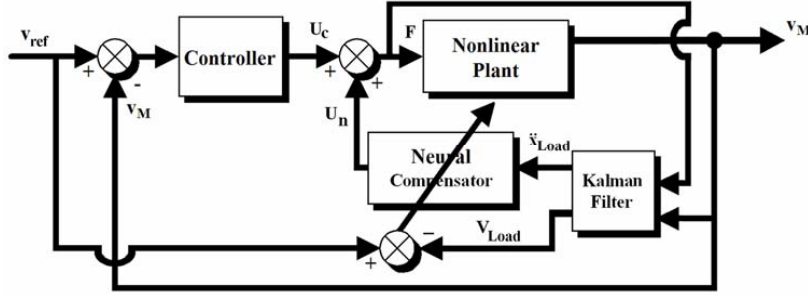


Fig. 5.3: The proposed neural network hybrid controller.

The extra term is defined as follows,

$$k_c = \frac{\partial V_{load}}{\partial U_n} = \frac{\partial V_{load}}{\partial t} \frac{\partial t}{\partial U_n} = \frac{a_{load}}{\dot{U}_n} \quad (5.4)$$

k_c is the extra gain for online updating of the neural network.

There are two parameters to find the amount of k_c . First parameter, a_{load} , is estimated by Kalman filter directly, and second parameter is calculated by the following equation,

$$\dot{U}_n = \frac{U_n(n) - U_n(n-1)}{T_s} \quad (5.5)$$

where T_s is the sampling time.

The new online updating for neural network as the compensator can be defined as,

$$\Delta w_{ji}(n+1) = \eta(\delta_{pj} o_{pi})k_c + \alpha k_c \Delta w_{ji}(n) \quad (5.6)$$

Following the proof for the new online updating formula, Eq. 5.6, for the neural network is provided.

Lets define the following terms;

$$E = \frac{1}{2}(v_{ref} - v_{load})^2 \quad (5.7)$$

$$I_n = \sum_j w_{nj} y_j \quad (5.8)$$

$$U_n = f(I_n) \quad (5.9)$$

Where E is the error, I_n is the net input to unit n , U_n is the output of the neural network, y is the output of the previous hidden layer and w_{nj} are the weights connected to the output layer to the previous hidden layer.

Focusing first on the weight updates for the output unit, we can use the actual error to find the update rule. We have

$$\frac{\partial E}{\partial w_{nj}} = \frac{\partial E}{\partial I_n} \frac{\partial I_n}{\partial w_{nj}} = \frac{\partial E}{\partial I_n} \left(\frac{\partial}{\partial w_{nj}} \sum_j y_j w_{nj} \right) \quad (5.10)$$

The term in the parentheses in Equation (5.10) can be evaluated directly. Thus,

$$\frac{\partial}{\partial w_{nj}} \sum_j y_j w_{nj} = y_j \quad (5.11)$$

Now, using the chain rule again, we can write

$$\frac{\partial E}{\partial I_n} = \frac{\partial E}{\partial U_n} \frac{\partial U_n}{\partial I_n} \quad (5.12)$$

From Equation (5.9) we have,

$$\frac{\partial U_n}{\partial I_n} = f'(I_n) \quad (5.13)$$

Using the chain rule for first term of Equation (5.12), we can write

$$\frac{\partial E}{\partial U_n} = \frac{\partial E}{\partial v_m} \frac{\partial v_m}{\partial U_n} = -(v_{ref} - v_{load}) k_c \quad (5.14)$$

where v_m is the motor velocity.

Define,

$$k_c = \frac{\partial V_{load}}{\partial U_n} = \frac{\partial V_{load}}{\partial t} \frac{\partial t}{\partial U_n} = \frac{a_{load}}{\dot{U}_n} \quad (5.15)$$

k_c is the extra gain for online updating of the neural network. There are two parameters needed in order to find the amount of k_c . The first parameter, a_{load} , is estimated by the Kalman filter directly, and the second parameter is calculated using the following equation:

$$\dot{U}_n = \frac{U_n(n) - U_n(n-1)}{T_s} \quad (5.16)$$

where T_s is the sampling time.

If we now define,

$$\delta_n = (v_{ref} - v_{load})f'(I_n)k_c \quad (5.17)$$

we can write the update rule for the output unit in the same form as the backpropagation, that is

$$\Delta w_{nj} = -\eta \frac{\partial E}{\partial w_{nj}} = \eta \delta_n y_j \quad (5.18)$$

This rule applies to all the hidden and output weights for a single pattern presentation. As shown in Equation (5.14), there is an extra term not found in the standard form of Backpropagation, k_c .

Turning now to the hidden layer weight v_{ji} , we see that in this case, we do not have target values from, which to compute the errors; instead, we must somehow use the error from the output unit to adjust the input to the hidden layer weights. For this, we can use the chain rule repeatedly to relate the output error to these weights.

Defined in the following terms,

$$H_j = \sum_i v_{ji} x_i \quad (5.19)$$

$$y_j = f(H_j) \quad (5.20)$$

Where H_j is the combined or net input to the hidden layer unit j , x_i is the output of unit i , and v_{ji} is the weight connecting hidden layer j to i .

Let's find an expression for

$$\Delta v_{ji} = -\eta \frac{\partial E}{\partial v_{ji}} = -\eta \frac{\partial E}{\partial H_j} \frac{\partial H_j}{\partial v_{ji}} \quad (5.21)$$

Note that last partial derivative in this expression can be evaluated directly from Equation (5.21), that is,

$$\frac{\partial H_j}{\partial v_{ji}} = \sum_i \frac{\partial}{\partial v_{ji}} (v_{ji} x_i) = x_i \quad (5.22)$$

To solve for the partial derivative term in (5.21), we used the chain rule to obtain

$$\frac{\partial E}{\partial H_j} = \frac{\partial E}{\partial y_j} \frac{\partial y_j}{\partial H_j} = \frac{\partial E}{\partial y_j} f'(H_j) \quad (5.23)$$

Differentiating $\frac{\partial E}{\partial y_j}$ directly we obtain,

$$\begin{aligned}
\frac{\partial E}{\partial y_j} &= \frac{1}{2} \sum_k \frac{\partial (v_{ref} - v_{load})^2}{\partial y_j} \\
&= - \sum_k (v_{ref} - v_{load}) \frac{\partial v_{load}}{\partial y_j} \\
&= - \sum_k (v_{ref} - v_{load}) \frac{\partial v_{load}}{\partial U_n} \frac{\partial U_n}{\partial y_j} \\
&= - \sum_k (v_{ref} - v_{load}) k_c f'(I_n) w_{nj}
\end{aligned} \tag{5.24}$$

Combining the above operations, we can write the update rule for the hidden layer units as

$$\Delta v_{ji} = \eta \delta_j x_i = \eta x_i k_c f'(H_j) \sum_n \delta_n w_{nj} \tag{5.25}$$

Where

$$\delta_j = f'(H_j) \sum_n \delta_n w_{nj} \tag{5.26}$$

Equations (5.14) and (5.25) are for updating the weights of the output and hidden layers respectively, the only difference with the standard form of backpropagation is the term k_c .

The k_c in Eq. (5.4) is bounded, because it is division of load acceleration by derivation of neural network output. The load acceleration is bounded and the derivative of network output can be estimated by Eq. (5.5). In the equation, the term of $U_n(n) - U_n(n-1)$ is difference of two sequence output of the neural network and it can not be zero during online training. The parameter of k_c is bounded if and only if the derivative of network output is not zero.

Following, the only two probable cases that term will be zero are discussed and showed they can not happened in our study.

Case one:

When the neural network is saturated then the output of neural network does not change to varying of inputs. The saturation problem happens when the initial weights of neural network are not selected correct (local minimum) or the amount of learning rate for online training is too big. These two conditions are not happened in our study, because of avoiding the saturation problem, Differential Evolution algorithm is used

to find the best values for the weights (global minimum) and during the online training, the amount of learning rate must be sufficient, too.

Case two:

When the system approaches to its steady state condition (the error becomes near to zero), the term of $U_n(n) - U_n(n-1)$ approaches to zero. In this case the online updating of weights is stopped, because in practical application of neural network, the online update only happens when the error is bigger than tolerance chosen by a designer (the neural network is trained until the error is in a tolerance), so as it discussed, the term of k_c is bounded and it can be used in online training.

As it shown in this chapter, the online training of the compensator network has an extra term, k_c compare to the standard backpropagation algorithm with momentum term. There are lots of criteria on the stability of backpropagation with a momentum term. The effect of the extra term on stability of the network has not been studied, but it can be considered as future study. To end top this section, comparing the Eq. (5.6) to the standard form, the following criteria must be satisfied;

$$\begin{cases} \eta k_c < 1 \\ \alpha k_c \leq 1 \end{cases} \quad (5.27)$$

Chapter 6

Adaptive Backstepping

6.1 Background

The backstepping procedure was first introduced by Kokotovic [Kokotovich, 1992]. The key idea of backstepping is to start with a system which is stabilised with a known feedback law for a known Lyapunov function, and then to add to its input an integrator. For the augmented system a new stabilising feedback law is explicitly designed and shown to be stabilising for a new Lyapunov function, and so on. The method used in the backstepping controller is called Lyapunov's direct method.

Nowadays nonlinear controllers like sliding mode [Furuhashi et al., 1992, Lin et al., 1998] and backstepping controllers [Krstic et al., 1995] are becoming more popular due to the development of design methods and computer software. For a certain class of nonlinear system a so called differentially flatness can also be used in stabilisation [Fliess et al., 1995]. A promising method for nonlinear control is the *control Lyapunov function* (CLF) whose derivative depends on the control and can be made negative by feedback. This is a difficult problem because no general design methods are available. For a large class of systems CLFs can be constructed by backstepping. Backstepping is a recursive design for a system with nonlinearities not constrained by linear bounds [Kokotovic and Arcak, 2001]. The key idea of backstepping is to derive an error equation and to construct a control law and a parameter adjustment law so that the state of the error equation goes to zero [Åström and Wittenmark, 1995]. The backstepping method can be successfully implemented for several applications [Dawson et al., 1994, Lin and Kanellakopoulos, 1997].

6.2 Lyapunov Stability method

There are basically two ways of using Lyapunov's direct method for control design, and both have a trial and error flavour. The first technique involves hypothesizing one form of control law and then finding a Lyapunov function to justify the choice. The second technique, used in the controller, works in such a way that

first a Lyapunov function $V(x)$ is selected and then a feedback control $u(x)$ that renders $\dot{V}(x, u(x))$ a negative definite is sought. With an arbitrary choice of $V(x)$ this attempt may fail, but if $V(x)$ is a *CLF*, a stabilising control law $u(x)$ can be found. The following example clarifies this.

The invariant system is

$$\dot{x} = f(x, u), \quad x \in \mathbb{R}^n, \quad u \in \mathbb{R}, \quad f(0, 0) = 0, \quad (6.1)$$

The goal is to design a feedback control law $\alpha(x)$ for the control variable u such that the equilibrium $x=0$ of the closed loop system

$$\dot{x} = f(x, \alpha(x)), \quad (6.2)$$

is Globally Asymptotically Stable (GAS). It is required that the derivative of a Lyapunov candidate $V(x)$ satisfy $\dot{V}(x) \leq -W(x)$, where $W(x)$ is a positive definite function. $\alpha(x)$ needs to be found to guarantee that $x \in \mathbb{R}^n$ for all

$$\dot{V}(x) = \frac{\partial V}{\partial x}(x) f(x, \alpha(x)) \leq -W(x), \quad (6.3)$$

A function is called a control Lyapunov function (CLF) for Eq. (6.1) if

$$\dot{V}(x) = \frac{\partial V}{\partial x}(x) f(x, u) < 0, \quad \forall x \neq 0, \quad (6.4)$$

For the nonlinear system

$$\dot{x} = f(x) + g(x)u, \quad (6.5)$$

the CLF inequality in Eq. (6.3) becomes

$$\frac{\partial V}{\partial x} f(x) + \frac{\partial V}{\partial x} g(x) \alpha(x) \leq -W(x), \quad (6.6)$$

It should be noted that Eq. (6.6) can be satisfied only if, for all $x \neq 0$,

$$\frac{\partial V}{\partial x} g(x) = 0 \Rightarrow \frac{\partial V}{\partial x} f(x) < 0, \quad (6.7)$$

When $V(x)$ is a *CLF*, there are many control laws that render $\dot{V}(x, u(x))$ a negative definite, and in the study a backstepping method is used.

6.3 Adaptive Backstepping

There are several well-known adaptive schemes that solve the control problem for linear systems with unknown parameters, but there are also restricted classes of

nonlinear systems for which the design problem is solvable. Adaptive controllers are dynamic and therefore more complex than static controllers. An adaptive controller guarantees not only that the plant state x remains bounded, but also that it tends towards a desired constant value or asymptotically tracks a reference signal. For systems with parametric uncertainties, a parameter update law is designed such that the closed loop stability is guaranteed when the estimator is used by the controller. This is achieved by extending the Lyapunov function $V(x)$ with a term penalising the estimation error. The idea is to employ backstepping to design a control law for the system as if all the parameters were known and then replacing the unknown parameters with their estimates.

This is achieved by extending the Lyapunov function $V(x)$ with a term penalising the estimation error ($\theta = \hat{\theta} + \xi$). The idea is to employ backstepping to design a control law for the system as if all the parameters were known and then replace the unknown parameters with their estimates, a certainty equivalence way of thinking. We can consider the plant [Hirvonen, 2006]

$$\dot{x} = u + \theta x, \quad (6.8)$$

where u is the control parameter and θ is the unknown constant parameter. The aim is to achieve regulation of the state $x(t)$: $x(t) \rightarrow 0, t \rightarrow \infty$. Here we are seeking a parameter update law for the estimate $\hat{\theta}$,

$$\dot{\hat{\theta}} = \tau(x, \hat{\theta}), \quad (6.9)$$

which, along with the control law $u = \alpha(x, \hat{\theta})$, will make the derivative of the *CLF* $V(x, \hat{\theta})$ negative. As mentioned in the preceding part of this section, one of the terms in the *CLF* is intended to penalise the estimation error ξ . A simple choice is the quadratic term $\frac{1}{2}\xi^2$. This results in the *CLF*

$$V(x, \hat{\theta}) = \frac{1}{2}x^2 + \frac{1}{2}(\hat{\theta} - \theta)^2, \quad (6.10)$$

which is a radially unbounded function of time. We express the derivative of V as a function of u and $\dot{\hat{\theta}}$ and seek $\alpha(x, \hat{\theta})$ and $\tau(x, \hat{\theta})$ to guarantee that $\dot{V} \leq -cx^2$ with $c > 0$.

$$\dot{V} = x(u + \theta x) + (\hat{\theta} - \theta)\dot{\hat{\theta}} \leq -cx^2, \quad (6.11)$$

By rearranging the terms we obtain

$$xu + \hat{\theta}\dot{\theta} + \theta(x^2 - \dot{\theta}) \leq -cx^2, \quad (6.12)$$

Since neither α nor τ is allowed to depend on the unknown θ , we must take $\tau = x^2$,

$$\dot{\theta} = x^2, \quad (6.13)$$

The remaining condition

$$xu + \hat{\theta}x^2 \leq -cx^2, \quad (6.14)$$

allows us to select u in various ways. For instance, we can choose

$$u = -(c + \hat{\theta})x, \quad (6.15)$$

Along with the update law, this controller renders the derivative of CLF negative, and the closed-loop system is guaranteed to be stable. Although this example applies to linear systems, it shows the principle of the Lyapunov-based design, which is also valid for nonlinear systems. This exhibits some features of the more general scheme. A block diagram of the adaptive control principle is presented in Figure 6.1.

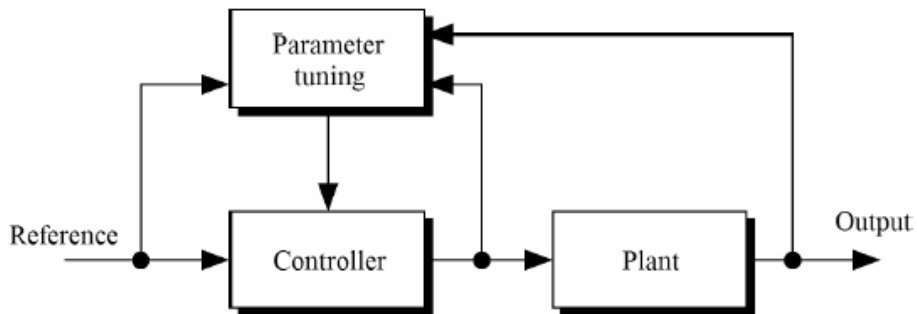


Fig. 6.1: Adaptive control scheme.

Chapter 7

Results and Conclusions

In this chapter the main results and conclusions of the journal papers are provided. The first section describes the differential evolution in system identification of a servo-hydraulic system with a flexible load. The second section is the application of fuzzy gain-scheduling in a servo-hydraulic system with a flexible load. The third section deals with a neural dynamics compensator in a PMLM system and the fourth section shows the some results and conclusions of applying an adaptive back stepping controller in position tracking of a PMLM system.

7.1 DE-Identification

In this study, the application of DE to solve the minimal representation problem in system identification of the servo-hydraulic system with a flexible load has been studied.

7.1.1 Results

In this section, the application results of DE in system identification of the servo-hydraulic with a flexible load are presented. Two tests in extension and retraction movements are studied. In the first test, by using DE and measured states of unit step response (extension movement), the unknown parameters of the system are accurately determined. In Publication I, the second test shows the estimated parameters are valid in retraction movement, too and finally, the effect of friction in extension and retraction is presented.

In extension movement, the unit step response of the system carried out is reported. The input and five states of the system for off-line system identification were measured. All states of system were measurable, but the following states ($X_p, X_l, \ddot{X}_p, p_1, p_2$) were used. Using more states makes the estimation of unknown parameters more reliable. Note that the state of \ddot{X}_l was not measured. Then, by using Matlab Simulink environment, the mathematical model of the system based on the parameters was found and the Differential Evolution code was written in C/C++ language and used as Matlab S-Function. To run the program the population size

and cost function was chosen as follows: The minimum and maximum of population size is 2 and 100 times the dimensionality of the problem respectively, and a recommended size is 5-10 times that of the dimensionality of the problem, because the number of unknown parameters were 16, so the selected size of population was $5 \times 16 = 90$. At the first step, all unknown parameters were initialized in their lower and upper bounds randomly. Then, the amount of cost function for each population was calculated and minimized. The cost function is defined as follows:

$$F_{Co}(x_i, G) = \sum_{i=1}^{4113} \sum_{j=1}^5 e_j^2 + \sum_{i=1}^D P(x_i, G), \quad (7.1)$$

where, $P(x_i, G)$ is penalty function and e_j indicates the normalized error between measured and simulation for each sampling point. The penalty function of each element is defined as follow,

$$P(x_i, G) = \begin{cases} 20((x_{i(\min)} - x_i) / x_{i(\min)})^2 & \text{for } x_i < x_{i(\min)} \\ 0 & \text{for } x_{i(\min)} \leq x_i \leq x_{i(\max)} \\ 20((x_{i(\max)} - x_i) / x_{i(\max)})^2 & \text{for } x_i > x_{i(\max)} \end{cases}, \quad (7.2)$$

where, $x_{i(\min)}$ and $x_{i(\max)}$ are minimum and maximum amount of element x_i respectively. If the generated parameter is in the tolerance the amount of penalty function is zero. The number of sampling points in the real test is 4113, so the simulation runs 4113 times at the same sampling period (one millisecond). At the end of each run (4.113 seconds), the cost function is calculated.

Constraints typically make optimization harder for DE; however they can eliminate the local minima of the cost function. In the study, all of the unknown parameters have acceptable ranges, so during the simulation run; all the child vector parameters must be in that ranges.

The important factors in DE are population size, F_1 and CR . The iterations number is incredibly related to these factors. If the size of population is small then saturation will happen and the DE can not find the global minimum of the cost function.

In general, F_1 and CR affect the convergence speed and robustness of the search process. Their optimal values depend both on objective function characteristics and the population size, NP , and thus, the selection of optimal parameter values is an application dependent task.

The results show that DE can find the optimum parameters to minimize the cost function. Depending on the expected value of the cost, the DE will find the proper values for unknown parameters.

Figure 2 shows the amount of cost against its iterations number. At earlier iterations the amount of the cost is large, but the algorithm finds new child generations with lower cost. As shown in figure 2, the minimum cost for the initial population is 35.6 and it is 5.6 when the number of iteration is 1484. The minimum cost approaches 1.28 at 29586 iterations and it will be constant, so 1.28 is the global minimum for the proposed cost function. The number of successful generations was 2073 times during 46020 iterations. Table 7.1 shows the best values for unknown parameters.

Figures 7.2 to 7.6 are plots of system states. It is obvious that the simulation and experimental results are close to each other and DE is a strong optimizer algorithm in finding the best parameters which constrain in nonlinear systems.

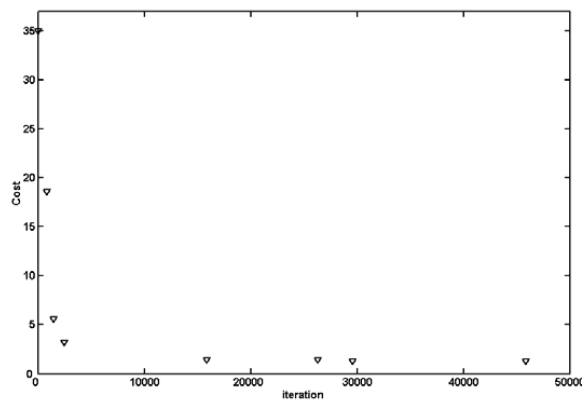


Fig. 7.1: Cost of system against the number of its iterations.

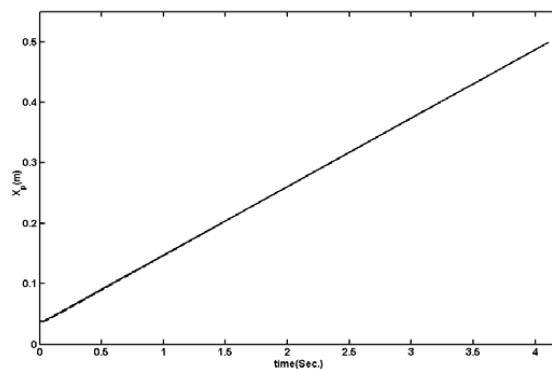


Fig. 7.2: Mass load position, experimental (solid line) and simulation (dash line).

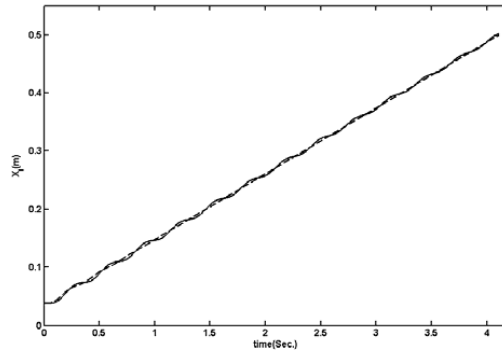


Fig. 7.3: Flexible load position, experimental (solid line) and simulation (dash line).

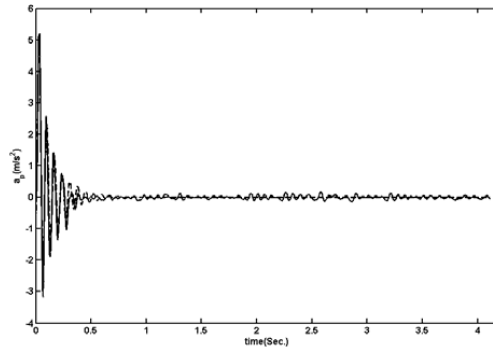


Fig. 7.4: Mass load acceleration, experimental (solid line) and simulation (dash line).

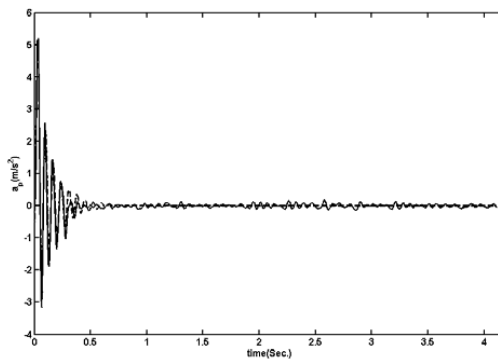


Fig. 7.5: First pressure, experimental (solid line) and simulation (dash line).

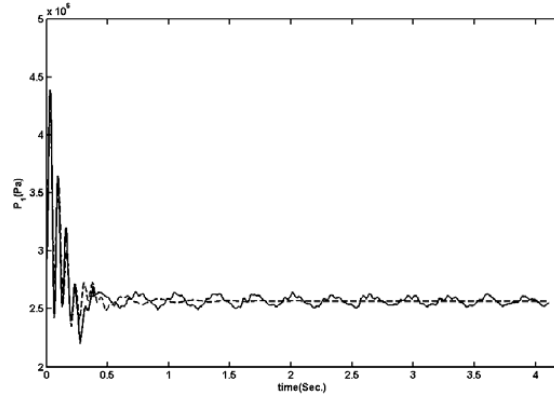


Fig. 7.6: Second pressure, experimental (solid line) and simulation (dash line).

Table 7.1: The values of unknown Parameters.

$\tau_1 = 470$	$\tau_2 = 0.490 \text{ 1/s}$
$b = 418.335 \text{ Ns/m}$	$Le1 = 2.401 \times 10^{-12}$
$c_v = 2.36 \times 10^{-5} \text{ m}^3 / (\text{sVPa}^{1/2})$	$Le2 = 1.029 \times 10^{-13}$
$Li = 1724 \times 10^{-13}$	$a_1 = 0.1972$
$a_2 = 124.368$	$a_3 = 26.814$
$\sigma_0 = 97525.459 \text{ m/s}$	$\sigma_1 = 385.167 \text{ Ns/m}$
$K_v = 376.613 \text{ Ns/m}$	$F_c = 247.804 \text{ N}$
$F_s = 7485.084 \text{ N}$	$V_s = 0.026318 \text{ m/s}$

7.1.2 Conclusions

In this study, the application of DE to solve the minimal representation problem in system identification of the servo-hydraulic system with a flexible load has been studied. The results demonstrated that DE can accurately identify the time delay, structure and parameters of the system with a fast convergence rate.

Constraints typically make optimisation harder for DE. They divide the search space into several disjoint islands. On the other hand, constraints can eliminate local

minima that might otherwise trap vectors, thereby reducing the chance that DE will prematurely converge.

The important factors in DE are population size, F_1 and CR. The iteration number is incredibly related to these factors. If the size of population is small then saturation will happen and the DE can not find the global minimum.

The advantage of DE over conventional system identification methods are simplicity, speed of calculation and convergence, even for identifying a number of unknown parameters with minimizing several functions at the same time.

7.2 Fuzzy Gain-Scheduling

In the study, the application of the fuzzy gain-scheduling controller in a servo hydraulic system was studied.

7.2.1 Results

In the study a fuzzy gain-scheduling to track the reference model is proposed. To verify the performance of the proposed controller, simulation and experimental results are compared with a commonly used p-controller. Both controllers are used in feedback acceleration to increase the system damping.

The desired input (X_d) is a pulse input with an amplitude of 0.2 (m) and period of 4 (sec.).

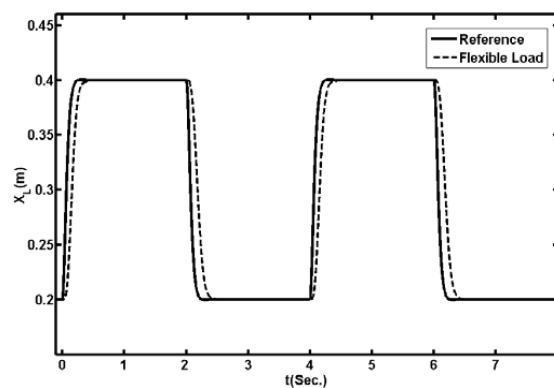


Fig. 7.7: Simulation result of flexible load tracking using the proposed controller.

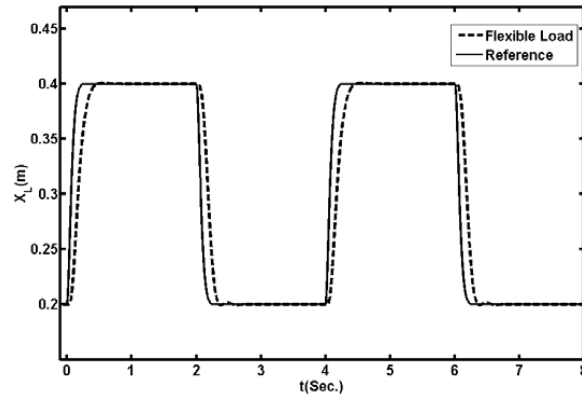


Fig. 7.8: Simulation result of flexible load tracking using a P-Controller.

In the section, the results of flexible tracking for both controllers are provided. In Publication II, the results (simulation/experiment) for piston load and flexible load are provided.

Figure 7.7 is the tracking of the reference model by the flexible load using the proposed controller. The result is satisfactory and illustrates the capability of the fuzzy controller in tracking of the reference model.

Figure 16 indicates the behaviour of the p-controller in tracking the reference model.

As illustrated in Fig. 7.8, the p-controller has good tracking in the extension and retraction movements, but there is a very small vibration. The Summation of the Absolute Amount of Error (SAAE) during one period (4 sec.) for each controller is calculated. The SAAE amounts are 69.7 and 74.17 for the proposed controller and modified p-controller respectively, so the proposed controller has 6.4 percent decrease in the amount of SAAE.

7.2.2 Conclusions

In the study, the application of the fuzzy gain-scheduling controller in a servo hydraulic system was studied. The results suggest that the proposed controller with an acceleration compensator shows good performance compared to the modified p-controller.

The architecture of the fuzzy controller is essential key to overcome the change in the systems dynamics during the extension and retraction tracking.

In light of the above, using feed-forward acceleration to compensate the dynamics of the hydraulic systems is an important factor to improve the performance of controllers. The results suggest that the proposed controller is very fast and more accurate compared to the P-controller in the tracking the reference model and the mount of the Summation of the Absolute Amount of Error (SAAE) for the proposed controller is 6.4 percent less than for the modified p-controller.

The comparison between the simulation results and the experimental results suggest that fuzzy gain scheduling is one of the best controllers for servo hydraulic systems.

7.3 Neural network in PMLM

In the study, a recurrent neural network hybrid controller for a PMLSM is introduced and successfully implemented in the physical linear motor application.

7.3.1 Results

In the study, a recurrent neural network hybrid controller for a PMLSM is introduced and successfully implemented in the physical linear motor application. The motor velocity is controlled by a conventional PI controller and the vibration of the load is suppressed by using the recurrent neural network compensator.

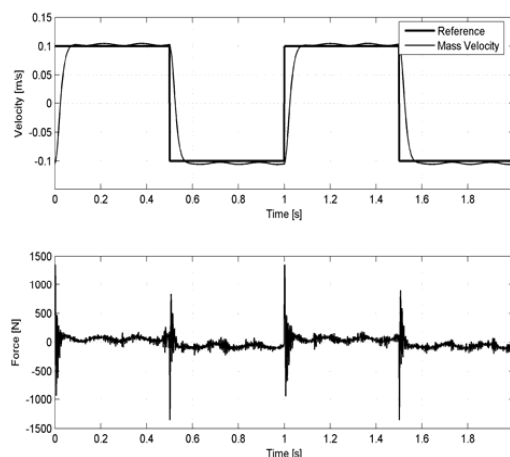


Fig. 7.9: Measured velocity of the load, and the driving force of the motor in the case of a neural network compensator and under a step excitation.

In the study by using the differential evolution, the final weights of neural network have been found. After the weights are finalized, they are transferred to the neural network compensator in the experimental setup. The states for the neural network compensator are estimated using the Kalman filter.

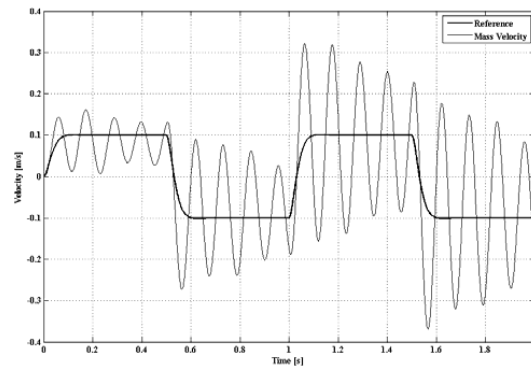


Fig. 7.10: Simulation of mass load velocity using PI controller without any compensator.

Figures 7.9 and 7.10 show a comparison of the velocity responses in the case of PI controller with proposed compensator and without any compensator respectively. The velocity of the load follows the reference command accurately; even the system stiffness is relatively loose. In this case, the reference has been a pulse function, the amplitude of which is 0.2 m/s and the frequency 1 Hz.

7.3.2 Conclusions

In the study, a recurrent neural network hybrid controller for a PMLSM is introduced and successfully implemented in the physical linear motor application. The motor velocity is controlled by a conventional PI controller and the vibration of the load is suppressed by using the recurrent neural network compensator. The differential evolution algorithm proved to be an efficient tool for finding initial weights for the recurrent neural network. More traditional teaching methods might have a local minimum problem due to their gradient based calculation, i.e. they frequently lead to local optimums. DE is not based on the gradient method, and therefore, the local minimum problem is avoided. The problem of DE is its speed in approaching the global minimum, and therefore, it is useful only in offline training. The

combination of a neural network hybrid controller and a state estimator reduces the vibration of the load considerably, and the proposed controller is perceived to be stable in all conditions.

7.4 Adaptive Backstepping in PMLM

In the study, adaptive backstepping for position tracking in a PMLSM is introduced and successfully implemented in the physical linear motor application. The damping ratio of the system is small (6 Ns/m), so to simplify the applying method, it is set to zero.

7.4.1 Results

In the study, adaptive backstepping controller is used to position tracking of a reference model for a PMLSM system. In the study the so called *observer based backstepping* controller is used. This means that unavailable states are estimated for the controller. The position and velocity of the load and the velocity of the motor that are needed in the proposed controller are estimated by using the Kalman filter.

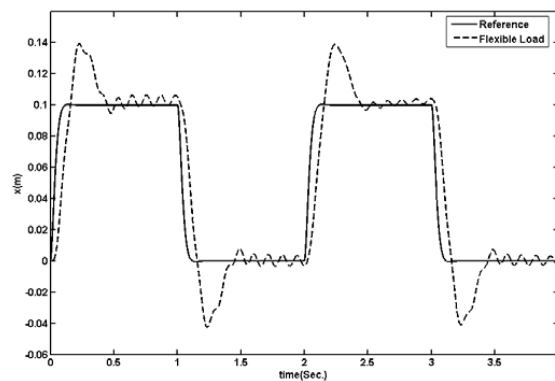


Fig. 7.11: Simulation result from position of the load in the case of PI control.

Figures 7.11 and 7.12 show a comparison of the position tracking responses in the case of the PI controller and the proposed load controller. Figure 7.11 corresponds to the simulation result of the flexible load position tracking by using a PI controller. The performance of this controller is not good and the flexible mechanism starts to vibrate. In Figure 7.12 there is a measured position tracking response of the flexible load when the adaptive backstepping is used. The load tracks the reference well and

steady state error is zero; even the system stiffness is relatively loose. In both cases the reference has been a step function, the amplitude of which is 0.1 m and the frequency 0.5 Hz.

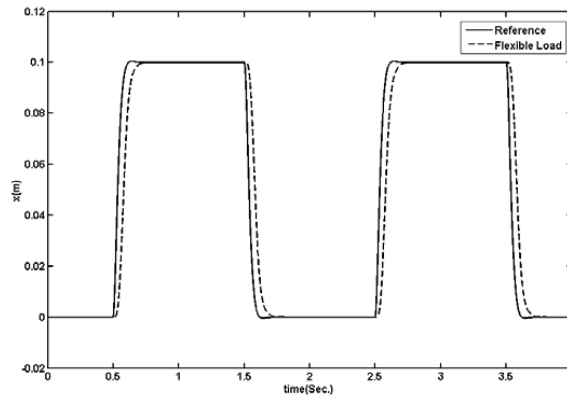


Fig. 7.12: Measured position of the load in the case of a Backstepping controller under a step excitation.

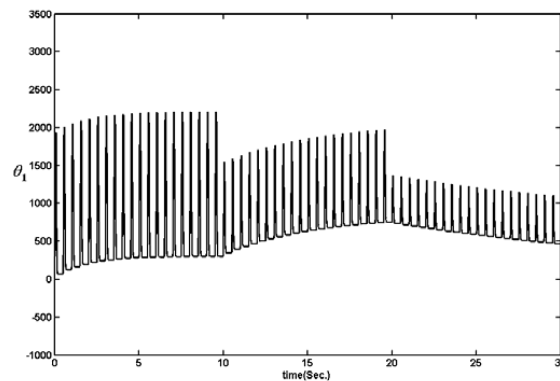


Fig. 7.13: The dynamics of the adaptation law in the case of varied load mass and spring constant.

The dynamics of the adaptation law is tested with the simulation model, which is excited with the step command, the amplitude of which is 0.1 m and the frequency 1 Hz. In the simulation the mass of the load is changed from 2 kg to 5 kg at 10 seconds, and the spring constant is changed from 13700 N/m to 6700 N/m at 20 seconds in the middle of the simulation. Figures 7.13 and 7.14 show the time varying

estimations of unknown parameters θ_1 and θ_2 from the previously mentioned simulation. From the figures we can perceive how fast the controller adapts new parameters. The interesting phenomenon in the proposed backstepping controller is that the estimated values are converging, but not to the right values. This is due to estimation equations, which are differential functions of the controller states and parameters.

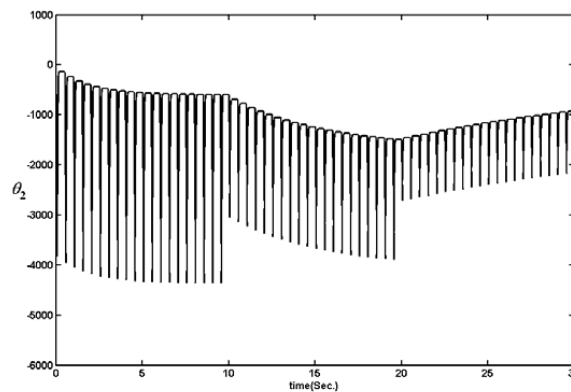


Fig. 7.14: The dynamics of the adaptation law in the case of varied load mass and spring constant.

7.4.2 Conclusions

In the study, a load control method for a PMLSM is introduced and successfully implemented in the physical linear motor application. The motor is controlled by the adaptive backstepping controller. The position and velocity of the load and the velocity of the motor for the controller are estimated using the Kalman Filter. As shown, the performance of the PI-Controller is poor at tracking the reference model. The cogging and friction forces are considered as disturbances, so the estimated parameters are converging, but because of the disturbance they do not approach to the final values. The vibration of the load is removed and the proposed controller is perceived to be stable in all conditions. The system model for the Kalman Filter can be described by using a transient or frequency response to define offline mass and load states. The proposed controller is designed base on the pulse reference input, so its derivative is zero when its amplitude is constant (most of the working time). The results of using pulse and sine inputs suggest that the above assumption is valid.

Bibliography

- Abbass HA., 2002.** An Evolutionary Artificial Neural Networks Approach for Breast Cancer Diagnosis. *Artificial Intelligence in Medicine*, (25):265–281.
- Ahmed, M.S., Bhatti, U.L., Al-Sunni, F.M. and El-Shafei, M., 2001.** Design of a fuzzy servo-controller, *Fuzzy Sets Syst*, (124):231–247.
- Akhyar, S. & Omatsu, S., 1992.** Neuromorphic self-tuning PID controller, in Proc. of IEEE International Conference on Neural Networks, 552-557.
- Akkizidis, I.S., Roberts, G.N., Ridao, P. and Batlle, J., 2000.** Designing a fuzzy-like PD controller for an underwater robot, *Contr Eng Pract*, (11):471–480.
- Alleyne, A., 1996.** Nonlinear force control of an electro-hydraulic actuator Proceedings of Japan/USA Symposium on Flexible Automation, (1):193–200.
- Antonio, C.C. & Pacifico, M.P., 2000.** Fuzzy control of heat recovery systems from solid bed cooling, *Appl Therm Eng*, (20):49–67.
- Åström K, Wittenmark B., 1995.** Adaptive Control. USA: Addison-Wesley.
- Avila, A. M., Loukianov G. A. and Sanchez, N. E., 2004.** Electro Hydraulic Actuator Trajectory Tracking, Proceeding of the American Control Conference, 2603-2608.
- Axelson, S. & Kumar, K., 1988.** Dynamic feedback linearization of a hydraulic valve actuator combination. *Proceedings of American Control Conference*, 2202–2205.
- Babu B.V. & Sastry KKN., 1999.** Estimation of Heat Transfer Parameters in a Trickle-Bed Reactor Using Differential Evolution and Orthogonal Collocation. *Computers and Chemical Engineering*, 23(3):327–339.
- Babu, B.V. & Angira, R., 2006.** Modified differential evolution (MDE) for optimization of non-linear chemical processes. *Computers and Chemical Engineering*, (30):989–1002.
- Babu BV & Jehan MML., 2003.** Differential Evolution for Multi-Objective Optimization. Proceedings of Conference on Evolutionary Computation (CEC-2003), Canberra, Australia, 2696–703.
- Bartos, F.J., 1996.** Fuzzy logic reaches adulthood, *Contr Eng*, (43) (10):50–56.
- Bergey PK., 1999.** An Agent enhanced Intelligent Spreadsheet Solver for Multi-Criteria Decision Making. *Proceedings of AIS AMCIS 1999: Americas Conference on Information Systems*, Milwaukee, USA, 966–8.

- Blackburn, J. F., Reethof, G. and Shearer, J. L., 1960.** Fluid Power Control. Massachusetts: The MIT Press.
- Booty WG, Lam CL, Wong IWS and Siconolfi P., 2000.** Design and Implementation of an Environmental Decision Support System. *Environmental Modeling and Software*, **16**(5):453–458.
- Branštetter, P. & Skotnica, M., 2000.** Application of artificial neural network for speed control of asynchronous motor with vector control, Proceedings of International Conference of Košice, EPE-PEMC 2000, (6):157-159.
- Burton, B., Harley, R. G.G. and Diana, J. L., 1998.** Rodgerson: Implementation of a Neural Network to Adaptively Identify and Control VSI-Fed Induction Motor Stator Currents, *IEEE Transaction on Industry Applications*, **34**(3).
- Cafolla AA., 1998.** A New Stochastic Optimization Strategy for Quantitative Analysis of Core Level Photoemission Data. *Surface Science*, **402–404**,561–565.
- Carelli, R., Camacho, E. F. and Patino, D., 1995.** A neural network based feedforward adaptive controller for robots, *IEEE Trans. Syst., Man, Cybern.*, (25):1281-1288.
- Chang TT & Chang HC, 2000.** An Efficient Approach for Reducing Harmonic Voltage Distortion in Distribution Systems with Active Power Line Conditioners. *IEEE Transactions on Power Delivery*, **15**(3):990–995.
- Chen, C.H. & Chang, M.H., 1998.** Optimal design of fuzzy sliding-mode control: a comparative study, *Fuzzy Sets Syst*, (93):37–48.
- Chen, F.-C. & Khalil, H. K., 1992.** Adaptive control of nonlinear systems using neural networks, *Int. J. Contr.*, **55**(6):299-1317.
- Chen, F.-C. & Khalil, H. K., 1995.** Adaptive control of a class nonlinear discrete-time system using neural networks, *IEEE Trans. Automat. Contr.*, **40**(5):791-801.
- Cheng SL & Hwang C., 2001.** Optimal Approximation of Linear Systems by a Differential Evolution Algorithm. *IEEE Transactions on Systems, Man and Cybernetics*, Part A, **31**(6):698–707.
- Chou, C.H. & Teng, J.C., 2002.** A fuzzy logic controller for traffic junction signals, *Inform Sci* **143**(2):73–97.

- Chun, JS., Jung, HK., Jung, JS., 2000.** Analysis of the end-effect of permanent magnet linear synchronous motor energized by partially excited primary. *ICEM, International Conference on Electrical Machines*, (1):333-337.
- Dawson D, Carroll J, Schneider M., 1994.** Integrator backstepping control of a brush DC motor turning a robot load. *IEEE Trans. on Control System Technology* 2(3):233-244.
- Elman, J., 1990.** Finding structure in time. *Cognitive Sci.* (14):179–211.
- El-Sherbiny, M.K., El-Saady, G. and Yousef, A.M., 2002.** Efficient fuzzy logic load frequency controller, *Energy Convers Manage* (43):1853–1863.
- Fliess M, Lévine J, Martin P, Rouchon P., 1995.** Flatness and Defect of Nonlinear Systems: Introductory Theory and Examples. *International Journal of Control*, 61(6):1327-1361.
- Fraichard, T. & Garnier, P., 2001.** Fuzzy control to drive car-like vehicles, *Robot Autonom. Syst.* (34):1–22.
- Furuhashi T., Sangwongwanich S., Okuma S., 1992.** A Position-and-Velocity Sensorless Control for Brushless DC Motor Using an Adaptive Sliding Mode Observer. *IEEE Trans. on Industrial Electronics*, 39(2):89-95.
- Ghiaus, C., 2000.** Fuzzy model and control of a fan-coil, *Energy Build*, (22):545–551.
- Goodson, R.E. & Leonard, R.G., 1972.** A survey of modelling techniques for fluid transients. *Journal of Basic Engineering, Trans. ASME*, 94(2):474–482.
- Harris, J., 2006.** *Fuzzy Logic Applications in Engineering Science*. Springer.
- Hirvonen, M., 2006.** On the analysis and control of a linear synchronous servomotor with a flexible load. *Acta Universitatis Lappeenrantaensis* 257.
- Horiuchi, J.I. & Kishimoto, M., 2002.** Application of fuzzy control to industrial bioprocesses in Japan, *Fuzzy Sets Syst*, (128):117–124.
- Hung, J.Y., Gao, W. and Hung, J.C., 1993.** Variable structure control a survey, *IEEE Transactions on Industrial Electronics* 40(1): 2–22.
- Hyun, DS., Jung, IS., Shim, JH., Yoon, SB., 1999.** Analysis of forces in a short primary type permanent magnet linear synchronous motor. *IEEE Trans. Energy Conversion*, 14(4):1265-1270.

- liguni, Y., Sakai, H. and Tokumaru, H., 1991.** A nonlinear regulator design in the presence of system uncertainties using multilayered neural network, *IEEE Trans. Neural Networks*, **2**(4):410-417.
- Jalali, M. and Kroll, A., 2004.** Hydraulic Servo-Systems; Modeling, Identification and Control, Springer.
- Jamshidi, M., Vadiie, N. and Ross, T.J., 1993.** Fuzzy logic and control: software and hardware applications, Prentice-Hall, Englewood Cliffs, NJ.
- Jordan, M. I., and Jacobs, R.A., 1990.** Learning to control an unstable system with forward modeling. D. Touretzky (Ed.), *Advances in Neural Information Processing Systems 2*. San Mateo, CA: Morgan Kaufmann.
- Jordan, M.I. and Rumelhart, D.E., 1992.** Forward models: Supervised learning with a distal teacher. *Cognitive Science*, **16**(3):307-354
- Jordan, M.I., 1986a.** Attractor dynamics and parallelism in a connectionist sequential machine. *In Proc. of the Eighth Annual Conference of the Cognitive Science Society*, 531-546.
- Jordan, M.I., 1986b.** A parallel distributed processing approach. *Technical report, Institute for Cognitive Science*, University of California.
- Khalid, M. & Omatu, S., 1995.** Temperature regulation with neural networks and alternative control schemes, *IEEE Trans. Neural Networks*, **6**(3):572-582.
- Kokotovic P, Arcak M., 2001.** Constructive nonlinear control: a historical perspective. *Automatica*, (37) 637-662.
- Kokotovich P., 1992.** The joy of feedback: Nonlinear and adaptive. *Control System Magazine*, (12):7-17.
- Kremer, G.G., & Thompson, D. F., 1998.** A bifurcation-based procedure for designing and analyzing robustly stable nonlinear hydraulic servo systems. *Proceedings of the Inst. of Mechanical Engineers: Part I, J. of Systems and Control Engr.*, (212):383-94.
- Krink, T. Filipic, B. and Fogel, G.B., 2004.** Thomsen, R. Noisy Optimization Problems. A Particular Challenge for Differential Evolution. *Proceedings of Congress on Evolutionary Computation*, (1): 332-339.
- Krstic M., Kanellakopoulos I., Kokotovic P., 1995.** Nonlinear and adaptive control design. USA: John Wiley & Sons.

- Krus, P., Weddfelt, K. and Palmberg, J.O., 1994.** Fast pipeline models for simulation of hydraulic systems. *Journal of Dynamic Systems, Measurement and Control*, (116):132-136.
- Ku, C. & Lee, K. Y., 1995.** Diagonal recurrent neural networks for dynamic systems control, *IEEE Trans. Neural Networks*, 6(1):144-156.
- Kugi, A. & Schlacher, K., 2002.** Non-linear control of a hydraulic piston actuator without velocity information. *Proceedings of Applied Mathematics and Mechanics*, 1(1):105-106.
- Kuo, K.Y. & Lin, J., 2002.** Fuzzy logic control for flexible link robot arm by singular perturbation approach, *Appl Soft Comput* (2):24–38.
- Jung, S.Y., Jung, H.K., 2002.** Reduction of force ripples in permanent magnet linear synchronous motor. *Proceedings of International Conference on Electrical Machines*, Brugge, Belgium, pp. 452.
- Lee, S.Y. & Cho, H.S., 2003.** A fuzzy controller for an electro-hydraulic fin actuator using phase plane method, *Contr Eng Pract.*, (11):697–708.
- Leonhard, W., 1995.** Control of electrical drives (2nd.), *Springer, Berlin*.
- Levin, A. U. & Narendra, K. S., 1996.** Control of nonlinear dynamical systems using neural networks - part II: Observability, identification, and control, *IEEE Trans. Neural Networks*, 7(1):30-42.
- Lewis, E. E. & Stern, H., 1962.** Design of Hydraulic Control Systems. McGraw Hill.
- Lewis, F. L., Lui, K., and Yesildirek, A., 1995.** Neural net robot controller with guaranteed tracking performance, *IEEE Trans. Neural Networks*, 6(3):703-715.
- Lewis, F. L., Yesildirek, A. and Lui, K., 1996.** Multilayer neural-net robot controller with guaranteed tracking performance, *IEEE Trans. Neural Networks*, 7(2):388-399.
- Lin F-J, Chiu S-L, Shyu K-K., 1998.** Novel Sliding Mode Controller for Synchronous Motor Drive. *IEEE Trans. on Aerospace and Electronic Systems*, 34(2):532-542.
- Lin J-S, Kanellakopoulos I., 1997.** Nonlinear design of active suspensions. *IEEE Control Systems Magazine* 17(3):45-59.
- Lin, F.C. & Yang, S.M., 2003.** Adaptive fuzzy-logic velocity observer for servo motor drives, *Mechatronics*, (13):229–241.
- Lin, S. & Aker, A., 1991.** Dynamic analysis of a flapper nozzle valve. *Journal of Dynamic Systems, Measurement and Control*, (113):163–167.

- Lischinsky, P., de Wit, C.C. and Morel, G., 1999.** Friction Compensation for an Industrial Hydraulic Robot. *IEEE Control Systems*, p.25-32.
- Mamdani, E. and Assilian, S., 1975.** An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies*, 7:(1):1-13.
- Mamdani, E.H., 1974.** Application of fuzzy algorithms for control of simple dynamic plant, *Proc IEE*, 121(12):1585–1588.
- Mayer, D.G., Kinghorn, B.P., and Archer, A.A., 2005.** Differential Evolution –An Easy and Efficient Evolutionary Algorithm for Model Optimisation. *Agricultural Systems*, (83):315–328.
- McCloy, D. & Martin, H. R., 1973.** The control of fluid power. *Longman*.
- Merritt, H. E., 1967.** Hydraulic control systems. John Wiley and Sons.
- Minsky, M., and Papert, S., 1969.** Perceptrons: An Introduction to Computational Geometry. The MIT Press. [Cited on pp.13, 26, 31, and 33].
- Mohamed, A.Z., Eskander, M.N. and Ghali, F.A., 2001.** Fuzzy logic control based maximum power tracking of a wind energy system, *Renew Energy*, (23):235–245.
- Munson, B.R., Okiishi, T. H. and Young, D. F., 1996.** A Brief Introduction to Fluid Mechanics. John Wiley and Sons Inc.
- Narendra, K. S. & Parthasarathy, K., 1990.** Identification and control of dynamical systems using neural networks, *IEEE Trans. Neural Networks*, 1(2):4-27.
- Olsson, H., Åström, KJ., de Wit, CC., Gäfvert, M., Olsson, PL., 1998.** Friction Models and Friction Compensation. *European Journal of Control*, 4(3):176-195.
- Otten, G., de Vries, TJA., van Amorengen, J., Rankers, AM., Gaal, EW., 1997.** Linear motor motion control using a learning feedforward controller. *IEEE/ASME Trans. Mechatron*, 2(3):179-187.
- Owen, W. S. and Croft, E. A., 2003.** The Reduction of Stick-Slip Friction in Hydraulic Actuators. *IEEE/ASME Transactions on Mechatronics*, 8(3), 362-371.
- Parker, D. B., 1985.** Learning-Logic (Tech. Rep. Nos. TR{47}. Cambridge, MA: Massachusetts Institute of Technology, Center for Computational Research in Economics and Management Science. [Cited on p.33].
- Passino, K. M. , Yukovich, S., K., 1998.** Fuzzy Control, *Addison-Wesley*.
- Pham D. T. and OH S. J., 1993.** Adaptive Control of Dynamic Systems Using Neural Networks. *Proc. Of 1993 Int. Conf. on Systems, Man and Cybernetics*, (4):97-102.

- Ploycarpou, M. M. & Helmicki, A. J., 1995.** Automated fault detection and accommodation: A learning systems approach, *IEEE Trans. Syst., Man, Cybern.*, (25)1447-1458.
- Plummer, A.R. & Vaughan, N.D., 1996.** Robust adaptive control for hydraulic servosystems. *Journal of Dynamic Systems, Measurements and Control*, 118(2):237–244.
- Price, K. V., Storn, R. M., Lampinen, J. A., 2005.** Differential Evolution: A Practical Approach to Global Optimization. Berlin. Springer.
- Radakovic, Z.R., Milosevic, V.M., and Radakovic, S.B., 2002.** Application of temperature fuzzy controller in an indirect resistance furnace, *Appl Energy*, (73):167–181.
- Rosenblatt, F., 1958.** The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain, Cornell Aeronautical Laboratory, *Psychological Review*, 65(6):386-408.
- Rovithakis, G. A. & Christodoulou, M. A., 1994.** Adaptive control of unknown plants using dynamical neural networks, *IEEE Trans. Syst., Man, Cybern.*, (24):400-412.
- Rovithakis, G. A. & Christodoulou, M. A., 1995.** Direct adaptive regulation of unknown nonlinear dynamical systems via dynamic neural networks. *IEEE Trans. Syst., Man, Cybern.*, (25):1578-1594.
- Rumelhart, DE., Hinton, GE., Williams, RJ., 1986.** Learning Internal Representations: by Error Propagation. *Parallel Distributed Processing Explorations in the Microstructures of Cognition* (1):318-362.
- Schmitz GPJ. & Aldrich C., 1998.** Neuro-Fuzzy Modeling of Chemical Process Systems with Ellipsoidal Radial Basis Function Neural Networks and Genetic Algorithms, *Computers & Chemical Engineering*, (22):1001–1004.
- Schwefel, H.P., 1995.** Evolution and Optimum Seeking, John Wiley.
- Shieh, H.J. & Shyu, K.K., 1999.** Nonlinear sliding-mode torque control with adaptive backstepping approach for induction motor drive, *IEEE Transactions on Industrial Electronics* 46 (2):380–389.
- Shukla, A., 2002.** Stability analysis and design of servo-hydraulic systems. *PhD thesis*, University of Cincinnati.

- Slotine, J.J. & Li, W., 1991.** Applied nonlinear control, Prentice-Hall, New Jersey.
- Slotine, J.-J. E. & Li, W., 1990.** Applied nonlinear control. *Englewood Cliffs*.
- Storn R., 1999.** System Design by Constraint Adaptation and Differential Evolution. *IEEE Transactions on Evolutionary Computation*, **3**(1):22–34.
- Storn, R. & Price, K.V., 1995.** Differential evolution - a Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces, *Technical Report TR-95-012, ICSI*.
- Stumberger G, Pahner U and Hameyer K., 2000.** Optimization of Radial Active Magnetic Bearings Using the Finite Element Technique and the Differential Evolution Algorithm. *IEEE Transactions on Magnetics*, **36**(4):1009–13.
- Vas, P., 1999.** Artificial-Intelligence-based Electrical Machines and Drives. *Oxford University Press, Oxford*.
- Verbruggen, H.B. & Bruijn, P.M., 1997.** Fuzzy control and conventional control: What is (and can be) the real contribution of fuzzy systems?, *Fuzzy Sets Syst*, (90):151–160.
- Viersma, T. J., 1980.** Analysis, synthesis, and design of hydraulic servosystems and pipelines. *Elsevier Scientific Pub. Co*.
- Vossoughi, G. & Donath, M., 1995.** Dynamic feedback linearization for electrohydraulically actuated control systems. *Dynamic Systems, Measurement and Control*, (117):469–477.
- Wai, R.J., Lin, C.H. and Hsu, C.F., 2004.** Adaptive fuzzy sliding-mode control for electrical servo drive, *Fuzzy Sets Syst* (143) (2):295–310.
- Wang, L.X., 1994.** Adaptive fuzzy systems and control: design and stability analysis, *Prentice-Hall, Englewood Cliffs*.
- Watton, J., & Braton, R., 1985.** Further contributions to the response and stability of electrohydraulic servo actuators with unequal areas - part1: System modeling. *Dynamic systems. ASME Journal of Modeling and Control*, (1):155–160.
- Weisstein, E. W. and Vassilis, P., 2006.** Differential Evolution. *From MathWorld, A Wolfram Web Resource* (<http://mathworld.wolfram.com/DifferentialEvolution.html>).
- Werbos, P.J., 1974.** Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences. Un published doctoral dissertation, Harvard University. [Cited on p. 33].