Analysis of the hierarchical fuzzy control using evolutionary algorithms

by

Juliusz Zajaczkowski

M.Sc.

A thesis submitted to:

School of Computing Sciences

Faculty of Arts, Business, Informatics and Education

CQUniversity

in fulfillment of the requirements for the degree of Doctor of Philosophy

April 2010

Abstract

The research presented in this thesis examines the construction of a fuzzy logic controller for complex nonlinear system by control system decomposition into hierarchical fuzzy logic sub-systems. This decomposition reduces greatly the number of fuzzy rules to be defined and addresses the problem of exponential increase of fuzzy rules with an increase in the number of input variables or membership functions, so called 'The curse of dimensionality'. The decomposition is not unique and often gives rise to variables with no physical significance. This fact causes difficulties in obtaining a complete class of rules from experts. Hence, a learning algorithm needs to be employed to learn fuzzy rules, for example an evolutionary algorithm.

In this thesis evolutionary algorithm (EA) based methods are proposed to determine the control system for the hierarchical fuzzy system (HFS). Evolutionary algorithm modifications include EA operators, such as crossover, mutation schedules, elitism, and other methods of passing the learned knowledge to the next generation. Variations on objective function formulation are also considered.

Different HFS topologies for a given dynamical system (such as the inverted pendulum system) are investigated and the controller (designed by the EA) performance is examined. Investigation into hierarchical structures is performed on the inverted pendulum system as the case study. For this particular dynamical system, a single layer, two layered, three layered, and four layered HFS, with different variable input configuration is investigated. Effects of different input configurations on controller performance are examined and discussed.

A new evolutionary algorithm based compositional method is proposed to control system over the whole set of user-defined initial conditions. The method addresses directly the problem of controlling the dynamical system from specific, user-defined initial conditions. In many practical applications there is no necessity to secure controllability over the large region in the state space, which is often difficult to achieve. Instead, a selected region of the state space, or even specific initial conditions can be considered. Further investigation is conducted into improvement of the compositional method by the use of the multiobjective optimisation formalism. The multiobjective evolutionary algorithm (MOEA) based compositional method is developed and tested on the example of the inverted pendulum system. The MOEA based compositional method provides good controller performance over large set of initial conditions and can be extended to other fuzzy logic control applications, especially in robotics applications.

Statement of originality

I certify that this thesis does not, to the best of my knowledge and belief:

1. Incorporate without acknowledgement any material previously submitted for a degree in any institution of higher degree;
2. Contain any material previously published or written by another person except where due references are made in the thesis text;
3. Contain any defamatory material.

Signature: [illegible]

Juliusz Zajaczkowski

Date: 21/03/2010

Acknowledgements

I would like to thank my supervisor, Prof. Brijesh Verma, for his contribution, for making possible completion of my thesis, and for his high and rigorous scientific standards that significantly contributed to the final form of this thesis. I would also like to thank my previous supervisor, Prof. Russel J. Stonier, for his role in the research work conducted for this thesis. Especially for his generous hospitality over long years it took to complete this research.

Many thanks to the High Performance Computing facility staff for their assistance in running all computer simulations for my research.

I would like to convey my gratitude to the Faculty of Arts, Business, Informatics and Education and the Research Office staff for their support, especially in the final stages of my PhD thesis submission, and also for making it possible for me to attend national and international conferences.

Last but not least, I would like to thank my parents, Czeslawa and Stanislaw Zajaczkowski, and my children Esmi and Matthew, for their unending support in completion of this thesis.

List of publications

1. Zajaczkowski, J, Verma, B. (2010). MOEA based hierarchical fuzzy control over the set of user-defined initial conditions. *2010 IEEE Conference on Fuzzy Systems, WCCI2010*, Barcelona, Spain (accepted).
2. Zajaczkowski, J, Verma, B. (2010). An evolutionary algorithm based approach for selection of topologies in hierarchical fuzzy systems. *2010 IEEE Congress on Evolutionary Computation, WCCI2010*, Barcelona, Spain (accepted).
3. Zajaczkowski, J, Verma, B. (2009). A compositional method using an evolutionary algorithm for finding fuzzy rules in 3-layered hierarchical fuzzy structure. *International Journal of Computational Intelligence and Applications IJCIA*, Vol. 8, No 4, pp. 467—485.
4. Zajaczkowski, J, Verma, B. (2009). A multiobjective evolutionary algorithm based compositional method for hierarchical fuzzy control. *ICMI 2009: International Conference on Machine Intelligence*, Bangkok, Thailand, pp. 1009—1016.
5. Zajaczkowski, J, Verma, B. (2008). Hierarchical fuzzy control for the inverted pendulum over the set of initial conditions. Xiaodong Li *et al*. eds *Simulated Evolution and Learning. LNCS 5361, SEAL2008*. Springer, Heidelberg, pp. 534—543.
6. Zajaczkowski, J., Stonier, R.J. (2008). Analysis of hierarchical control for the inverted pendulum. *Complexity International*, Vol. 12, Paper ID: msid49, URL: http://www.complexity.org.au/vol12/msid49/
7. Zajaczkowski, J., Stonier, R.J. (2006). Co-evolutionary algorithm for hierarchical fuzzy control of the inverted pendulum. *Proceedings of WCCI2006, IEEE International Conference on Fuzzy Systems*, Vancouver, Canada, pp. 737—744.
8. Zajaczkowski, J., Stonier, R.J. (2004). Analysis of hierarchical control for the inverted pendulum. *Proceedings of Complex2004*, Cairns, pp. 350—374.
9. Stonier, R.J., Zajaczkowski, J. (2003). Hierarchical fuzzy controllers for the inverted pendulum. *Proceedings of CIRAS 2003*, on CD ISSN: 0219-613, PS01-4-03, Singapore.
10. Stonier, R.J., Zajaczkowski, J. (2003). Model reference control using sliding mode with Hamiltonian dynamics. *The ANZIAM Journal (The Australian & New Zealand Industrial and Applied Mathematics Journal)* Vol. 45 Part E, Dec. 14, pp. E1—E40.

Contents

List of figures

List of tables

List of abbreviations and symbols

Abbreviations:

EA – Evolutionary algorithm

FLC – Fuzzy logic control

FS – Fuzzy system

GA – Genetic algorithm

HFS – Hierarchical fuzzy system

MF – Membership function

MIMO – Multiple-input multiple-output

MOEA – Multi-objective evolutionary algorithm

NN – Neural network

TR – Target region

Symbols:

A_i^l – normalised fuzzy set for input variable x_i corresponding to l^{th} fuzzy rule

B^l – normalised fuzzy set for output variable (u_1, u_2, or u) corresponding to l^{th} fuzzy rule

f_i – objective function or one of the objective MOEA functions

M_p – population size

M – number of elements in the individual string in the population

N_c – number of initial conditions in the user-defined set of initial conditions

N_{max} – the maximum number of iterations

n_T – tournament selection size

p_c – crossover probability

p_m – mutation probability

$\vec{p}_k$ – individual string from the population defined as $\vec{p}_k = (a_1, \ldots, a_M)$, where a_j is an integer $\in [1,7]$ and M is the size of the population

$P(t)$ – population at time t, $P(t) = \{ \vec{p}_k : k = 1, \ldots, M \}$

PF – Pareto optimal front

PS – Pareto optimal set

T – time for which the simulation of the system is run in objective function evaluation

T_f – time for which the controller is expected to act on the system in consideration

T_S − survival time

u – control output

u_i – intermediary control output from different layers in the hierarchical fuzzy structure

x_i − state variable

$\vec{x}$ − state vector

ω_i − positive weights in the objective function

Chapter 1 INTRODUCTION

1.1 Overview

Conventional modelling of physical systems is to develop a mathematical model and examine its behaviour by running computer simulations of the model. However, for many highly non-linear dynamical systems this method fails to adequately reflect physical reality. This is especially true in the dynamical systems that can develop chaotic characteristics. In other words, their dynamics matches chaotic motion. Examples of such systems can be found in robotic systems and large input-output data systems due to high non-linearity and large uncertainty in system parameters. A number of methods have been developed to address control problems in such systems. Some of them have become widely accepted, especially the artificial intelligence paradigms such as Neural Networks and Fuzzy Logic often coupled with Evolutionary Algorithms, see for example (Anderson 1989), (Thrift 1991), (Cheng *et al.* 1996), (Magdalena 1998), (Wang 1999), (Belarbi and Titel 2000), (Cordon *et al.* 2001a), (Hsu *et al.* 2001), (Lee *et al.* 2003), (Lei and Langari 2003), (Leung *et al.* 2003), (Kumar and Garg 2004), (Stonier and Mohammadian 2004), (Lin and Mon 2005), (Castillo *et al.* 2006).

Fuzzy control provides a practical alternative to conventional control techniques as it has capability to incorporate heuristic information, coming from human experts either in directly augmenting fuzzy rule base or by influencing automatic acquisition of fuzzy knowledge base. However, apart from the obvious advantages of fuzzy control techniques there might be also disadvantages and some of them were stated by K.M. Passino and S. Yurkowich as early as 1998, see (Passino and Yurkovich 1998).

Hierarchical fuzzy control is now a growing area of research in control systems. Increasing the number of input variables or input fuzzy sets results in an exponential increase in complexity of the rule base. The decomposition of the system into a layered or hierarchical fuzzy logic system is intended to reduce the size of the rule base while maintaining an adequate accuracy. The decomposition reduces greatly the number of fuzzy rules to be defined, as it is known that in the single-layered control system, the number of fuzzy rules exponentially increases with an increase in the number of input variables. It is so called 'The curse of dimensionality' that was first identified by Bellman (Bellman 1961).

The curse of dimensionality can be handled in variety of ways (Bellman 1961), (Wang 1997), (Stonier and Mohammadian 2004):

- Clustering input variables in the inference engine to reduce the number of rules in the knowledge base.
- Grouping the rules into prioritised levels to design hierarchical or multi-layered structures.
- Reducing the size of the inference engine directly using notions of passive decomposition of fuzzy relations.
- Decomposing the system into a finite number of reduced-order subsystems and thus eliminating the need for a large-sized inference engine.

The research presented in this thesis examines the construction of a fuzzy logic control system to control complex nonlinear system by its decomposition into hierarchical/multi-layered fuzzy logic sub-systems.

Layered fuzzy logic systems utilize the modularity characterizing many physical systems and their mathematical models. Specifically, in the context of fuzzy logic rule bases, the output influenced by one closely related group of input variables may be largely independent of the values of other variables. Therefore, a layered fuzzy logic system can decompose the rule base along the weak interdependence between state variables and still maintain a high level of accuracy. The decomposition is not unique and may give rise to variables with no physical significance. This can cause difficulties in obtaining a complete class of rules from human experts. Therefore, the rules need to be learnt by some learning algorithm, for example an evolutionary algorithm (EA). These fuzzy rules are typically evolved with no previous knowledge other than input-output data, or the physical system model.

Evolutionary algorithm, with various modifications, is used to determine the control system for the hierarchical fuzzy system (HFS). Evolutionary algorithm modifications include EA operators, such as crossover, mutation schedules, elitism, and other methods of passing the learned knowledge to the next generation. Variations on objective function formulation are also considered. Furthermore, the EA behaviour on the pre-define set of initial conditions is examined.

In this thesis different HFS topologies for a given dynamical system (such as the inverted pendulum system) are investigated and the performance of controllers designed by the EA for these topologies are examined. Examination of the topologies gives insight into the workings of the physical system and its control system. Investigation into hierarchical structures is performed on the inverted pendulum system as the case study. For this particular dynamical system, a single layer, two layered, three layered, and four layered HFS with different input configurations are examined and controllers' performances compared.

The curse of dimensionality affects both fuzzy logic and evolutionary algorithms. Its effect in fuzzy logic systems can be mitigated by using a hierarchical or multi-layered structure. The same problem needs to be addressed for evolutionary algorithms. One approach being so called co-evolutionary algorithms (De Jong and Potter 1995), (Pena-Reyes and Sipper 2001), (Young and Stonier 2003). This approach is investigated further in this thesis for fine-tuning of the fuzzy system membership functions.

The evolutionary algorithm based compositional method addresses a problem of designing the control system over the user-defined set of initial conditions. The method is successfully applied to the inverted pendulum system, see (Zajaczkowski and Verma 2008). Significant improvement in the controller performance was observed after implementing multiobjective optimisation formalism in the compositional method (Zajaczkowski and Verma 2009a). Application of the hierarchical fuzzy control and evolutionary algorithm based compositional method can be made to articulated robot arms and multi-robot systems in particular, although it has much wider application.

1.2 Significance of research

Fuzzy logic control methods have been successfully applied in industrial, scientific, and business-related problems. However, fuzzy systems suffer from dimensionality predicament, which limits their practical applications due to computational limitations of hardware/software. To enhance the usefulness of fuzzy logic control the research presented in this thesis investigates hierarchical fuzzy systems approach. The investigation focuses on two aspects:

- Overcoming high dimensionality problems by selecting fuzzy system representation scheme and by using hierarchical fuzzy structures.
- Application of evolutionary algorithms to learn hierarchical fuzzy system knowledge base by an automated process.

The presented research contributes to development of practical applications in robotics. The use of hierarchical fuzzy systems makes possible developing a fuzzy control system for real-time industrial applications, for example – a robotic manipulator. Standard single-layer fuzzy control systems require a large number of rules and thus an immense computing power that renders them impractical in real life applications. Approach to the control problem of the inverted pendulum presented in this thesis makes practical applications of fuzzy systems control more feasible.

The advantage of using hierarchical fuzzy systems is obvious in extreme case of decomposition of a given fuzzy logic system into its subsystems (standard input configuration, see Chapter 3), with two input variables in the first layer and one input per layer in all subsequent layers. In such a case, the size of the rule base is a linear function of the number of input variables.

The topologies of the hierarchical fuzzy systems are examined and analysed to address the problem of how input configuration in multi-layered structure affects the output, i.e., controller performance. There are various approaches to building a topology of the HFS and no consistent guidelines how to do it. In most cases it is human intuition or expert knowledge.

The problem of selecting the best topology and input configuration is examined on the example of the inverted pendulum system. Extensive examination of the inverted pendulum as a case study aims at developing methods for designing hierarchical fuzzy control system. Such a case study can be very instructive for extending some of the relevant results to other dynamical systems, in particular to a robotic manipulator. Research into inverted pendulum problem includes designing architectures of the HFS and testing the best configuration of the fuzzy system. Furthermore, it includes fine-tuning the evolutionary algorithm for the hierarchical fuzzy control.

The problem of designing fuzzy control system over a user-defined set of initial condition is investigated. Two methods can be used for this purpose: amalgamation and compositional method. Amalgamation method develops fuzzy rule knowledge base for every initial condition taken from a grid covering the region of state space that is under consideration and then amalgamates them into a single knowledge base. Compositional method develops knowledge base for the whole grid of initial conditions at once – by modifications done to a fitness function and tuning the evolutionary algorithm. In this thesis, a novel compositional method for hierarchical fuzzy system control has been developed and successfully applied to the inverted pendulum system.

Further improvements to the compositional method are investigated by applying multiobjective optimisation formalism to find the control system. The method is developed on the example of the inverted pendulum system but with wide range of possible applications. The multiobjective evolutionary algorithm based compositional method shows good controller performance with possible further improvements by implementation of a more sophisticated MOEA. The method shows potential for wide applications in robotics.

1.3 Aims of research

The main aim of the research presented in this thesis is to investigate the different topologies of the hierarchical fuzzy control system using evolutionary algorithms and to examine their impact on the controller performance. Specifically, the research presented in this thesis aims to:

- Investigate the different topologies of a hierarchical fuzzy control structure with single layer, two layers, three layers, and four layers with different variable input in the two layered, three layered, and four layered structures. Determine how hierarchical fuzzy structure affects controller performance.
- Investigate the role of membership functions in the co-evolutionary algorithm for the hierarchical fuzzy control.
- Propose and investigate an evolutionary algorithm based compositional method for hierarchical fuzzy control over the whole set of user-defined initial conditions.
- Propose and investigate a multiobjective evolutionary algorithm based compositional method for hierarchical fuzzy control over the whole set of initial conditions.

1.4 Original contributions

The original contributions of this research are as follows:

- Extensive investigation of hierarchical fuzzy structures for the inverted pendulum. Examination of the performance of the control system in different input configurations and hierarchical fuzzy structures. The result of such analysis is a controller with the ‘best’ performance that is established arbitrary for the task.
- Co-evolutionary approach to the hierarchical fuzzy control of the inverted pendulum system involving the membership functions adjustment for improved controller performance. Investigation into different methods of co-evolutionary mechanisms and their application to the fuzzy control.

- Evolutionary algorithm based compositional method for the hierarchical fuzzy control over the user-defined set of initial conditions and its testing on the example of the inverted pendulum system.
- The multi-objective approach to the problem of hierarchical fuzzy control over the set of initial conditions. Implementation of the multi-objective evolutionary algorithm for the compositional method.

1.5 Organisation of the thesis

This thesis is divided into nine chapters followed by the Appendix and References section.

Chapter 1: An introduction to the thesis is presented in this chapter. It provides an overview of the research field, thesis objectives, its original contribution to scientific knowledge, and thesis organisation.

Chapter 2: This chapter presents background knowledge and literature review.

Chapter 3: The inverted pendulum system is described as a test system. System dynamics is given as a set of first order differential equations. System parameters are defined. Finally, the hierarchical fuzzy systems and its application in fuzzy logic control are introduced.

Chapter 4: This chapter describes evolutionary algorithm that is used to find fuzzy rules for the hierarchical fuzzy control system. Single and multiobjective evolutionary algorithms are briefly introduced.

Chapter 5: The chapter provides detailed investigation of different topologies for the inverted pendulum system and their impact on the controller performance.

Chapter 6: The chapter describes co-evolutionary approach to the inverted pendulum problem that includes membership functions adjustments within the EA.

Chapter 7: This chapter presents an evolutionary algorithm based compositional method.

Chapter 8: The multiobjective optimization evolutionary algorithm based compositional method is introduced as an improvement on the single objective compositional method.

Chapter 9: The final chapter provides final conclusions and discussion of results. Future directions are briefly discussed.

Summary

In this chapter an overview of the thesis' topic is provided. Hierarchical fuzzy control is a growing area of research in control systems as it addresses an important issue in fuzzy control theory: increasing the number of input variables results in an exponential increase in the size of the rule base (the curse of dimensionality). The decomposition of the system into a hierarchical fuzzy logic system reduces the size of the rule base while maintaining an adequate accuracy.

Significance of research is discussed. Research in this thesis being focused on two major subjects: overcoming high dimensionality problems by using hierarchical fuzzy structures, application of evolutionary algorithms to learn hierarchical fuzzy system knowledge base.

Objectives and original contributions of the research presented in this thesis are stated. Finally, the organization of the thesis is given.

Chapter 2 BACKGROUND KNOWLEDGE AND LITERATURE REVIEW

2.1 Review of the existing techniques

2.1.1 Fuzzy logic control

The design of control systems for complex and high dimensional dynamical systems relies on the availability of a system model under consideration. It is often difficult to create an adequate model of the system or process due to a limited availability of mathematical theory in case of very complex systems. Approximate models are often employed in such cases but with the growing discrepancy between physical system and its mathematical (or experimental) model. However, very complex systems can be controlled by human operators with only a rudimentary knowledge of the dynamic model. This kind of control problems has given rise to new intelligent control methods, fuzzy logic and neural networks being most widely used.

Fuzzy logic origins lie in multi-valued logic concepts that were basis for the fuzzy theory developed by Lofti A. Zadeh. The theory gained wider approval in the scientific community in the 1980's with growing number of practical applications that proved its effectiveness. In 1990's the fuzzy logic control has become a viable and attractive alternative to classical control techniques. It is a fast growing area of research, diversified in many sub-disciplines and hybrid techniques.

There are two main problems with the design of the intelligent control methods. The first is to obtain an adequate knowledge base for the controller, usually obtained from expert knowledge, and the second problem is selection of key parameters defined in the method. Evolutionary algorithms are often used for automated knowledge acquisition for fuzzy logic controllers (Cordon *et al.* 2001a), (Cordon *et al.* 2002), (Konar 2005), (Mohammadian and Stonier 1996a). However, there are a number of methods employed to knowledge base acquisition (Cordon *et al.* 2001c):

- Fuzzy rule base derived from human experts. The expert specifies the linguistic labels associated with linguistic variables, structure of the rule base, and the meaning of each label.

- Fuzzy rule base derived from automated learning methods. There are many different design techniques for automated learning methods apart from evolutionary algorithms, for example: *ad hoc* data driven generation methods, variants of the least squares method, descent method, neural networks, and clustering techniques.

Considering the fuzzy logic control methods record in engineering applications, its popularity is not surprising. Success of fuzzy logic control can be attributed to a few factors:

- Fuzzy logic control is capable of using both sensor data and human expertise. It can use both sources of information, or just one if the other is not available. This makes the FLC a flexible control system with wide range of applications when other, more rigorous methods, struggle to meet their control objectives.
- Fuzzy logic control is model-free approach; fuzzy logic techniques are not dependant on the model of the physical system under consideration. This is important feature as in some cases mathematical model of the system is not available.
- Fuzzy systems are universal approximators, which makes them suitable for non-linear control system design.
- Fuzzy logic control provides good compromise between performance and cost. The fuzzy logic control systems are easy to design and thus cutting development costs. Most importantly, the fuzzy logic control is easy to understand which is important for non-experts in the field.

However, fuzzy logic applications are limited by the heuristic nature of their knowledge bases. Fuzzy logic control is often based upon knowledge derived from imprecise heuristic knowledge of human operators. Some methods for transforming human knowledge into the fuzzy logic knowledge base (rule base and database of a fuzzy inference system) are described in (Harris *et al.* 1993).

The most popular application of fuzzy set theory are fuzzy rule-based systems as they provided the framework for engineering applications. There are three major types of rule-based systems (Babuska 2009):

- Linguistic fuzzy model in which both the antecedent and consequent part of *IF-THEN* rule are fuzzy propositions (Zadeh 1973).
- Fuzzy relational model in which a particular antecedent proposition can be associated with several different consequent propositions via a fuzzy relation (Pedrycz 1984).
- Takagi-Sugeno fuzzy model in which the consequent is a crisp function of antecedent variables (Takagi and Sugeno 1985).

A large number of research papers still assume the original Zadeh's model of fuzzy rule base due to its enduring flexibility in handling various practical applications.

There are also various classifications of fuzzy logic systems based on differences in fuzzy rules and methods of their generation (Feng 2006):

- Fuzzy proportional-integral-derivative (PID) control.
- Hybrid techniques encompassing fuzzy logic, neural networks, evolutionary algorithms, etc.
- Fuzzy-sliding mode control.
- Adaptive fuzzy control.
- Takagi–Sugeno model-based fuzzy control.
- Conventional fuzzy control.

2.1.1.1 Fuzzy PID control

PID controllers are still used in industrial applications due to their simplicity and low cost of implementation. The reason for combining fuzzy logic control and PID control is that the latter does not handle well highly nonlinear and uncertain systems. There are different types of fuzzy PID controllers, one of the most efficient is so called 'gain-scheduling' fuzzy controller (Chiu 1998). In general, fuzzy PID controllers perform better than conventional PID controllers but with the electronics becoming less and less expensive both are being replaced by 'intelligent' control systems. A good review of PID controllers can be found in (Chen 1996).

2.1.1.2 Hybrid techniques

Hybrid techniques are often represented by neural networks control techniques combined with fuzzy logic control. Such techniques are among most popular intelligent control methods (Feng 2006). There are two major types of such hybrid systems: neuro-fuzzy systems (see above) and fuzzy-neural systems, depending on which component is dominant in the hybrid system. In neuro-fuzzy systems (combination of fuzzy logic control and neural networks) the dominant component is the fuzzy control with NN fulfilling the role of the adaptation mechanism. Neuro-fuzzy control method usually uses NN to find fuzzy rules and find/adjust membership functions associated with fuzzy rules. Typically, fuzzy logic parameters are represented by weights in NN nodes. Early applications of neuro-fuzzy systems were as learning techniques to find/adjust the membership functions in the fuzzy control system (Ichihashi and Tokunaga 1993).

Popularity of neuro-fuzzy control methods is due to their relatively robust, model-free control techniques capable of storing and using knowledge for control decisions. NN control acquires such knowledge by data training while fuzzy control acquires knowledge from human experts. Both approaches have their advantages and disadvantages. NN control derives knowledge from objective data sample but if the training data is not sufficiently representative for the analysed problem it may fail in its objective or incur large errors. Fuzzy logic knowledge is based on qualitative and imprecise human knowledge and therefore subject to its limitations but at the same time it gives this approach relatively high robustness and possibility of applications where other techniques are not applicable. Many researchers decided to combine advantages of both approaches to achieve better control outcomes. Further information about neuro-fuzzy control methods can be found in (Mitra and Hayashi 2000).

Fuzzy-neural systems are basically NN in which the imprecision represented by fuzzy sets is applied to pattern recognition. Other fuzzy-neural systems NN augmented by fuzzy operators, are based on the use of logical operators in the neural nodes (Gupta 1992). Another popular approach to control problem is by using fuzzy logic and/or neural networks, evolutionary algorithms, and even more hybridised techniques. Evolutionary algorithms were used to optimise synaptic weights in NN but this approach has become obsolete with emergence of better techniques, such as improved gradient methods. For elaboration on hybrid system with NN and fuzzy system components see (Cordon *et al.* 2001a).

2.1.1.3 Fuzzy sliding mode control

Sliding mode control techniques proved themselves in many applications as robust control systems for uncertain nonlinear systems, (Utkin 1992). Sliding mode techniques are often used in robotic applications, and generally in MIMO (multiple-input multiple-output) systems, as they display robustness in dealing with parameter uncertainty and external disturbances. Inherent problems with sliding mode applications, namely chattering control characteristics are dealt by a number of techniques, most popular being supervisory controller (Wang 1993). Fuzzy sliding mode control eliminates chattering by defining fuzzy boundary layers that replace crisp switching surfaces (Ha *et al.* 2001), (Feng 2006). The stability analysis for sliding mode techniques is well developed which is another advantage of such techniques. Sliding mode techniques are often combined with EAs to find/adjust membership functions (Chen and Chang 1998), (Lin and Chen 1997) or with decomposition of the system into several subsystems (Lo and Kuo 1998). See also (Kaynak *et al.* 2001) for further information on sliding mode techniques.

2.1.1.4 Adaptive fuzzy control

Adaptive fuzzy system is defined as a fuzzy logic system with training algorithm. The fuzzy logic system is designed from the fuzzy *IF-THEN* rules using fuzzy logic principles while training algorithm adjusts the parameters and/or structure of the fuzzy logic system based on numerical information (Wang 1994). In control theory many adaptive methods assume linear or linearised systems and only for certain cases of non-linear dynamical systems adaptive methods were developed (Ioannou and Sun 1995), (Krstic *et al.* 1995). Fuzzy systems are capable of approximating any smooth function on the compact interval and this fact is used by L.X. Wang to design an adaptive fuzzy controller for affine nonlinear systems with unknown functions (Wang 1993). The parameters of the fuzzy system (including membership functions) are updated according to the adaptive law derived from the Laypunov stability theory. See also (Zeng and Singh 1994), (Zeng and Cai 2002) for other fuzzy systems approximation examples.

A number of works in adaptive fuzzy control uses an idea of approximating unknown nonlinear function by a fuzzy system and representing the fuzzy system in the form of linear regression with respect to unknown parameters to use adaptive control techniques, which are well developed for linear or near-linear systems as mentioned before, for example see (Anderson *et al.* 1997), (Campos and Lewis 1999), (Han *et al.* 2001), (Koo 2001), Tong and Li 2003), (Lee and Zak 2004), (Velez-Diaz and Tang 2004).

2.1.1.5 Takagi–Sugeno model-based fuzzy control

In many engineering applications both input and output values are numerical and therefore fuzzy logic systems usually use fuzzifier and defuzziefier combination to translate the problem into fuzzy logic formalism and back into crisp numerical values. T. Takagi and M. Sugeno (Takagi and Sugeno 1985) proposed fuzzy system in which *IF* part in *IF-THEN* rules is fuzzy but the *THEN* part is a linear combination of input variables:

If $(x_1 \text{ is } A_1^l)$ *and* $(x_2 \text{ is } A_2^l)$ *and* $(x_3 \text{ is } A_3^l)$ *and* $(x_n \text{ is } A_n^l))$ *Then* $u = a_0+a_1x_1+ a_2x_2+ \ldots +a_nx_n$.

Takagi-Sugeno fuzzy systems are fuzzy dynamic models, for more details see (Cao *et al.* 1995), (Cao *et al.* 1997). The fuzzy dynamic model idea is based on using a set of local linear models which are connected to a global nonlinear system by membership functions, (Johansen *et al.* 2000), (Tanaka and Wang 2001), (Sugeno 1999). This approach does not suffer from the ‘curse of dimensionality’ as much as conventional fuzzy logic systems as it reduces significantly the number of fuzzy rules. Another of its advantages is its stability analysis that can be performed using classical stability theory including Lyapunov stability analysis.

2.1.1.6 Conventional (Mamdani type) fuzzy logic control

The conventional fuzzy control, also called: Mamdani type fuzzy control, was widely used in many practical applications, see for example (Tong *et al.* 1980), (Holmblad and Ostergaard 1982), (Larkin 1985), (Lee *et al.* 1994), (Kandel *et al*. 1999), (Baturone 2004), (Xiao 2004). Advantage of conventional approach is that is heuristic and basically model-free. However, this approach lacks developed stability analysis and consistency in controller design. Those issues still need to be comprehensively resolved.

In spite of advantages of Takagi-Sugeno fuzzy systems they also suffer from serious drawbacks, it is more difficult to incorporate expert knowledge in such systems and the structure of the rule consequents is difficult to interpret for human experts. One of the major advantages of using fuzzy logic approach in the first place is therefore diminished. The conventional approach provides also more flexibility with using different fuzzy operators to perform fuzzy inference (Cordon *et al.* 2001a). This is one of the reason the conventional fuzzy logic control is still more popular in many practical applications.

The fuzzy logic application as a control system seems most successful in highly nonlinear dynamical systems with large parameter uncertainty. Modeling of such systems is difficult (if not impossible) and classical control methods are often inadequate. Fuzzy logic control can be considered as a real-time expert system that employs fuzzy logic to analyse system input to output. Fuzzy logic approach provides means to convert a linguistic control system derived from the expert knowledge into automatic control system. Furthermore, fuzzy logic control system provides means of controlling the control system evolution and its performance, see (Konar 2005), (Wang 1997), (Stonier and Mohammadian 1996). A typical fuzzy logic control system components are (see Figure 2.1):

- Fuzzification interface which converts crisp input values into fuzzy linguistic values used in fuzzy reasoning mechanism.
- Knowledge base which is the collection of expert control knowledge required to achieve the control objective.
- Fuzzy reasoning mechanism which employs various fuzzy logic operations to infer

the control action from the given fuzzy inputs.

- Defuzzification interface which converts the inferred fuzzy control action into the crisp control values to be entered into the system process.

In general, fuzzy reasoning in the decision making unit is usually expressed as rules with conjunctives *and* , *or* and *else*:

If $(x_1 \text{ is } A_1^l)$ *and* $(x_2 \text{ is } A_2^l)$ *and* $(x_3 \text{ is } A_3^l)$ *and* $(x_n \text{ is } A_n^l))$ *Then* $(u_1 \text{ is } B_1^l)$ *else* $(u_2 \text{ is } B_2^l)$ *else ... else* $(u_m \text{ is } B_m^l)$ (2.1)

where A_k^l , $k = 1, \dots , n$ are fuzzy sets for n input variables x_k , $k =1, \dots , n$, and where B_k^l , $k = 1, \dots , m$ are fuzzy sets for m output variables u_k, $k =1, \dots , m$. In this thesis, the consequent part is assumed to be a single value:

If $(x_1 \text{ is } A_1^l)$ *and* $(x_2 \text{ is } A_2^l)$ *and* $(x_3 \text{ is } A_3^l)$ *and* $(x_n \text{ is } A_n^l))$ *Then* $(u \text{ is } B_1^l)$ (2.2)

Basic elements in fuzzy logic design are described below (Cordon *et al.* 2002a), (Cordon *et al.* 2001b), (Cordon *et al.* 2001c), (Cordon *et al.* 2002).

Fuzzifiers and defuzzifiers are used to convert ‘crisp’ values (such as state variables values) into fuzzy membership functions and vice versa:

- Fuzzifier is a mapping from a real valued point x^* in U to a fuzzy set A' in U (which is called a universe of discourse containing all elements in each particular context). Typical fuzzifiers: singleton, Gaussian, triangular.
- Defuzzifier is a mapping from a fuzzy set B' in V to a crisp value y^* in V. Typical defuzzifiers: centre of gravity defuzzifier, centre average defuzzifier, maximum defuzzifier.

Inference engine lies at the heart of the fuzzy logic control system. In a fuzzy inference engine, fuzzy logic principles are used to combine the fuzzy rules in the rule base into a mapping from a fuzzy set A' in U to a fuzzy set B' in V. Two methods are used to infer with a set of rules: composition based inference and individual-rule based inference. In composition based inference, all rules in the fuzzy rule base are combined into a single fuzzy relation in $U \times V$, and used as a single fuzzy *IF-THEN* rule. In individual-rule based inference, each rule in the fuzzy rule base determines an output fuzzy set and the output of the whole fuzzy inference engine is the combination of the M (size of the rule base) individual fuzzy sets. Combination may be taken either by union or intersection of fuzzy sets. If fuzzy rules are viewed as independent conditional statements then the operator union is used. If fuzzy rules are viewed as strongly coupled conditional statements then the operator intersection is used.

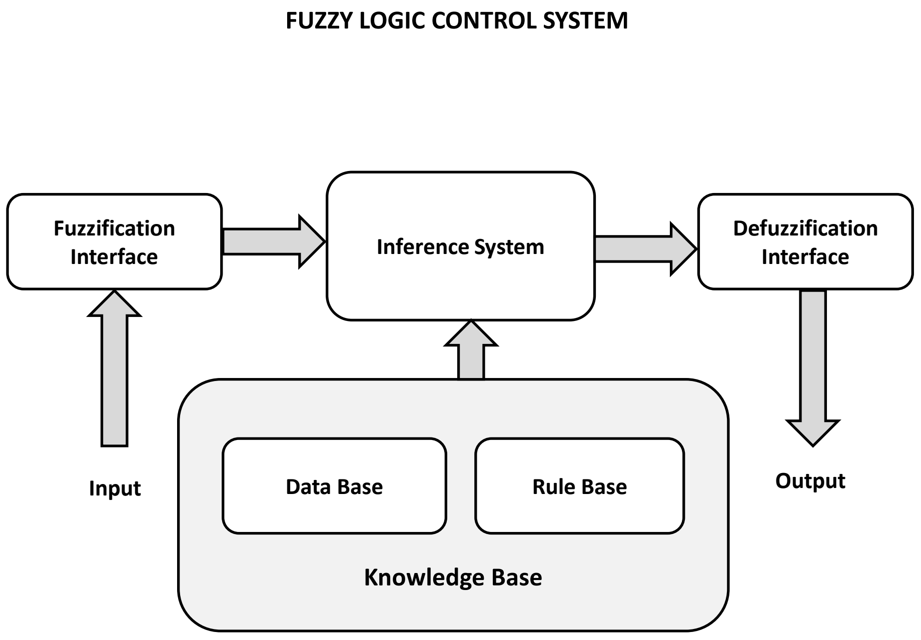

Figure 2.1 Basic diagram of the Fuzzy Logic Control System.

Typical fuzzy system design process can be described as follows:

- Determine the input and output variables.
- Decide on the number of input variables and fuzzification.
- Decide on the number of the output variables and their defuzzification.
- Create the fuzzy knowledge base and inference engine.

Evolutionary algorithm is used to learn the fuzzy rules in the knowledge bases. The important issue in determining the right type of evolutionary algorithm for a control problem is the fuzzy rule base encoding method. Often, fuzzy rule base can be represented as a multidimensional decision table. For a problem with n input variables taking m_i values, $i = 1, \dots, n$, the table has dimensions: $m_1 \times m_2 \times \dots \times m_n$. This decision table can be converted into a linear string. The entire knowledge base is encoded uniquely as a string of integer numbers representing the fuzzy rules. In this way, each fuzzy rule is uniquely defined by the consequent part of the fuzzy rule. In some control problems it is more convenient to encode strings using real valued string elements. However, in this thesis only integer encoding is used.

2.1.2 Hierarchical fuzzy control and evolutionary algorithm

2.1.2.1 Hierarchical fuzzy control

In a hierarchical fuzzy logic structure, typically the most influential parameters are chosen as the system variables in the first level, the next most important parameters are chosen as the system variables in the second level, and so on (Raju *et al.* 1991). In this hierarchy, the first level gives an approximate output which is then modified by the second level rule set, this procedure can be repeated in succeeding levels of hierarchy.

In general, with *n* input variables and *m* fuzzy sets defined for each input variable, there is m^n fuzzy rules in the rule base. In the hierarchical structure, the number of rules in a complete rule set is so reduced to a linear function of the number of variables, but this number may still be high. But which variables in a given system are the most influential and in what order should variables be chosen? These questions in general are yet unanswered and an understanding of fundamentals is required to determine the level of interdependence of input variables. Also, given that different hierarchical structures can exist, how can the fuzzy knowledge base and associated parameters in each layer be effectively learnt?

The decomposition into hierarchical fuzzy logic sub-systems reduces greatly the number of fuzzy rules to be defined and to be learnt but such decomposition is not unique and it may give rise to variables being output from one layer and input into the next layer, which do not have any physical significance (Magdalena 1998). This can raise difficulties in obtaining a complete class of rules from experts even when the number of variables is small (Stonier and Mohammadian 2004), (Mohammadian 2003), (Kingham *et al.* 1998).

Hierarchical fuzzy systems combined with evolutionary algorithms (for learning fuzzy knowledge base) can be used not just to reduce the size of the knowledge base but also to improve model accuracy in regions where non-hierarchical models do not provide sufficient performance (Cordon *et al.* 2004).

It is worth mentioning that reduction of the rule base can be achieved using other methods than hierarchical decomposition, for example clustering approach presented by S. Chopra (Chopra *et al.* 2005). K.S. Tang (Tang *et al.* 1998) proposed a scheme to reduce number of fuzzy rules and membership functions by using a hierarchical genetic algorithm. Note that the genetic algorithm structure is assumed hierarchical, not the fuzzy system topology. The method does not require a priori knowledge of the fuzzy system topology.

2.1.2.2 Evolutionary algorithm

The evolutionary algorithm is a heuristic search technique that maintains a population of individuals. Each individual can be considered to represent a potential solution to a given problem. Each individual is assigned a measure of fitness which determines how accurate it is as a potential solution to the problem. The new population is obtained from the old one by the use of genetic operators such as crossover, and mutation. An elitism strategy is used to pass the fittest individuals to the new population, so that the information encapsulated in the best individual is not lost and passed to the next generation.

A selection process is used to obtain parents for mating in the current generation. The most popular is proportional selection to select randomly two parents based on their fitness in proportion to the overall total fitness of the population. Another is tournament selection in which a specified number of possible parents are selected at random from the population. A tournament is then held to select the two fittest strings and they are used as parents in the next process of crossover to generate children to be passed into the next generation.

In the crossover operation a number of ‘parent’ strings, typically two, are recombined to create ‘child’ strings. The most popular crossover operator is the one-point, arithmetic, and uniform crossover. The crossover operator plays a role of sexual reproduction in which two individuals exchange parts of their strings to produce offspring.

With a given probability the mutation operator mutates elements of the individual in the population. This ensures satisfactory diversity within the population which is required for the EA to find better approximate solutions to the problem.

With an appropriate selection of EA parameters and operators, the algorithm is allowed to evolve. It is terminated when pre-defined termination condition is satisfied; usually at a fixed number of generations or until there is minimal change or no change to the string which has the best fitness. The fittest individual is taken as the best possible solution learnt by the algorithm.

2.2 Literature review

Foundations of fuzzy theory were laid by Lofti A. Zadeh in 1965 in his famous paper “Fuzzy sets” (Zadeh 1965). Initially, the new theory did not attract much attention and remained outside the mainstream of control theory techniques. After a series of publications (Zadeh 1968), (Bellman and Zadeh 1970), (Zadeh 1973), the fuzzy control theory started to emerge as a serious competitor to the conventional control techniques. Zadeh's 1973 paper (Zadeh 1973) introduced the concept of linguistic variable and *IF-THEN* rules to encompass human expert knowledge. Research work by E. Mamdani and S. Assilian (Mamdani and Assilian 1975) provided a groundwork for future fuzzy logic applications. With the first successful application in the 1978 experiment in the cement kiln in Denmark, the fuzzy control theory started to be seen as a practical alternative to classical control methods (Holmblad and Ostergaard 1982). However, not much progress was made, mostly due to lack of funding and interest from the industry, until successes of Japanese researchers and engineers (Fuji Electric water purification plant, Sandai subway) in 1980's paved way to rapid increase in funding and research in fuzzy control area. Fuzzy logic systems proved to be an excellent framework for representing both human expert derived knowledge or/and automatically acquired via some learning mechanism.

The successful real-life applications made fuzzy logic control recognised as one of the major control theory techniques. Initially few papers were published on fuzzy logic control with momentum gaining in 1980's. For examples of early publications on fuzzy logic control methods see (Mamdani and Assilian 1975), (Mamdani 1976), (Kickert and Lemke 1976), (Mamdani 1976), (Kickert and Mamdani 1978), (Procyk and Mamdani 1979), (Czolgala and Pedrycz 1981), (Czolgala and Pedrycz 1982), (Ray and Majumder 1984), (Kiszka *et al.* 1985), (Takagi and Sugeno 1985), (Daley and Gill 1986), (Graham and Newell 1988), (Chen and Tsao 1989). With increasing number of practical applications, such as washing machines, image stabilisers, self-parking cars, etc, the researches encountered growing problems related to complexity of the systems to be controlled. One of the most challenging was exponential growth of fuzzy rules with the increase of input variables or number of fuzzy sets associated with them ('The curse of dimensionality').

There is a vast literature on fuzzy control systems, especially with applications to the inverted pendulum (cart-pole system) as it is often used as a test system for proposed methods. However, there is much less publications on hierarchical fuzzy control systems. A large number of control systems (especially from 1980's and 1990's) rely on local linearization of the dynamical system under consideration. Design of the stabilizing fuzzy logic controllers is achieved via piece-wise linearization of the non-linear system, especially when authors are implementing Lyapunov direct method. Lyapunov method can be used not just for stability analysis but also to design fuzzy controllers, for example (Chen and Chen 1998), (Chen *et al.* 1999), (Zhong and Rock 2001).

It is difficult to study the problem of hierarchical decomposition for a large class of fuzzy systems but it is possible to analyse such architectures on the example of a particular fuzzy system. Obviously, topology of the HFS must be selected according to the physical properties of the dynamical system under consideration. The selection process is subject to human decision. It might be possible to design the EA for finding the most suitable (optimal or near-optimal) topology for any particular problem (hierarchical EA), so the process can be automated. There is a number of research projects dealing with variable control structures, see for example (Hsu *et al.* 2001).

Fuzzy logic control applied to the inverted pendulum system can be found in many research papers (as the inverted pendulum is often used as a test-system). In spite of providing obvious advantages in reduction of the knowledge base size, hierarchical fuzzy control does not often appear in research literature involving the inverted pendulum problem. Furthermore, in most research papers implementing conventional Mamdani approach to fuzzy rules a single or relatively few test initial conditions are examined (Takagi-Sugeno approach allows use of Lyapunow stability analysis). Therefore, it was decided to investigate a method that could control the inverted pendulum from a wide range of initial conditions, including nonzero initial cart velocity and pole angular velocity (usually assumed to be zero).

Research papers on fuzzy control fall generally into five categories: sliding mode, adaptive, EA & NN fuzzy control, and hybrid techniques encompassing two or more categories, see also Section 2.1.1 for a general classification of fuzzy logic control techniques.

The research presented in this thesis has originated in research work by R.J. Stonier (Stonier *et al.* 1998) and by R.J Stonier (Stonier 1999). In (Stonier *et al.* 1998) a two layered HFS is investigated and the development of GA for knowledge base design is discussed in detail. Research presented in this thesis is based on a modified algorithm from this paper, with different parameters, crossover procedure and mutation schedule. Strong elitism is introduced to bring the average of the population in the GA close to the desired minimum of objective function.

Amalgamation and compositional method was investigated in (Stonier 1999). However, attempts at replicating the results for amalgamation method achieved in the abovementioned paper failed. The original simulations were run on Borland Turbo Pascal, with different parameters and included variable scaling. Exact re-creation of simulations was not attempted as Borland Turbo Pascal is not supported on newer platforms. The method of fuzzy amalgamation relies on developing knowledge bases for every initial condition from a configuration set separately and then amalgamating them into one final (global) knowledge base by averaging the outputs for each rule in the knowledge bases. The compositional method determines a single fuzzy knowledge base directly from a modified EA that learns fuzzy rules over the grid of initial conditions.

2.2.1 Literature review: fuzzy logic control

Development of fuzzy logic control techniques began in earnest in 1980's with a large number of papers published. The growth of fuzzy logic techniques was partly due to fuzzy controllers solving previously intractable or very difficult control problems.

In this thesis, the inverted pendulum system is used as a case study therefore special consideration is given to publications that solve the pole-cart problem, either as a research paper dedicated to this particular problem or just using it for testing the proposed control method.

In M.J. Desylvia's MSc thesis (Desylva 1994) a fuzzy logic controller was developed for the task of balancing the inverted pendulum from an arbitrary set of initial conditions. The rule base developed was based on intuition and logic rather than any mathematical model. This approach made the control process much simpler as there was no need to solve nonlinear differential equations. However, results achieved are specific to the inverted pendulum system and have no much value as a method that can be extended to other dynamical systems.

In the paper by K.J. Astrom and K. Furuta (Astrom and Furuta 2000) the authors discuss simple strategies, based on Lyapunov analysis, for swinging up an inverted pendulum and show that the cart-pendulum system critically depends on the ratio of the maximum acceleration of the pivot to the gravity acceleration. Comparison of energy based strategies with minimum time strategies are provided. In the paper, a designed controller is capable of bringing the pendulum to the upright position in one swing, providing that the control force satisfies $u > 2g$. Instead of controlling position and velocity of the cart-pendulum system, the method relies on the control task being achieved by controlling the energy of the system, namely acceleration of the pivot.

M. Margaliot and G. Langholz (Margaliot and Langholz 1999) used classical Lyapunov approach to design fuzzy controllers. They used as one of their case studies the inverted pendulum system. The system variables were restricted to the pole angle and its angular velocity. The fuzzy rules were derived from the Lyapunov function derivative condition (negative definite derivative). Five static initial conditions were selected to test the control system and one with the initial angular velocity of 1rad/s. The method main advantage is that it required very little knowledge about the system under consideration. The dynamic equations do not need to be known, only the state of the system must be known (state variables) and proportional dependence of input variable (angular speed) to the system control output. Furthermore, the authors demonstrated that for linear time-invariant plants of arbitrary order, four Mamdani-type fuzzy rules suffice to guarantee local asymptotic stability.

W.S. Yu and C.J. Sun (Yu and Sun 2001) developed a fuzzy adaptive control for a class of nonlinear systems and verified it on the example of the inverted pendulum. The control algorithm guarantees global stability of the system with the output of the system approaching the origin if there are no disturbances and uncertainties, converging to the neighbourhood of the origin for all realisations of uncertainties and disturbances.

Similar approach was presented by T. J. Koo (Koo 2001) using reference model adaptive fuzzy control. N. Muskinja and N. Tovornik (Muskinja and Tovornik 2006) designed an adaptive fuzzy controller for a real inverted pendulum and compared various control strategies. Their investigation showed advantages of using fuzzy control theory in real-time applications, specifically for the inverted pendulum system with the aim of fast stabilization of the pendulum and the pendulum cart.

The most relevant to the compositional method described in this thesis are results by J. Yi and N. Yubazaki (Yi and Yubazaki 2000), (Yi *et al.* 2002). They developed fuzzy controller based on the single input rule modules (SIRM) and dynamic importance degree (DID). The method was tested for a wide range of system parameters (cart and pole mass, pole's length, etc.). They reported that for specific system parameters the inverted pendulum can be stabilised for initial pole angle within $[-30.0^{\circ}, 30.0^{\circ}]$. In (Yi *et al.* 2002) the authors investigate parallel inverted pendulum system and design the fuzzy control system that stabilizes the system for angles within $[-20.0^{\circ}, 20.0^{\circ}]$ defined as lower and upper limit of controllability region. The control system is effective for initial angles up to 10.0° (depending on the angle of the other parallel pendulum). Dynamic initial conditions are set to zero.

Sliding mode technique is one of the most popular techniques used to control the inverted pendulum problem. W. Chang (Chang *et al.* 2002) used the inverted pendulum system as a case study for the robust fuzzy-model-based sliding mode controller and tested it on several initial conditions but with limited scope. The cart's position was fixed and located at the origin, only pole angle varied: 0.08*π*, *π/60,* 89*π*/180*, π*/4. The focus of their research was on system uncertainties not on the region of controllability. Sliding mode control provides a robust controller but with inherent chattering problem that various techniques seek to overcome, supervisory controller being one of the relatively simple solutions. F. Qiao (Qiao *et al.* 2003) found efficient fuzzy sliding mode control for discrete nonlinear systems in the presence of noise and tested it on the example of the inverted pendulum. Other control methods utilizing sliding mode can be found in (Brunetti and Dotoli 2004), (Wai *et al.* 2003), and (Allamehzadeh and Cheung 2002).

W.J. Wang (Wang *et al.* 2003) proposed a new GA based method to construct a fuzzy rules base which does not require initialisation of the fuzzy rules' number, the positions of the antecedent and the consequent fuzzy sets. Only the length and the structure of the chromosome are set. With the specific structure of the chromosome (two strings encoding the antecedent part of the *IF-THEN* rule), the special mutation operation (hierarchical mutation) and the adequate fitness function (based on measured output error), GA generates the fuzzy rule base spontaneously. The generated rule base has the small number of rules and arranges the suitable placement of the premise's fuzzy sets and chooses the proper location of the consequent singletons. The method is developed for multi-input multi-output systems with arbitrary number of input and output variables.

Research by O. Castillo (Castillo *et al.* 2006) examined stability issues in fuzzy control theory on the example of the inverted pendulum problem. In many real-life applications the reliability of the controller is considered more important than stability issues but development of the stability analysis allows the use of model based approach in fuzzy logic control. The authors used Lyapunov theory to develop stable Mamdani type 2 fuzzy logic controllers.

P.A. Phan and T. Gale (Phan and Gale 2007) presented two-mode adaptive fuzzy control with approximation error estimator. In the learning mode adaptive law is used to tune the fuzzy system parameters and in the operating mode the fuzzy parameters are fixed and only the estimator is updated.

A. Di Nola (Di Nola *et al.* 2007) used the inverted pendulum to demonstrate the control system based on algebraic analysis of fuzzy systems. As the authors used it only as a demonstration of the method no significant result was produced for the inverted pendulum problem apart from providing another original approach to the fuzzy control problem.

A new fuzzy control method, scalar fuzzy control (SFC) was discussed in (Mlynski and Zimmermann 2008). In general, the method is concerned with the problem of representing imprecise statements and knowledge, and processing it to draw conclusions from them. The method goes back to the principles of the multi-valued logic and introduces axiomatic framework to develop SFC. The method is based on so called: Calculus of imprecise knowledge and deals with linguistic variables. SFC is used to solve the inverted pendulum problem. Surprisingly, the results achieved by SFC are quite similar to presented in this thesis, especially state variables convergence. The initial conditions investigated being: pole angle = 0 and 1 rad ($\approx 57^{\circ}$), and cart's position x = 0.0m and x = 10.0m. The rate of stabilisation is slightly slower then achieved in this project. The restrictions imposed on the system in this thesis do not allow for cart's position to be larger than 1 m from the origin therefore it is difficult to compare SFC results. The results achieved in (Mlynski and Zimmermann 2008) make this method very robust and provide another proof of fuzzy logic solving real-life problems.

2.2.2 Literature review: fuzzy logic control and evolutionary algorithms

A major problem for fuzzy systems is that they lack a learning mechanism. Coupling fuzzy system with evolutionary algorithm provides a solution to the automated acquisition of the fuzzy rule base. The fuzzy knowledge base can be derived by other learning mechanism but evolutionary algorithms proved to be very successful search mechanism as they are efficient global search techniques and capable of incorporating a priori knowledge, such as knowledge derived from human experts.

An example of early work on combination of fuzzy control system and genetic algorithms is the paper by M.G. Cooper and J.J. Vidal (Cooper and Vidal 1994). The focus of the paper is on the problem of representation of the fuzzy rule base (encoding the individuals in the genetic population). The compact encoding scheme is discussed and its implementation. This scheme allows for a smaller fuzzy rule base size and is aimed at overcoming the 'curse of dimensionality' as the search space increases exponentially with string size. In the compact encoding scheme, each of the input variables requires only two one byte integers: one giving the centre and the other the half-length of the base of the isosceles triangle representing the membership function. In general, each rule requires $2 \cdot m$ bytes where m is the total number of control variables in the system. The authors use random initial population and the size of the rule base emerges as a result of the genetic algorithm evolution. For an early example of a method of simultaneous design of membership functions and fuzzy rule base see (Homaifar and McCormick 1995). For other examples of fuzzy control and evolutionary learning see (Lee and Takagi 1993), (Lee and Takagi 1993a), (Shimojima *et al.* 1995), (Mohammadian and Stonier 1996a), (Stonier *et al.* 1998). (Matellan *et al.* 1998), (Mao *et al.* 2001), (Damousis *et al.* 2002), (Kumar and Garg 2004). A good review of fuzzy logic control and evolutionary algorithms can be found in (Cordon *et al.* 2001a) where the authors describe in detail different approaches and methods in automated knowledge acquisition with particular emphasis on rule-based systems and different variants of genetic/evolutionary algorithms. They include both Michigan and Pittsburgh approach among other methods.

2.2.3 Literature review: hierarchical fuzzy control

The curse of dimensionality remains an unsolved problem in fuzzy logic control theory (Abraham 2005). The problem is subject of many research papers with some authors focusing on systematic design of fuzzy logic systems (Chen *et al*. 2007). There are also research projects with focus on finding automatically fuzzy structure and parameters of the fuzzy system, for example see (Huang and Wang 2000), (Wu and Chen 1999).

One of early methods to reduce the size of the rule base by introducing hierarchical fuzzy control was developed G.V.S. Raju (Raju *et al.* 1991) and G.V.S. Raju and J. Zhou (Raju and Zhou 1993). G.V.S. Raju proposed a hierachical structure in which the most influential system variables were input in the first layer, the next most important variables as input in the next layer, and so on. The first layer control output is an approximation of the controller and is modified by the fuzzy rule base of the next layer, until the final layer produces the final control output.

Another early attempt to overcome the dimensionality problem was made by M. Brown (Brown *et al*. 1995) who proposed a low-dimensional rule base in a hierarchical structure. Automatic determination of the fuzzy rule base in a hierarchical structure was proposed in (Shimojima *et al.* 1995).

L.X. Wang (Wang 1997) provided proof that the HFS are universal approximators to any continuous function on a compact set and analyses the sensitivity of the fuzzy system output with respect to small perturbations in its input. This was further elaborated in (Wang 1999).

M.G. Joo and J.S. Lee (Joo and Lee 2002) extended Wang's results to a case where the intermediary control outputs (between layers) are not part of the antecedent part of the *IF-THEN* rules but only in the consequent part. This approach removed the problem of intermediary control outputs that had little physical meaning and were difficult to interpret.

L.C. Lin and G.Y. Lee (Lin and Lee 1999) investigated optimization of the hierarchical structure and its parameters for a five input variables fuzzy system in a low-speed control problem. Input configuration analysis was performed by J.C. Duan and F.L. Chung (Duan and Chung 2002), see also (Chen *et al.* 2007).

K.Y. Tu (Tu *et al.* 2000) presented a method for designing of a multilayer fuzzy logic controller for multi-input multi-output systems. In this paper, we propose a multi-layer fuzzy logic controller (MLFLC) for multi-input–multi-output (MIMO) systems. For the convenience of analysis, the structure of the multi-layer fuzzy logic controller is divided into multi-input–single-output (MISO) controllers. Each multi-input–single-output controller consists of many fuzzy logic controllers (FLC). In the fuzzy logic controller the linguistic rules are designed as a suction controller. A theorem shows that such fuzzy logic controller has a switching line. The results of analysis show that the multi-layer fuzzy logic controller has a switching manifold and its parameters are the scaling factors which normalize the input variables. Moreover, a theorem which shows the stability of the proposed multi-layer fuzzy logic controller can easily be formulated by properly selecting the denormalising scaling factors. A cart-pole system with two links is used as an illustrated example for demonstration. The demonstrations also include the links controlled to track a set of desired trajectories. Simulation results show that the fuzzy control system is asymptotically stable, and the desired trajectories can be followed very closely.

Y.J. Mon and C.M Lin (Mon and Lin 2002) proposed a hierarchical fuzzy sliding-mode control to achieve asymptotic stability of the system. The nonlinear system is decomposed into several subsystems and the state response of each subsystem can be designed to be governed by a corresponding sliding surface. The whole system is controlled by a hierarchical sliding-mode controller. The inverted pendulum system is used to test the proposed method. Later they improved their hierarchical fuzzy sliding-mode controller with decoupling of the nonlinear inverted pendulum system into several subsystems, see (Lin and Mon 2005).

M.L. Lee (Lee *et al.* 2003) addressed the intermediate output from hierarchical fuzzy layers that usually do not have a physical meaning. They proposed a new mapping of the rule base that allows treating the intermediary control values as intermediary mapping variables. The method aims at reducing the rule base size.

Z.M. Yeh and K.H. Li (Yeh and Li 2004) proposed a multistage control system for the inverted pendulum system that reduced the number of rules. However, the fuzzy rules formulation/generation was not clearly elaborated.

R.J. Stonier and M. Mohammadian (Stonier and Mohammadian 2004) presented introduction to hierarchical fuzzy control with the use of evolutionary algorithms on several examples: interest rate prediction, inverted pendulum, collision-avoidance in a robot system, micro-robot control, and co-evolutionary algorithm.

L.X. Wang (Wang *et al.* 2005) designed a sliding mode controller for one-input multiple-output system where sliding surfaces are organized in a cascade thus creating a hierarchical system.

F. Cheong and R. Lai (Cheong and Lai 2007) addressed problems with the use of hierarchical fuzzy logic controllers, especially in the automatic design of controllers. This includes the coordination of intermediary outputs (approximate controllers) of sub-controllers at lower levels of the hierarchy. The authors describe a method for the automatic design of a hierarchical fuzzy logic controller using an evolutionary algorithm called differential evolution. The method is developed for a wide class of control systems and the feasibility of the method is demonstrated by developing a two-layered HFS for controlling the inverted pendulum system.

Other examples of hierarchical fuzzy control applied to the inverted pendulum system can be found in (Magdalena 1998) and (Lei and Langari 2003).

2.2.4 Literature review: hybrid control systems

Hybrid techniques are part of the wider quest for an intelligent control methodology. Many researchers found that merging different control paradigms results in more efficient, often adaptable, control techniques. Different approaches were tried to merge control technique. An early example can be found in (Chiaberge *et al.* 1995) in which several control paradigms are merged: fuzzy control, neural networks, linear control, optimisation algorithms (simulated annealing and genetic optimisation), and finite state automata. The presented method, for designing hybrid intelligent controllers, is based on an implementation of the fuzzy logic control with real and binary weights. The learning is performed by the genetic algorithm and is defined as a mixed integer constrained dynamic optimization. Training of the controller is performed in a closed-loop simulation with the controller in the loop.

E.S. Sazonov (Sazonov *et al.* 2003) developed a hybrid control system, including neural controller and linear quadratic Gaussian (LQG) controller. The neural controller was optimised by genetic algorithms on the inverted pendulum system. The optimisation process stipulated a region of asymptotic stability of the neural controller around the regulation point. This paper has little relevance to the research presented in this thesis but it shows a variety of approaches taken to solve control problems.

In (Saifizul *et al*. 2006) the authors presented a neuro-fuzzy controller for the inverted pendulum. The mathematical model of the inverted pendulum gives a good representation of the physical system taking into account a large number of system parameters, including electrical characteristics of the actuators, damping forces, viscous damping, etc. The control system, SESIP (self-erecting single inverted pendulum), consists of two control loops: swing-up controller and stabilisation controller. Position-velocity controller is used to design swing-up control and Takagi-Sugeno fuzzy controller with adaptive neuro-fuzzy inference system is used to stabilise the pendulum at the unstable equilibrium position. The authors pay special attention to control the cart's position which is returned to its original position. The control system stabilises the pendulum in about 3.5s. The initial condition tested was pendulum angle at 10 degrees and other initial values set to zero.

S. Khwan-on (Khwan-on *et al.* 2004) developed neuro-tabu-fuzzy controller to stabilize a wide range of inverted pendulum systems using the same SIRM technique. They investigated relation between the pendulum length and the initial angle of the pendulum in terms of stabilization times.

M. Kumar and D.P. Garg (Kumar and Garg 2004) compared fuzzy logic control using GA-fuzzy and neuro-fuzzy models. Neuro-fuzzy approach was faster but with higher PI value while GA-fuzzy was slower but with lower PI value, where PI was defined as sum of squared angle errors over a simulation period of 10s. Y. Gao and M.J. Err (Gao and Err 2003) proposed an approach by combining neural networks with adaptive techniques and designed a fuzzy controller for the inverted pendulum. J.A.K. Suykens (Suykens *et al.* 2001) successfully implemented neural network techniques to achieve fast and smooth convergence of state variables. Other examples of successful application of GAs and NNs in controlling the inverted pendulum system can be found in (Sazonov *et al.* 2003), (Wu and Tam 2000).

A hybrid controller for stabilisation of the rotary inverted pendulum proposed by P. M. Melba and N.S. Marimuthu (Melba and Marimuthu 2008) is another example of effectiveness of hybrid methods. In this case the control system is designed in two parts: PD position control to swing up the pendulum to approximately upright position and then FLC used to stabilise the pendulum in the upright position. LQR (linear quadratic regulator) feedback control is used for pendulum stabilisation.

Hybrid control methods is a very active research field with many researches trying to design practical intelligent control systems when a single control paradigm fails to provide satisfactory results.

2.2.5 Literature review: MOEA related work

There is a fast growing literature on multiobjective optimisation in the last decade. One of the most influential are publications by Kalyanmoy Deb (Deb 2001) and by Carlos A. Coello Coello (Coello Coello *et al.* 2002). More recent methodology can be found in (Abraham *et al*. 2005).

The first application of the EA in finding multiple trade-off solutions was made by D. Shaffer (Shaffer 1985) in which non-Pareto approach was used. After an idea about domination in multi-objective optimization in D.E. Goldberg book (Goldberg 1989) a number of multiobjective evolutionary algorithms were developed by different authors, for example see (Fonseca and Fleming 1993), (Srinivas and Deb 1994), (Horn *et al.* 1994). E. Zitzler and L. Thiele (Zitzler and Thiele 1999) proposed a Pareto based method, so called the strength Pareto evolutionary algorithm (SPEA). The main characteristics of SPEA can be summarized as:

- Sorting non-dominated solutions externally.
- Evaluating an individual's fitness on the basis of external non-dominated individuals that dominate it.
- Preserving population diversity by using the Pareto dominance.
- Using clustering technique to reduce the non-dominated set.

H.A. Abbas (Abbass *et al.* 2001) developed differential evolution (DE) that is an EA designed to handle optimization problems over continuous domains. The paper introduces a novel Pareto-frontier differential evolution (PDE) algorithm to solve multiobjective optimization problem (MOP). The solutions provided by the proposed algorithm for two standard test problems, outperform the SPEA. The Pareto differential evolution (PDE) algorithm performance varies according to the crossover and mutation rates. In (Abbas 2002) a new version of PDE was presented with self-adaptive crossover and mutation. This new version is called self-adaptive Pareto differential evolution (SPDE).

M. Laumanns (Laumanns *et al*. 2002) addressed the important problem of convergence to the true Pareto set in MOEAs, which is related to the problem of maintaining diversity in solution space. I.F. Sbalzarini (Sbalzarini *et al.* 2001) investigated similar problem, namely: How to accomplish fitness assignment and selection in order to guide MOEA towards the Pareto set and how to maintain a diverse population to prevent premature convergence. Further research material on MOEA convergence can be found in (Deb *et al.* 2000), (Deb and Goel 2001), (Zitzler *et al.* 2001) just as an example of the vast literature on the MOEA subject.

Multi-objective evolutionary algorithms usually perform well for problems with two or three objectives. However, for many-objective optimisation with more than three objectives, the algorithms applying Pareto optimality as a ranking metric may loose their effectiveness. This problem is addressed by E.J. Hughes (Hughes 2005) who compares three different approaches to generating Pareto surfaces on both multi and many objective problems. In the first approach a Pareto ranking method (NSGA II) is used. The second approach combines multiple single objective optimisations in a single run (MSOPS). The third uses multiple runs of a single objective optimiser. The results show advantages of generating the entire Pareto set in a single run compared to repeated single objective optimisations. NSGA II loses its effectiveness as the problem dimensionality increases.

The growing number of MOEA methods required new methodology for method comparison. Initial investigation into comparison of various MOEA methods was done by E. Zitzler and L. Thiele (Zitzler and Thiele 1999). This issue was also addressed by D.A. Van Veldhuizen and G.B. Lamont, (Van Veldhuizen and Lamont 2000). E. Zitzler and L. Thiele expanded their work into more comprehensive study (Zitzler *et al.* 2000). E. Zitzler in (Zitzler *et al.* 2000) provided systematic comparison of various evolutionary approaches to multiobjective optimisation based on selected six test functions. Selected test functions are known for causing difficulties in implementation of multiobjective optimisation, mostly in convergence to the Pareto front, see Chapter 8 for terminology. The authors introduce metrics to measure the methods performance, addressing specifically three major objectives:

- Minimisation of the distance of the solutions set to the Pareto-optimal front.
- A good (that means in most cases uniform) distribution of the solutions.
- The extent to which the nondominated solutions should be maximised.

Eight algorithms are compared:

- RAND: a random search algorithm.
- FFGA: Fonseca and Fleming's multiobjective EA.
- NPGA: the niched Pareto genetic algorithm.
- HLGA: Hajela and Lin's weighted-sum based approach.
- VEGA: the vector evaluated genetic algorithm.
- NSGA: the nondominated sorting genetic algorithm.
- SOEA: a single objective evolutionary algorithm using weighted-sum aggregation.
- SPEA: the strength Pareto evolutionary algorithm.

Generally, it was found that multiobjective EAs performed better than random search algorithm. It was also observed that NSGA outperforms the other nonelitist multiobjective algorithms. The best overall performance is demonstrated by SPEA. The results also show the importance of elitist strategies. Elitism plays important part in performance. Furthermore, other methods when supplemented by SPEA elitism show improved performance.

Another comparison can be found in the paper of D.A. Van Veldhuizen and G.B. Lamont (Veldhuizen and Lamont 2000). Four methods are compared: MOGA (same as FFGA, see above), MOMGA (Van Veldhuizen and Lamont method incorporating fitness sharing and Horn's tournament selection), and finally NPGA and NSGA as mentioned in (Zitzler *et al.* 2000). The authors performed in depth analysis of the investigated methods using three MOP comparison metrics (generational distance, overall nondominated vector generation, and spacing) and nonparametric statistical analyses. They showed that NSGA performance is statistically worse than the other tested methods.

Summary

In this chapter the background knowledge and literature review is presented. The existing techniques in fuzzy logic control, hierarchical fuzzy control, hybrid methods, and MOEA methods are reviewed. Different approaches to fuzzy logic control are briefly described: fuzzy proportional-integral-derivative (PID) control, hybrid techniques, fuzzy-sliding mode control, adaptive fuzzy control, Takagi–Sugeno model-based fuzzy control, and conventional fuzzy control. Many of the FLC methods encompass two or more different techniques. The review was focused on fuzzy logic control coupled with evolutionary algorithm as a learning method. Selected papers of interest are briefly described.

Compositional method, presented in Chapter 8, is formulated using multiobjective optimisation formalism and for this reason MOEA methods are reviewed in a separate section.

Chapter 3 HIERARCHICAL FUZZY SYSTEMS

3.1 Introduction

Hierarchical fuzzy systems are used not only to overcome the curse of dimensionality but also to improve the control system performance. The decomposition into hierarchical structure that reflects the physical properties of the system under investigation simplify the control system and it might greatly improve its performance.

The fuzzy control methods are often tested on the example of the inverted pendulum system. In some cases, especially when the control method is dependent on the physical properties of the system, the method is developed on the example of the inverted pendulum and then extended to other dynamical systems.

In the following sections, the case study setup for all experiments with the inverted pendulum is described.

3.1.1 Case study: inverted pendulum system

The control of the inverted pendulum system has been undertaken using linear and nonlinear dynamics and include both classical and fuzzy logic control techniques, see for example (Slotine 1991), (Anderson 1989), (Lee and Takagi1993), (Stonier *et al.* 1998), (Magdalena 1998).

The inverted pendulum system consists of the cart and a rigid pole hinged to the top of the cart, see Figure 3.1. The cart moves left or right on a straight bounded track and the pole swings in the vertical plane determined by the track. The dynamics of the system is modelled by the following equations:

$$\begin{aligned}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= u + m_p\, L\, (\sin(x_3)\, x_4^2 - \dot{x}_4 \cos(x_3))/(M_c + m_p) \\
\dot{x}_3 &= x_4 \\
\dot{x}_4 &= \frac{g \sin(x_3) + \cos(x_3)\,(u - m_p\, L\, x_4^2 \sin(x_3))/(M_c + m_p)}{L\,(4/3 - m_p \cos(x_3)^2/(M_c + m_p))}
\end{aligned} \tag{3.1}$$

where x_1 is the position of the cart, x_2 is the velocity of the cart, x_3 is the angle of the pole, x_4 is the angular velocity of the pole, u is the control force on the cart, m_p is the mass of the pole, M_c is the mass of the cart, L is the length of the pole, and g is gravitational acceleration. The control force is applied to the cart to prevent the pole from falling while keeping the cart within the specified bounds on the track. The system has the following parameters: m_p = 0.1kg, M_c = 1.0kg, L = 0.5m, g = 9.81ms^{-2}, with state limits: $-1.0 \le x_1 \le 1.0$ and $-\pi/6 \le x_3 \le \pi/6$. Even though the above system equations do not represent fully the physical dynamics of the inverted pendulum system they are a good and relatively accurate approximation of the real system.

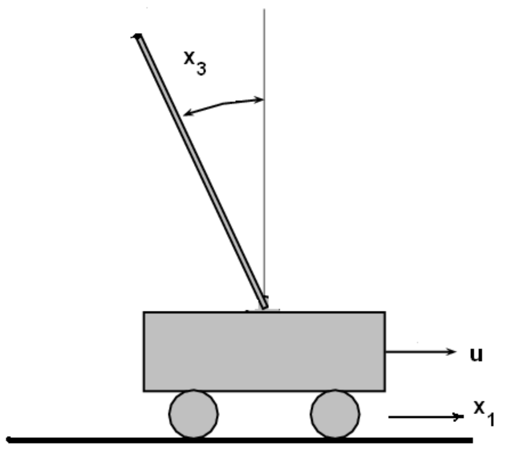

Figure 3.1 Inverted pendulum system.

Fuzzy controller is to stabilise the system about the unstable reference position $\vec{x} = \vec{0}$ as quickly as possible, whilst maintaining the system within the target region (TR) defined by the following bounds: $|x_1| \le 0.1$, $|x_2| \le 0.1$, $|x_3| \le \pi/24$, $|x_4| \le 3.0$. The fuzzy controller is required to control the system such that the state variables converge to the TR and are maintained within TR for a prescribed time limit T_f, with T_f = 20.0s.

3.1.2 Case study: fuzzy system for the inverted pendulum problem

The control of the inverted pendulum (sometimes referred to as pole-cart system) has been undertaken using linear and nonlinear dynamics and include both classical and fuzzy logic control techniques, see for example (Slotine 1991), (Anderson 1989), (Lee 1993), (Stonier *et al.* 1998), (Magdalena 1998).

In initial experiments all fuzzy membership functions are assumed to be triangular. After experiments with co-evolutionary algorithm all fuzzy membership functions are assumed to be Gaussian functions with their centres evenly spaced over the range of input and output variables. The only exception being some experiments with co-evolutionary EA in which centres of the membership functions are randomly generated.

It is worth noting that with the increase of the number of membership functions covering the domains of input and output variables a better accuracy of the control system can be achieved. However, it comes at a price, namely with a larger number of rules and computational times increased dramatically. For this reason a compromise must be struck between accuracy and computational requirements.

Each domain region for x_i is divided into five overlapping intervals and assigned linguistic values:

- NB – Negative Big
- NS – Negative Small
- NE – Neutral
- PS – Positive Small
- PB – Positive Big.

The defined linguistic values are associated with membership sets A_i^k, k =1, ... , 5, which are encoded numerically as integers from 1 to 5 respectively. Membership sets for x_1 and x_2 are assumed the same. The set of five membership functions provides relatively small knowledge base while maintaining a good controller performance.

As the output variable u range was found larger (by experiments), it is divided into seven overlapping regions:

- NB – Negative Big
- NM – Negative Medium
- NS – Negative Small
- NE – Neutral
- PS – Positive Small
- PM – Positive Medium
- PB – Positive Big.

The seven linguistic values are associated with seven membership sets B^k, k =1, ... , 7, with output being an integer number from the interval [1,7]. It is more convenient to refer to the linguistic variables values by their encoded integer values than by their linguistic values and therefore this approach is assumed for the reminder of the thesis.

3.1.3 Triangular membership functions

The seven centres associated with the output sets B^l are: −10.0, −4.5, −2.5, 0.0, 2.5, 4.5, 10.0. These values are obtained by examining the values of u_1, u_2 and u obtained as output from the integration of the state equations, see Equation 3.1. Triangular membership functions are shown in Figure 3.2 − Figure 3.5.

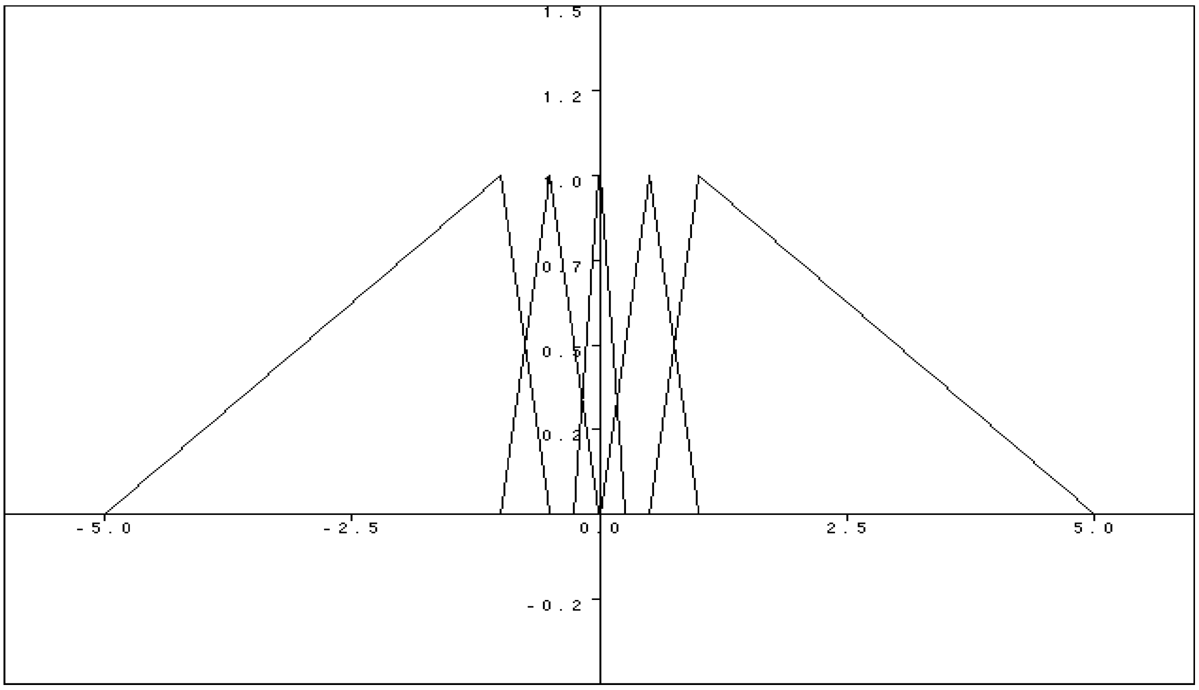

Figure 3.2 Triangular membership functions for x_1 and x_2.

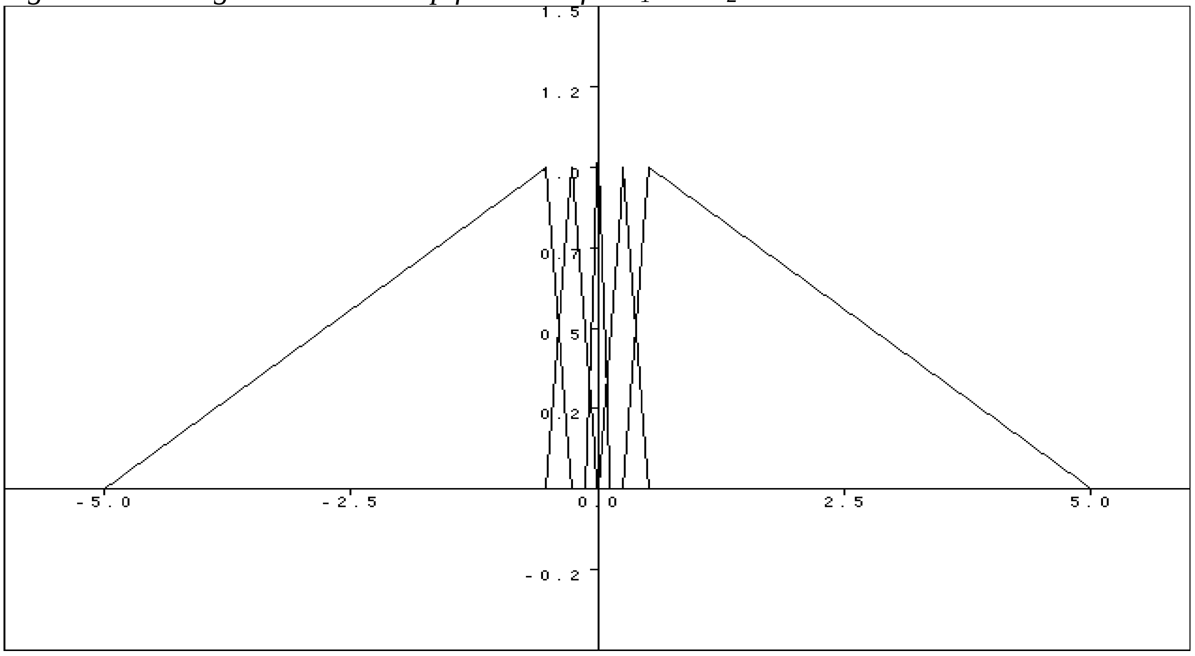

Figure 3.3 Triangular membership functions for x_3.

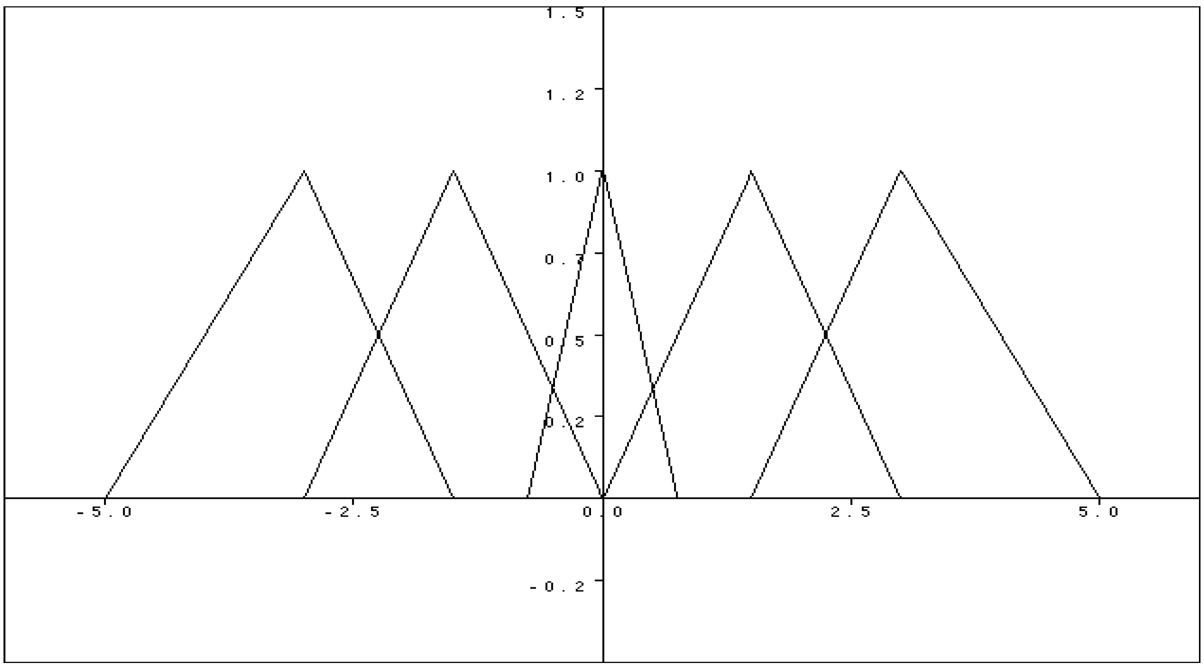

Figure 3.4 Triangular membership functions for x_4.

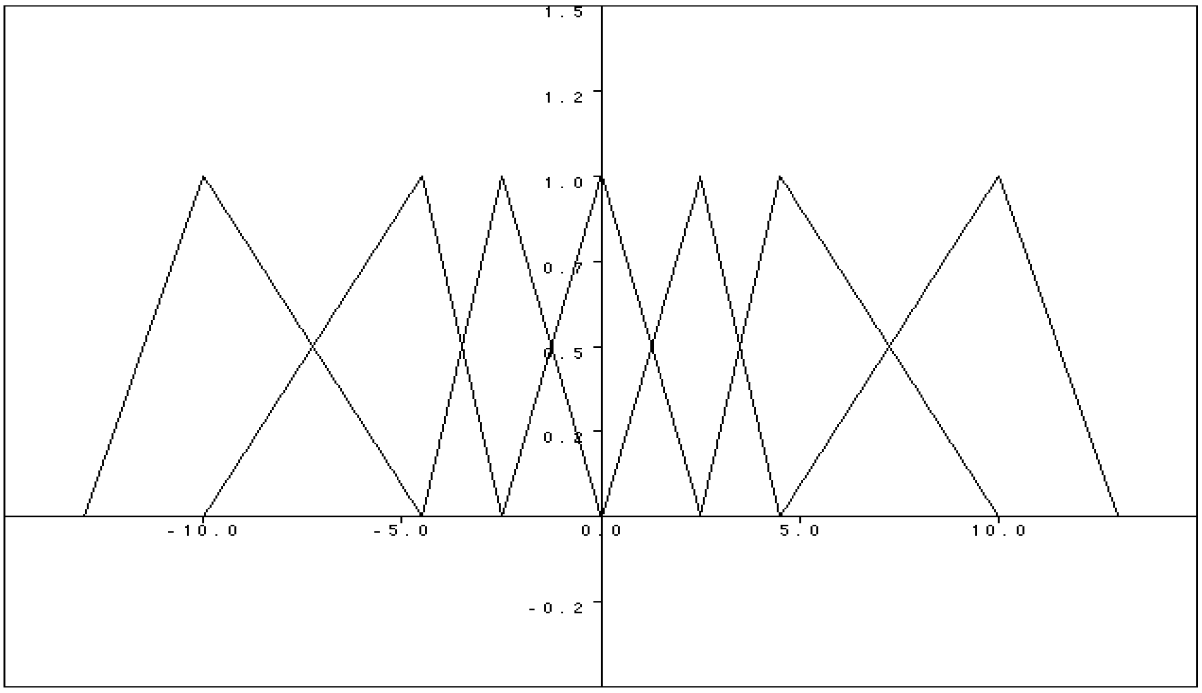

Figure 3.5 Triangular membership functions for u.

3.1.4 Gaussian membership functions

Each domain region for x_i is divided into five overlapping intervals and each assigned membership sets: A_i^k, k =1, ... , 5, which are encoded numerically as integers from 1 to 5 respectively. As with the triangular functions, the set of five membership functions provides small knowledge base while maintaining a good controller performance. The output variable

u range is divided into seven overlapping regions covered by seven membership sets B^k, k = 1, ... , 7. All fuzzy membership functions are assumed to be Gaussian functions with their centres evenly spaced over the range of input and output variables. Thus, for x_1 and x_2 there are five Gaussian membership functions covering [−2.0, 2.0], see Figure 3.6. For x_3 there are five Gaussian membership functions covering [−π/2.0, π /2.0], see Figure 3.7. For x_4 there are five Gaussian membership functions covering [−4.0, 4.0], see Figure 3.8. For u there are seven Gaussian membership functions covering [−15.0, 15.0], see Figure 3.9.

Each Gaussian membership function is defined by three numbers: its centre and the two centres of the neighbouring membership functions (which define standard deviation σ): xm_L – the centre of the neighbouring MF to the left, xm – the centre, and xm_R – the centre of the neighbouring MF to the right. Because five membership functions are used to cover each input variable – it suffices to have three centres to define five Gaussian functions covering the range of input variable with left and right boundary of the range acting as extreme left and right centres. If the value of the variable x falls within interval defined by xm_L and xm_R the membership function value is calculated as:

$$MF(x) = e^{-d\left(\frac{x-xm}{xm_R-xm_L}\right)^2} \tag{3.2}$$

where d = 10.0 is a stretching parameter in the Gaussian Function.

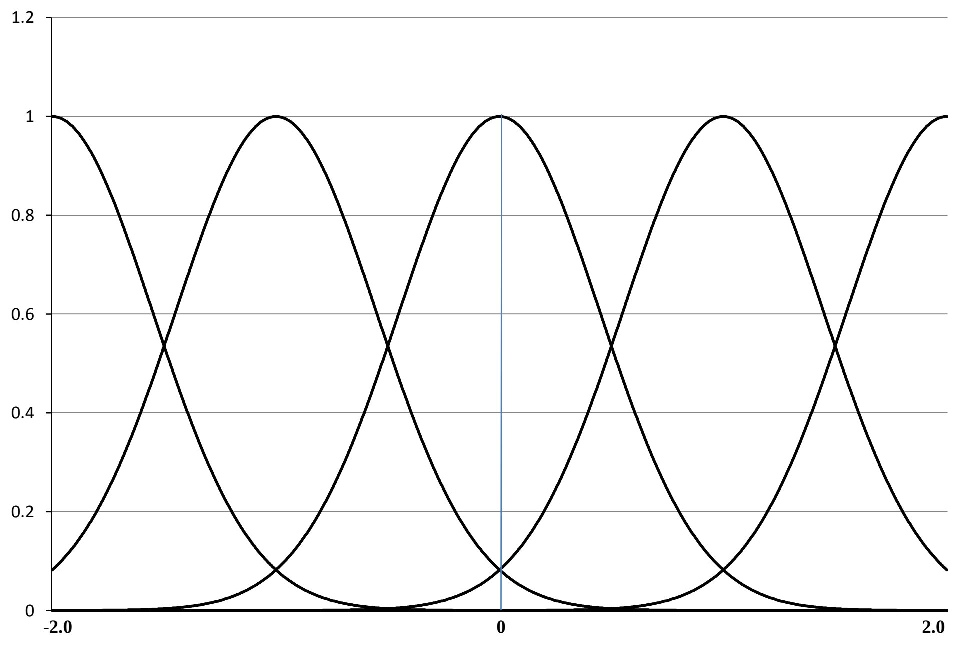

Figure 3.6 Gaussian membership function for x_1 and x_2 input variables.

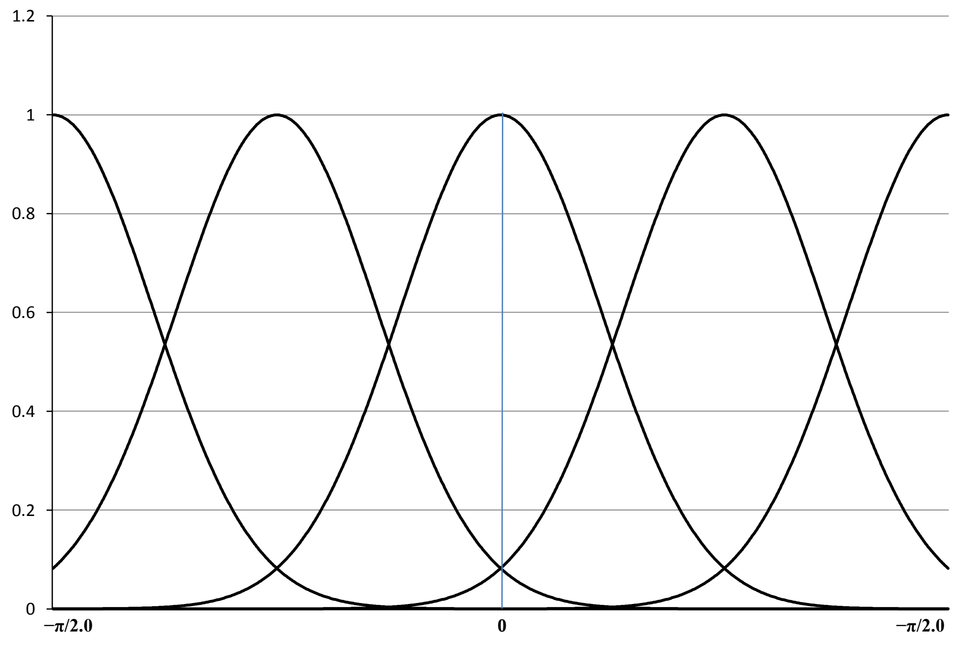

Figure 3.7 Gaussian membership function for x_3 *input variable.*

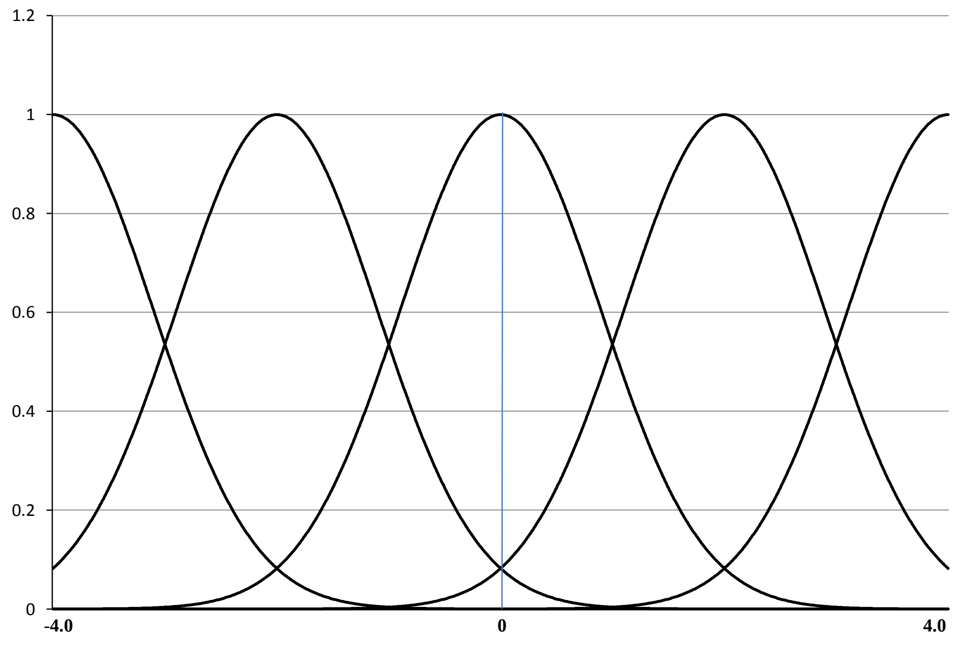

Figure 3.8 Gaussian membership function for x_4 *input variable.*

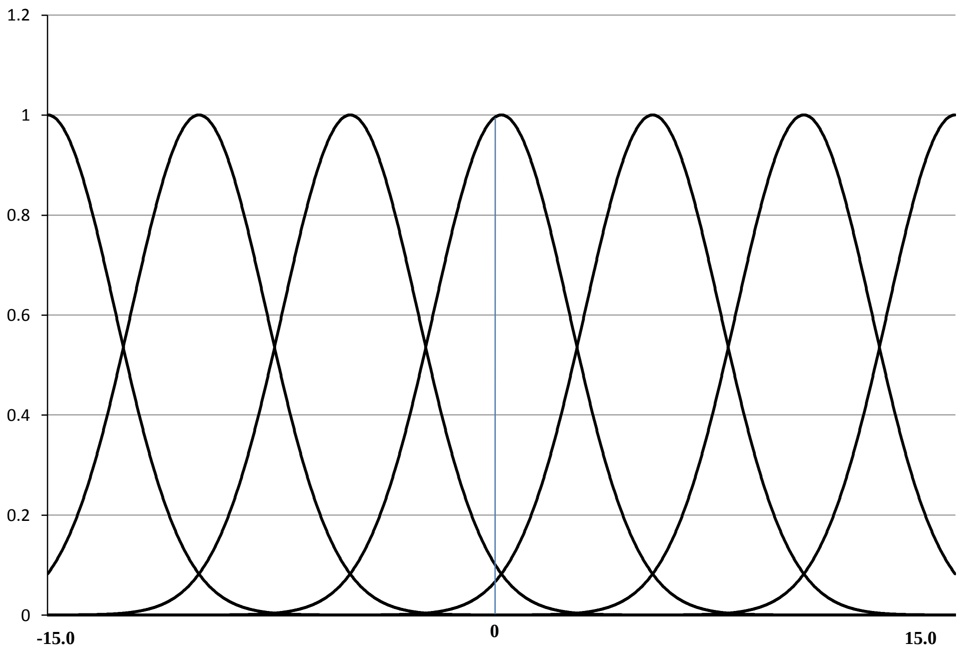

Figure 3.9 Gaussian membership function for output variable.

3.2 Control output

At the heart of the fuzzy logic control system lies the inference engine that applies principles of intelligent reasoning to interpret the rules to output an action from inputs. There are many known types of inference engines in the literature, including the most popular Mamdani and minimum inference engine (Wang 1997).

In all experiments presented in this thesis the control output *u* is calculated using either the Mamdani product or minimum inference engine. In general, other inference engines can be applied, see (Wang 1994). In experiments with the inverted pendulum problem, the product Mamdami and minimum inference engines are used for their specific characteristics.

Given a fuzzy rule base with *M* rules and *n* antecedent variables, a fuzzy controller as given in Equation 3.2 (with Mamdani inference engine) or Equation 3.3 (with minimum inference engine) uses a singleton fuzzifier and centre average defuzzifier to determine output variables.

$$u = \frac{\sum_{l=1}^{M} \bar{u}^l \left(\prod_{i=1}^{n} \mu_{A_i^l}(x_i)\right)}{\sum_{l=1}^{M} \left(\prod_{i=1}^{n} \mu_{A_i^l}(x_i)\right)} \tag{3.3}$$

$$u = \frac{\sum_{l=1}^{M} \bar{u}^l \left(min_{i=1}^{N} \mu_{A_i^l}(x_i)\right)}{\sum_{l=1}^{M} \left(min_{i=1}^{N} \mu_{A_i^l}(x_i)\right)} \tag{3.4}$$

where $\bar{u}^l$ are centres of the output sets B^l and $\mu_{A_i^l}$ are membership functions associated with fuzzy sets A_i^l.

3.3 Single layer fuzzy system

An initial analysis of the learning of fuzzy rules in a single layered system is given in (Stonier and Stacey 1998). In that paper a max-min inference engine is used and all variables are normalised to have their values lie in the interval [−1, 1]. Nevertheless, the formalisation of the knowledge base is similar to the approach used in this thesis.

The l^{th} fuzzy rule for a single layer has the form:

If and (x_1 is A_1^l) and (x_2 is A_2^l) and (x_3 is A_3^l) and (x_4 is A_4^l)) Then (u is B^l) (3.5)

Assuming there are n input variables and there are m fuzzy sets defined for each input variable, the number of fuzzy rules is given by $N = m^n$. The number of rules increases exponentially with the increase of input variables. This is a common problem in all complex systems; the complexity of the system growing exponentially with the number of variables describing the system, and is not unique to fuzzy systems (Wang 1997).

With $n = 4$ input variables and m = 5 fuzzy sets defined for every input variable there are 625 rules in the rule base: $m^n = 5^4 = 625$. Obviously there is only one possible topology for the single layer fuzzy system with all variables as input into a single layer fuzzy system.

Given a fuzzy rule base with M rules and n antecedent variables, a fuzzy controller as given in Equation 3.3 or 3.4 uses a singleton fuzzifier, Mamdani product or minimum inference engine and centre average defuzzifier to determine output variables.

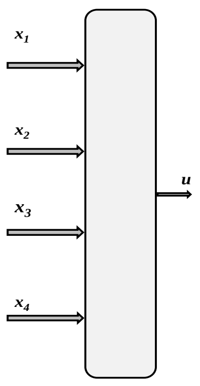

Figure 3.10 Single-layer topology.

3.4 Hierarchical fuzzy systems

The number of rules in the hierarchical fuzzy system is a linear function of the number of input variables. Let assume that there are n input variables in L-layered structure. For every input variable there are m fuzzy sets associated with that variable. Assume further that in the first layer there is n_1 input variables, $2 \leq n_1 < n$, and $n_i + 1$ in the i-th layer, $n_i \geq 1$. If $n_1 = n_i + 1 = c$ is constant for $i = 2, \ldots, L$, then the total number of rules in the hierarchical fuzzy system is given by:

$$M = \frac{m^c}{c-1}\,(n-1)$$

Furthermore, if $m \geq 2$ the number of rules M is minimized when $c = 2$, which means that there are two input variables in every layer (Wang 1997).

Let input configuration that has two input variables in every layer be called 'standard configuration' for convenience. Such a standard input configuration has in the first layer two state variables as input and for successive layers: one state variable and one intermediary variable that can be considered as control approximation in each layer. Such a configuration is shown in Figure 3.11. This standard input configuration provides the minimal number of fuzzy rules in the knowledge base but it does not necessarily provide the best configuration from the control system perspective. Often, such systems do not provide sufficient control performance, especially in complex high-dimensional systems.

Obviously, the 2-layered HFS does not constitute a standard configuration in the above sense as there are 3 input variables in the second layer: two state variables and one intermediary variable as output from the first layer. Therefore, the number of fuzzy rules in the 2-layered HFS for the inverted pendulum system does not provide minimal size of the rule base. However, for the sake of simplicity, this 2-layered HFS will be considered among standard configurations even though it is not optimising the number of rules in the knowledge base.

As mentioned in Section 1.2, the size of the rule base changes exponentially with the increase/decrease of the input variables. In the hierarchical fuzzy structure the size of the rule base becomes a linear function of the number of input variables (Wang 1997). However, standard input configuration can cause deterioration of the control performance. The decomposition needs to be performed along the weak interdependencies between input variables. Obviously, it requires certain knowledge of the physical system in the absence of any automated method of the HFS decomposition.

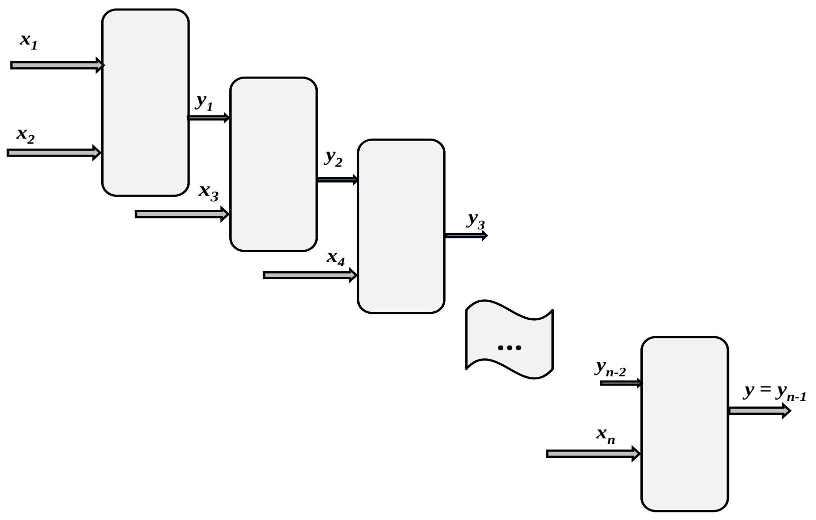

Figure 3.11Standard hierarchical fuzzy system input configuration.

There exist other topologies, both in 2 and 3-layered structures, with a different number of input variables in any particular layer than in the standard configuration. There are more or less than two input variables in the first layer, and different input configurations in subsequent layers for the 3-layered HFS.

Another possible topology for the inverted pendulum problem is the 4-layered HFS with a single input variable in the first layer. However, the size of the knowledge base is larger than in standard 3-layered HFS configuration.

The output for each layer is obtained using the Mamdani or minimum inference engine as given in Equation 3.3 and 3.4, with the appropriate change of variable and associated membership functions for that variable.

Please note that the size of the knowledge base does not automatically translate into better controller performance. The size of the rule base is of paramount importance in high-dimensional systems when the number of rules can make the control system completely impractical due to a long computation time. This consideration is however case dependant.

3.4.1 Two layers

There are six different topologies of the two layered hierarchical fuzzy system in the 'near-standard' configuration: two input variables in the first layer and two input variables plus intermediary control variable from layer 1 in layer 2. This decomposition does not exhaust all possibilities as different input configurations can be considered (some of them discussed later), for example: three input variables in layer 1 and one input variable plus intermediary control from layer 1 in layer 2. Another possibility is to have one input in layer 1 and three input variables plus intermediary control in layer 2 which would result in knowledge base consisting of 880 rules (with current number of membership functions covering domains of input and output variables) and this is more rules than in a single layer architecture thus deeming it impractical. The number of rules may increase (the latter example) or decrease (the first example) in different topologies. However, the two input configuration in each layer seems the most reasonable. The architecture of the 2-layered HFS is shown in Figure 3.12.

For the inverted pendulum system the first knowledge base KB1 has the two inputs to produce as output a first approximation of the control u_1. This u_1 together with another state input x_i and x_j, $i, j \in [1,4]$ are used as input in the second knowledge base KB2 to produce the final control output u.

In the first layer there are $25 = 5^2$ rules in the knowledge base. The lth fuzzy rule for the first layer has the form:

$$\textit{If } (x_i \textit{ is } A_i^l) \textit{ and } (x_j \textit{ is } A_j^l) \textit{ Then } (u_1 \textit{ is } B^l) \tag{3.6}$$

where A_k^l, k = 1,2,3,4, are fuzzy sets for input variables x_k, k = 1,2,3,4, respectively, and where B^l are fuzzy sets for output variable u_1.

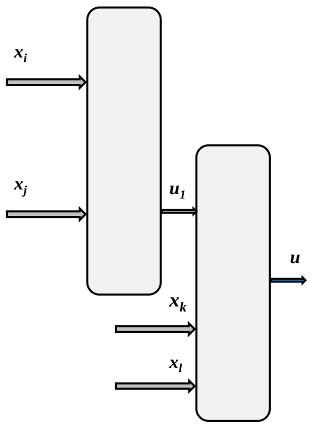

Figure 3.12 HFS: 2-layered input configuration.

For the second layer there are $175 = 7 \cdot 5^2$ rules in the knowledge base. The lth fuzzy rule for the second layer has the form:

$$\textit{If } (u_1 \textit{ is } C^l) \textit{ and } (x_i \textit{ is } A_i^l) \textit{ and } (x_j \textit{ is } A_j^l)\ ) \textit{ Then } (u \textit{ is } B^l) \qquad (3.7)$$

where C^l are fuzzy sets for the input control variable *u*. There are a total of 200 fuzzy rules in this hierarchical structure with the output variable of each layer calculated using the appropriate Equation 3.3 or 3.4.

This hierarchical system is different to that described in (Stonier *et al*. 1998) where the output variable of the first layer is an offset angle added to x_3 before input into the second layer. Furthermore, this new fuzzy system uses a product inference engine rather than a min-max inference engine and results in a different range of values for u_1 and control *u*, requiring adjustments to membership functions and centres.

The approximate control from the first layer u_1 may have not any physical representation. It is an artificial variable connecting the two layers, (Magdalena 1998). Only the final value of control from layer two has actual physical meaning.

3.4.2 Three layers

In standard configuration there are twelve different topologies for the three layered hierarchical fuzzy system with the following input configuration: two input variables in the first layer, and one input variable plus intermediary control value in layer 2 and 3. Again, this decomposition does not exhaust all possibilities, as different input configurations can be considered but with an increased number of rules in the knowledge base. The architecture of the 3-layered HFS is shown in Figure 3.13.

Standard hierarchical fuzzy logic structure has two input variables in the first layer. Then there is one input variable in second and third layer of the 3-layered HFS. This standard input configuration provides the minimal number of fuzzy rules in the knowledge base.

For this system the first knowledge base KB1 has the two inputs, x_i and x_j, $i, j \in [1,4]$ to produce as output a first approximation of the control u_1. This u_1 together with x_k are used as input in the second knowledge base KB2. Then the second layer produces another approximation of control u_2 which with x_l is used as input to the third (and final) layer to produce the final control output *u*.

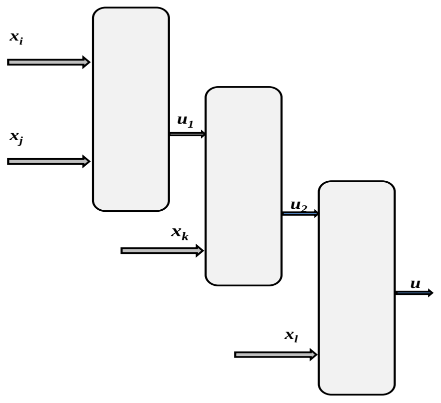

Figure 3.13 HFS: 3-layered standard input configuration.

In the first layer there are 25 = 5^2 rules in the knowledge base. The l^{th} fuzzy rule for the first layer has the form:

If (x_i *is* A_i^l) *and* (x_j *is* A_j^l) *Then* (u_1 *is* B^l) (3.8)

where A_k^l, k = 1,2,3,4, are fuzzy sets for input variables x_k, k = 1,2,3,4, respectively, and where B^l are fuzzy sets for output variable u_1. For the second layer there are 35 = 7 · 5 rules in the knowledge base and the l^{th} fuzzy rule for the second layer has the form:

If (u_1 *is* C^l) *and* (x_k *is* A_k^l)) *Then* (u_2 *is* B^l) (3.9)

where C^l are fuzzy sets for the input control variable u. Similarly, there are 35 rules in the third layer and the l^{th} fuzzy rule has the form:

If (u_2 *is* C^l) *and* (x_l *is* A_l^l) *Then* (u *is* B^l) (3.10)

There are a total of 95 fuzzy rules in this hierarchical structure. Each domain region for x_i is divided into five overlapping intervals and each assigned membership sets: A_i^k, k = 1, ... , 5; which are encoded numerically as integers from 1 to 5 respectively. The set of five membership functions provides relatively small knowledge base while maintaining a good controller performance. The output for each layer is obtained using the Mamdani inference engine as given in Equation 3.3 with the appropriate change of variable and associated membership functions for that variable.

3.4.3 Four layers

There are twenty four different topologies for the four layered hierarchical fuzzy system for the inverted pendulum with one input in every layer plus intermediary control value in layer 2, 3, and 4. The architecture of 4-layered HFS is shown in Figure 3.14. The first knowledge base KB1 has one input x_i to produce as output a first approximation to the control u_1. This u_1 together with x_j are used as input in the second knowledge base KB2. Then the second layer produces another approximation of control u_2 which with x_k is used as input to the third layer to produce the approximate control output u_3. Finally, the input x_l and u_3 in the fourth layer produce the final control value u.

In the first layer there are only five rules in the knowledge base. The l^{th} fuzzy rule for the first layer has the form:

If $(x_i$ *is* $A_i^l)$ *Then* $(u_1$ *is* $B^l)$ (3.11)

where A_k^l, k = 1,2,3,4 are fuzzy sets for input variables x_k, k = 1,2,3,4, respectively, and where B^l are fuzzy sets for output variable u_1.

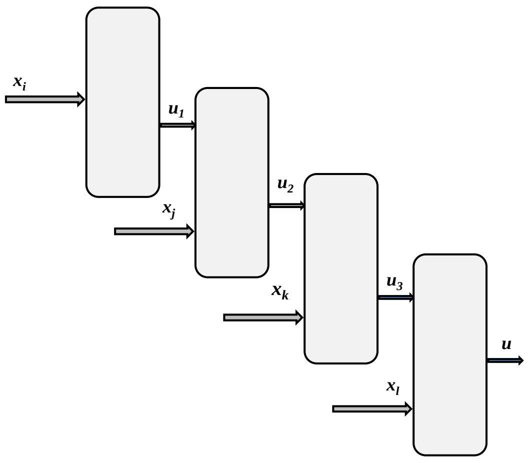

Figure 3.14 HFS: 4-layered non-standard input configuration.

For all the other layers there are 35 = 7 · 5 rules in their respective knowledge bases. For the second layer the l^{th} fuzzy rule has the form:

If $((u_1$ *is* $C^l)$ *and* $(x_k$ *is* $A_k^l))$ *Then* $(u_2$ *is* $B^l)$ (3.12)

where C^l are fuzzy sets for the input control variable u_1.

Fuzzy rules for the third layer has a similar form.

If $((u_2$ *is* $C^l)$ *and* $(x_k$ *is* $A_k^l))$ *Then* $(u_3$ *is* $B^l)$ (3.13)

In the final fourth layer the l^{th} fuzzy rule has the form:

If $(u_3$ *is* $C^l)$ *and* $(x_l$ *is* $A_l^l)$ *Then* $(u$ *is* $B^l)$ (3.14)

There are a total of 110 fuzzy rules in this 4-layered hierarchical structure.

3.5 Non-standard hierarchical topologies

There are other, non-standard, possible input configurations within two and three layered hierarchical structures. They can be called ‘alternative topologies’ as they do not satisfy the condition for the smallest rule base, namely they do not have two inputs in the first layer.

3.5.1 Non-standard two layers HFS

Two examples of alternative input configurations are shown in Figure 3.15 and Figure 3.16. In the first configuration there are three input variables in layer 1 and one input variable plus intermediary control from layer 1 in layer 2 which results in 160 rules in the knowledge base, less than in the previous 2-layered topologies. In the second configuration there is one input in layer 1 and three input variables plus intermediary control in layer 2 which results in a knowledge base of 880 rules - more than in a single layer knowledge base!

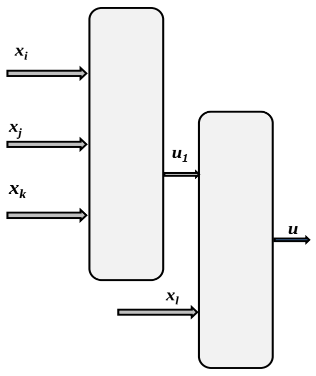

Figure 3.15 HFS: 2-layered non-standard input configuration.

3.5.2 Non-standard three layers HFS

Two non-standard 3-layered input configurations considered for performance examination are shown in Figure 3.17 and Figure 3.18. In the first configuration there is one input variable in layer 1, two input variables plus intermediary control from layer 1 in layer 2, and one input variable plus intermediary control from layer 2 in layer 3, which results in 215 rules in the knowledge base. In the second configuration there is one input variable in layer 1, one input variable plus intermediary control from layer 1 in layer 2, and two input variables plus intermediary control from layer 2 in layer 3, which results again in 215 rules in the knowledge base.

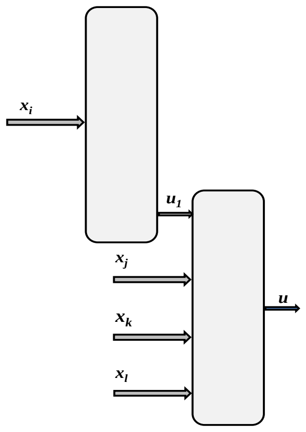

Figure 3.16 HFS: 2-layered non-standard input configuration.

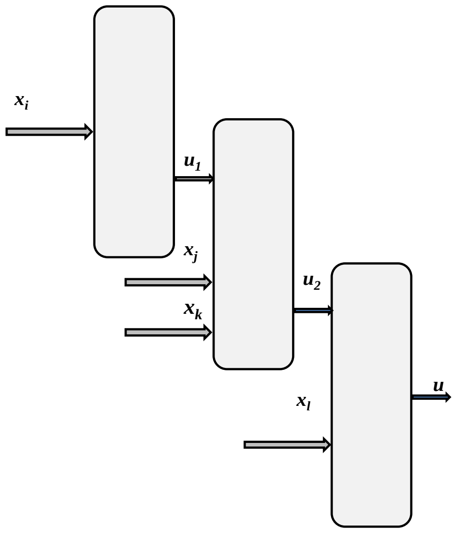

Figure 3.17 HFS: 3-layered non-standard input configuration.

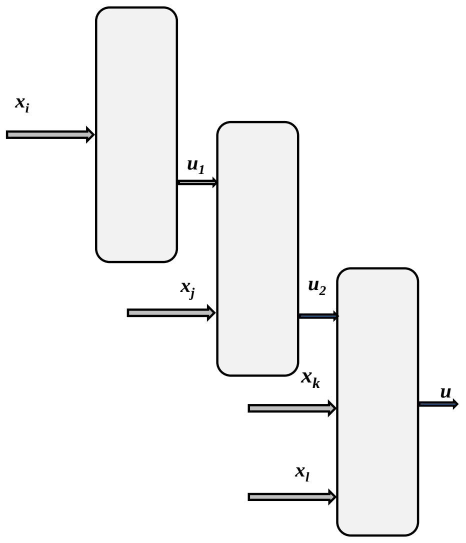

Figure 3.18 HFS: 3-layered non-standard input configuration.

Summary

The mathematical model of the inverted pendulum is introduced. The dynamics and parameters of the system are described. The control problem is defined. The bounds imposed on the state variables are stated. The inverted pendulum is the case study to test the new control methods throughout the rest of this thesis.

Basic concepts of the hierarchical fuzzy control systems are introduced. The minimum and Mamdani inference engine is introduced. Membership functions used in this thesis are defined: triangular and Gaussian membership functions. Different topologies are briefly described. The terminology and notation used in the thesis is introduced. The hierarchical fuzzy systems described in this chapter are used to define control problems in the following chapters.

Chapter 4 EVOLUTIONARY ALGORITHM

4.1 Introduction

Evolutionary algorithms were introduced by J.H. Holland (Holland 1975), mimicking Darwinian evolution and genetics in a mathematical model. Basically, every EA consist of a population of competing individuals and employs the principle of the survival of the fittest. Population evolves in a consecutive generations by applying evolutionary operators such as selection method, crossover, and mutation. Evolutionary algorithms have been successfully applied to a variety of engineering and scientific problems. In many cases they are used as search methods in the solution space. An early practical application of genetic algorithms can be found in D.L. Caroll paper (Caroll 1996).

4.2 Basic evolutionary algorithm

In this section a basic evolutionary algorithm (Michalewicz 1994) is described that is subsequently used to learn (with various modifications) the fuzzy rules in the knowledge bases. There are three different approaches to evolutionary learning:

- Michigan approach: Each individual encodes a single fuzzy rule. The knowledge base is represented by the entire population. Crossover serves to provide a new combination of rules and mutation yields new rules.
- Pittsburgh approach: Each individual encodes the entire rule base. The population is then evolved maintaining a population of candidate rule sets and using genetic operators such as selection, crossover and mutation to produce new rule sets. The solution is found as best individual in the population.
- Iterative approach: Individuals code separate rules, and a new rule is adapted and added to the rule set, in an iterative fashion, at every generation of the evolutionary algorithm.

Both Michigan and Pittsburgh approaches have their advantages. Michigan approach is less computationally demanding and therefore is often used for on-line learning. Pittsburgh approach is more suitable for off-line learning because of its relatively large search space.

In this thesis Pittsburgh approach is used. The entire knowledge base is encoded uniquely as a string of integer numbers representing the fuzzy rule base (Stonier and Mohammadian 2004) and used to find the best rule base by using evolutionary algorithm. Each of the fuzzy rule bases in the HFS can be represented as a linear string of M consequents (the size of all knowledge bases of the HFS). This is possible because each fuzzy rule is uniquely defined by the consequent part represented by an integer defining the output linguistic set. Each rule is identified by the element's position in the linear string.

Evolutionary algorithm is a heuristic search technique that maintains a population of individuals $P(t) = \{ x_1^t, \dots, x_n^t \}$ at iteration t to the next $t + 1$. Each individual can be considered to represent a potential solution to a given problem. Each individual is assigned a measure of fitness (fitness function) which defines how accurate it is as a potential solution to the problem. Depending on how it is defined, either as a maximisation or minimisation problem, the best solution may be that string with the highest or lowest fitness value.

An initial population is created (often random-generated) from a pre-defined number of strings and the fitness of each string is evaluated. Fitness of a given string (called individual or chromosome) is evaluated by a fitness function (sometimes called also an objective function). Typically the population is then ordered or ranked in terms of the fitness value of each string. The new population $P(t + 1)$ is obtained from the old one by the use of genetic operators such as selection, crossover, and mutation. Full replacement policy, if implemented, requires that the population size remains constant from one generation to the next.

An elitism strategy is typically used to pass the fittest individuals or copies of the fittest individual to the new population, so that the information encapsulated in the best individual is not lost and the fittest individuals are passed into the next generation. In many applications this is not necessary but influences the EA convergence.

A selection process is used to obtain parents for mating in the current generation. The most popular is proportional selection to select randomly two parents based on their fitness in proportion to the overall total fitness of the population. Another is tournament selection in which a number of possible parents, say four are selected at random from the population. A tournament is then held to select the two fittest strings and they are used as parents in the next process of crossover to generate children to be passed into the next generation. In the crossover operation a number of 'parent' strings, typically two, are recombined to create 'child' strings. The procedure can be explained on the example of the one-point crossover. Assume that parent strings are:

$\overrightarrow{p_1} = (p_{11}, p_{12}, \dots , p_{1M})$

$\overrightarrow{p_2} = (p_{21}, p_{22}, \dots , p_{2M})$

A random point, an integer, is then selected in the range [1,M] with a certain probability. Assume that the integer is k. The two child strings are then formed by swapping over the tail ends of the two parent strings after the k-th position, that is:

$\overrightarrow{c_1} = (p_{11}, p_{12}, \dots, p_{2k}, p_{2(k+1)}, \dots, p_{2M})$

$\overrightarrow{c_2} = (p_{21}, p_{22}, \dots, p_{1k}, p_{1(k+1)}, \dots, p_{1M})$

Other forms of crossover exist in the literature and are popular in many applications, these include multi-point crossover and variants of the arithmetic crossover, for a full discussion see (Michalewicz 1994). The crossover operator plays a role of sexual reproduction in which two individuals exchange parts of their strings to produce offspring. The children are then added to complete the new population. They also undergo mutation by a mutation operator which perturbs or mutates the strings' structure.

With a given probability, usually small, the mutation operator mutates elements of the strings in the population. This ensures satisfactory diversity within the population which is required for the EA to find better approximate solutions to the problem.

Depending on whether the problem is defined as a maximisation or minimisation problem, the best solution may be the string with the highest or lowest fitness value respectively. The inverted pendulum problem is defined as minimisation problem in this thesis.

The general structure of the evolutionary algorithm may be written as:

```
begin
  t = 0
  Create random P(0)
  Evaluate Fitness of P(0)
  while (not Terminated) do
  begin
    Evaluate Fitness of P(t)
    Create P(t+1) from P(t)
    t = t + 1
  end
end
```

With the right EA parameters and operators, the algorithm converges to a desired solution, i.e., the fittest individual from the last generation satisfying predefined conditions. However there are other possible techniques to produce the final solution, for example averaging the top best individuals.

4.3 String encoding

Fuzzy rule base can be defined as a multidimensional fuzzy decision matrix (or decision table) with values representing consequent part of fuzzy rules. Consider *n* input variables, each taking m_i values, $i = 1, \dots, n$, the matrix would have dimensions: $m_1 \times m_2 \times \dots \times m_n$. Such a multidimensional fuzzy decision table representing the set of fuzzy *IF THEN* rules can be decomposed into a linear string of rows (or columns depending on the assumed convention). The string elements represent consequent part of fuzzy rules. Each rule is identified by the element's position in the string (corresponding to the matrix structure). Hence, the linear string represents the whole fuzzy rule base in the form that allows convenient use of the classical EA's operators.

In the hierarchical knowledge base of any layer each fuzzy rule is also uniquely defined by the position of the consequent part in the string. This consequent part is identified by a particular output fuzzy set, for example, B^k. Such a fuzzy set can be identified by the integer *k*, which has a value in the set $\{1, \dots, N_{MF}\}$, where N_{MF} is the number of linguistic output variable values (or the number of membership functions covering the system output domain). Therefore, each individual string in the evolutionary controller population uniquely represents the hierarchical structure of the fuzzy system.

The above explanation can be formalized as follows: the fuzzy rule bases can be represented as a linear individual string of *M* consequents, $\vec{p}_k = (a_{1, \dots,} a_M)$, where a_j is an integer $\in [1, N_{MF}]$ for $j = 1, \dots, M$. The population can be defined as set of M_p strings: $\vec{p}_k = (a_{1, \dots,} a_M)$, where a_j is an integer $\in [1, N_{MF}]$ for $j = 1, \dots, M$, and N_{MF} is the number of linguistic output variable values (or the number of membership functions covering the system output domain).

For example, the two fuzzy rule bases for the inverted pendulum system can therefore be represented as a linear individual string of $M = 25 + 175 = 200$ consequents, $\vec{p}_k = (a_1, \dots, a_{200})$, where a_j is an integer $\in [1,7]$ for $j = 1, \dots, 200$. The three fuzzy rule base structure can be represented as a linear individual string of $M = 25 + 35 + 35 = 95$ so the population can be defined as follows: $P = \{ \vec{p}_k : \vec{p}_k = (a_{1, \dots,} a_{95}), k = 1, \dots, M_p, a_j \in [1,7] \}$. Similarly, other hierarchical fuzzy structures can be encoded as linear strings of the length depending on their topology.

4.4 Evolutionary population and evolutionary operators

For experiments with the inverted pendulum system the EA setup is described below. The initial population $P(0) = \{ \vec{p}_k : k = 1, \dots , M_p \}$, where M_p is the number of strings (the size of the evolutionary population), is determined by choosing the a_j as a random integer $\in [1,7]$. M_p has been usually set at 300 or 500. In determining successive populations full replacement policy is used and tournament selection with size $n_T = 4$ and a modified mutation operator. An elitism policy is also used with copies of the best string from a given generation passed to the next generation. The number is dependent upon the size of the population. Typically, for a population of $M_p = 100$, two or four copies of the best individual are passed to the next generation. For a population of size $M_p = 500$, four copies of the top five individuals are passed to the next generation.

To maintain diversity of the population crossover operators of parent strings to form two children in the next generation are used. In initial experiments for a single initial condition the crossover is taken as the usual one-point crossover with $p_c = 0.6$. In examination of different topologies and co-evolutionary algorithm so called random crossover is implemented. In later experiments with the compositional method arithmetic and uniform crossover are used.

The random crossover procedure creates *child1* from *parent2* by copying it, then randomly selecting *m*-genes in the parent1 string to copy them in the corresponding positions in the *child1* string. The procedure is repeated for the *child2* string with parent strings roles reversed. The children are then added to complete the new population. The random crossover operator gives more control over crossover process as the number of genes subject to exchange can be arbitrarily determined. They also undergo mutation by a mutation operator which perturbs or mutates the string structures. The pseudo-code below illustrates the procedure:

```
i = 1
mgenarray[i]=rnd(1,lchrom)
while ( i <= mgen-1 )
 {i = i +1
 check = 0
 while (check == 0)
 {   // random number generation
 temp = rnd(1,lchrom)
 check = checkunique(temp,mgenarray,i-1)}
```

```
  mgenarray[i] = temp;}
  for ( i =1; i <= lchrom; i++)
  { child1[i] = parent2[i]
    child2[i] = parent1[i] }
// exchange of genes from parent1 to child1:
for ( i =1; i <= mgen; i++)
 child1[mgenarray[i]]=parent1[mgenarray[i]]
// end of creation of child1
```

where `lchrom` is a length of a string. Please note, that crossover operator performance is case dependent and other operators may perform better.

With a given probability p_m the mutation operator mutates elements of the strings in the population. This ensures satisfactory diversity within the population which is required for the EA to find better approximate solutions to the problem. Mutation is undertaken with probability p_m whose value is determined by a mutation schedule that decreases typically from 0.8 to 0.001 over 1000 generations. Below is the typical mutation schedule used in the computer simulations:

```
 if ( gen ≥ 0   & gen < 100 ) pm= 0.8
 if ( gen ≥ 100 & gen < 200 ) pm= 0.7
 if ( gen ≥ 200 & gen < 300 ) pm= 0.6
 if ( gen ≥ 300 & gen < 400 ) pm = 0.4
 if ( gen ≥ 400 & gen < 500 ) pm = 0.2
 if ( gen ≥ 500 & gen < 600 ) pm= 0.1
 if ( gen ≥ 600 & gen < 800 ) pm = 0.01
 if ( gen > 800) pm = 0.001
```

where `gen` denotes the generation number. The operator is defined by the following pseudo code:

```
 if (mutate)  {
   if (ak = 7)  ak = ak - rnd(1,3)
   else if (ak = 1) ak = ak + rnd(1,3)
   else if (flip(0.5)) ak= ak + rnd(1,3)
        else  ak = ak - rnd(1,3)
   if (ak > 7) ak = 7
   if (ak < 1) ak = 1  }
```

Full replacement policy is implemented and requires that the population size remains constant from one generation to the next.

A selection process is undertaken using tournament selection in which a number of possible parents are selected at random from the population. A tournament is then held to select the two fittest strings and they are used as parents in the next process of crossover to generate children to be passed into the next generation. Tournament selection with size $n_T = 4$ is used in all experiments.

4.5 Objective function

4.5.1 Objective function for the single initial condition EA

The fitness for a single initial condition is evaluated as follows: given an initial condition of the system each string $\vec{p}_k$ can be decoded into the two or more components defining the fuzzy knowledge base for each layer, then the Mamdani or minimum inference formula is used to evaluate u_1, u_2, and u (or only some of the control outputs depending on the selected hierarchical structure) to find the final control to be applied at each value of the state $\vec{x}$. Given an initial state the system state equations are integrated by the Runge-Kutta algorithm (RK4) with step size 0.02 over a sufficiently long time interval $[0,T]$. The fitness f to be minimised, is then calculated based on certain measures of the behaviour of the system over the time interval. These include, the accumulated sum of normalised absolute deviations of x_1 and x_3 from zero, the average deviation from vertical, the average deviation from the origin or $T - T_S$ where T_S (the survival time) is taken to mean the total time before the pole and cart break defined bounds. A penalty is added to the objective if the final state breaks the following bounds: $|x_1| \leq 0.1$, $|x_2| \leq 0.1$, $|x_3| \leq \pi/24$, $|x_4| \leq 3.0$, i.e., leaves the designated target region.

The objective function has the following form:

$$f = \omega_1 F_1 + \omega_2 F_2 + \omega_3 F_3 + \omega_4 F_4 + \omega_5 F_5 \tag{4.1}$$

with:

$$F_1 = \frac{1}{N} \sum_1^N \frac{|x_1|}{x_{max}},\ F_2 = \frac{1}{N} \sum_1^N \frac{|x_2|}{\dot{x}_{max}},\ F_3 = \frac{1}{N} \sum_1^N \frac{|x_3|}{\theta_{max}},\ F_4 = \frac{1}{N} \sum_1^N \frac{|x_4|}{\dot{\theta}_{max}},$$

$$F_5 = \frac{1}{N}(T - T_S) \tag{4.2}$$

where $x_{max} = 1.0$, $\theta_{max} = \pi/6$, $\dot{x}_{max} = 1.0$, $\dot{\theta}_{max} = 3.0$, N is the number of iteration steps. Survival time is defined as: $T_S = 0.02 \cdot N$, with $T = 0.02 \cdot N_{max}$, where the maximum number of iterations $N_{max} = 1000$. The weights ω_k in the fitness function are all positive real numbers. The first and second terms determine the accumulated sum of normalised absolute deviations of $x_1$and x_2 from zero, similarly for the third term and fourth terms in relation to x_3 and x_4, and the last term when minimised, maximises the survival time.

4.5.2 Objective function for the compositional method EA

The fitness f_i of a given string $\vec{p}_k$ is evaluated first for every single initial condition, $i = 1, \ldots ,$ N_c, where N_c denotes the number of initial conditions . Then overall fitness f is determined from the values f_i calculated for every single initial condition and assigned to the string.

A simple evaluation method is selected for the compositional method: the fitness function is evaluated as arithmetic average over all fitness values f_i , $i = 1, \ldots , N_c$, calculated for every single initial condition:

$$f = \frac{1}{Nc} \sum_{i=1}^{Nc} f_i \qquad (4.3)$$

A penalty is added to the objective if the final state breaks the following bounds (i.e., leaves the designated target region):

$|x_1| \leq 0.1$, $|x_2| \leq 0.1$, $|x_3| \leq \pi/24$, $|x_4| \leq 3.0$

Fitness function can be modified in order to reward strings which successfully control the system from a large number of initial conditions. One of the simple methods is to establish threshold values for the objective function and penalize strings that exceed those threshold values (for each init. condition). For example:

Penalty schedule-A:

```
if ObjFun ≥ 0.3·avg and ObjFun < 0.5·avg  then ObjFun = ObjFun + 500.0
if ObjFun ≥ 0.5·avg and ObjFun < 0.8·avg  then ObjFun = ObjFun + 1000.0
if ObjFun ≥ 0.8·avg then  ObjFun = ObjFun + 2000.0
```

Penalty schedule-B:

```
if ObjFun ≥ 0.2·avg and ObjFun < 0.3·avg  then ObjFun = ObjFun + 500.0
if ObjFun ≥ 0.3·avg and ObjFun < 0.5·avg  then ObjFun = ObjFun + 1000.0
if ObjFun ≥ 0.5·avg and ObjFun < 0.6·avg  then  ObjFun = ObjFun + 2000.0
if ObjFun ≥ 0.6·avg and ObjFun < 0.8·avg  then  ObjFun = ObjFun + 3000.0
if ObjFun ≥ 0.8·avg then ObjFun = ObjFun + 5000.0
```

where *avg* is a variable representing average fitness value of the previous population.

Please note, that increasing penalty values might ‘derail’ the evolutionary algorithm. Therefore penalties need to be fine-tuned to focus the EA on selecting strings that perform well for the large number of initial conditions.

Summary

Basic concept of evolutionary algorithm for hierarchical fuzzy control is introduced. Michigan, Pittsburgh and iterative approach are explained briefly. Pittsburgh approach is assumed in this thesis. The structure of the evolutionary algorithm and its functioning is described: population creation, population size, its evolution from generation to generation. Handling of the evolutionary population is described, including the use of the elitist strategy. Basic evolutionary operators are described: selection method, crossover, and mutation. The inverted pendulum problem encoding is explained. This string encoding is used in all further methods presented in this thesis.

Chapter 5 TOPOLOGIES FOR HIERARCHICAL FUZZY SYSTEMS: CASE STUDY

5.1 Introduction

In the context of the HFS, topology means both structure (layers architecture) and input configuration of the hierarchical fuzzy system. In the inverted pendulum problem there are four possible layer structures: 1-layer, 2-layers, 3-layers, and 4-layers, with different input configuration (except single layer topology that have obviously only one possible input configuration).

In this chapter, a single layer fuzzy system and different HFS structures are examined; two layered, three layered, and four layered HFS, with different input configuration. For clarity, some basic facts about HFS are elaborated further.

5.2 Single layer fuzzy system

As described in Section 3.1, each domain region for input variables x_i is divided into five overlapping intervals covered by membership sets A_i^k, k = 1, ... , 5, encoded as integers from 1 to 5. The output variable u is divided into seven regions covered by membership sets B^k, k = 1, ... , 7. All fuzzy membership functions are assumed to be triangular, see Section 3.1.3 and Figure 3.2 – Figure 3.5.

There are 625 rules in the single layer rule base: $5^4 = 625$. Given a fuzzy rule base with M rules and n antecedent variables, a fuzzy controller as given in Equation 3.3 uses a singleton fuzzifier, Mamdani product inference engine and centre average defuzzifier to determine output variables.

Obviously, there is only one possible topology for the single layer fuzzy system with all variables as input into a single layer fuzzy system.

5.3 Hierarchical fuzzy systems

If hierarchical fuzzy structure has two input variables in every layer it optimises the size of the rule base (Wang 1997). For reasons stated in Chapter 3, the 2-layered HFS for the inverted pendulum system does not provide the minimal number of fuzzy. However, the 2-

layered HFS with two input variables in the first layer can be considered as ‘near standard configuration’.

There exist other topologies with a different number of input variables in any particular level than in the standard configuration. This means that different number of input variables in the first layer, less than 2 or more than 2, is considered. Similarly the configuration of input variables in subsequent layers may vary from the standard configuration, from ‘2—2 input variables’ for the 2-layered HFS and from ‘2—1—1 input variables’ for the 3-layered HFS (ignoring intermediary variables in this notation).

Another possible topology for the inverted pendulum problem is the 4-layered HFS with single input variable in each layer. The size of the knowledge base increases in such topologies compared to the standard configuration.

5.3.1 Two layered HFS

There are six different architectures or topologies of the two layered hierarchical fuzzy system with input configuration as follows: two input variables in the first layer and two input variables plus intermediary variable from layer 1 in layer 2. Other possible configurations include:

- Three input variables in layer 1 and one input variable plus intermediary control from layer 1 in layer 2.
- One input in layer 1 and three input variables plus intermediary control in layer 2 with knowledge base of 880 rules.

For the standard configuration in the inverted pendulum system the first knowledge base KB1 has the two inputs to produce as output a first approximation u_1 to the control variable u. This approximate control variable u_1 is used with input variables x_i and x_j, $i, j \in \{1,2,3,4\}$ as input in the second knowledge base KB2. The final control output u is given by Equation 3.3. The l^{th} fuzzy rule in KB1 is given by Equation 3.6 and the l^{th} fuzzy rule in KB2 is given by Equation 3.7. In both knowledge bases KB1 and KB2 there are a total of 200 fuzzy rules.

5.3.2 Three layered HFS

There are twelve different topologies for the three layered hierarchical fuzzy system in standard input configuration: two input variables in the first layer, and one input variable plus intermediary control value in layer 2 and 3. Other, non-standard, input configurations can be considered but with an increased number of rules in the knowledge base.

For the inverted pendulum system, in standard configuration the first knowledge base KB1 has two inputs, x_i and x_j, $i, j \in \{1,2,3,4\}$. The first layer produces a first approximation of the control u_1. This u_1 together with x_k are used as input in the second knowledge base KB2. Then the second layer produces another approximation of control u_2 which with x_l is used as input to the third, and final, layer to produce the final control output u.

The l^{th} fuzzy rule in the first knowledge base KB1 has the form given by Equation 3.8. The l^{th} fuzzy rule in the second knowledge base KB2 is given by Equation 3.9. The l^{th} fuzzy rule in the third knowledge base KB3 is given by Equation 3.10. In all three knowledge bases there are a total of 95 fuzzy rules. The output for each layer is obtained using the Mamdani inference engine as given in Equation 3.3.

5.3.3 Four layered HFS

The four layered topology for the inverted pendulum system has by necessity a non-standard input configuration, see Figure 3.14. There are twenty four possible topologies in the four layered structure. Every layer has one state variable input and additional control approximation in layer 2, 3, and 4. The l^{th} fuzzy rule for every knowledge base KB1—KB4 is given by Equations 3.11—3.14. There are a total of 110 fuzzy rules in all knowledge bases KB1—KB4.

5.4 Non-standard hierarchical topologies

As mentioned before, non-standard input configurations within two and three layered hierarchical structures can be considered as viable hierarchical fuzzy structures. Some of them are examined in search for the best performing control system for the inverted pendulum.

5.4.1 Non-standard two layered HFS

Figure 3.15 shows non-standard topology with three input variables in layer 1 and one input variable plus intermediary control from layer 1 in layer 2. This input configuration produces a total of 160 rules in the knowledge base; a smaller knowledge base than in ‘near standard configuration’. In the second configuration, see Figure 3.16, there is one input in layer 1 and three input variables plus intermediary control in layer 2 which generates a knowledge base of 880 rules.

5.4.2 Non-standard three layered HFS

In the first configuration, shown in Figure 3.17, there is one input variable in layer 1, two input variables plus intermediary control from layer 1 in layer 2, and one input variable plus intermediary control from layer 2 in layer 3, which results in 215 rules in the knowledge base.

In the second configuration, shown in Figure 3.18, there is one input variable in layer 1, one input variable plus intermediary control from layer 1 in layer 2, and two input variables plus intermediary control from layer 2 in layer 3, which results again in 215 rules in the knowledge base.

5.5 Fuzzy systems

The fuzzy system for the HFS topology investigation is described in Chapter 3. All fuzzy membership functions are assumed to be triangular, see Section 3.1.3.

Given a fuzzy rule base with M rules and n antecedent variables, a fuzzy controller as given in Equation 3.3 uses a singleton fuzzifier, Mamdani product inference engine and centre average defuzzifier to determine output variables.

5.6 Evolutionary algorithm

As stated in Chapter 4, the two fuzzy rule base structure can be represented as a linear individual string of M = 25 + 175= 200 consequents, $\vec{p}_k = (a_1 , \dots , a_{200})$, where a_j is an integer ∊ [1,7] for $j = 1, \dots , 200$. The three fuzzy rule base structure can be represented as a linear individual string of $M = 25 + 35 + 35 = 95$ consequents, $\vec{p}_k = (a_1 , \dots , a_{95})$, where a_j is an integer ∊ [1,7] for $j = 1, \dots , 95$. Other hierarchical fuzzy system structures can be represented in a similar fashion.

The fitness f_k of a given string $\vec{p}_k$ can be evaluated as described in Section 4.5.1. The Mamdani formula is used to evaluate u_1, u_2, u_3 and u (depending on the fuzzy logic topology) to find the final control to be applied at each value of the state $\vec{x}$. Given an initial state the system state equations are integrated by the Runge-Kutta algorithm (RK4) with step size 0.02 over a time interval $[0,T]$, where $T = 0.02 \cdot N_{max}$ with the maximum number of iterations N_{max}

= 1000. The fitness f_k to be minimised, is then calculated according to Equation 4.1 and 4.2. A penalty of 1000 is added to the objective if the final state leaves the designated TR.

The initial population $P(0) = \{ \vec{p}_k : k = 1, \ldots , M_p \}$ is determined by choosing the a_j as a random integer $\in [1,7]$. M_p denotes the number of strings – the size of the evolutionary population. The new population $P(t + 1)$ is obtained from the old one by the use of genetic operators. Full replacement policy is implemented and requires that the population size remains constant from one generation to the next. A selection process is undertaken using tournament selection with size $n_T = 4$.

An elitism policy is implemented with four copies of the ten top individuals (forty copies altogether) passed to the next generation. In investigation of the HFS topologies random crossover is used, see Section 4.4 for details.

In experiments with various topologies for the inverted pendulum system the mutation is undertaken with probability p_m. Its value is determined by a mutation schedule that decreases from 0.8 to 0.001 over 300 generations.

```
if ( gen ≥ 0   & gen < 50  ) pm = 0.8
if ( gen ≥ 50 & gen < 100  ) pm = 0.7
if ( gen ≥ 100 & gen < 150 ) pm = 0.6
if ( gen ≥ 150 & gen < 200 ) pm = 0.3
if ( gen ≥ 200 & gen < 250 ) pm = 0.1
if ( gen ≥ 250 & gen < 300 ) pm = 0.01
if ( gen > 300) pm = 0.001
```

where `gen` denotes generation number. This mutation schedule is different from the schedule described in Chapter 4.

The above described evolutionary algorithm is used to learn fuzzy rules in the HFS that constitutes a control system for the inverted pendulum system.

5.7 Experimental setup

The relatively low number of all possible topologies enables their examination one by one and finding the topology with the best controller performance. Which topology provides the best controller is decided by considering the various aspects of controller performance:

- State variables convergence history (for example: undesired oscillations).
- Time in which the system reaches the target region.
- Control action magnitude and degree of controller smoothness.

For each topology ten simulations are run with randomly generated initial populations.

5.7.1 Initial condition

The initial state is that given in (Stonier *et al* 1998): $\vec{x}_0$ = (0.5, 0.0, 0.01, 0.0) in order to make results comparable.

5.7.2 Initial population

Initial population is randomly generated. Every string element (representing an individual in EA population) is assigned randomly generated integer value ∊ [1,7].

5.7.3 Population size

The population size is set at $M_p = 500$ and maintained at this level for all generations.

5.7.4 Termination condition

The evolutionary algorithm is terminated after 300 generations; except for single layer FS that is terminated after 500 generations, as there is little or no change in the minimum value of the objective function in the following generations. The best controllers at this generation are seen not to break defined constraints and the system is stabilised within the determined target region, see Section 3.1.1.

5.7.5 Fitness function

The following fitness function parameters: $\omega_1 = 3000$, $\omega_2 = 2000$, $\omega_3 = 0$, $\omega_4 = 0$, $\omega_5 = 5000$, are selected for all simulations except single layer fuzzy system with: $\omega_1 = 1000$, $\omega_2 = 0$, $\omega_3 = 1000$, $\omega_4 = 0$, $\omega_5 = 5000$. The values were determined by experimentation.

5.8 Computer simulations

The minimum, average and maximum of objective function are examined for every topology over consecutive generations. The results are fairly similar, both for the 2 and 3-layered HFS. Examples for typical simulations are given in Figure 5.1 and Figure 5.2. To illustrate the controller performance (for each HFS topology) one of the best performing controllers from ten simulation results is selected, see Figure 5.3 – Figure 5.56.

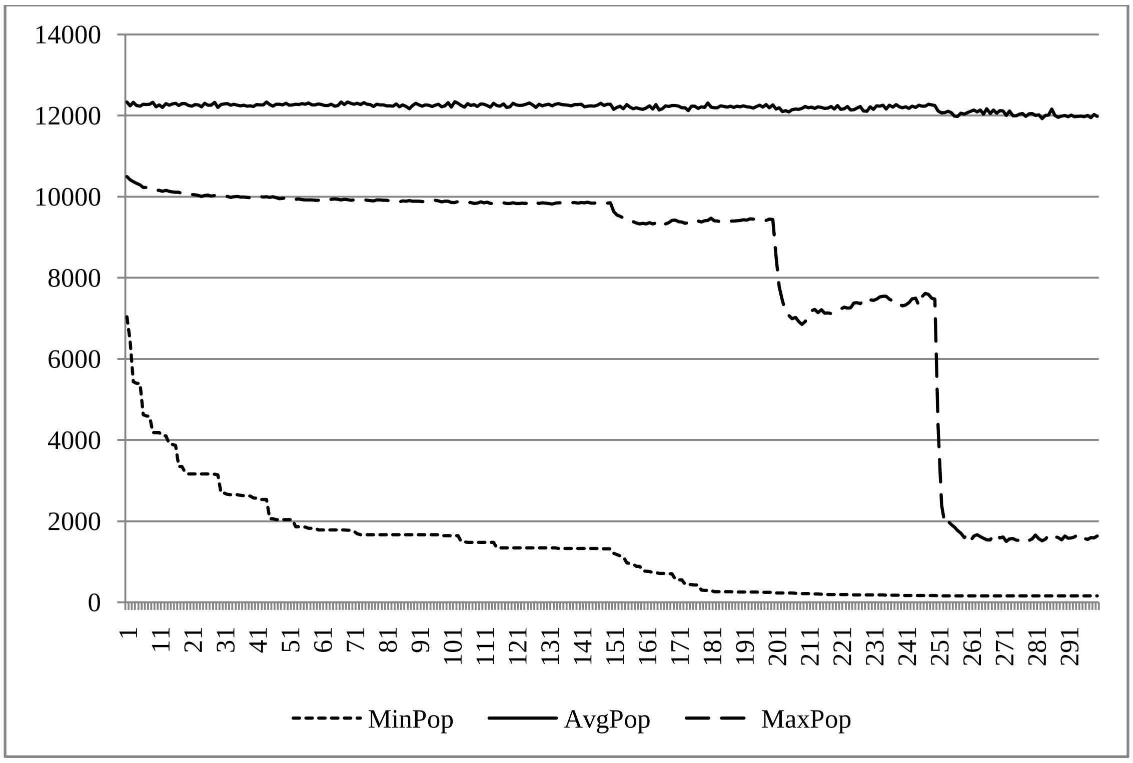

Figure 5.1 Minimum, average and maximum objective function values over 300 generations for L2-34-12.

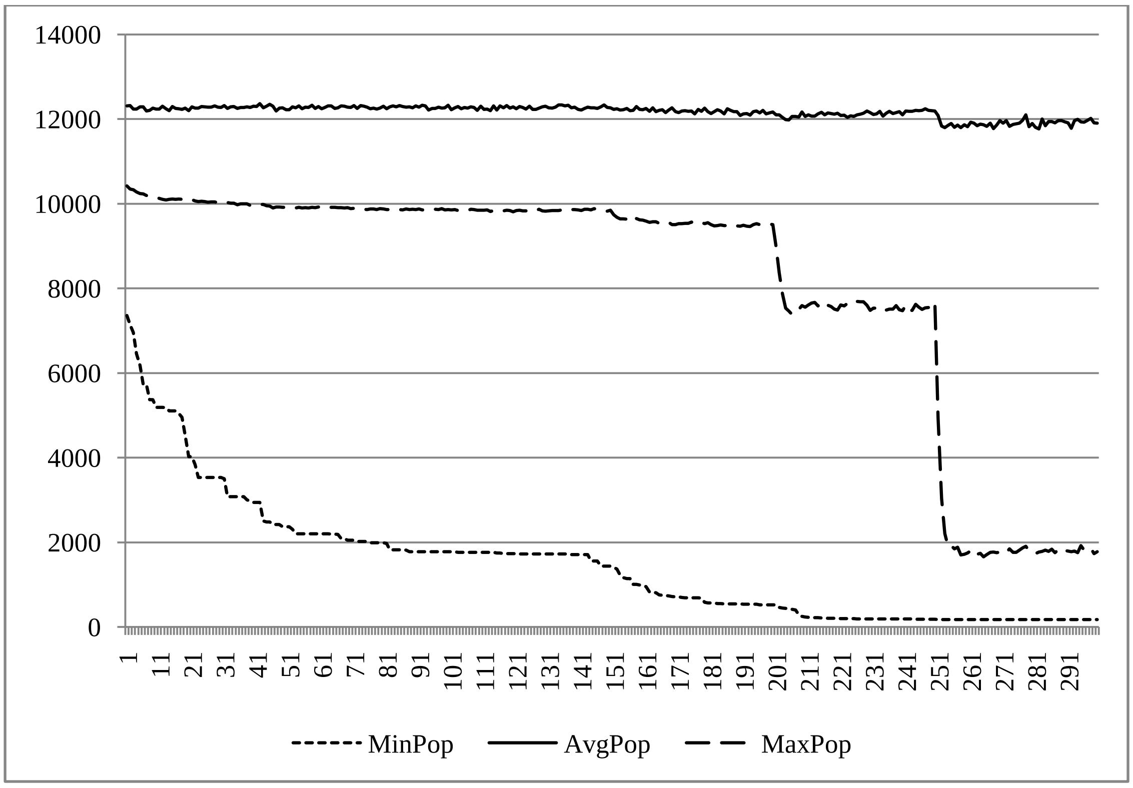

Figure 5.2 Minimum, average and maximum objective function values over 300 generations for L3-34-1-2.

5.8.1 One layer FS results

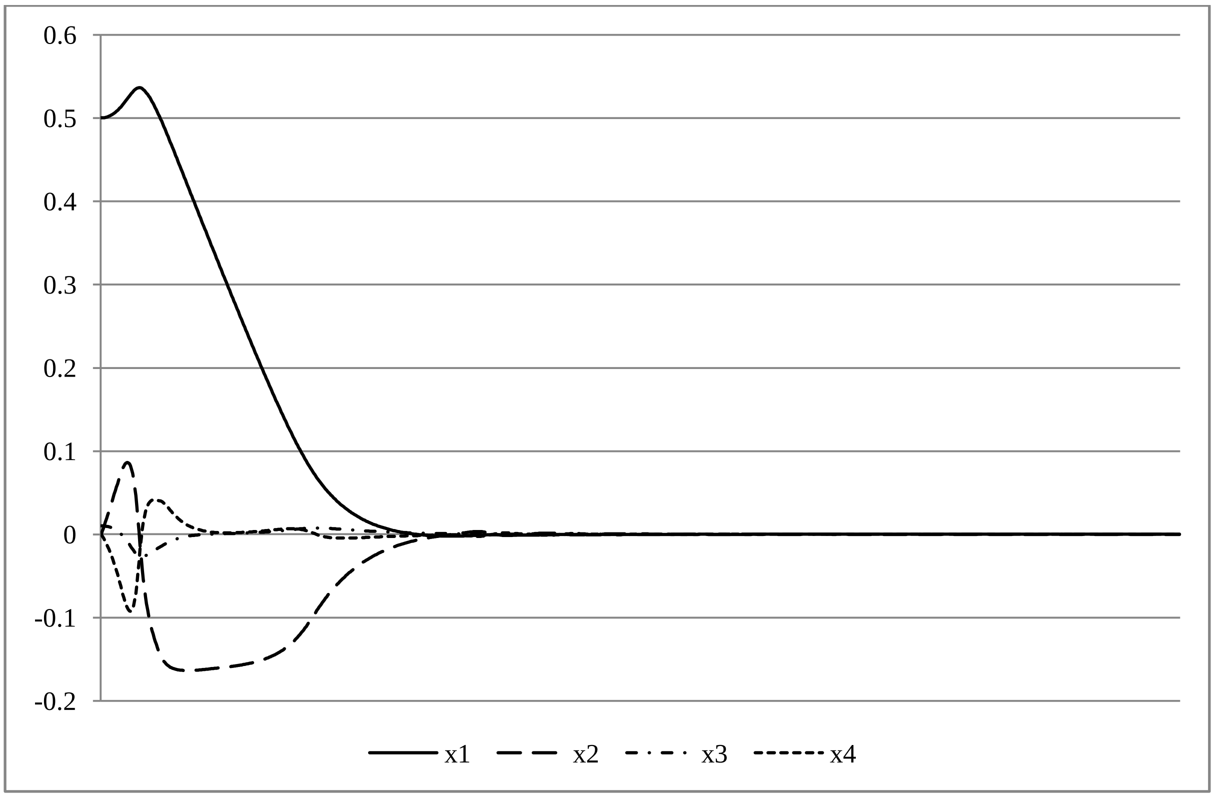

Figure 5.3 State variables convergence L1-1234.

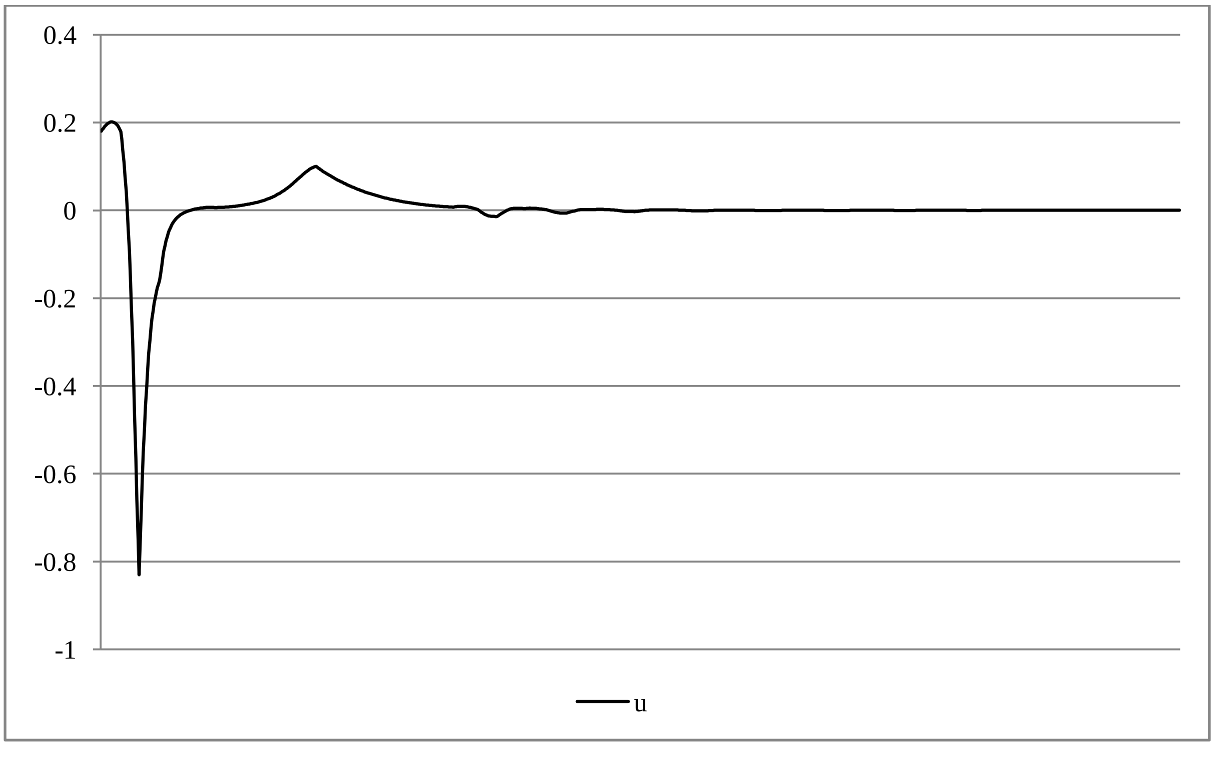

Figure 5.4 Controller L1-1234.

The EA is run ten times for the single layer FS with different initial random populations. Within around 200 generations the best fuzzy controller at each generation achieved convergence of state variables to the designated TR and maintained it within this region for the remainder of the prescribed time T_f. Typical convergence and controller output values are shown in Figure 5.3 and Figure 5.4. As it can be seen from Figure 5.3 the stabilisation is smooth and regular for all state variables. The controller is ‘frugal’, with values lying in [−0.83, 0.202], the best simulation results in terms of control magnitude. In ten simulations for L1-1234 there are controllers with faster stabilisation times but with higher control magnitude.

5.8.2 Two layered HFS results

The EA is run for the 2-layered HFS with many different initial random populations and the controller, evolved within 110—170 generations, achieves convergence of the state variables to the target region and maintains them within specified TR bounds until final time T_f. The evolutionary algorithm converges thus faster than in (Stonier *et al* 1998), (Stonier and Zajaczkowski 2003).

The best result is shown in Figure 5.5 and Figure 5.6, where pole angle x_3 and its angular velocity x_4 are input to layer 1, and cart position x_1 and its speed x_2 as input variables to layer 2. It is also the easiest knowledge base to learn. Stabilisation is very quick for a wide range of parameters $\omega_{k,}$ and it is typically to 5 decimal place accuracy for all variables. This result gives the first indication as to which input configuration provides the best controller performance. Furthermore in some simulations a small control effort is required, one of the best results being with control values in [−2.0, 1.6]. The controller shown in Figure 5.6 has control magnitude in [−6.4, 4.1].

Because the HFS decomposition should match weak interdependency between input variables, this results shows that the inverted pendulum system can be split into two subsystems:

- Pole, represented by input variables x_3 and x_4.
- Cart, represented by input variables x_1 and x_2.

Obviously, if topology *L2-34-12* provides good control structure then *L2-12-34* controller is expected to achieve similar performance as it is decomposed along the same weak interdependence between input variables only with pole variables replaced with cart variables as input in the first layer. Indeed, the *L2-12-34* controller performance is good and very similar in both state variable convergence and controller magnitude to *L2-34-12* controller. However, results for controllers with topology *L2-12-34* were much less consistent in ten simulation runs than for *L2-34-12*. On average, controller with topology *L2-34-12* provides slightly better performance and consistency of the EA solutions. Comparison between controllers with *L2-14-23* topology and *L2-23-14* shows that swapping input variables between layer 1 and layer 2 can have a significant effect on the controller performance. This indicates that the HFS topology is a decisive factor in the controller performance. This assertion is confirmed by the 3-layered HFS results presented in the following section.

The most 'frugal' is controller with topology *L2-23-14*, with values in [−0.87, 0.97] but state variable convergence to the TR is not considered satisfactory, see Figure 5.11. An acceptable convergence of the state variables to the TR is achieved by some of the *L2-13-24* controllers but it is affected by continuing small oscillations in all state variables, see Figure 5.15.

The relatively good controller performance is delivered by controller with topology *L2-14-23* characterised by regular and smooth convergence of all state variables. A typical for ten simulations result is shown in Figure 5.9 and Figure 5.10, where the cart position x_1 and angular velocity of the pole x_4 are input to layer 1. This topology is an example of configuration of 'mixed' input variables where decomposition breaks strong interdependence of state variables. A poor controller performance can be expected in such cases but if most significant state variable is an input in the first layer then it has positive moderating effect on the control process, see (Raju and Zhou 1993), (Zajaczkowski and Stonier 2008). The controllers with topologies *L2-23-14* and *L2-24-13* do not exhibit the desired convergence to the TR and their performance is erratic, see for Figure 5.11 and Figure 5.13 respectively.

Stabilisation times (the time at which all state variables reach the TR and remain within its bounds) for each HFS are grouped in the Table 5.1. The stabilisation times are given in seconds.

Table 5.1 Stabilisation times for 1 and 2-layered HFS

Run No	L1-1234	L2-12-34	L2-13-24	L2-14-23	L2-23-14	L2-24-13	L2-34-12
10	3.92	9.12	4.16	3.1	9.02	9.36	2.74

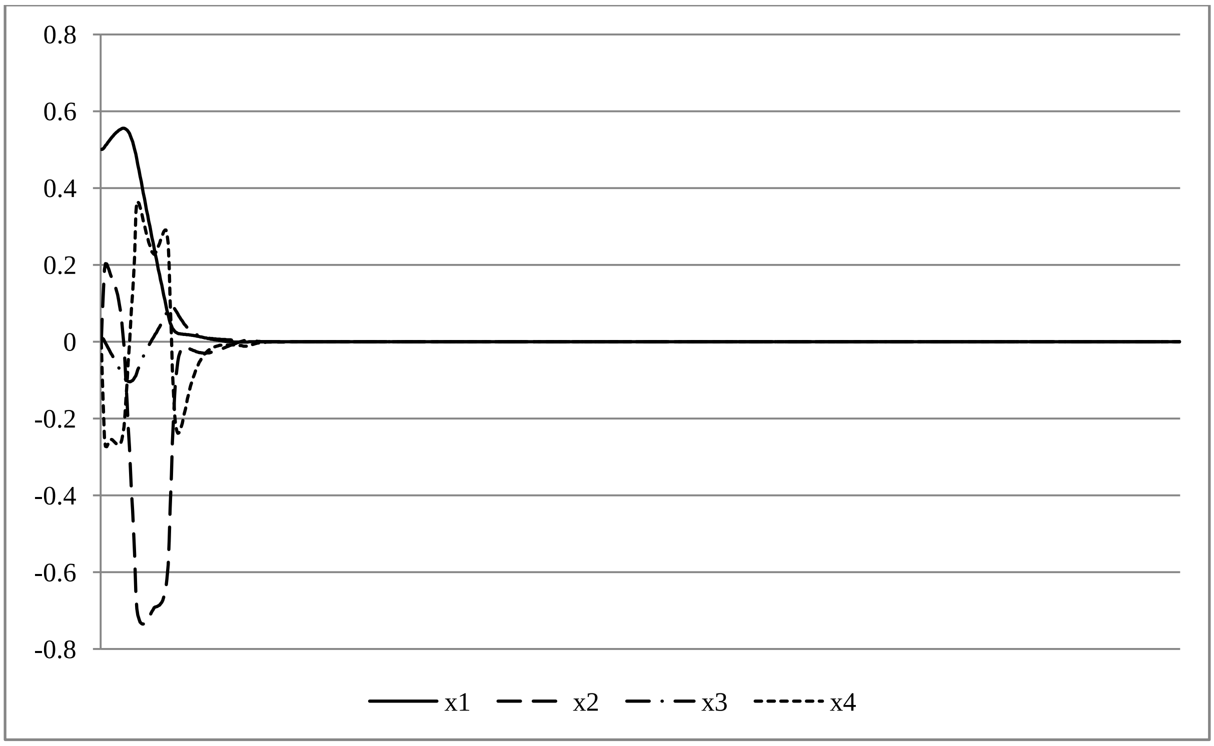

Figure 5.5 State variables convergence L2-34-12.

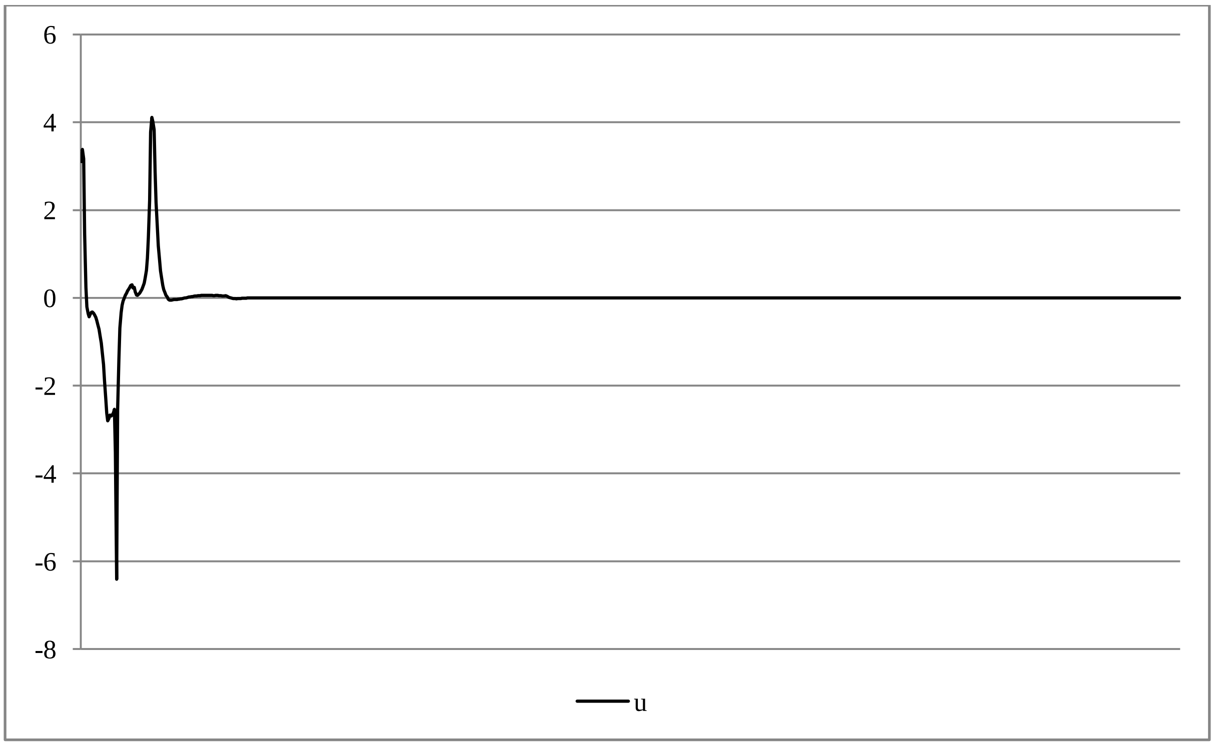

Figure 5.6 Controller L2-34-12.

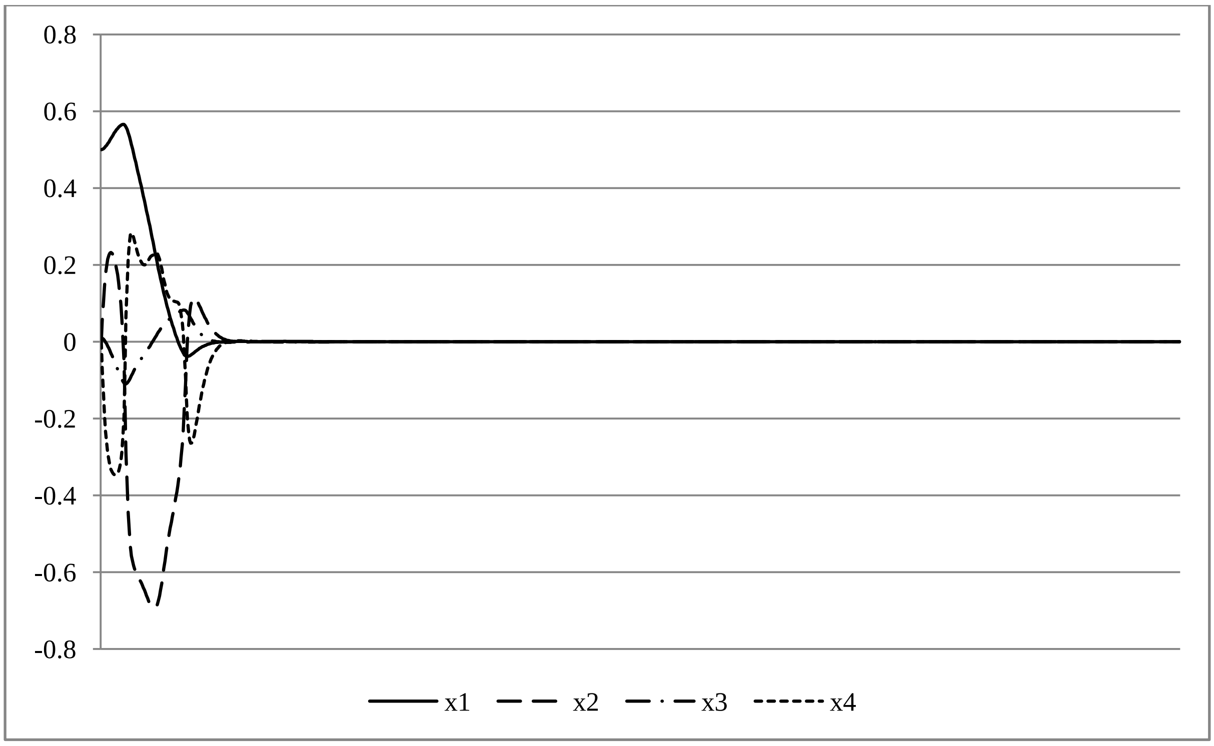

Figure 5.7 State variables convergence L2-12-34.

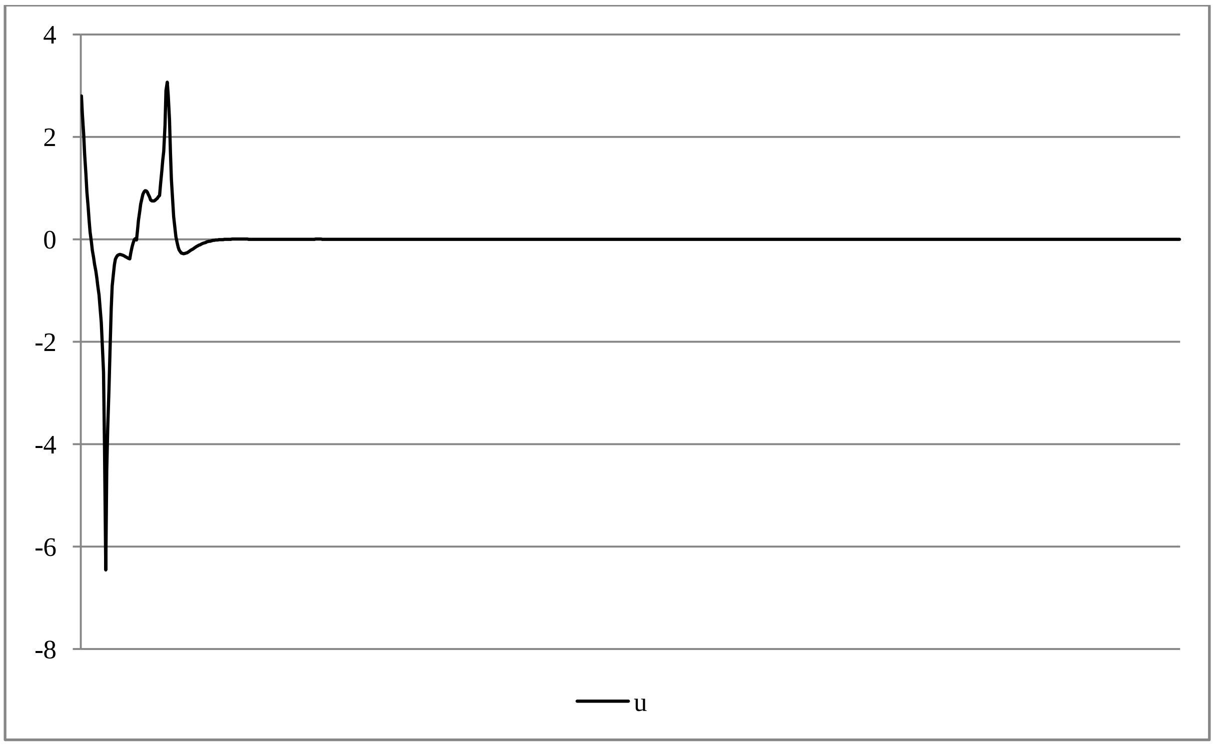

Figure 5.8 Controller L2-12-34.

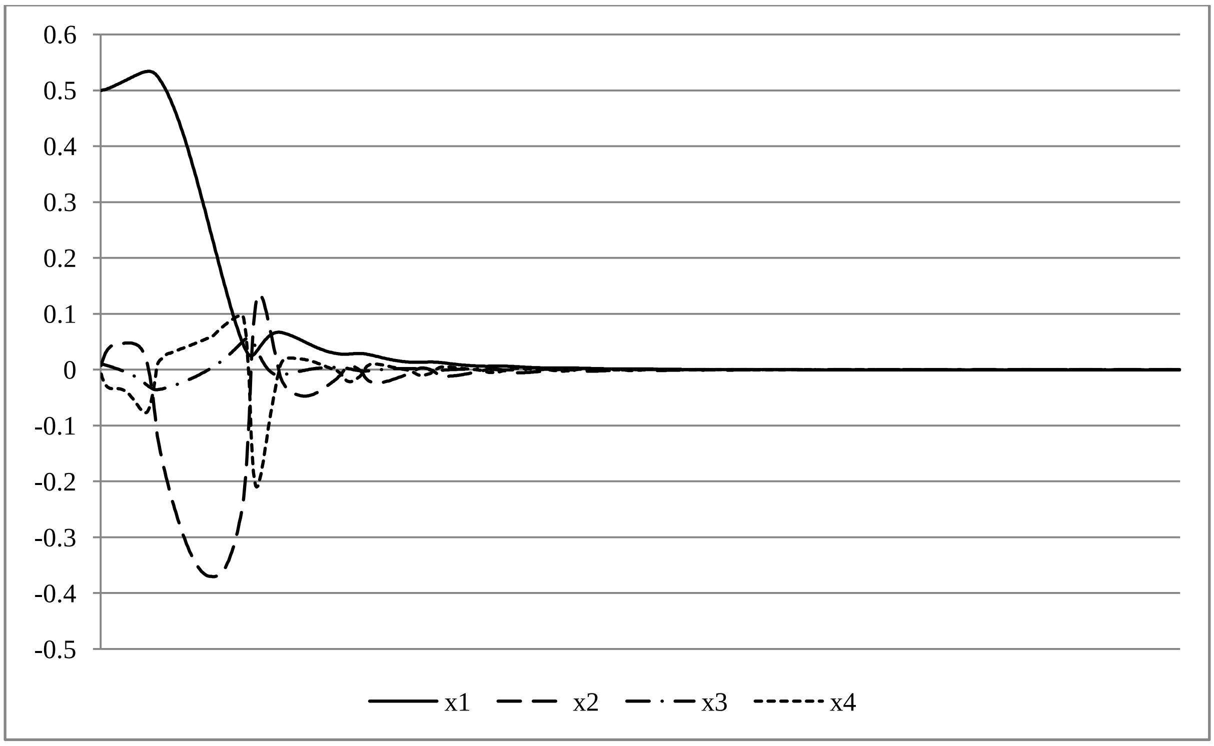

Figure 5.9 State variables convergence L2-14-23.

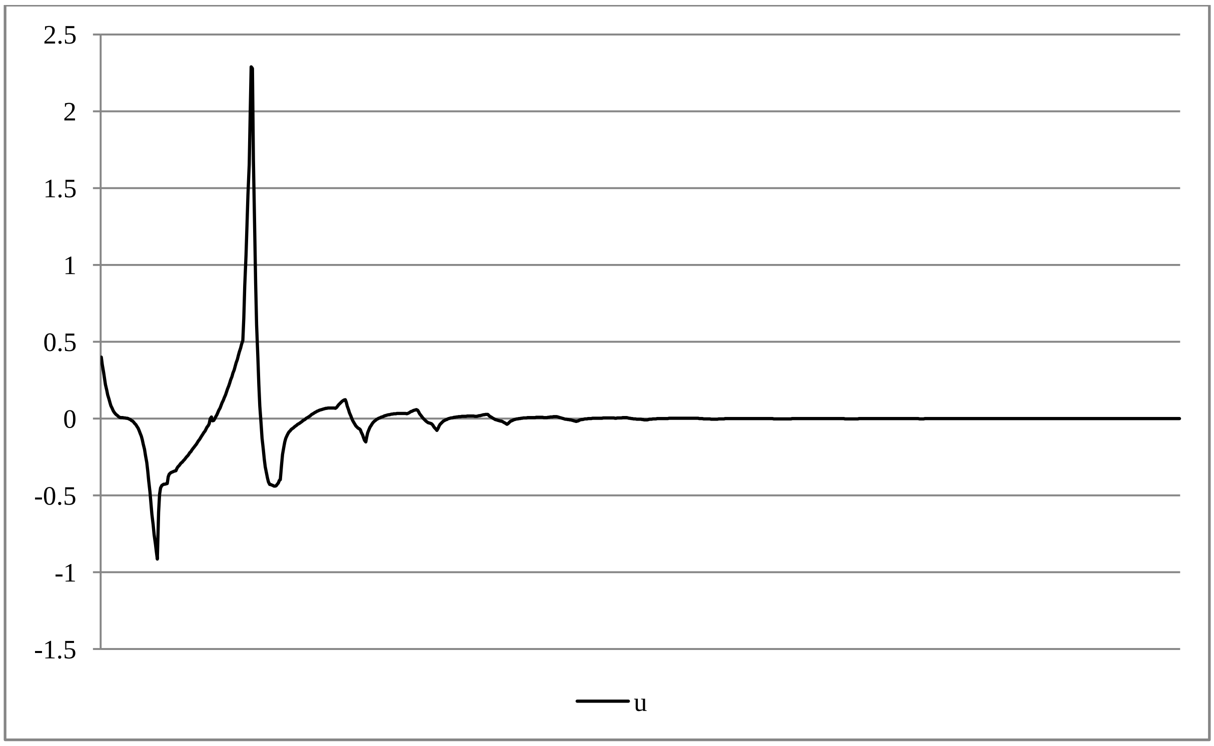

Figure 5.10 Controller L2-14-23.

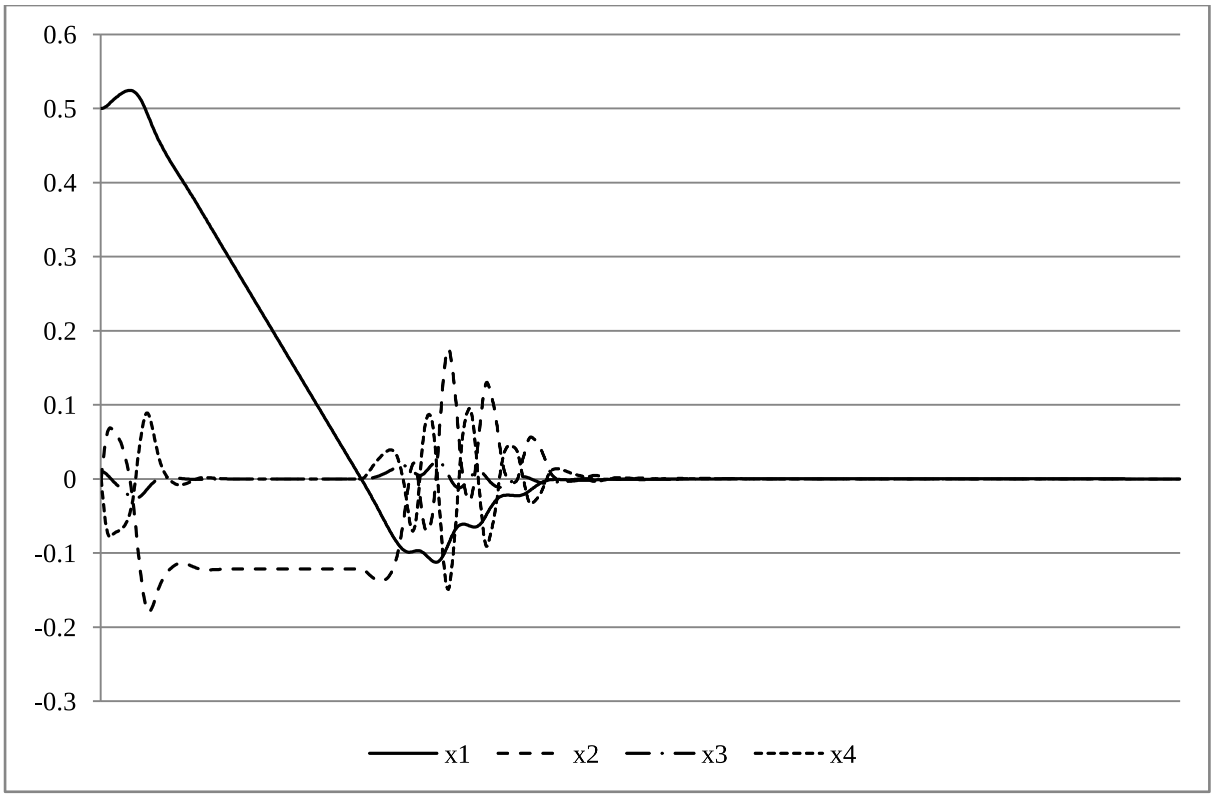

Figure 5.11 State variables convergence L2-23-14.

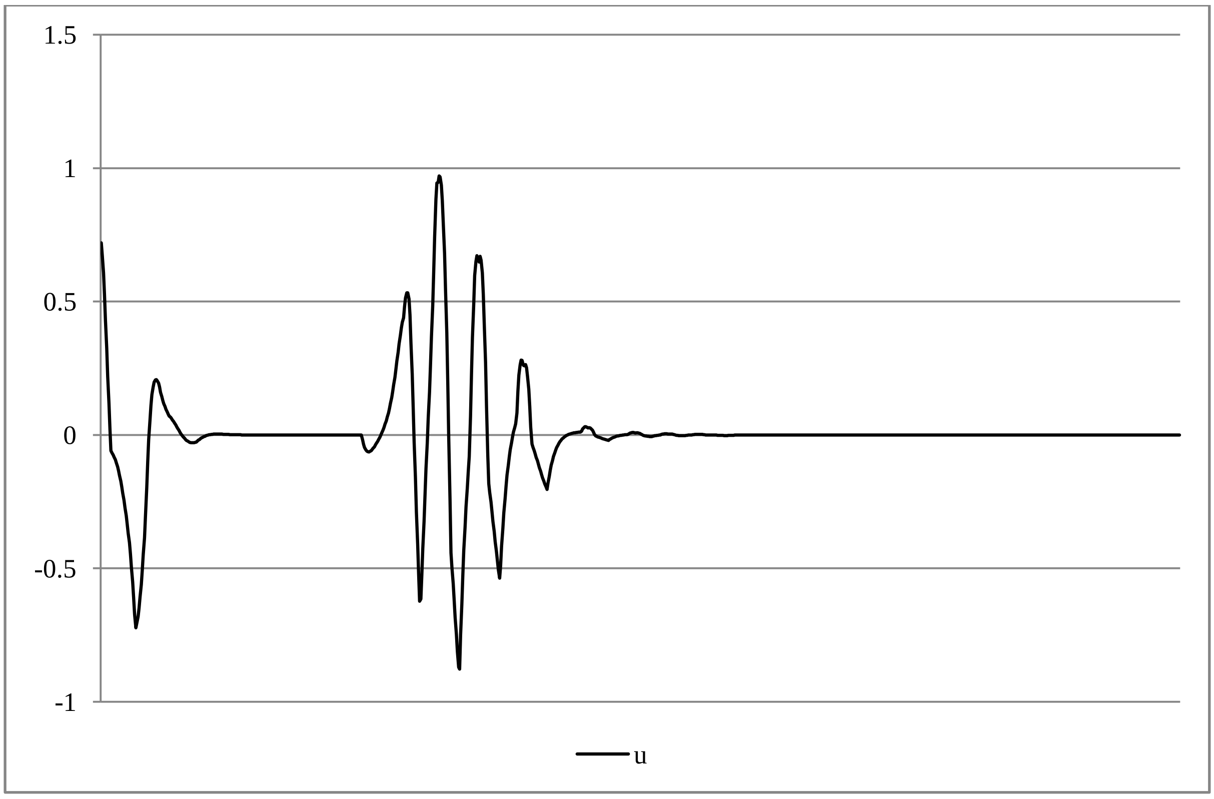

Figure 5.12 Controller L2-23-14.

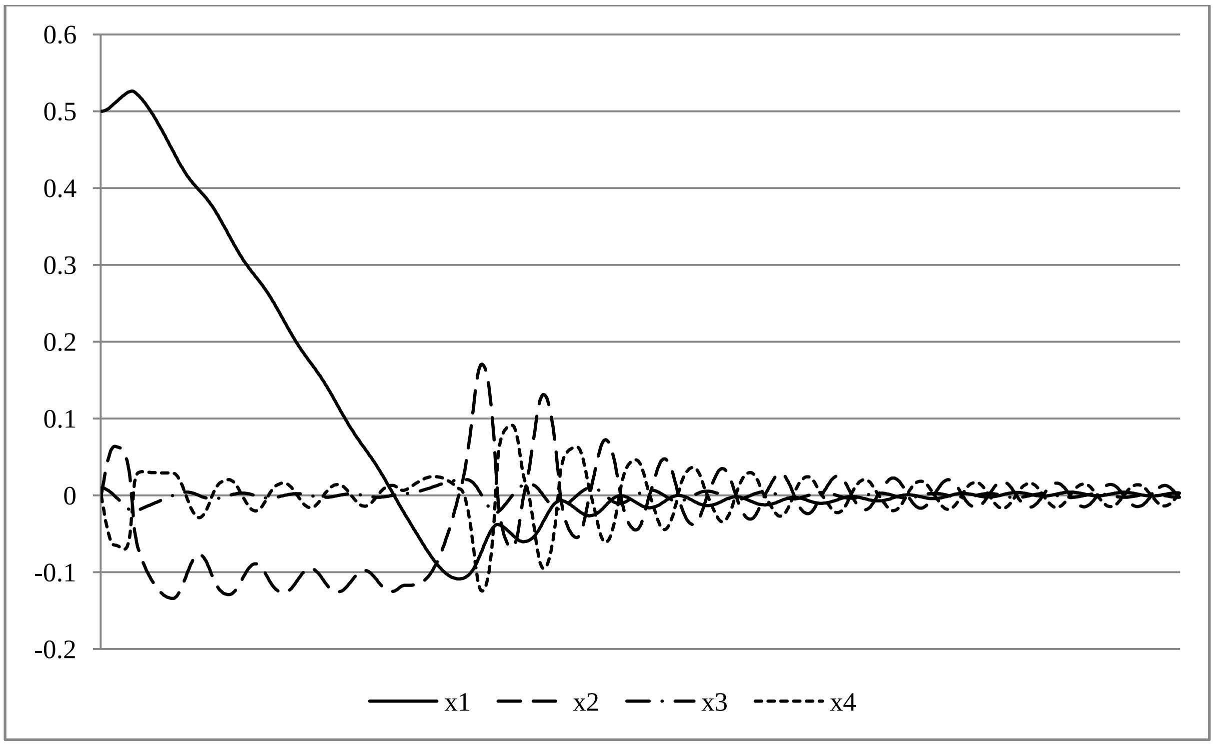

Figure 5.13 State variables convergence L2-24-13.

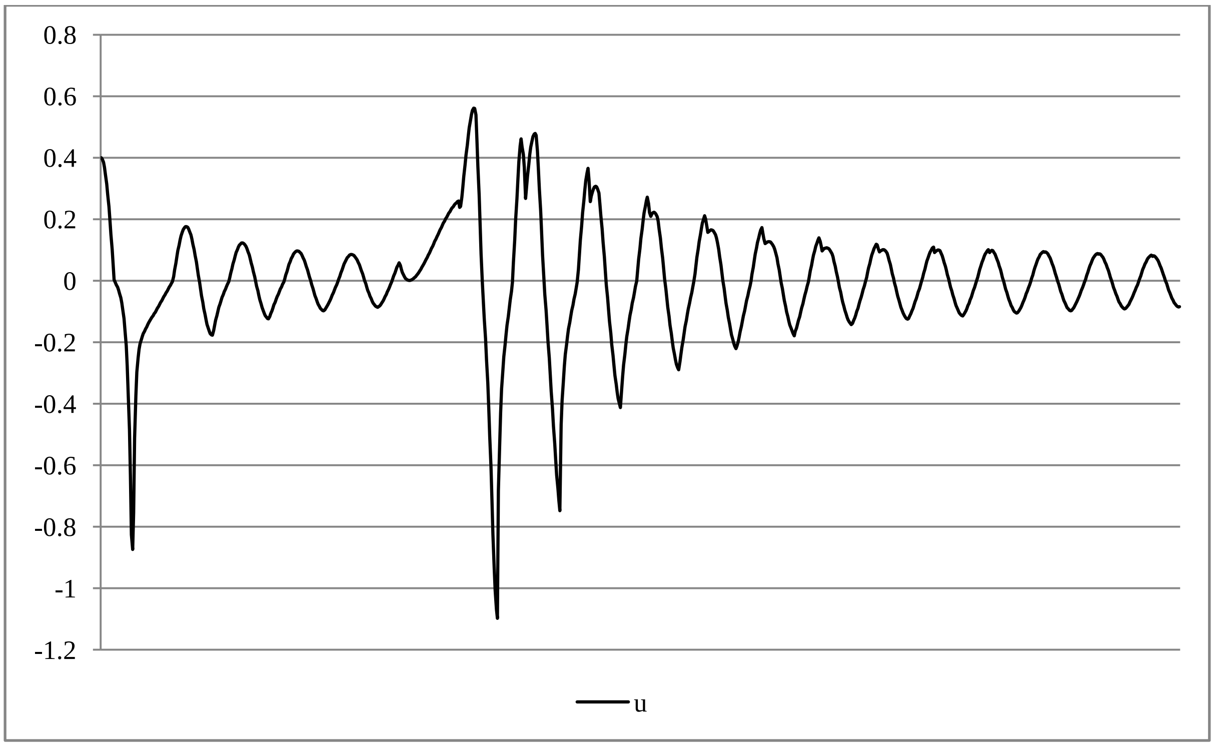

Figure 5.14 Controller L2-24-13.

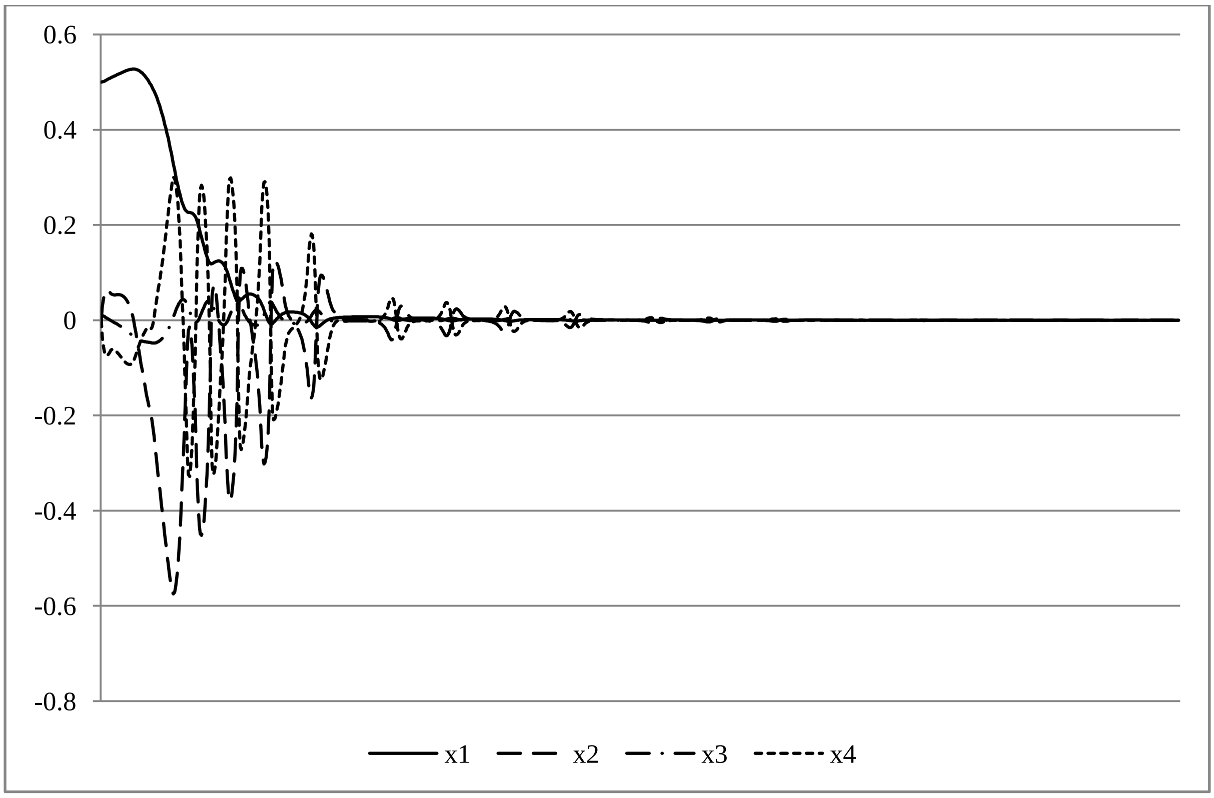

Figure 5.15 State variables convergence L2-13-24.

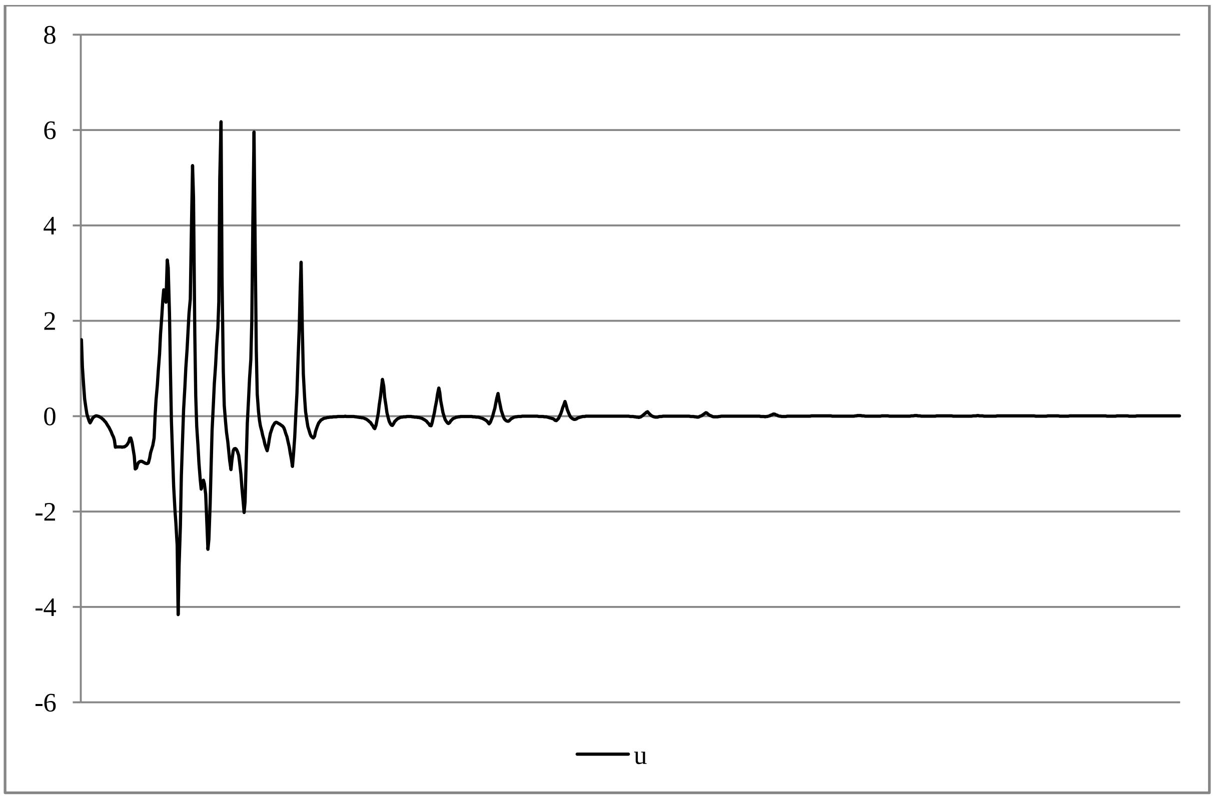

Figure 5.16 Controller L2-13-24.

5.8.3 Three layered HFS results

For the 3-layered HFS, the evolutionary algorithm evolves within 150—200 generations and the controller achieves convergence of the state variables to the target region and maintains them within the specified bounds for the remainder of the prescribed time T_f. Different topologies vary significantly in the number of generations that the EA takes to find the controller capable to perform such a task. For *L3-34-1-2*, it takes the EA about 140—160 generations, for *L3-12-3-4* it is 150—160, but for *L3-13-2-4* it usually takes about 200 generations. On average, in case of the 3-layered HFS the evolutionary algorithm takes longer to evolve to a satisfactory controller than in case of the 2-layered HFS.

Controller defined by topology *L3-34-1-2* achieves all state variables smooth convergence to the origin without any large oscillations. The controller performance is smooth after initial peak of −2.396 at t = 1.06, see Figure 5.20. The magnitude of control is reasonably low, within range of [−2.4, 1.4]. It is one of the best controllers among the 3-layered HFS. However, in ten simulations the controllers *L3-34-2-1* generally outperformed the controllers *L3-34-1-2* and therefore should be considered as the best controller amongst the 3-layered HFS.

Comparing convergence of the state variables and control values of *L2-34-12* and *L3-34-1-2* it can be seen that 3-layered HFS provides smoother convergence and control. The best result is shown in Figure 5.19. Note, that the state variables convergence to the TR is faster for *L2-34-12* than for *L3-34-1-2*.

As can be seen from comparison of *L3-13-2-4* and *L3-13-4-2* controllers, a seemingly insignificant change in input configuration in layer 2 and layer 3 results in a significant change in the controller performance, see Figure 5.21 and Figure 5.23 respectively. A similar effect can be seen by comparing controller performance for topologies *L3-14-2-3* and *L3-14-3-2*.

Controller with topology *L3-23-4-1* exhibits satisfactory convergence but with a long period of settling time, see Figure 5.25. The magnitude of the controller is in the range of [−4.83, 9.5], and thus relatively large compared to the other controllers. Controller with topology *L3-14-3-2* provides very smooth good convergence of the state variables but with some oscillations in x_4, which is barely visible in the diagram, see Figure 5.31, but is manifested more visibly in numerical values of the state variables convergence.

Surprisingly, topology *L2-14-23* that provided good stabilisation for the 2-layered HFS in most of the ten simulations does not have a match in *L3-14-2-3* for the 3-layered HFS. A typical result from ten simulations is shown in Figure 5.29 and Figure 5.30. On the other hand, the controller with topology *L3-14-3-2* performs much better, being one of the better performing controllers, see Figure 5.31 and Figure 5.32. This illustrates how important the selection of topology of the HFS is for controller performance.

The worst results are shown in Figure 5.21 and Figure 5.22; and also Figure 5.27 and Figure 5.28. The controllers with topologies *L3-13-2-4, L3-23-1-4, L3-23-4-1* and *L3-24-1-3* do not produce satisfactory state variables convergence and control characteristics.

Convergence times to the target region are grouped in Table 5.2. Empty space for *L3-23-1-4* means no convergence in the prescribed time for the controller in the simulation no 10 (some of the *L3-23-1-4* controllers achieve convergence to the TR). The numeric values for convergence times can be deceptive as the character of convergence must be taken into account and in some cases short convergence time does not necessarily translate into 'good convergence'. Another candidate topology *L3-23-4-1,* see Figure 5.25 and Figure 5.26, requires much more power expenditure and its characteristic is not as smooth as the controllers with topologies *L3-34-1-2* or *L3-34-2-1*.

Three-layered topology breaks strong interdependence between state variables in layers 2 and 3. This does not have adverse effect on the controller performance for the 'best' topologies *L3-34-1-2* and *L3-34-2-1* because decomposition reflects physical properties of the system. However, for *L3-12-3-4* or *L3-12-4-3* it has slightly detrimental effect because the HFS decomposition breaks state variables interdependence. In some other cases, for example *L3-14-2-3* or *L3-14-3-2*, it has a profound effect. As mentioned earlier, physical properties of the system under consideration require grouping of the input variables along weak state variables interdependence. In case of the inverted pendulum this grouping corresponds to two subsystems: the cart represented by x_1 and x_2, and the pole represented by x_3 and x_4. Swapping the input variables between the layers but preserving to some extent abovementioned groupings has little effect on the controller performance. When this grouping principle is broken, the results are often detrimental (depending which variables are more influential in the dynamical system). In case of *L3-14-2-3* and *L3-14-3-2* it seems that controlling the angle of the pole is more crucial than controlling the cart's velocity as it is reflected in both topologies.

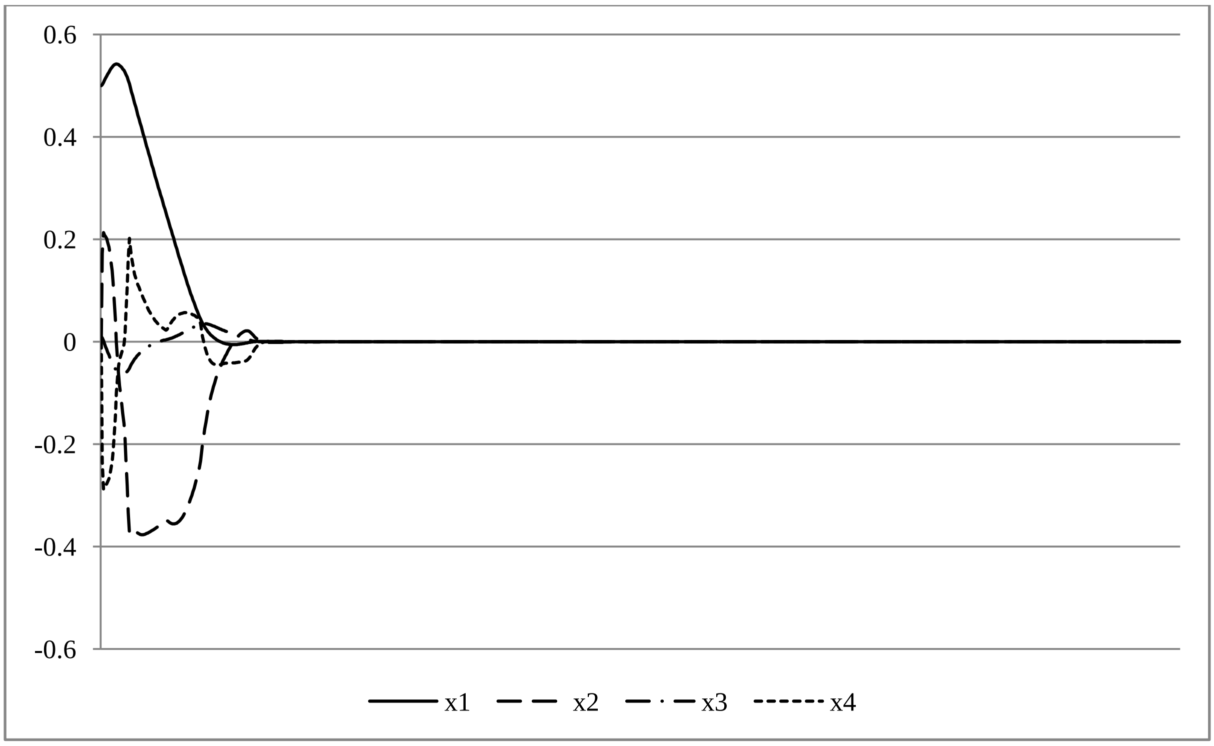

Figure 5.17 State variables convergence L3-34-2-1.

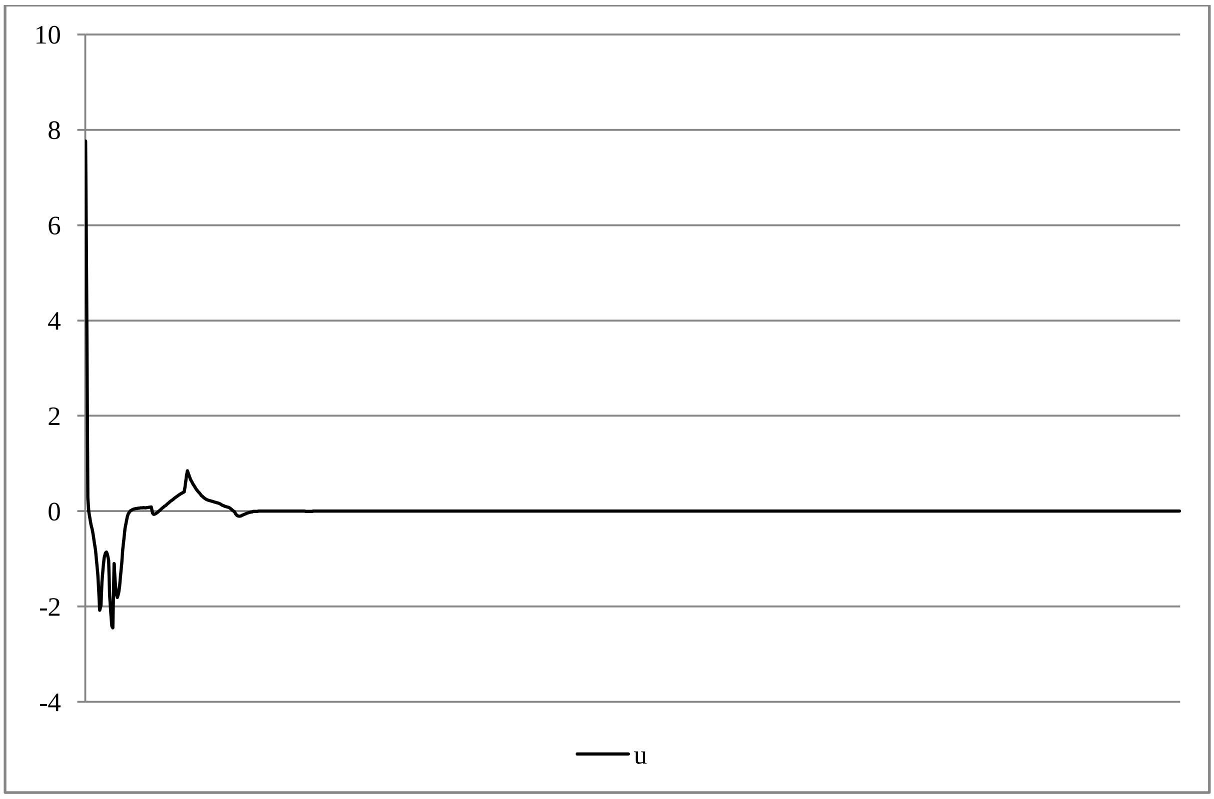

Figure 5.18 Controller L3-34-2-1.

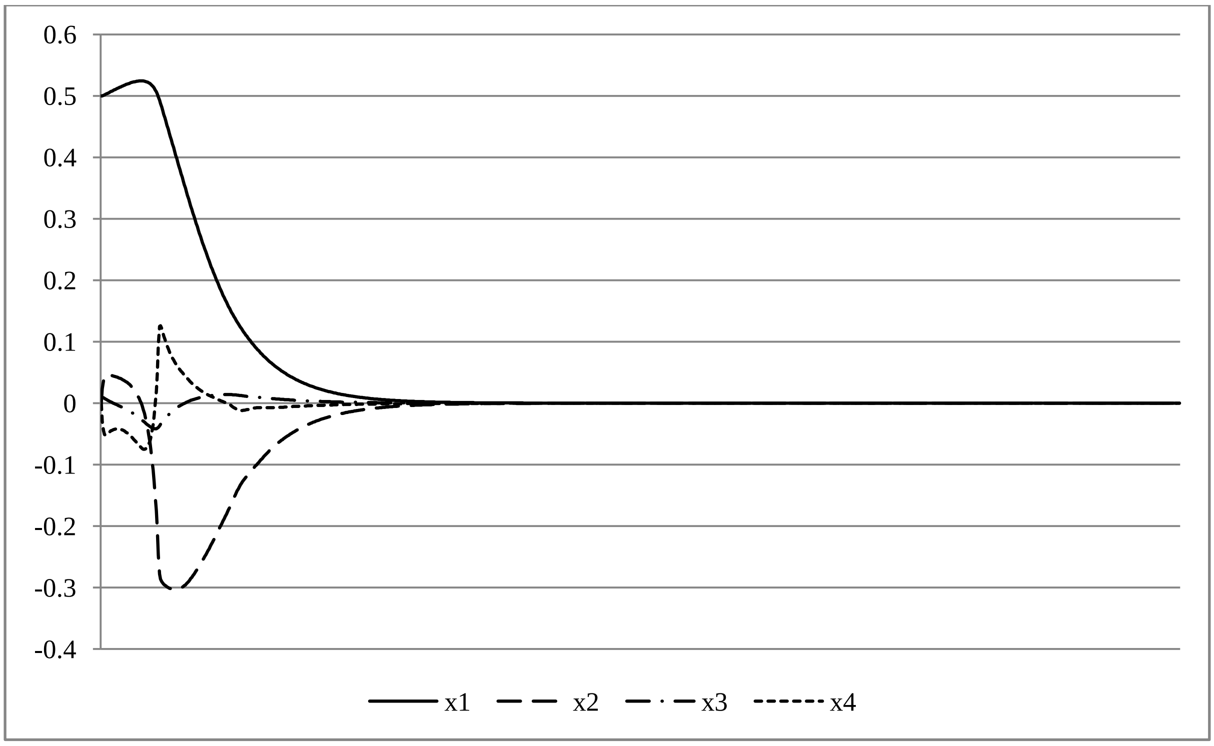

Figure 5.19 State variables convergence L3-34-1-2.

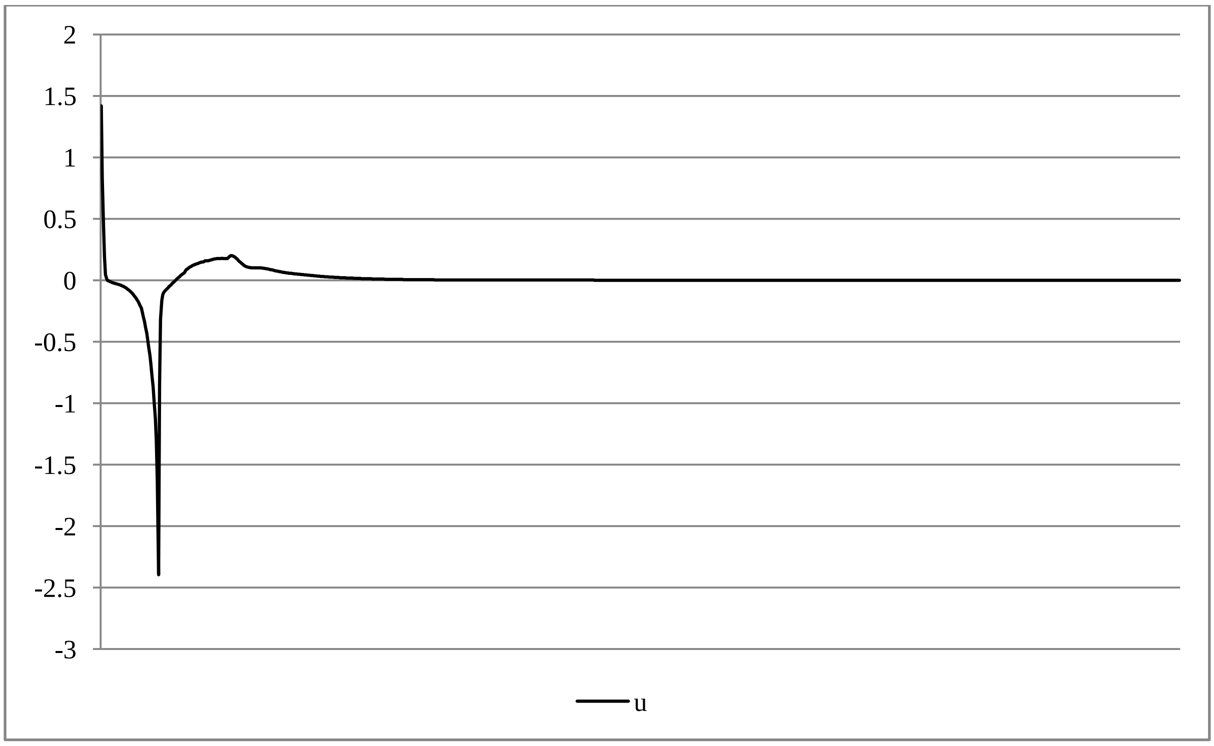

Figure 5.20 Controller L3-34-1-2.

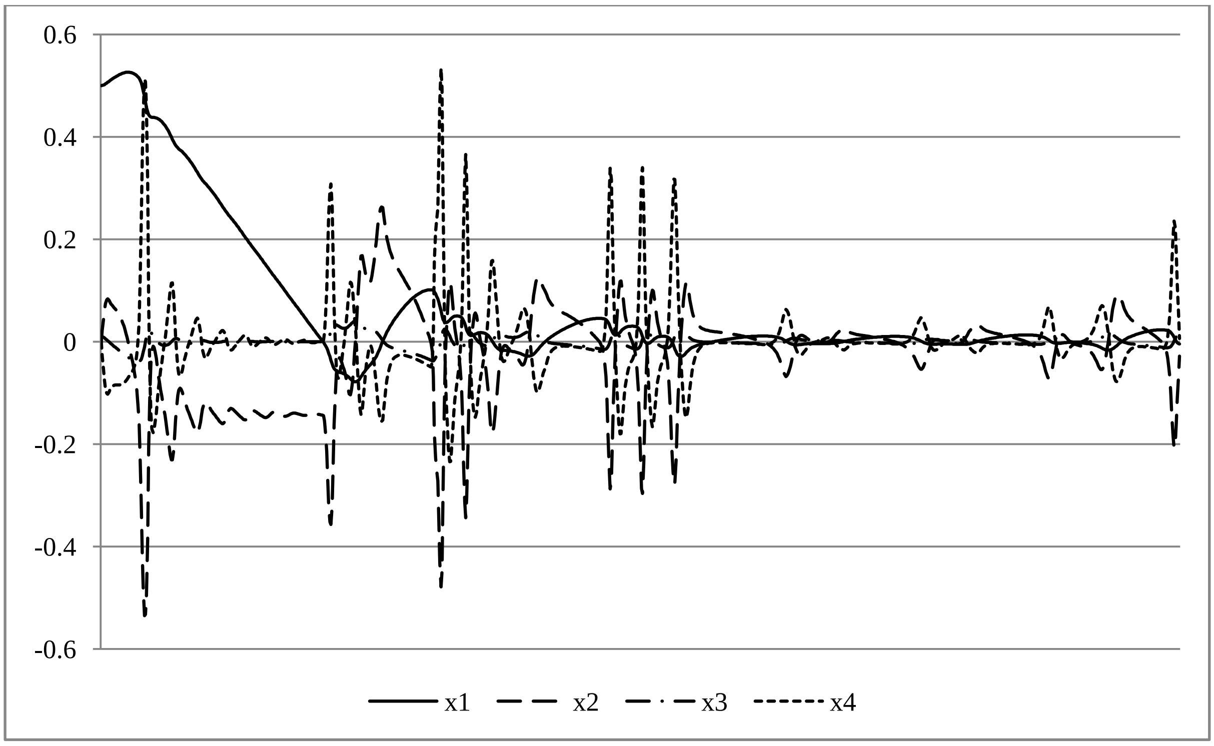

Figure 5.21 State variables convergence L3-13-2-4.

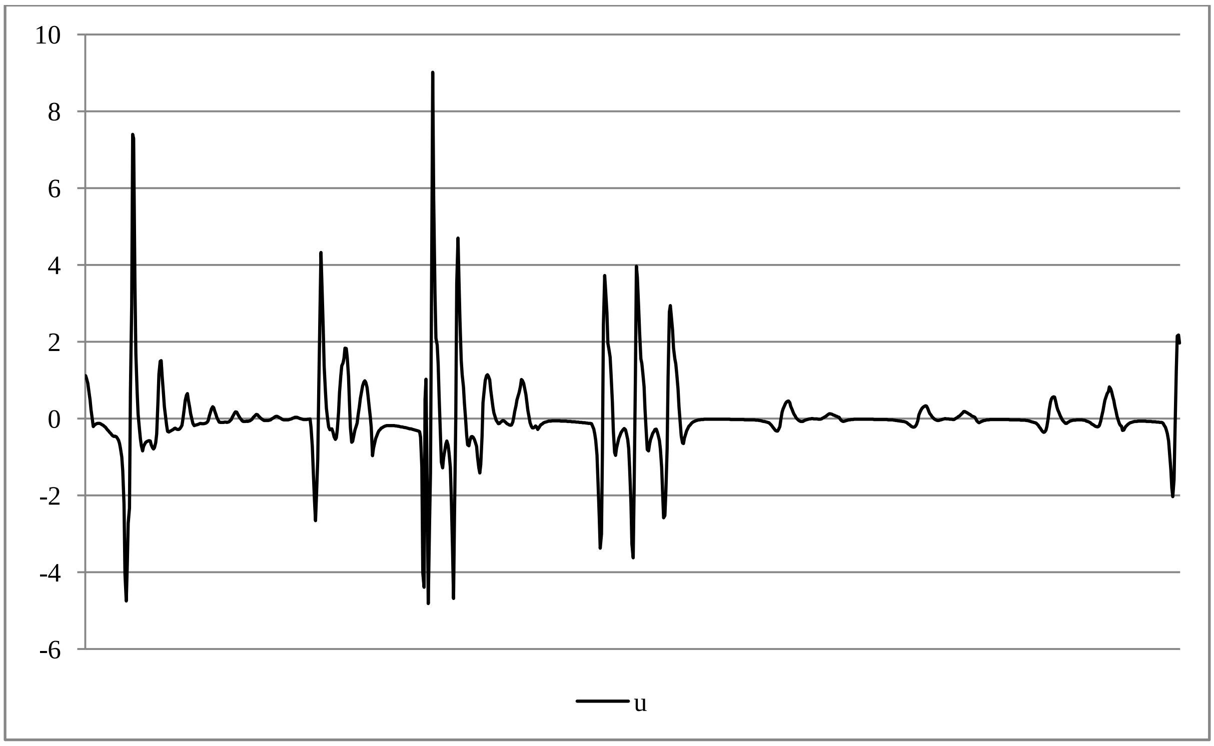

Figure 5.22 Controller L3-13-2-4.

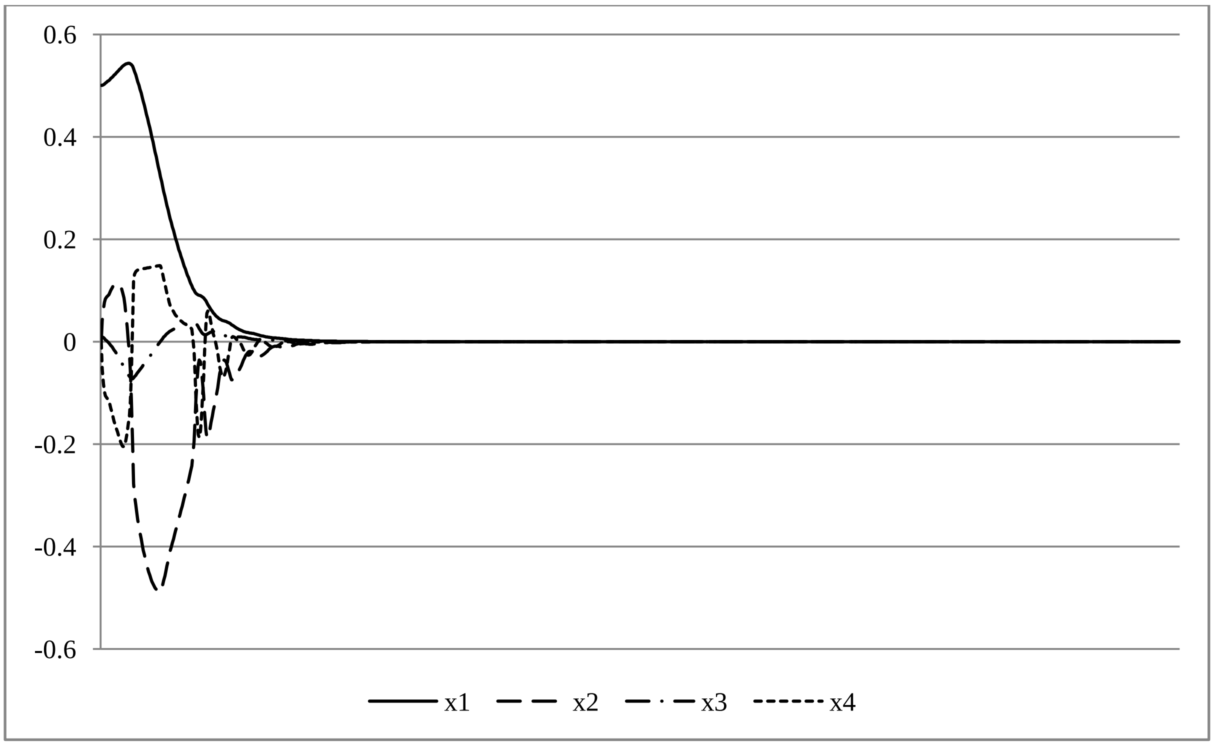

Figure 5.23 State variables convergence L3-13-4-2.

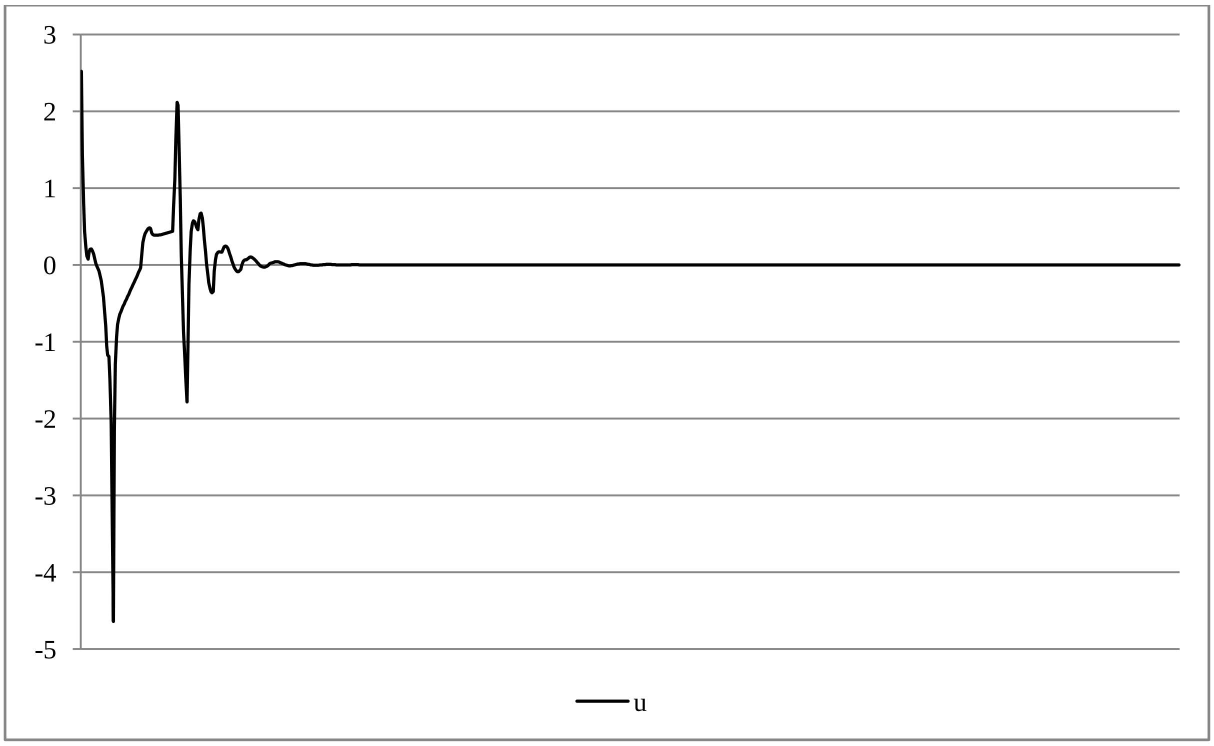

Figure 5.24 Controller L3-13-4-2.

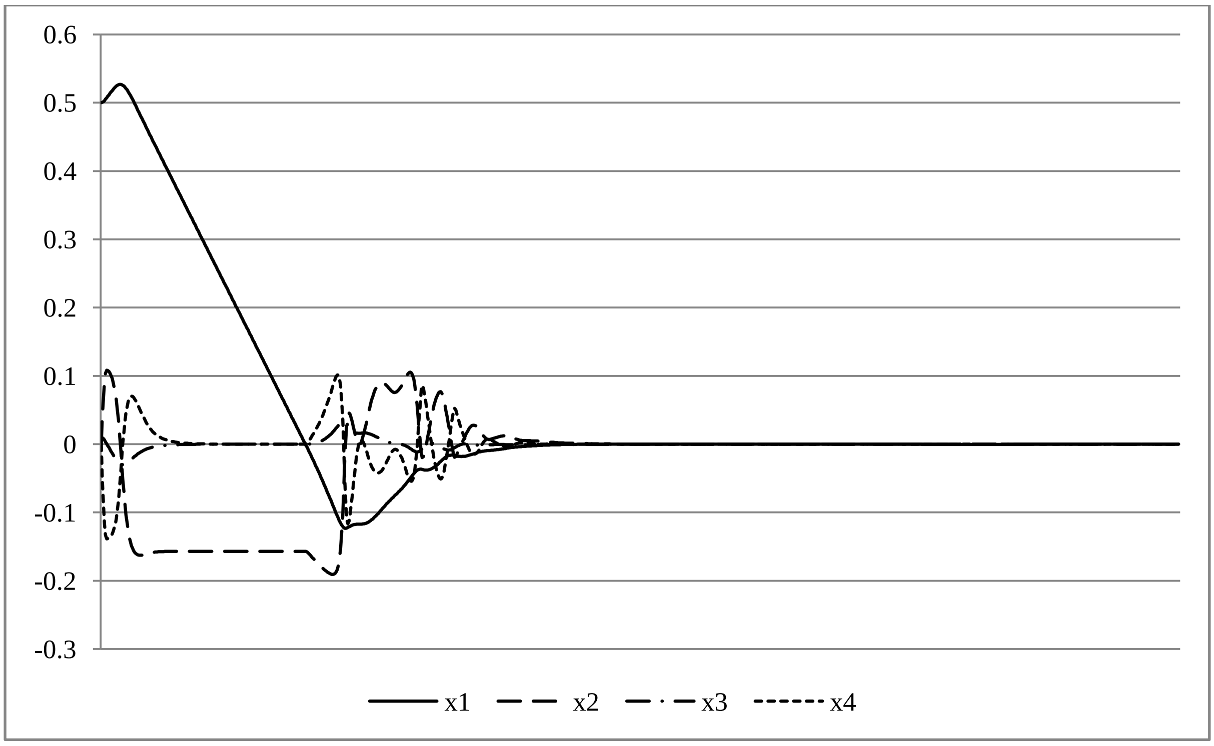

Figure 5.25 State variables convergence L3-23-4-1.

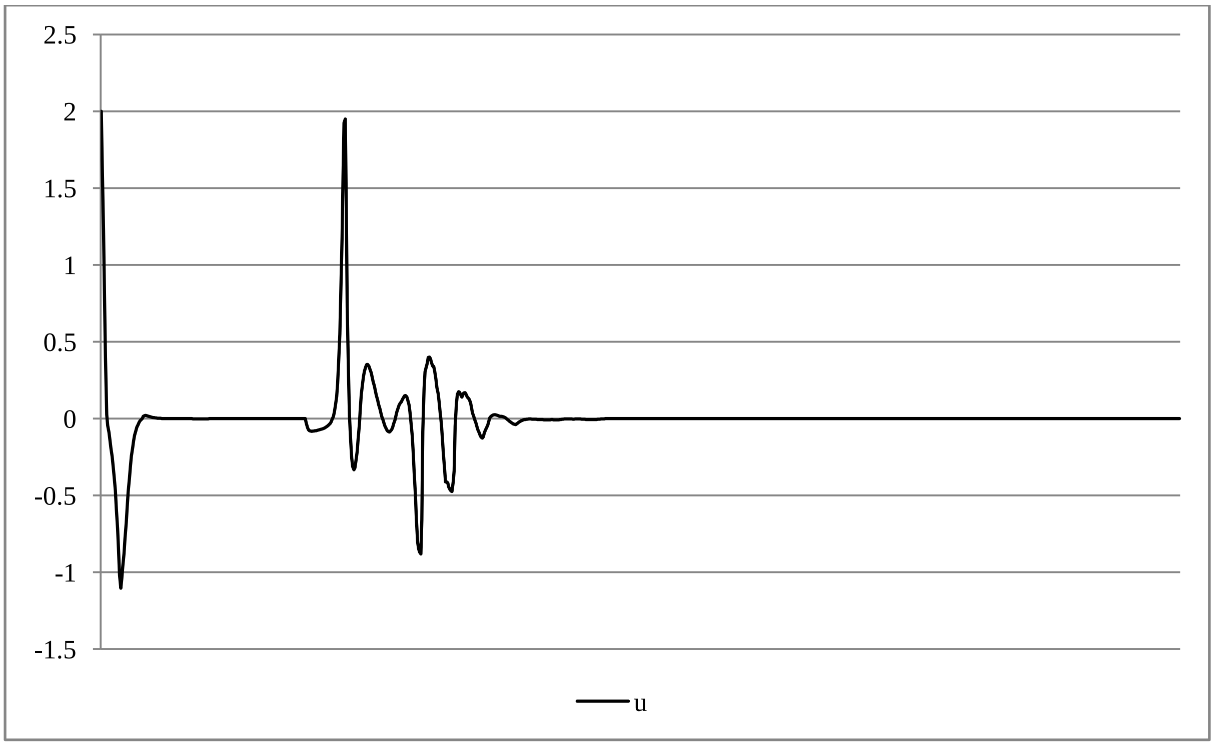

Figure 5.26 Controller L3-23-4-1.

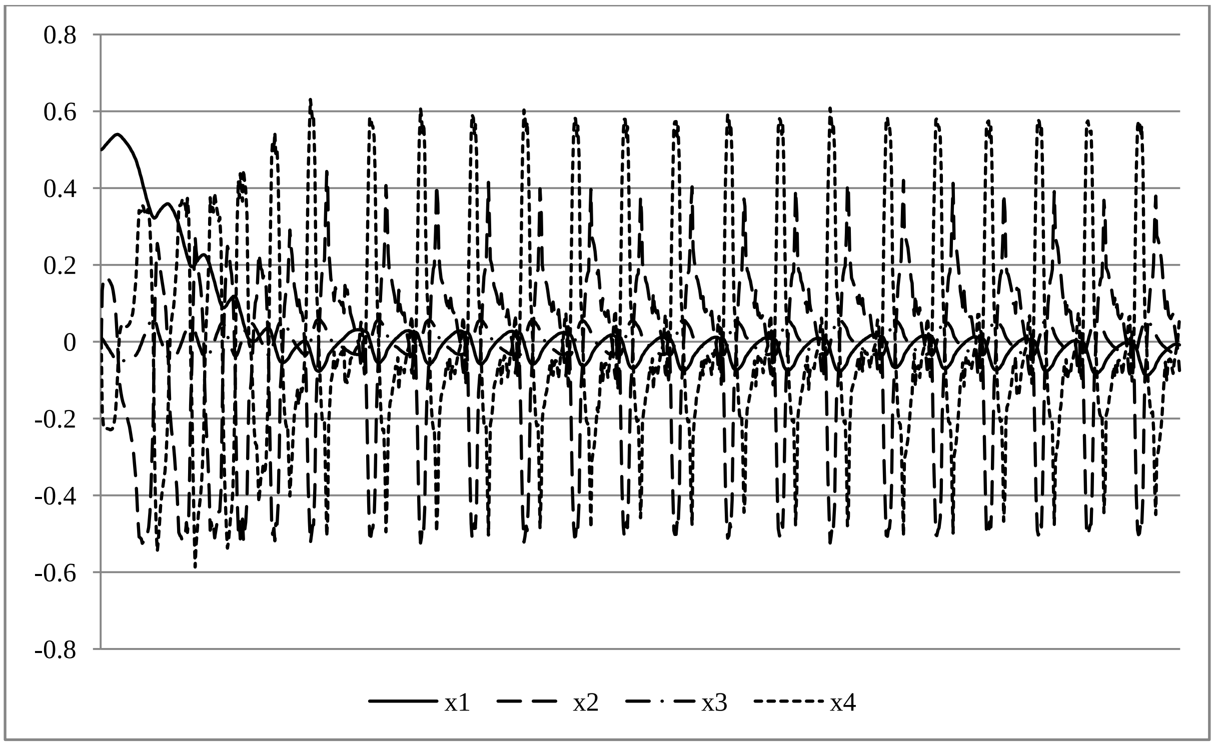

Figure 5.27 State variables convergence L3-23-1-4.

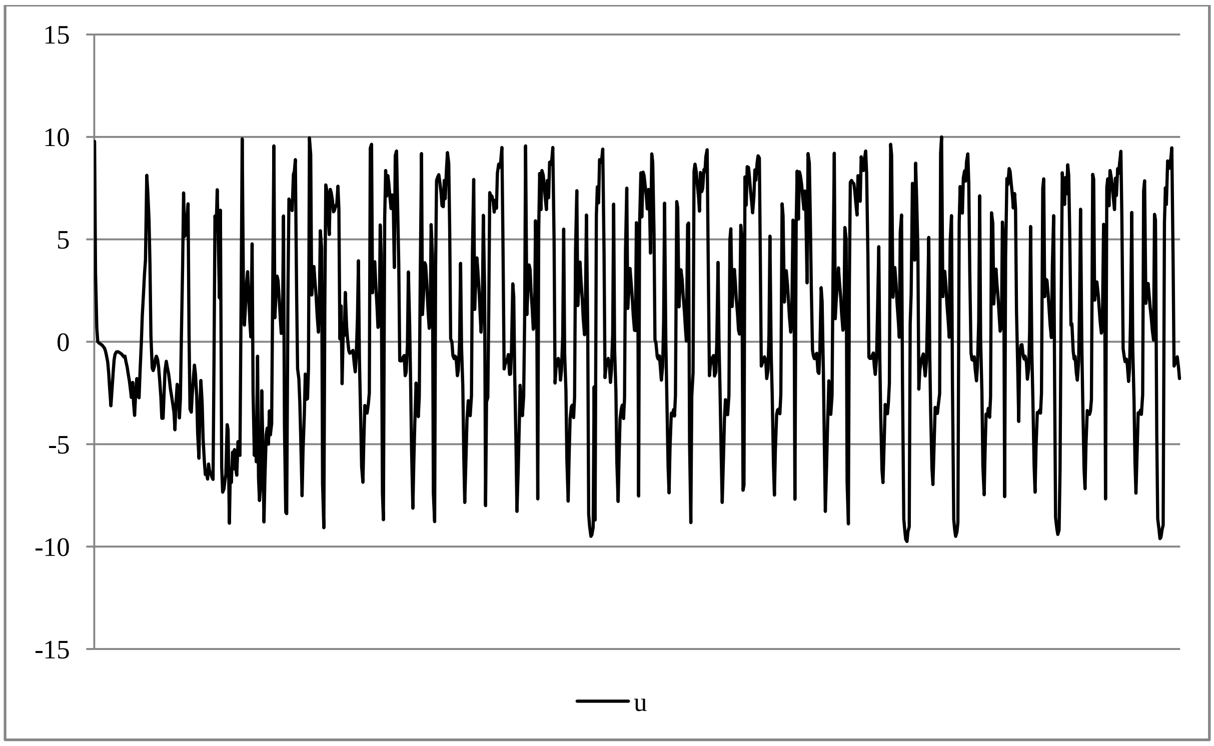

Figure 5.28 Controller L3-23-1-4.

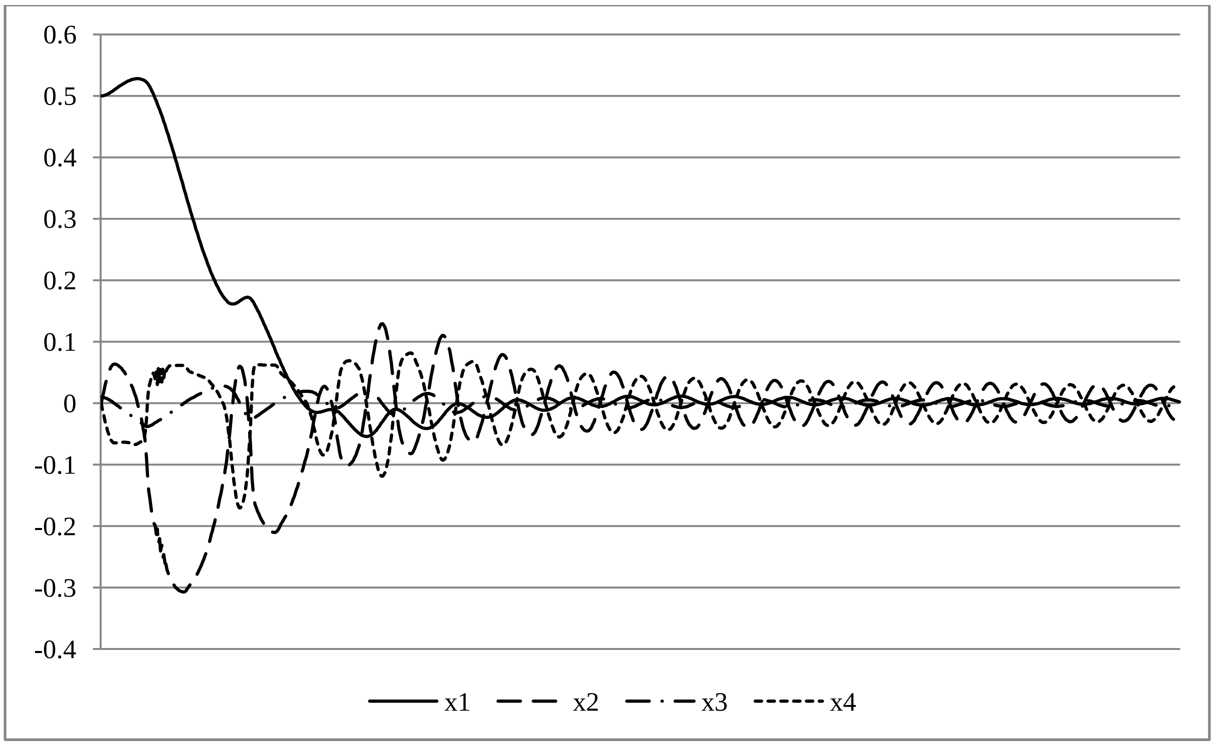

Figure 5.29 State variables convergence L3-14-2-3.

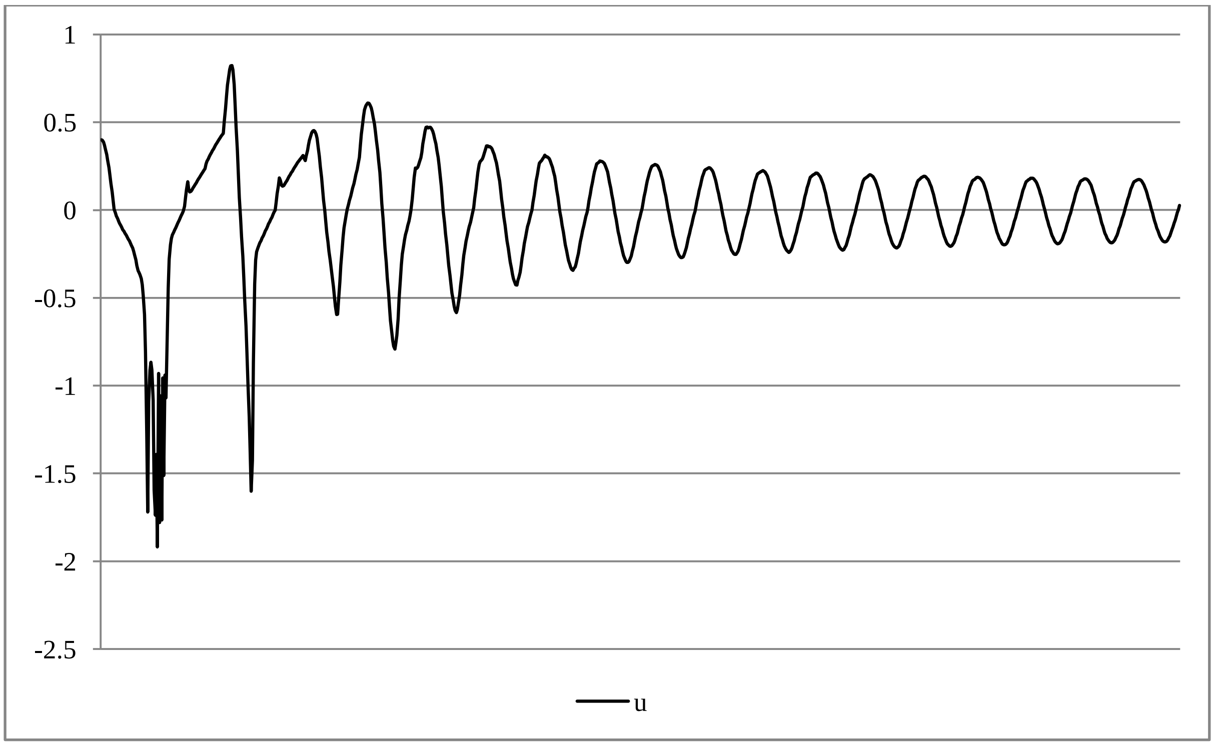

Figure 5.30 Controller L3-14-2-3.

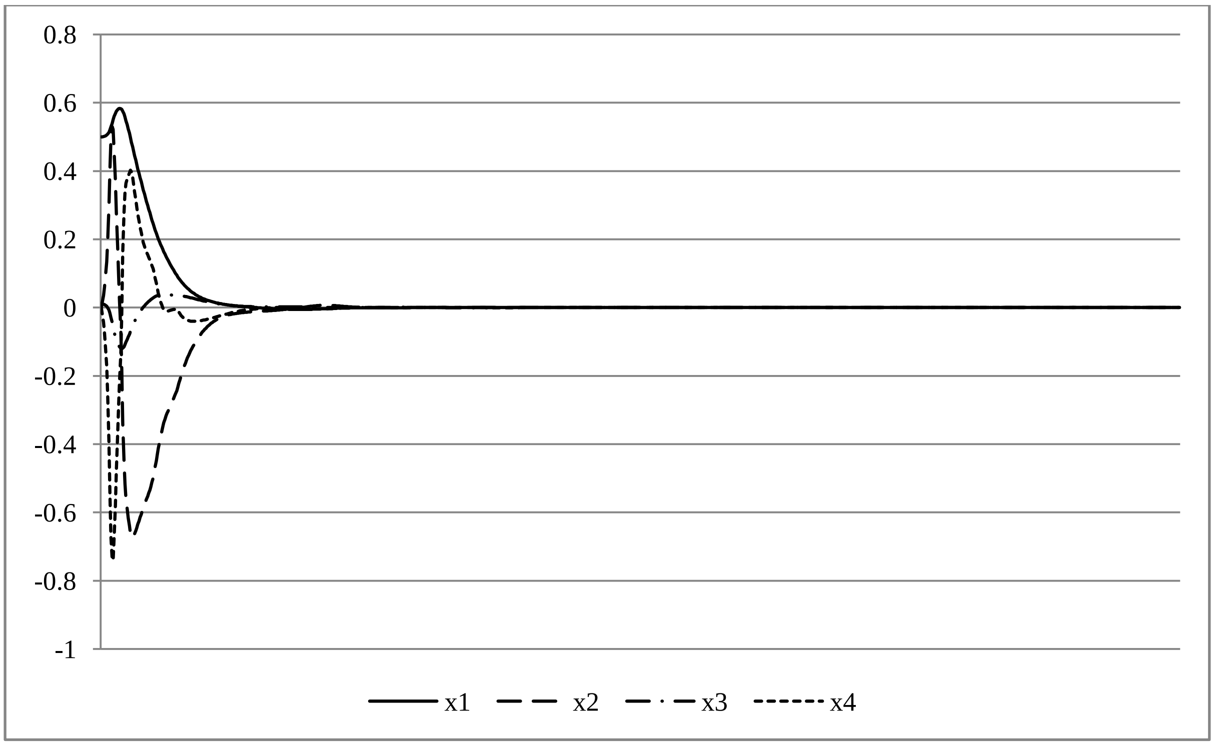

Figure 5.31 State variables convergence L3-14-3-2.

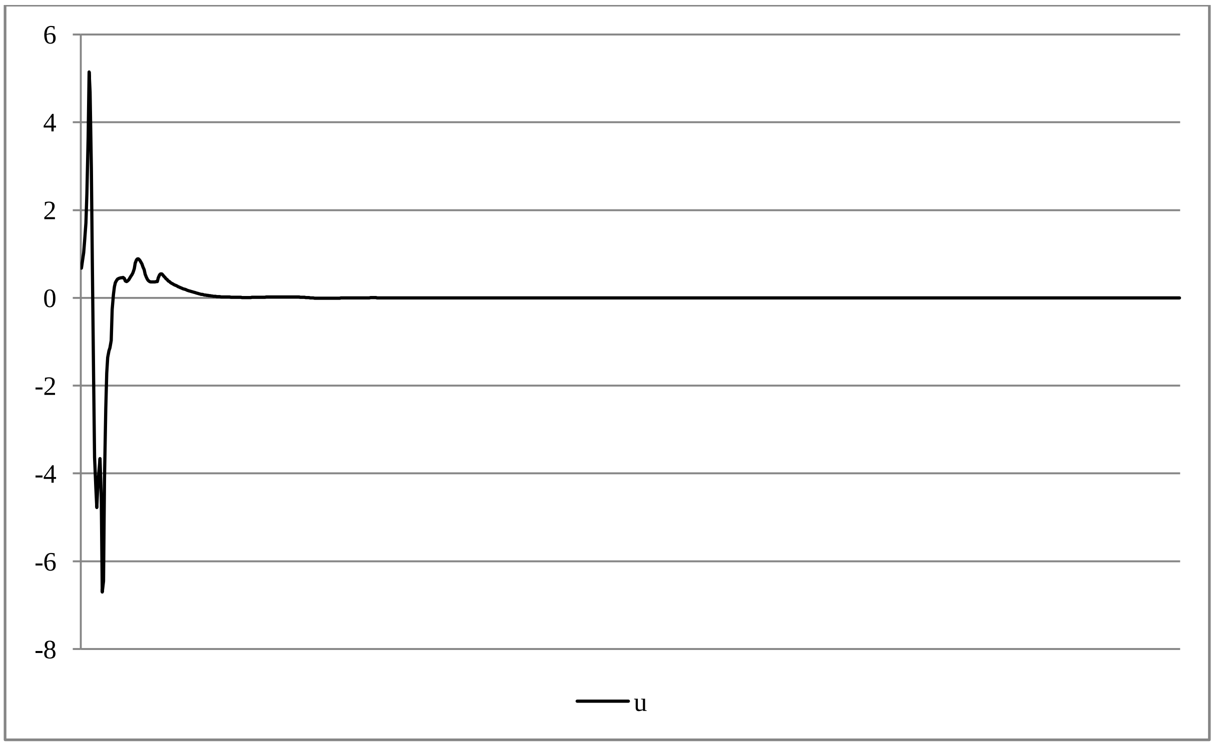

Figure 5.32 Controller L3-14-3-2.

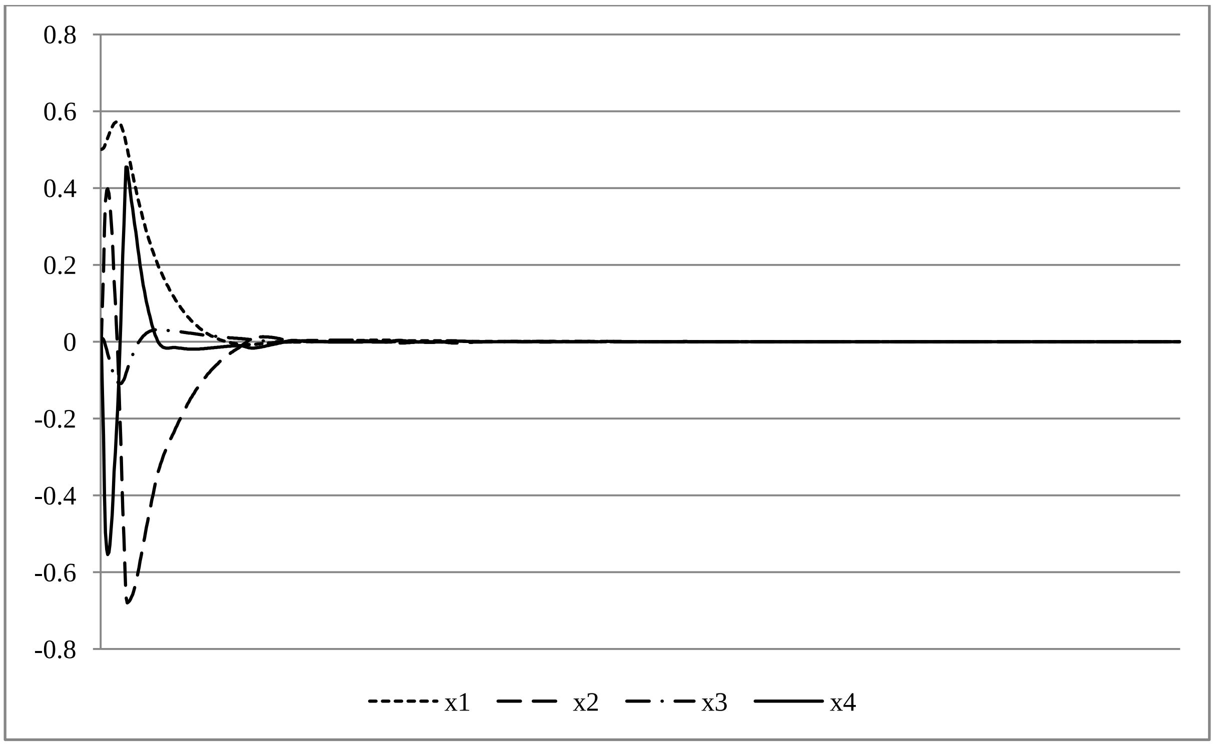

Figure 5.33 State variables convergence L3-12-3-4.

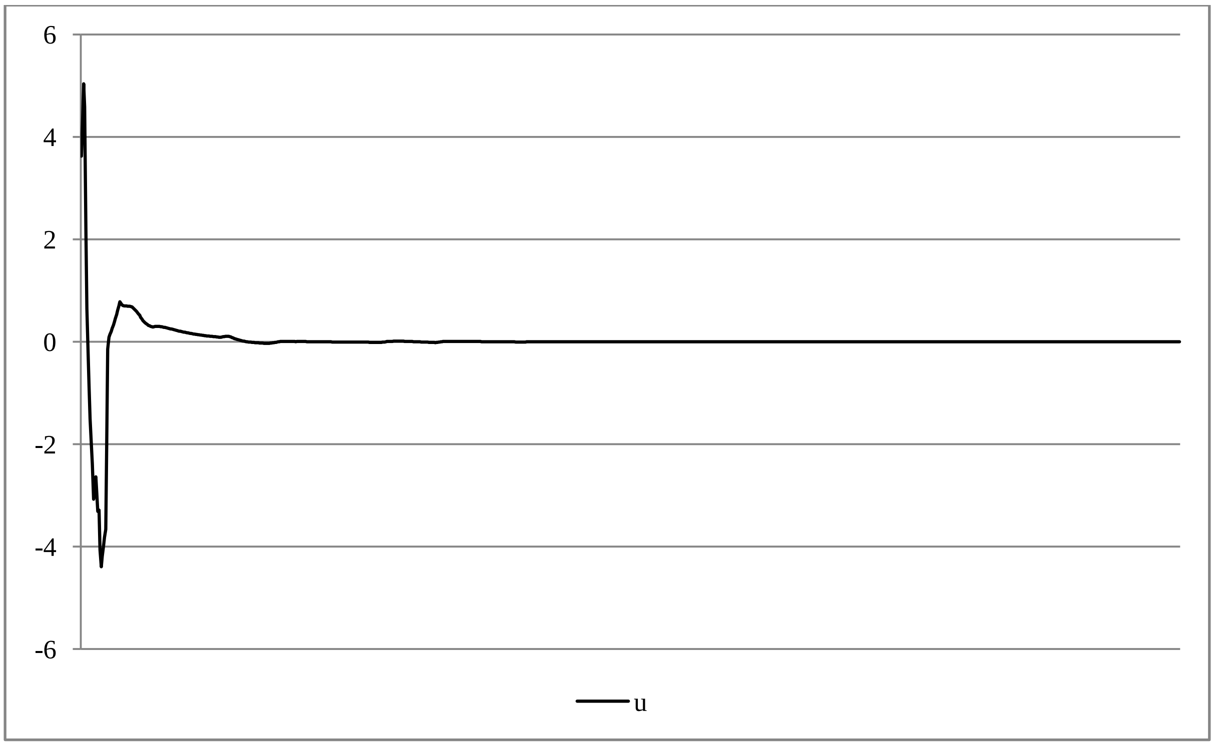

Figure 5.34 Controller L3-12-3-4.

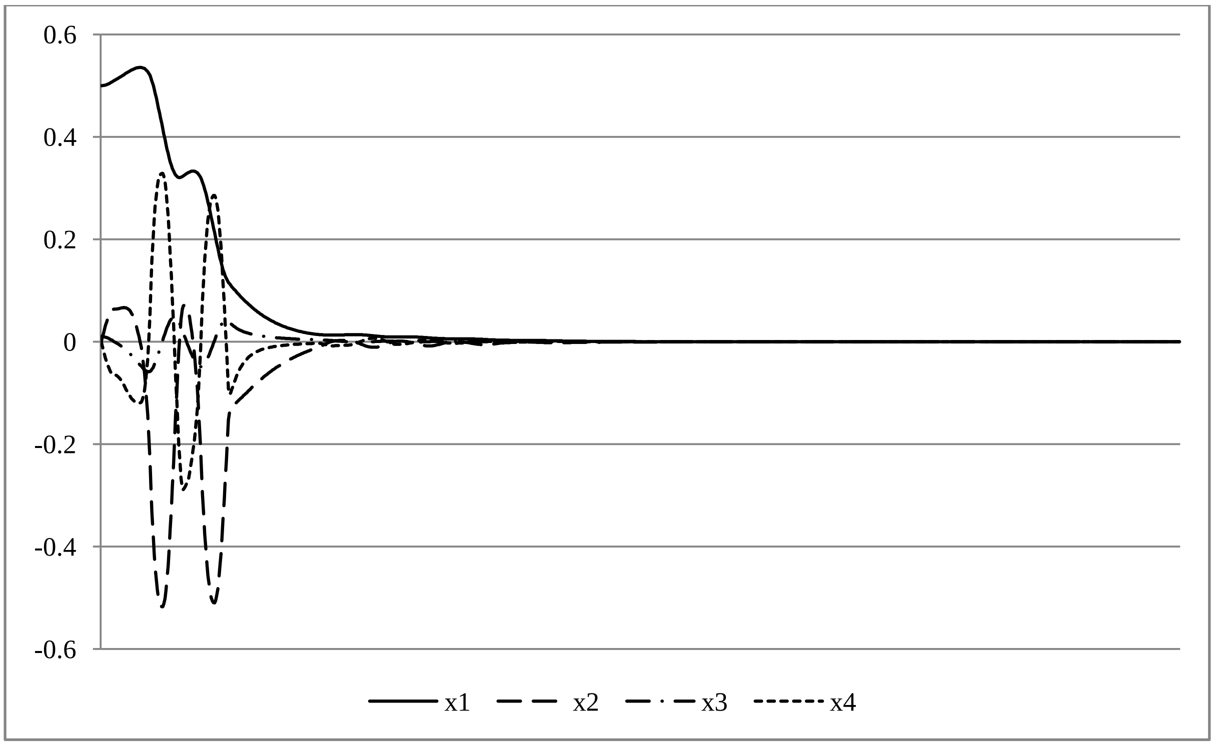

Figure 5.35 State variables convergence L3-12-4-3.

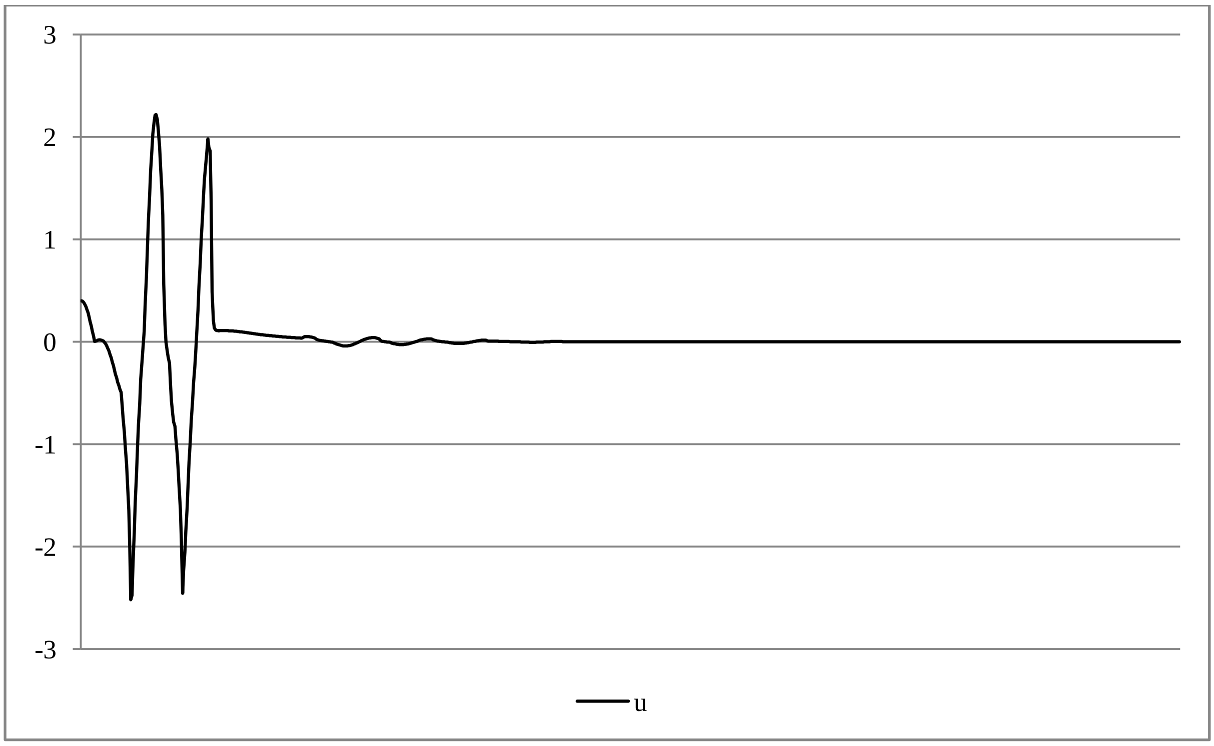

Figure 5.36 Controller L3-12-4-3.

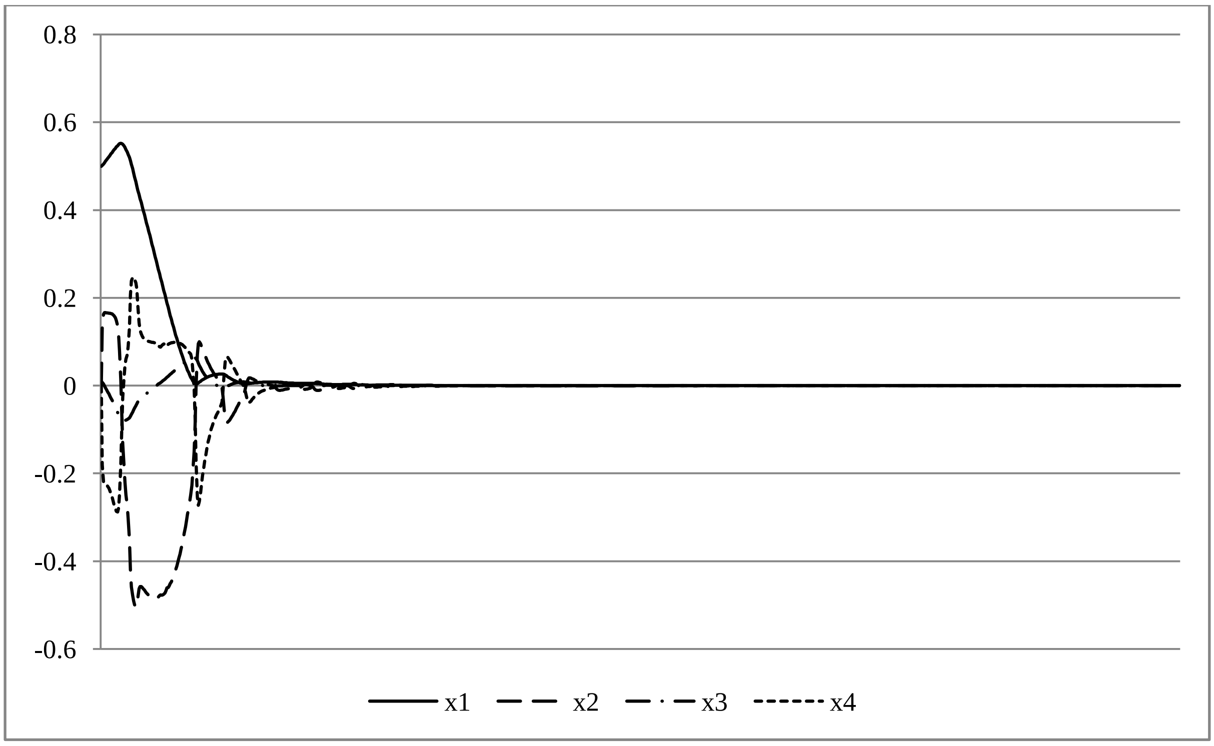

Figure 5.37 State variables convergence L3-24-3-1.

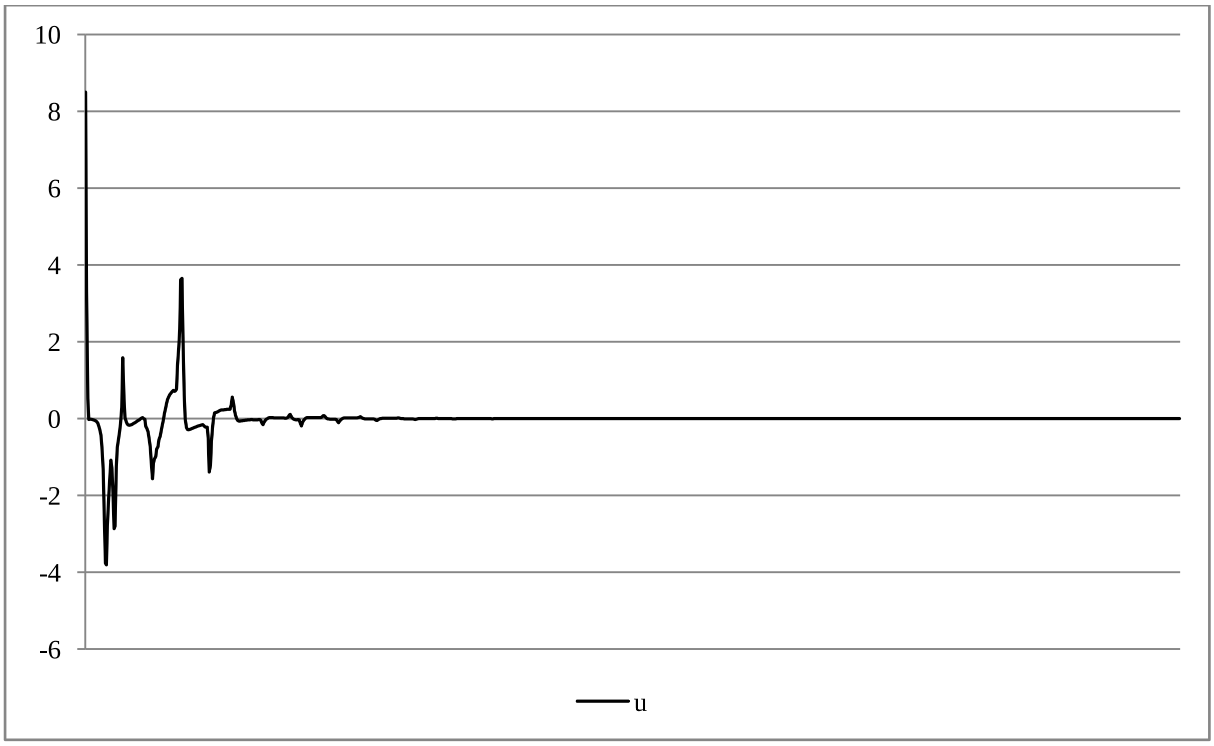

Figure 5.38 Controller L3-24-3-1.

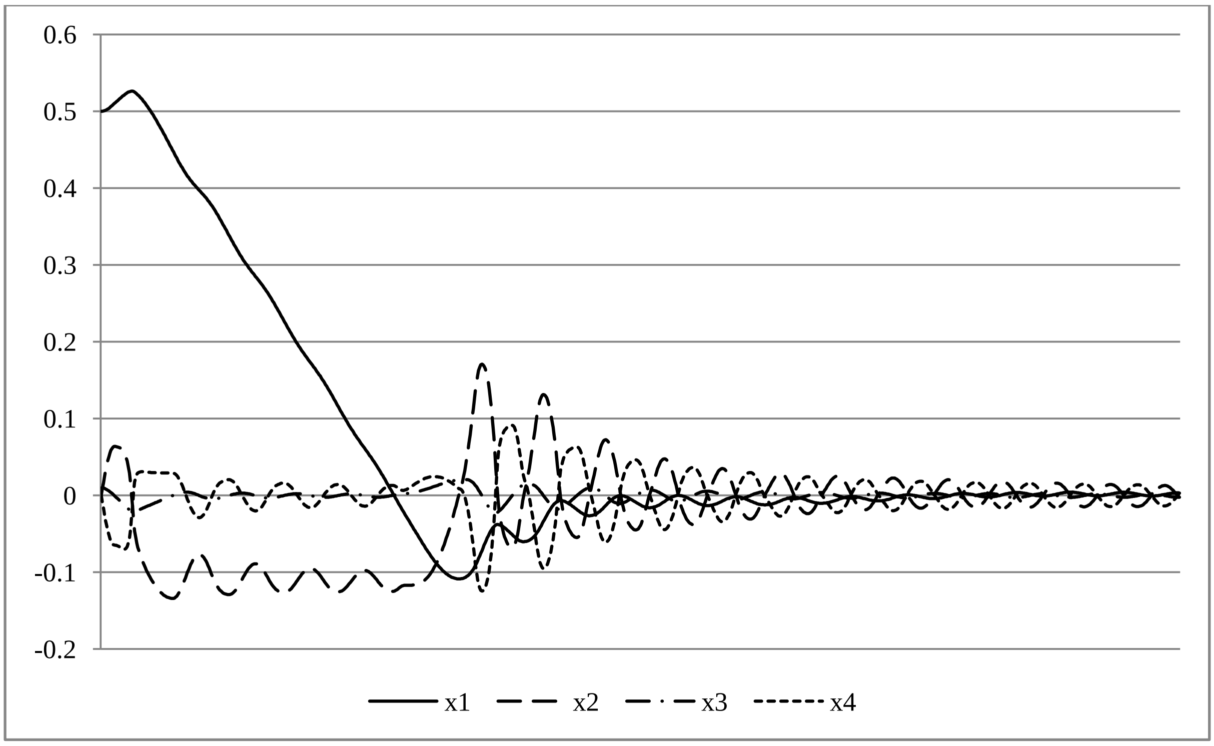

Figure 5.39 State variables convergence L3-24-1-3.

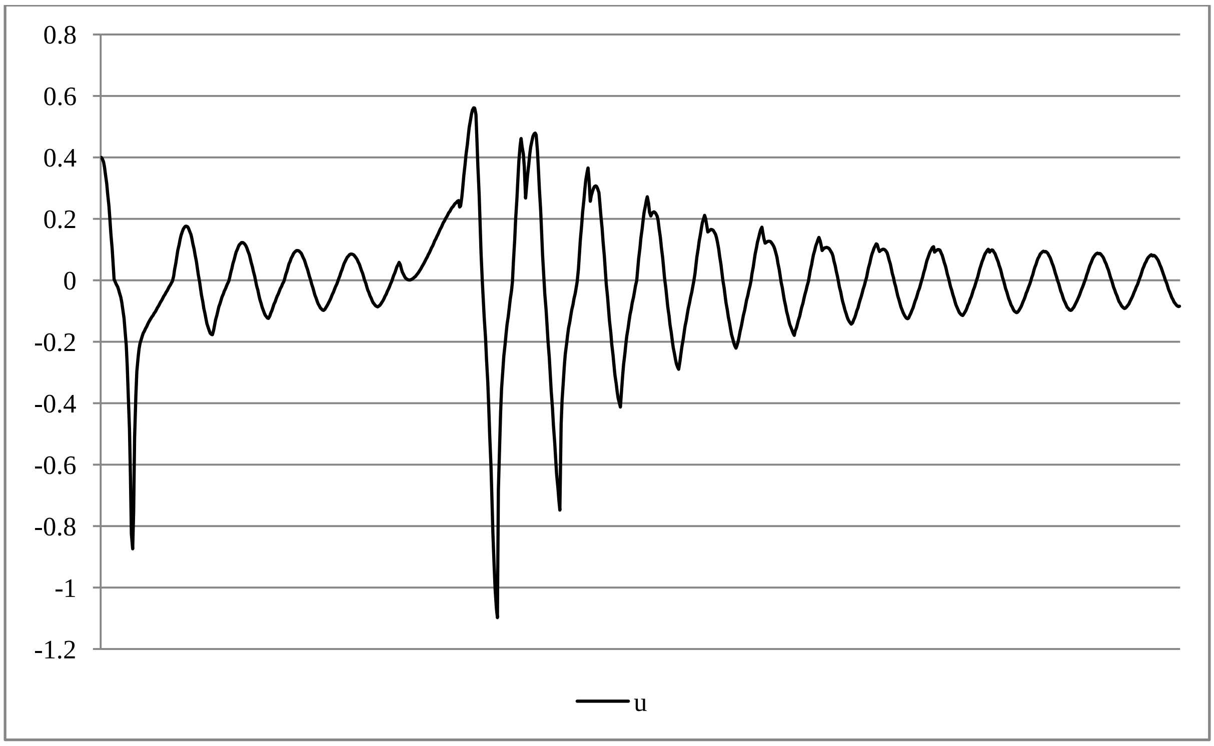

Figure 5.40 Controller L3-24-1-3.

Table 5.2 Stabilisation times for 3-layered HFS

Run No	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12
10	9.18	13.72	2.52	2.54	3.68	3.5		2.14	12.2	6.96	2.9	2.06

where C1 denotes: *L3-12-3-4*, C2: *L3-12-4-3*, C3: *L3-13-2-4*, C4: *L3-13-4-2*, C5: *L3-14-2-3*, C6: *L3-14-3-2*, C7: *L3-23-1-4*, C8: *L3-23-4-1*, C9: *L3-24-1-3*, C10: *L3-24-3-1*, C11: *L3-34-1-2*, C12: *L3-34-2-1*.

5.8.4 Four layered HFS results

Four variants of the 4-layered topology: *L4-3-4-1-2*, *L4-3-4-2-1*, *L4-4-3-1-2*, and *L4-4-3-2-1* are investigated, see Figure 5.41—Figure 5.48. The last two topologies produced good performance of the controllers. One of the controllers representing topology *L4-3-4-1-2* produced a very reasonable control with magnitude in the range [−2.0, 2.8], see Figure 5.48.

By examining the 4-layered topologies it can be found which input variables are most influential in the inverted pendulum system. It was found that topologies *L2-34-12, L3-34-2-1, L3-34-1-2* provide the best performing controllers. The simulation results show that the topology *L4-4-3-2-1* is the most consistent in producing well performing controllers for ten different initial populations with *L4-4-3-1-2* close behind, see Figure 5.41 and Figure 5.43 respectively. This clearly indicates that the most influential input variable is the angular speed of the pole x_4, second - the angle of the pole x_3, and then cart's speed x_2 and its position x_1.

After comparing the 4-layered topologies controller performance with previously analysed controllers it was found that the 4-layered HFS are outperformed by the HFS controllers with lower number of layers using criteria stated in Section 5.7. This indicates that for the inverted pendulum problem this ladder-like structure of the 4-layered HFS does not produce the best performing controllers.

Stabilisation times are relatively fast for the 4-layered HFS, see Table 5.3.

Table 5.3 Stabilisation times for 4-layered HFS.

L4-3-4-1-2	L4-3-4-2-1	L4-4-3-1-2	L4-4-3-2-1
3.22	1.68	2.64	1.82

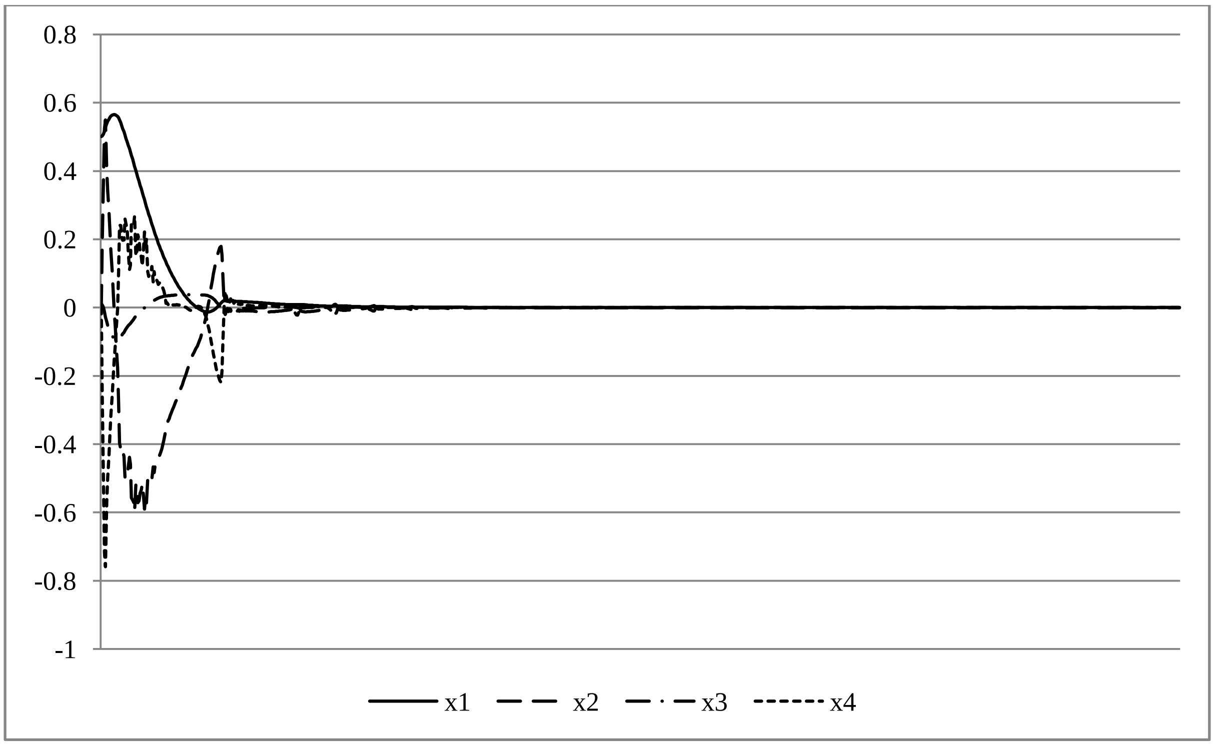

Figure 5.41 State variables convergence L4-4-3-2-1.

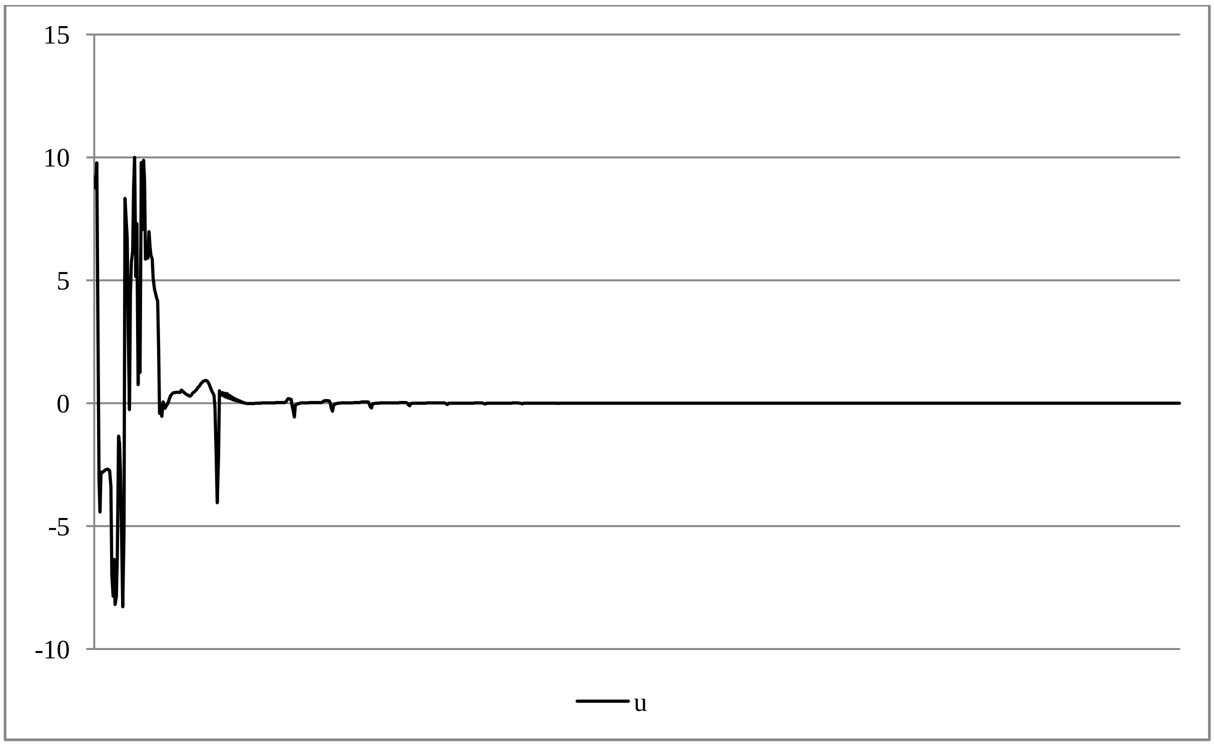

Figure 5.42 Controller L4-4-3-2-1.

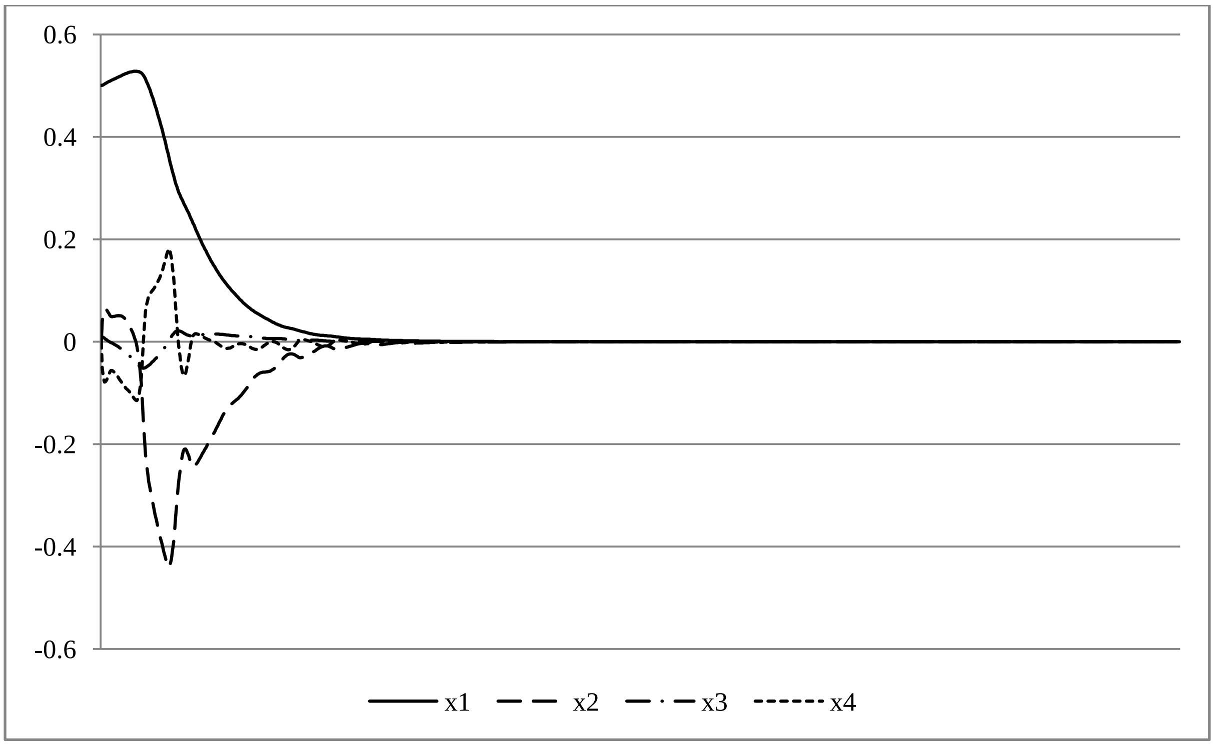

Figure 5.43 State variables convergence L4-4-3-1-2.

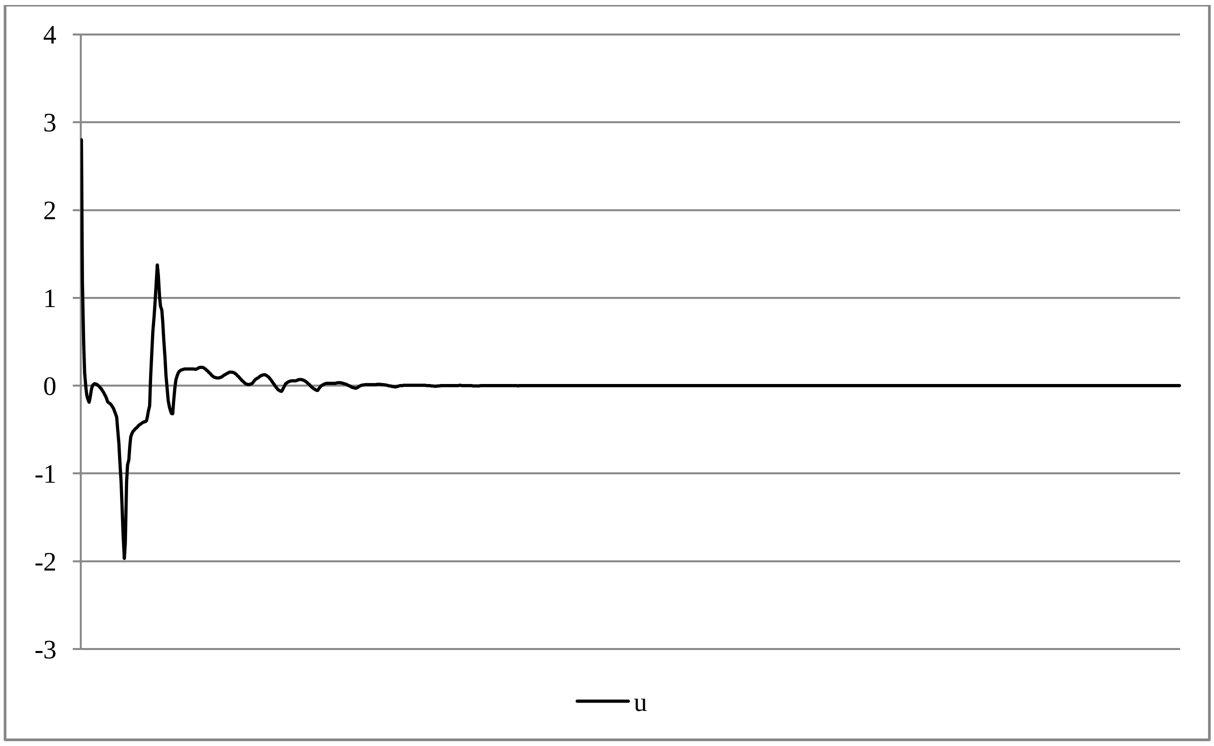

Figure 5.44 Controller L4-4-3-1-2.

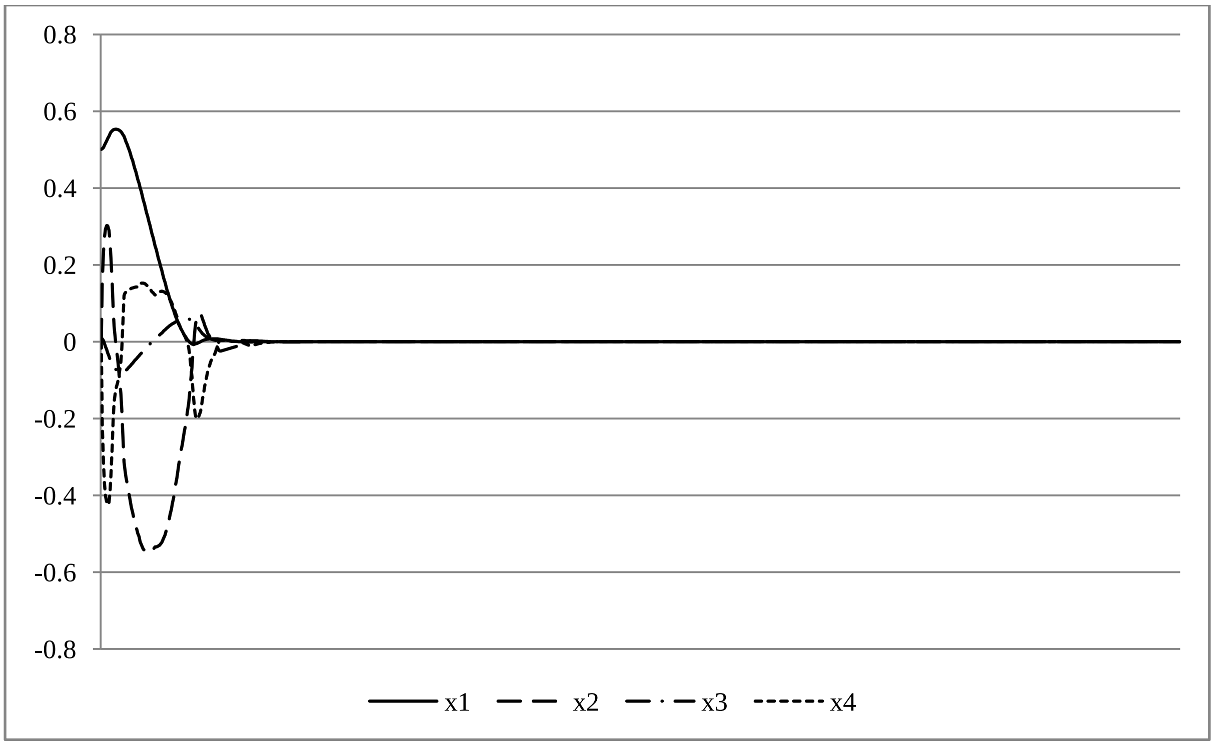

Figure 5.45 State variables convergence L4-3-4-2-1.

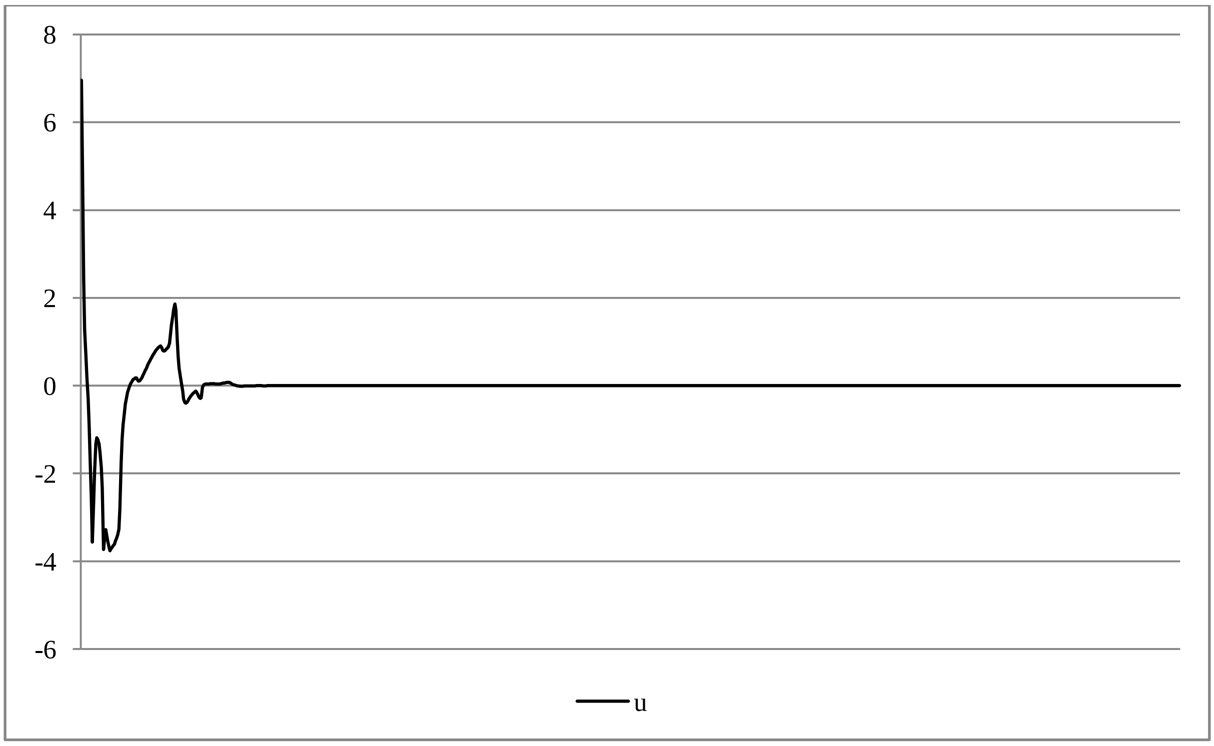

Figure 5.46 Controller L4-3-4-2-1.

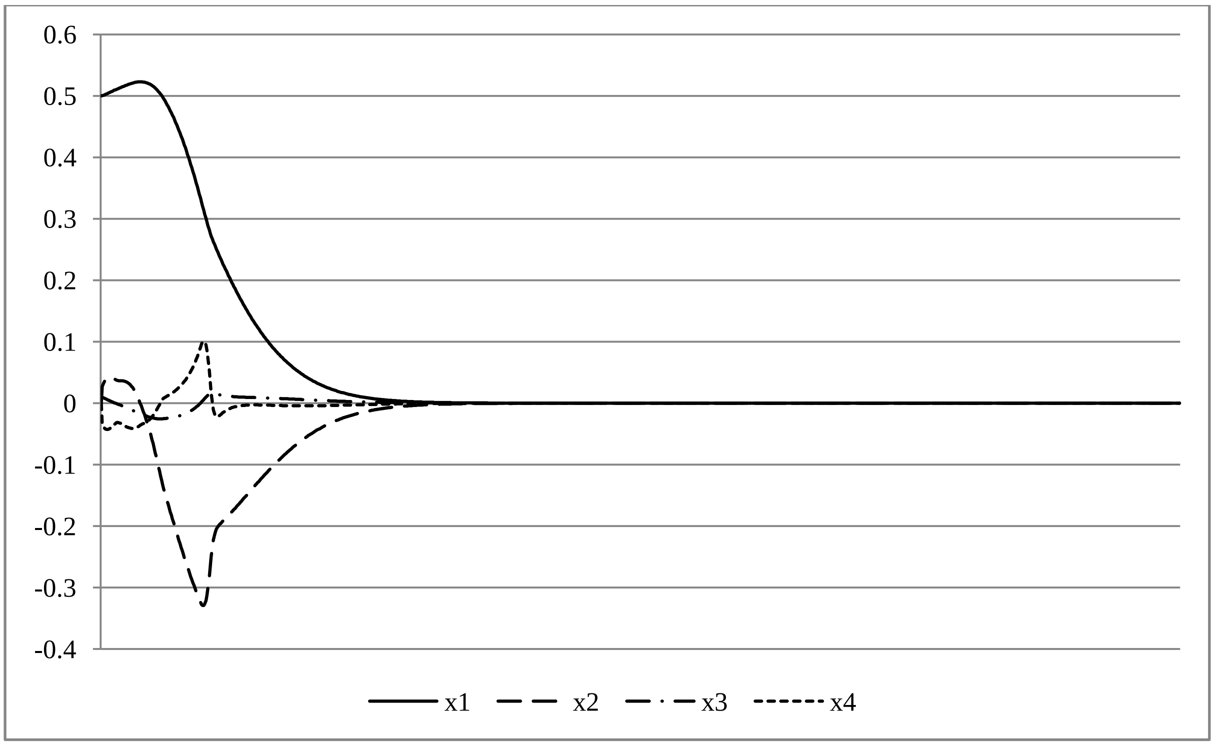

Figure 5.47 State variables convergence L4-3-4-1-2.

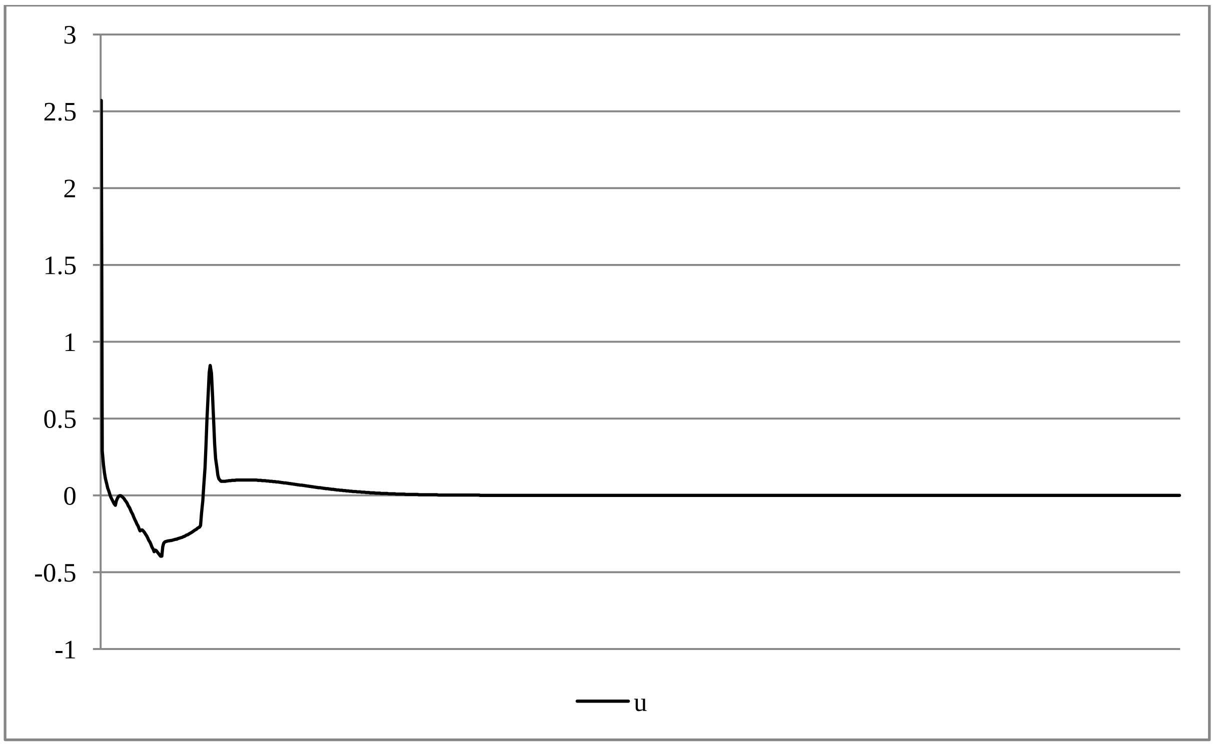

Figure 5.48 Controller L4-3-4-1-2.

5.8.5 Different topologies results

Simulations are conducted for selected 2 and 3-layered HFS with modifications to the number of inputs in the first layer, i.e., with more/less than two inputs in the first layer 1. The performance of controllers with such topologies is tested. The tested topologies are: *L2-3-412*, *L2-341-2, L3-3-41-2* and *L3-3-4-12*.

Results are shown in Figure 5.49—Figure 5.56. As can be seen from the figures, the controller performance is better than expected, especially for topologies *L3-3-41-2*, see Figure 5.53, and for *L3-3-4-12*, see Figure 5.55, as it provides smooth control with reasonable control magnitude and fast stabilisation of the system. In *L3-3-41-2* the variable x_1 (cart's position) does not converge to a near-zero value but stays in the target region, which is satisfactory.

Stabilisation times for the selected examples of different topologies are shown in Table 5.4. Surprisingly, they are faster than corresponding stabilisation times for *L3-34-1-2*.

Considering the performance and speed of these controllers they cannot be ruled out just because of the high number of fuzzy rules in their knowledge bases. The large number of rules does not hamper performance. They also prove that the HFS topology is crucial in achieving good control performance.

Table 5.4 Stabilisation times for 2 and 3-layered HFS: different topologies

L2-341-2	L2-3-412	L3-3-4-12	L3-3-41-2
4.58	2.4	1.4	1.74

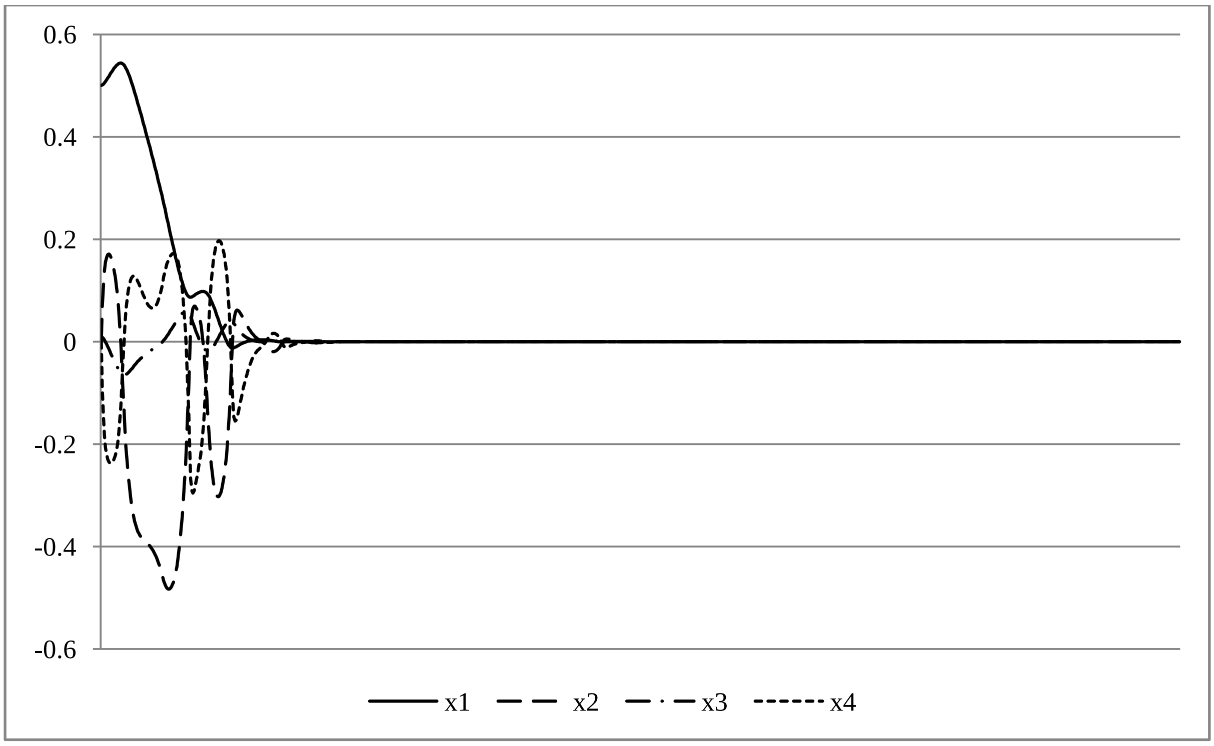

Figure 5.49 State variables convergence L2-3-412.

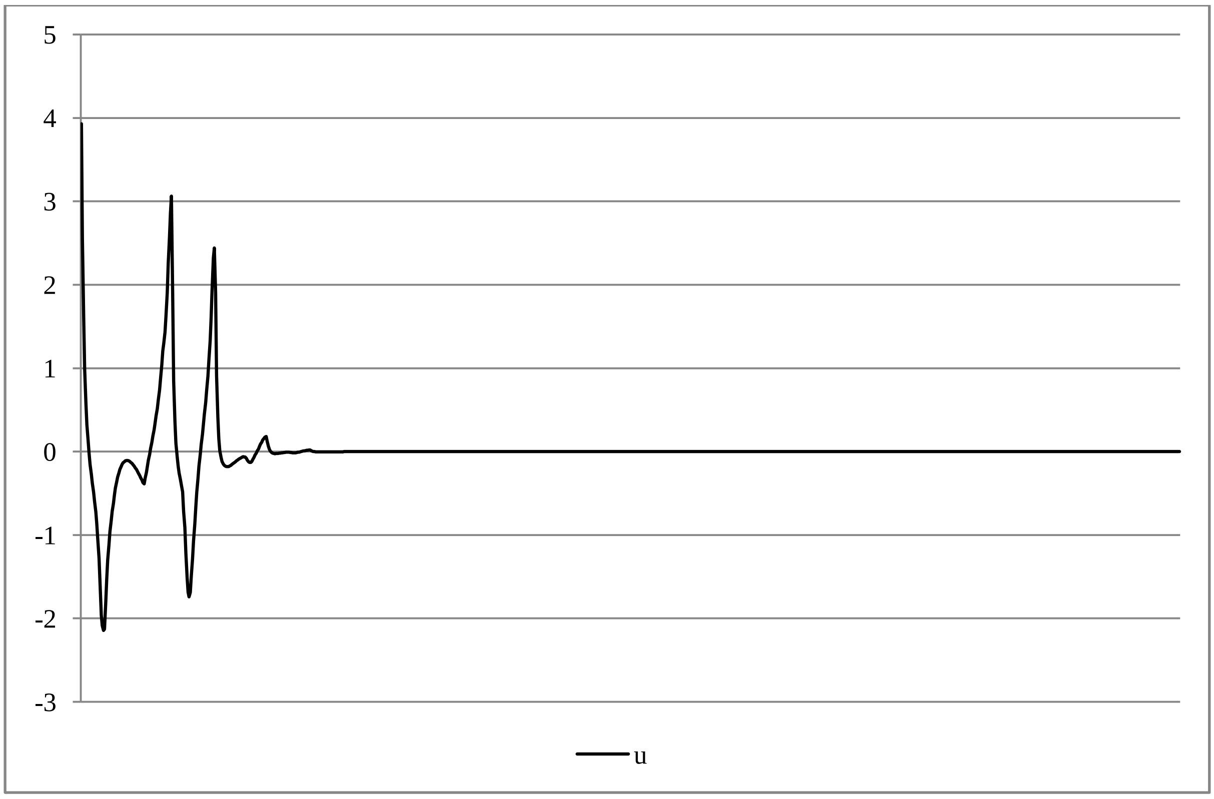

Figure 5.50 Controller L2-3-412.

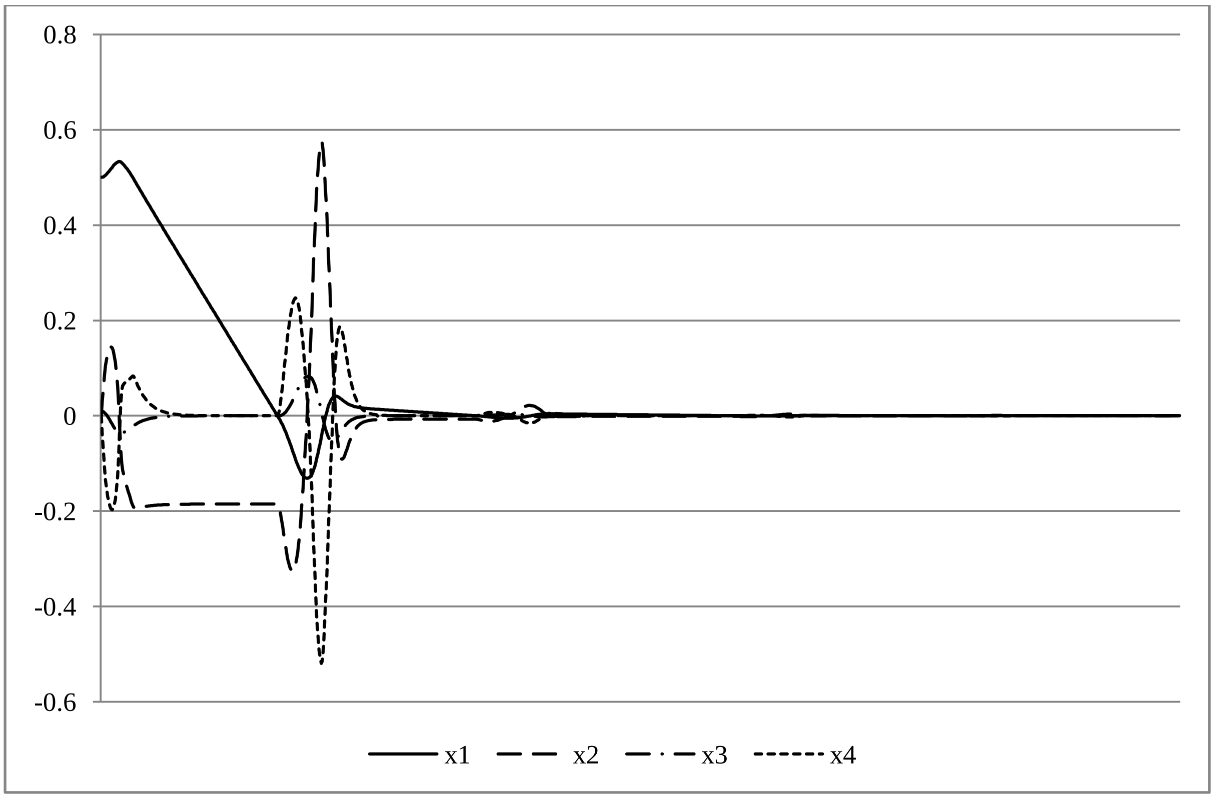

Figure 5.51 State variables convergence L2-341-2.

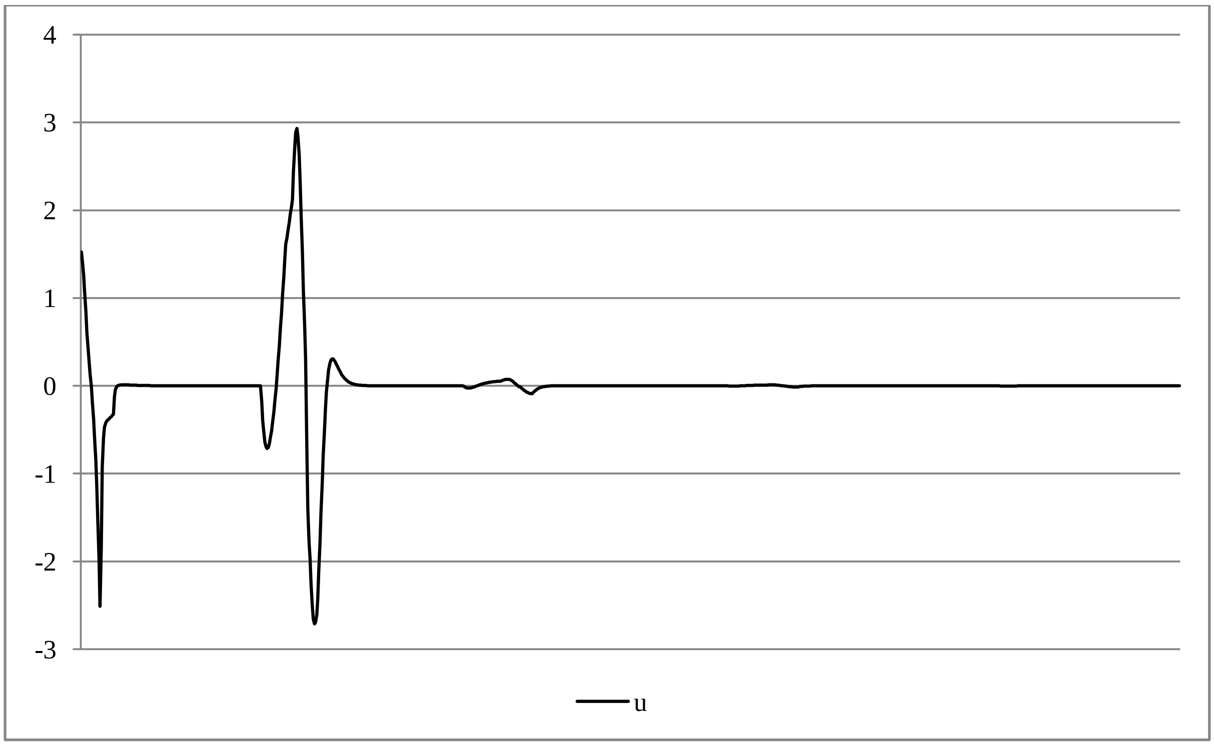

Figure 5.52 Controller L2-341-2.

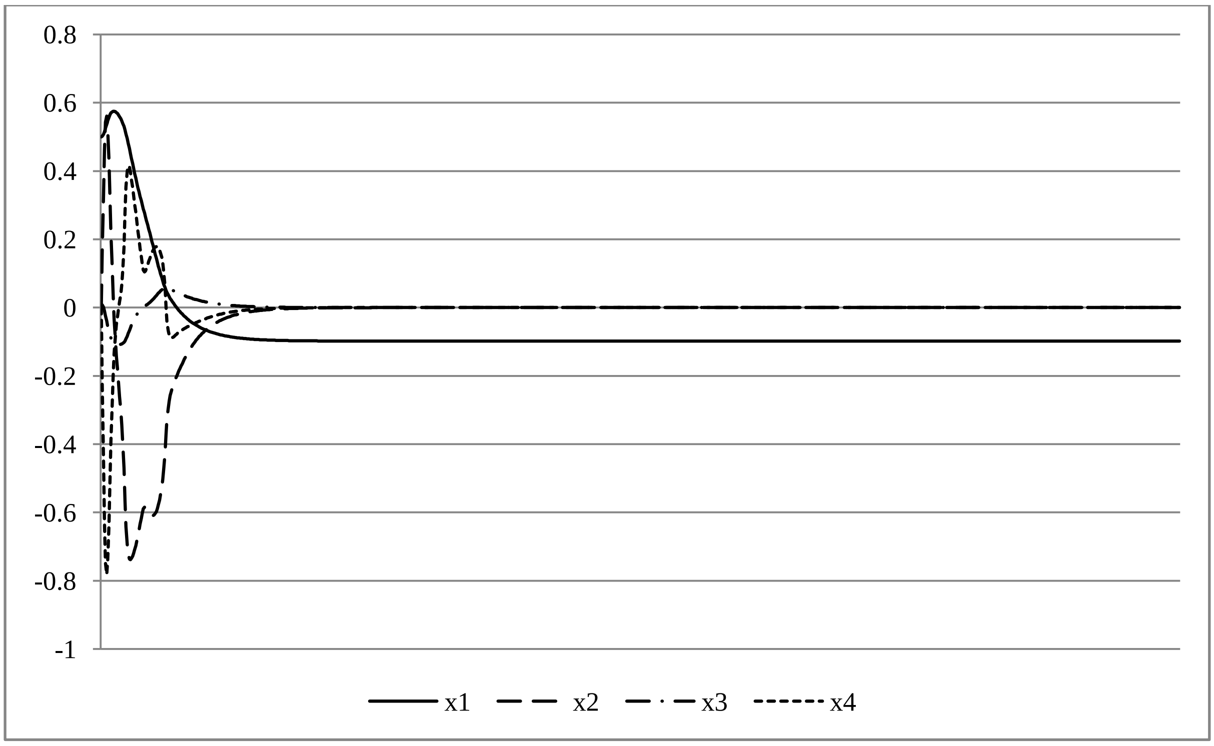

Figure 5.53 State variables convergence L3-3-41-2.

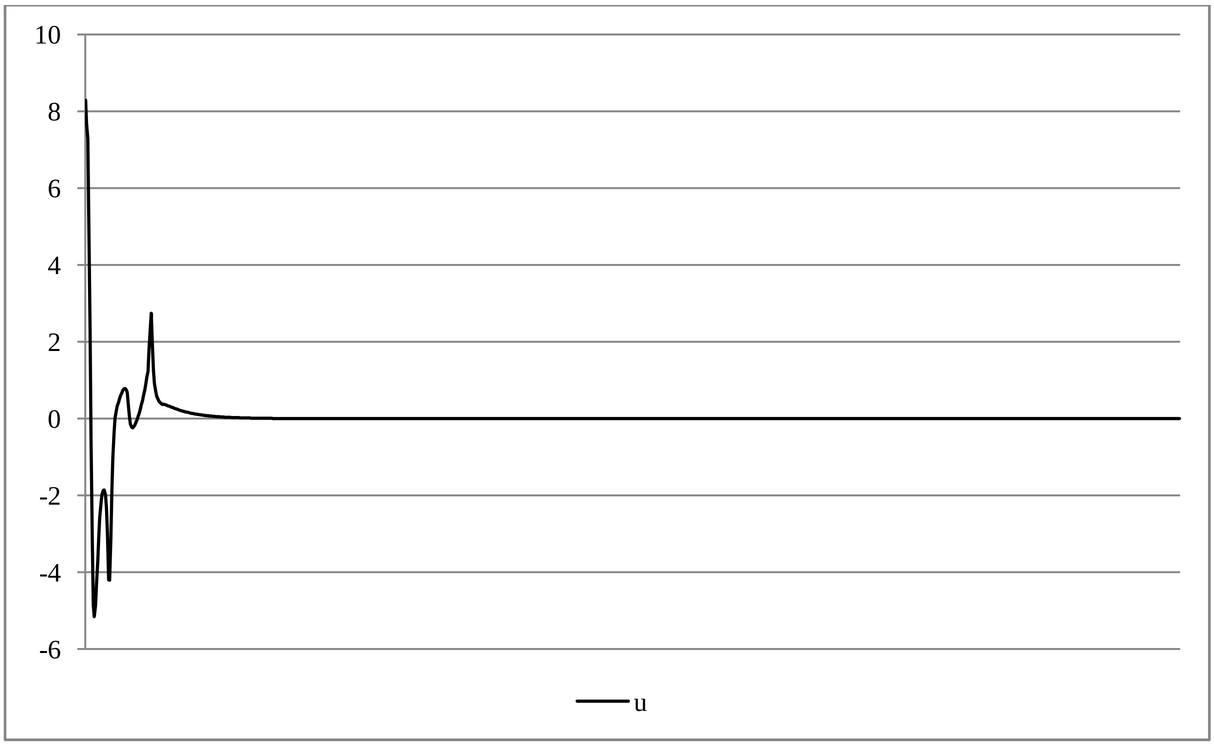

Figure 5.54 Controller L3-3-41-2.

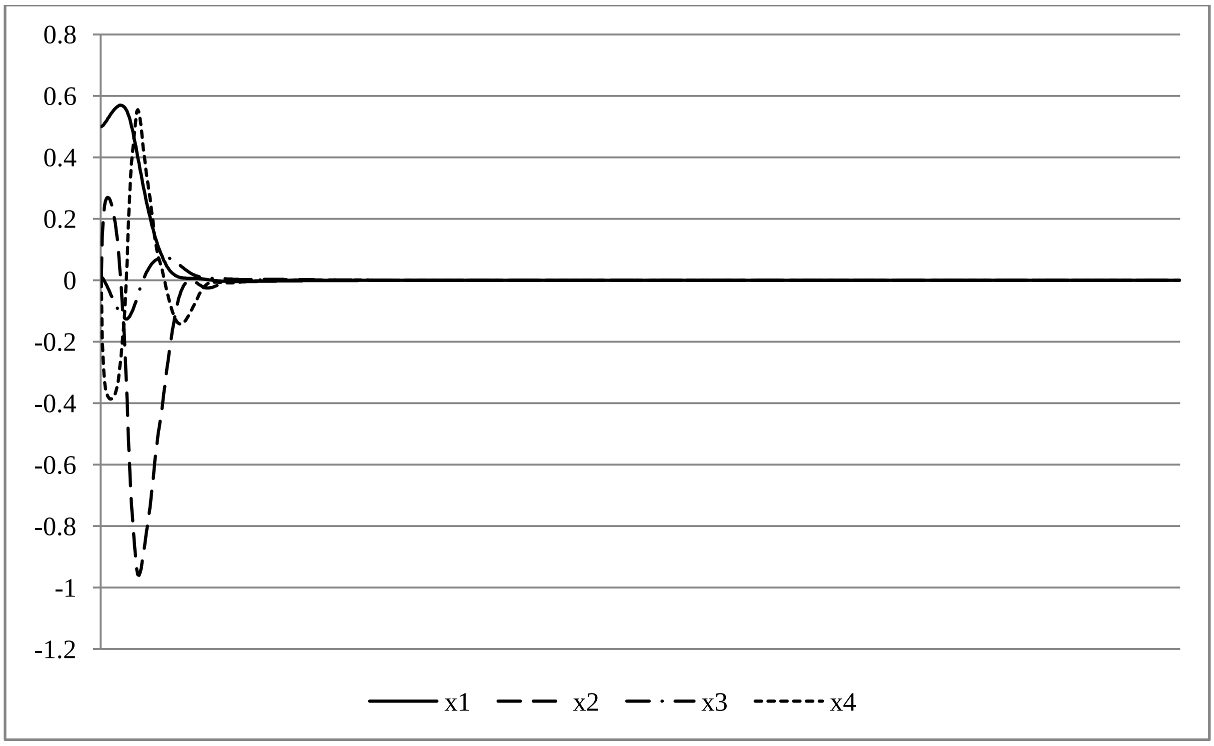

Figure 5.55 State variables convergence L3-3-4-12.

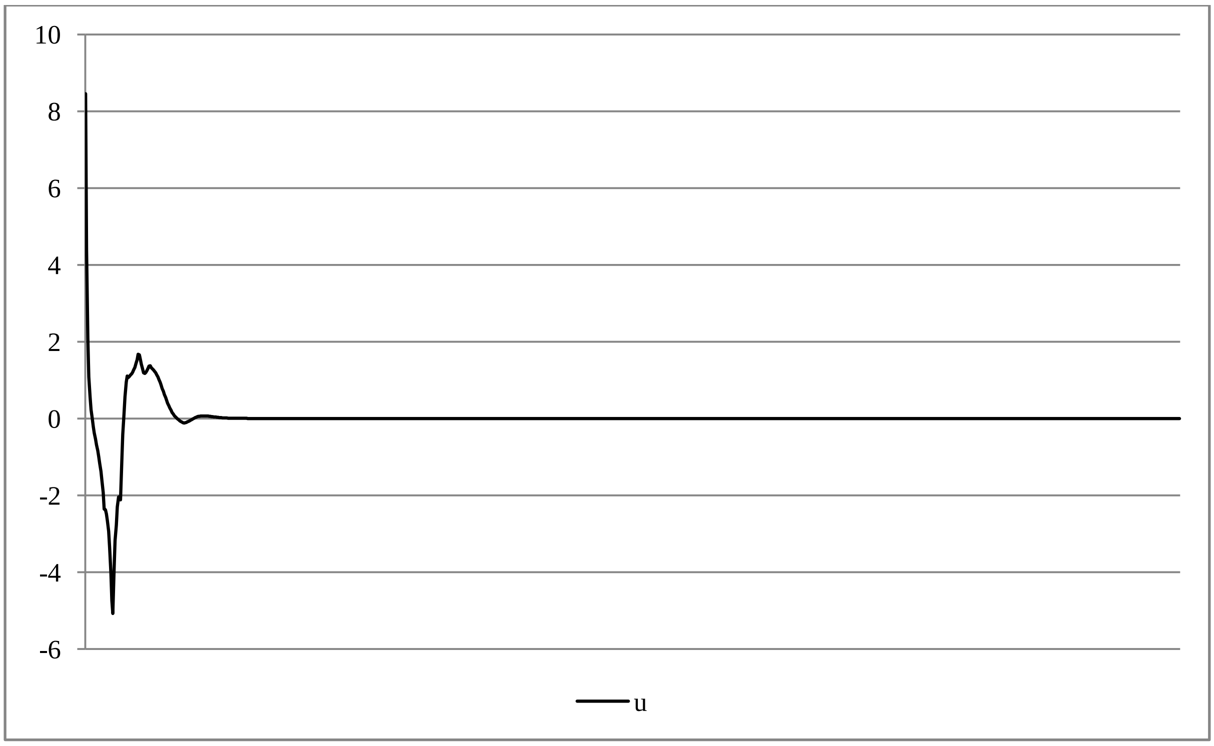

Figure 5.56 Controller L3-3-4-12.

5.9 Controller tests

As mentioned earlier, intermediate variables u_1 and u_2 may have not any physical representation. They can be considered as approximations to the controller action. By testing their performance, with one or two layers removed from the HFS one can expect to determine the robustness of the controller. Examining the approximate control provides insight into 'working parts' of the HFS.

When one layer is eliminated it is denoted by *L2-mn* or *L3-mn-k*. In a 3-layered HFS *L3-mn* denotes elimination of the two last layers.

For controller tests the best controllers for the 2 and 3-layered HFS are selected. The best 2-layered HFS are: *L2-34-12*, *L2-14-23*, *L2-13-24* and *L2-12-34*. The best 3-layered HFS are: *L3-34-1-2*, *L3-14-3-2*, and *L3-13-4-2*.

5.9.1 Two layers HFS controller test results

The best performing among the 2-layered HFS is the controller with topology *L2-34-12*, see Figure 5.5 and Figure 5.6. The controller is tested with its last layer removed: topology *L2-34*. The controller stabilises the system for 1.6 time units before 'crashing', i.e., until breaking the state bounds: $|x_1| \leq 1.0$ and $|x_3| \leq \pi/6$, see Figure 5.57. Otherwise the controller exhibits very regular behaviour.

Then the controller with topology *L2-14-23* is tested, see Figure 5.9 and Figure 5.10, with its last layer removed. One layer version *L2-24* 'crashed' at $t = 0.8$, the angular velocity of the pole x_4 rising steeply, Figure 5.59.

The controller with topology *L2-13-24* is tested with its last layer removed. As can be seen in Figure 5.61 and Figure 5.62, the approximate controller with only one layer rule base attempts to stabilise the system for the whole period of time $T_f = 20$: the angle of the pole x_3 is stabilised in a narrow band around the origin, the angular velocity x_4 oscillates but the values of x_4 remain within $[-1, 1]$ band. Therefore, the approximate controller, with second layer removed, performs well, while the controller with full rule base performs poorly, see Figure 5.15.

The controller with topology *L2-12-34* with its last layer removed is one of the worst performing controllers, see Figure 5.63 and Figure 5.64. This poor performance comes in spite of relatively good performance of the controller with the full rule base, see Figure 5.7.

The controller stabilises the system for 3.76 time units until the angle of the pole x_3 breaks the state limits.

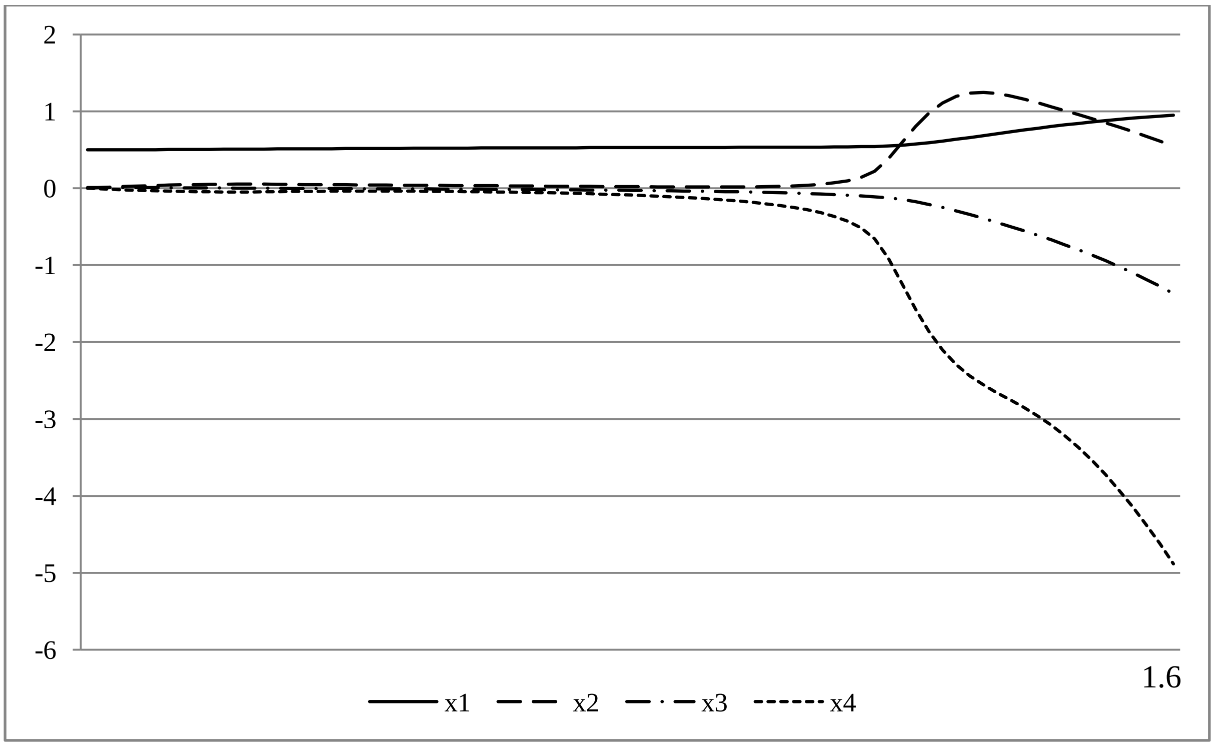

Figure 5.57 State variables convergence L2-34.

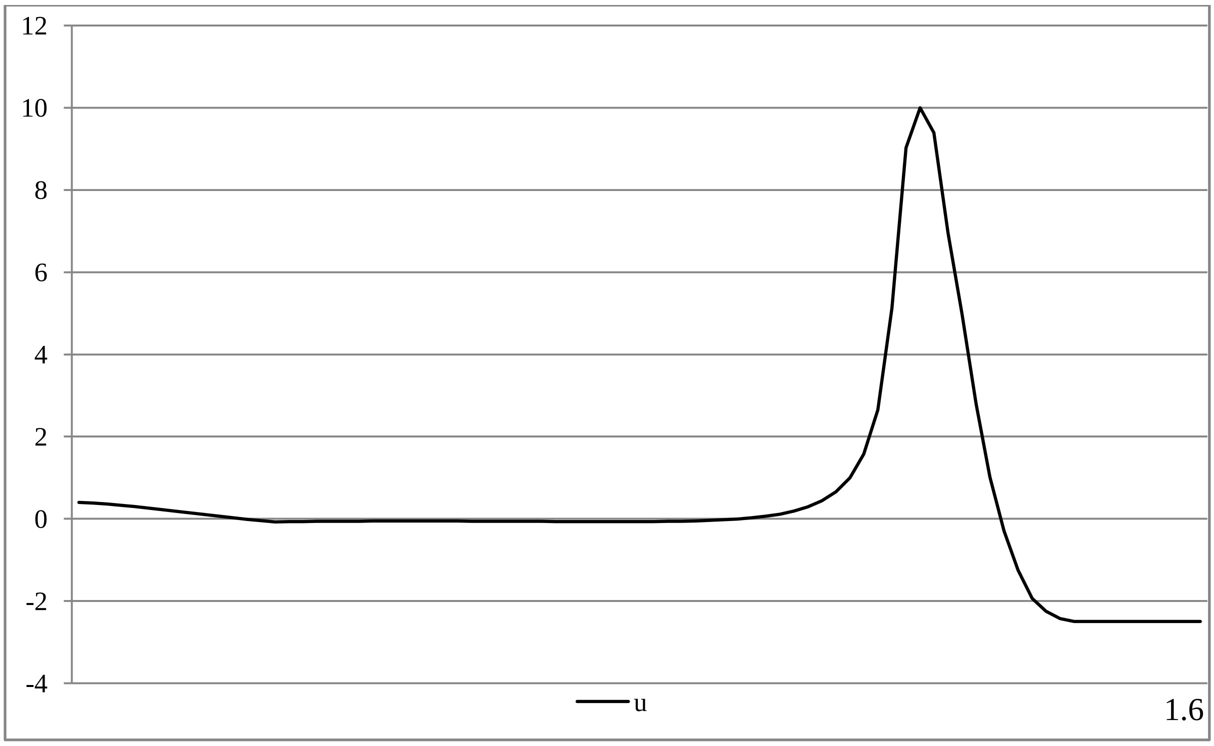

Figure 5.58 Controller L2-34.

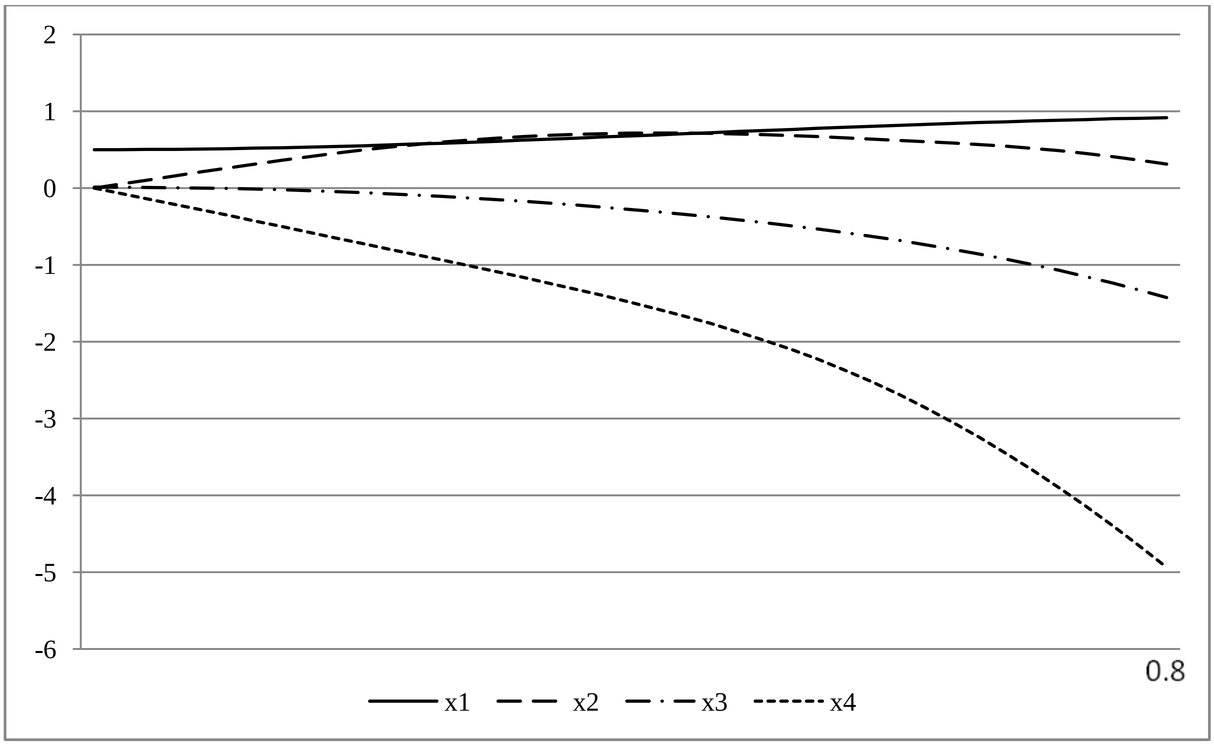

Figure 5.59 State variables convergence L2-14.

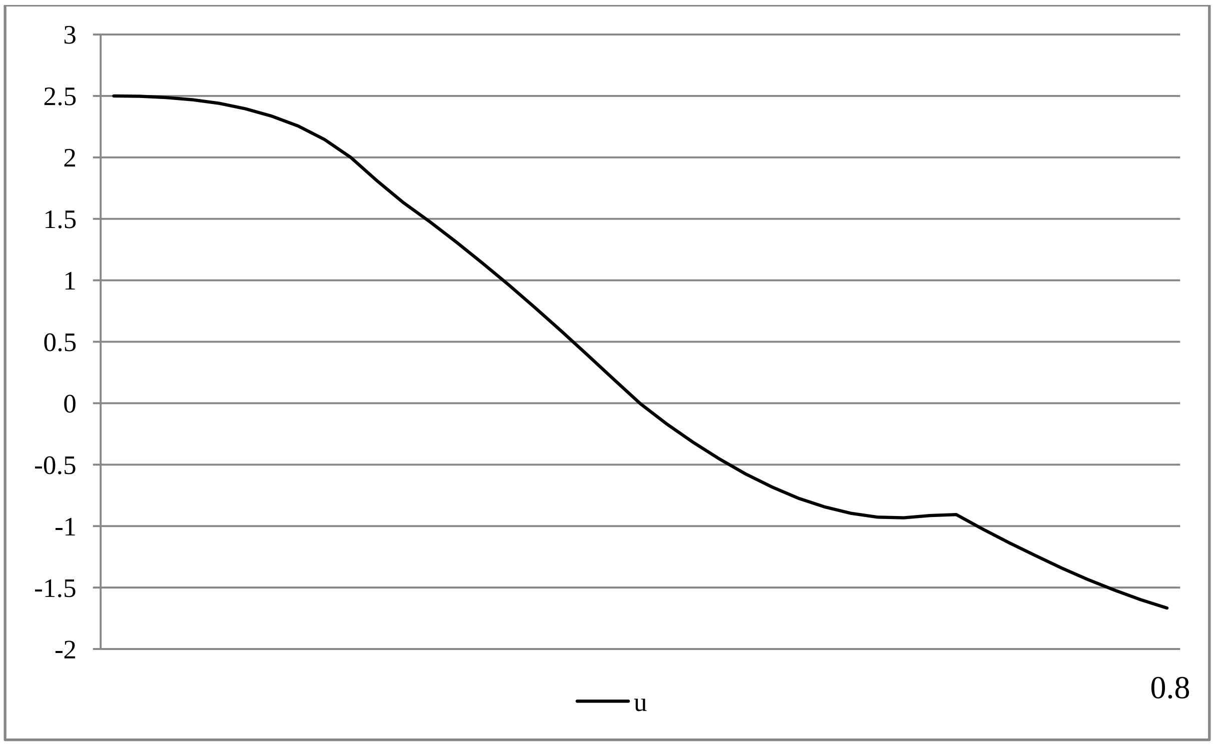

Figure 5.60 Controller L2-14.

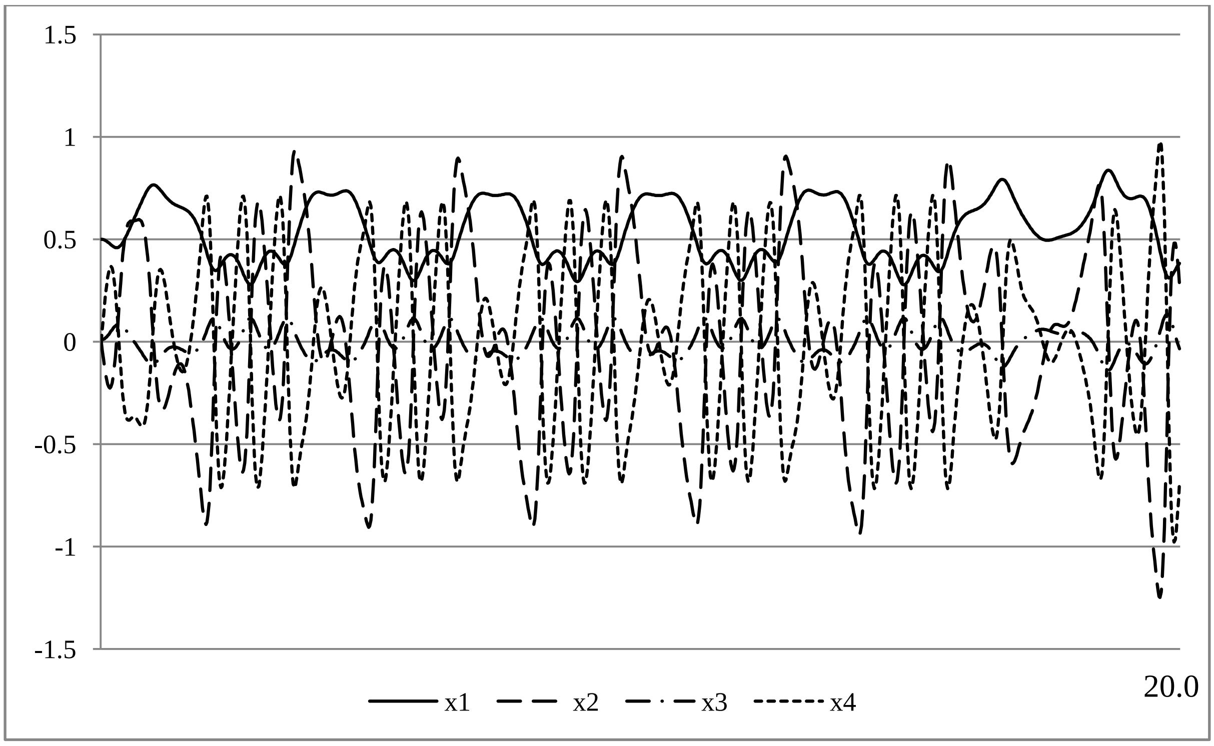

Figure 5.61 State variables convergence L2-13.

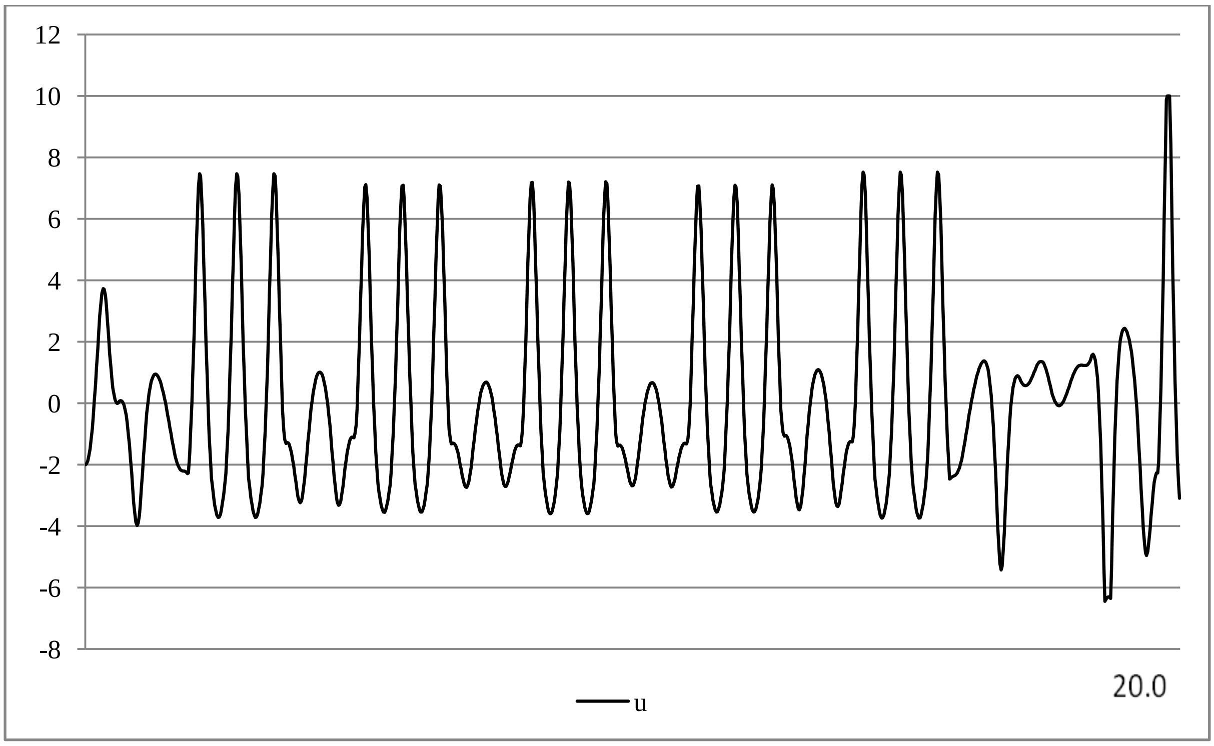

Figure 5.62 Controller L2-13.

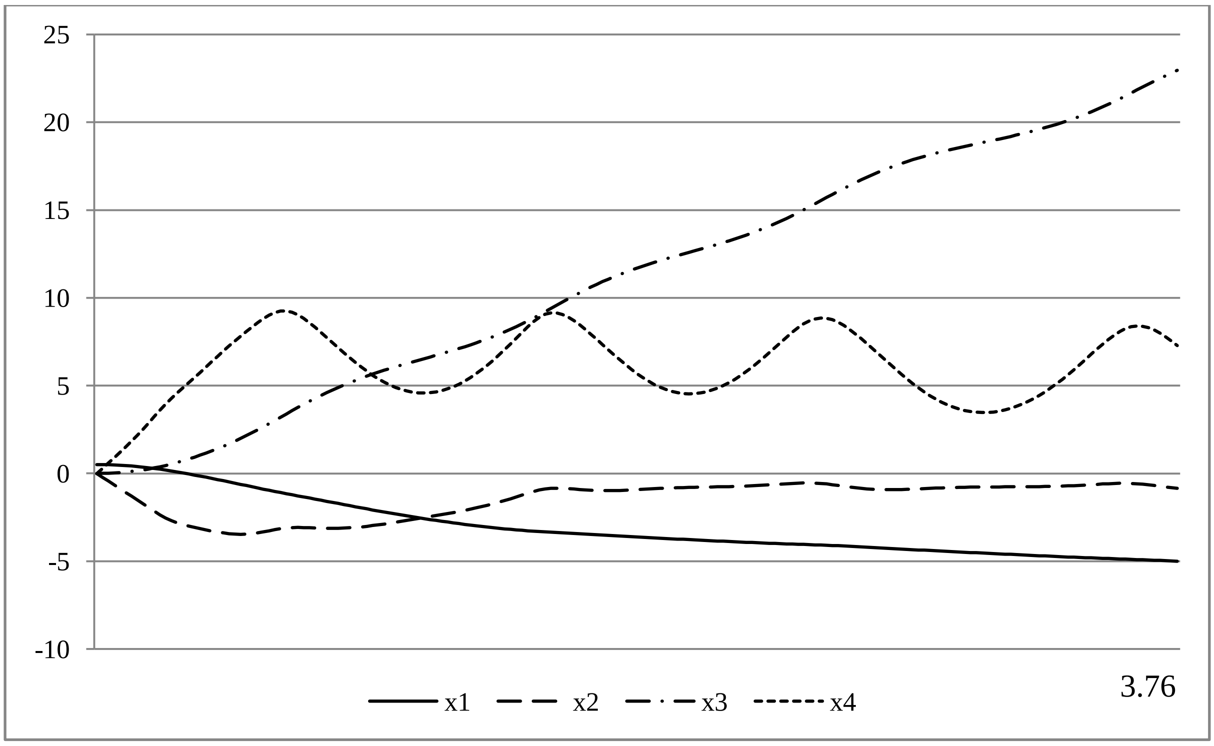

Figure 5.63 State variables convergence L2-12.

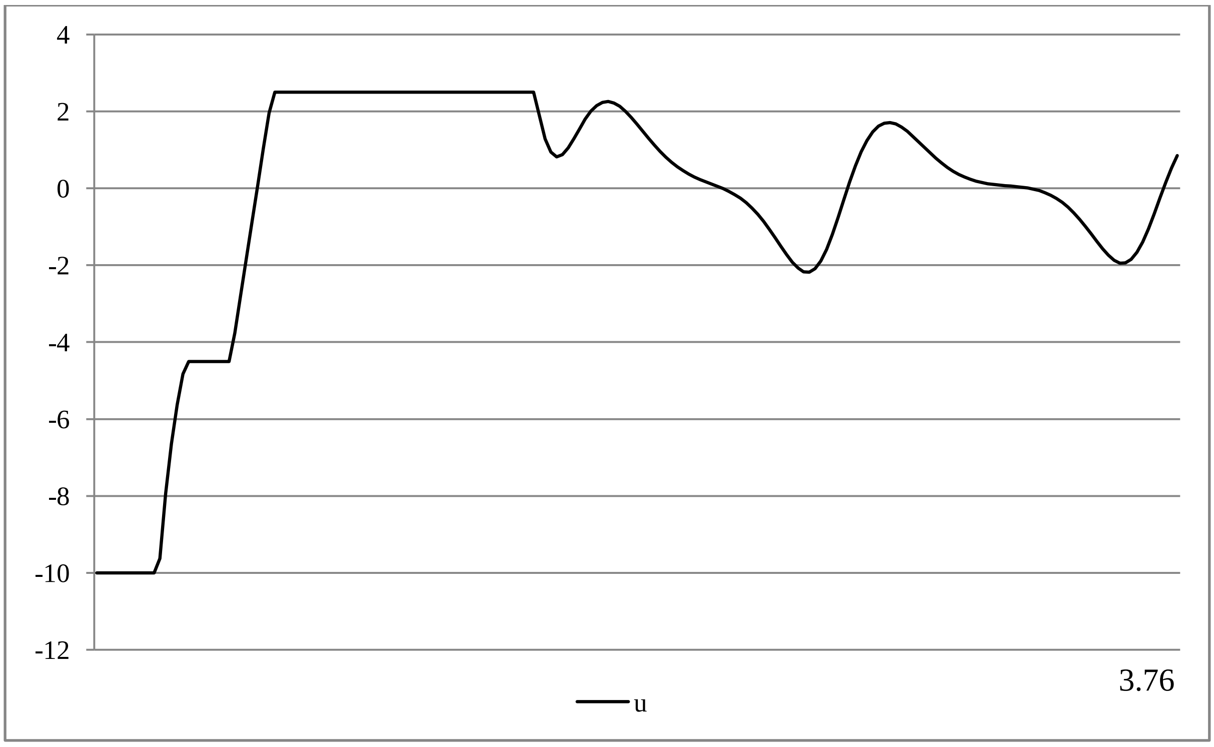

Figure 5.64 Controller L2-12.

5.9.2 Three layers HFS controller test results

In the 3-layered HFS the controllers are tested by removing the last layer or the two last layers. The controller with topology *L3-34-1-2* is analysed first. The simulation results are shown in Figure 5.67 and Figure 5.68 (the two last layers removed from the HFS) and in Figure 5.65 and Figure 5.66 (the last layer removed from the HFS). The approximate controller u_2 maintains control of the system for the whole prescribed time $T_f = 20$ and exhibits very regular behaviour.

The controller with topology *L3-23-4-1* is analysed and simulation results are shown in Figure 5.71 and Figure 5.72 (with two last layers removed) and in Figure 5.69 and Figure 5.70 (with the last layer removed). The control pattern is regular but the 'crash-time' for this controller is relatively short (1.2 and 0.78 respectively).

The well performing controller *L3-14-3-2*, see Figure 5.31 and Figure 5.32, is analysed and simulation results are shown in Figure 5.75 and Figure 5.76 (with two last layers removed) and in Figure 5.73 and Figure 5.74 (with the last layer removed). The additional rule base in the control system *L3-14-3* produces a better result than smaller rule base in *L3-14*.

In general, the approximate controller u_2 (intermediary control between layer 2 and layer 3) maintained control of the system for longer periods of time than approximate controller u_1 (intermediate control between layer 1 and layer 2), which is not surprising as the controller u_2 has a larger knowledge base to rely on.

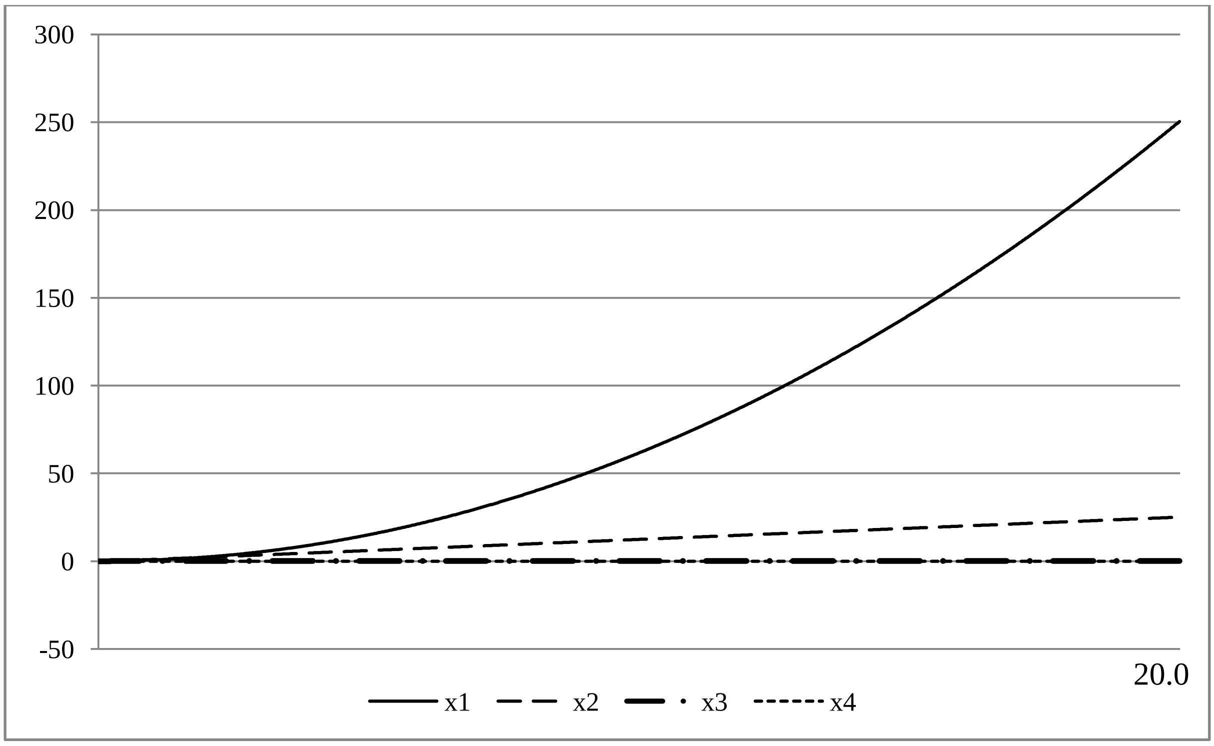

Figure 5.65 State variables convergence L3-34-1.

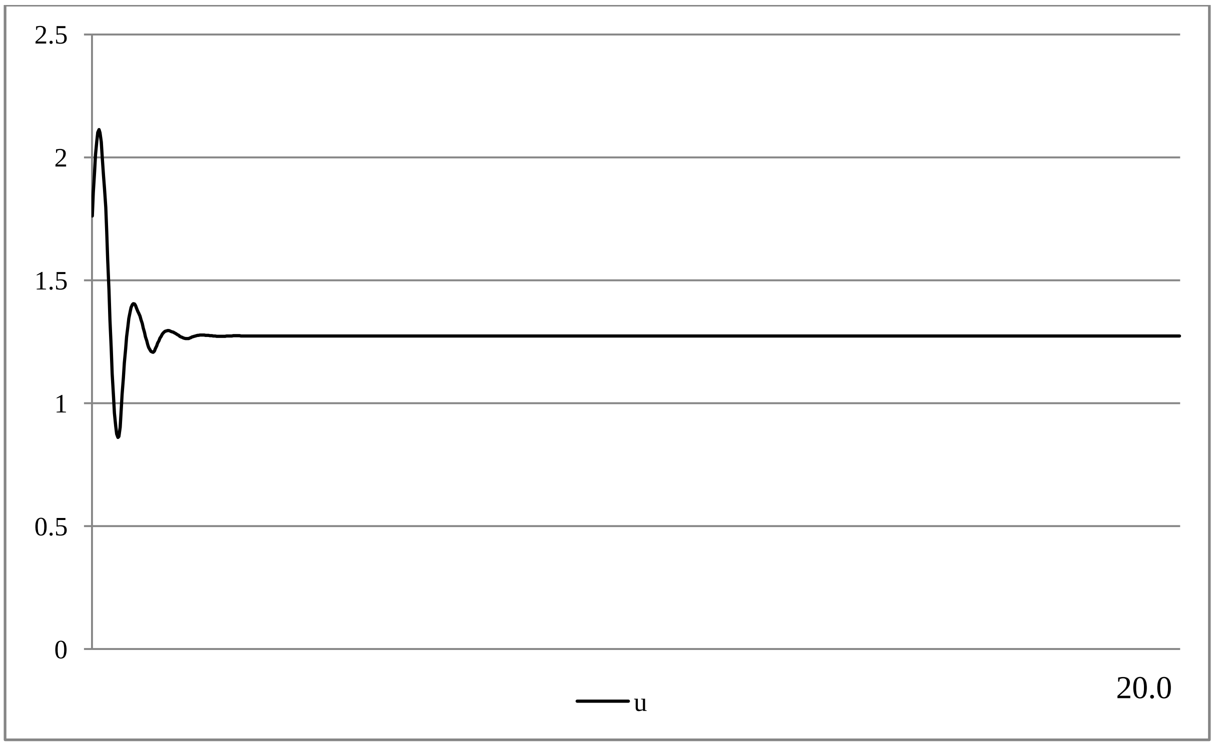

Figure 5.66 Controller L3-34-1.

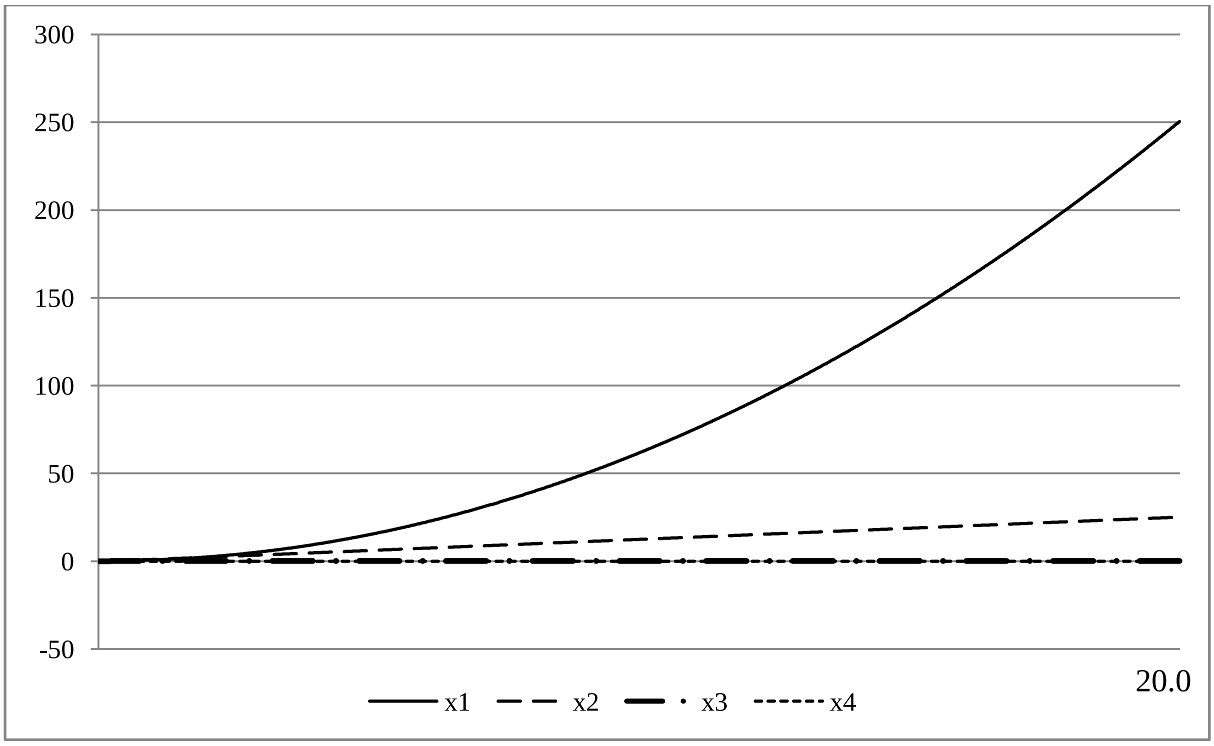

Figure 5.67 State variables convergence L3-34.

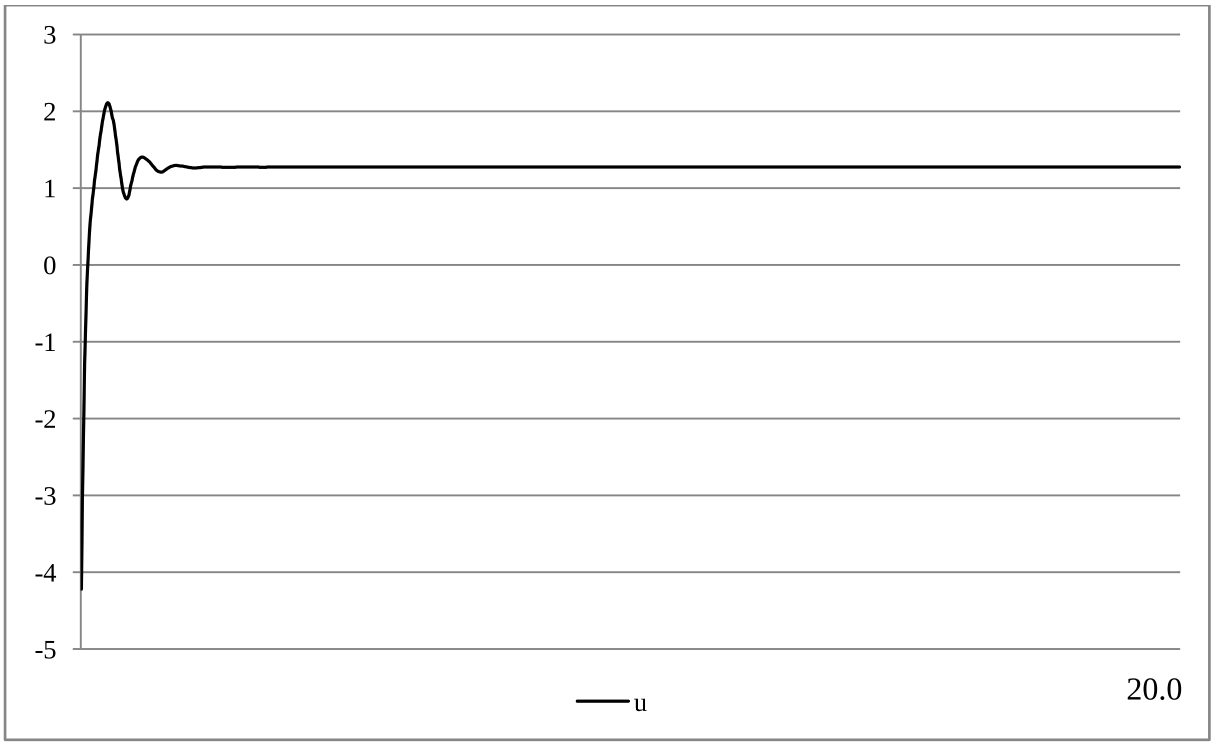

Figure 5.68 Controller L3-34.

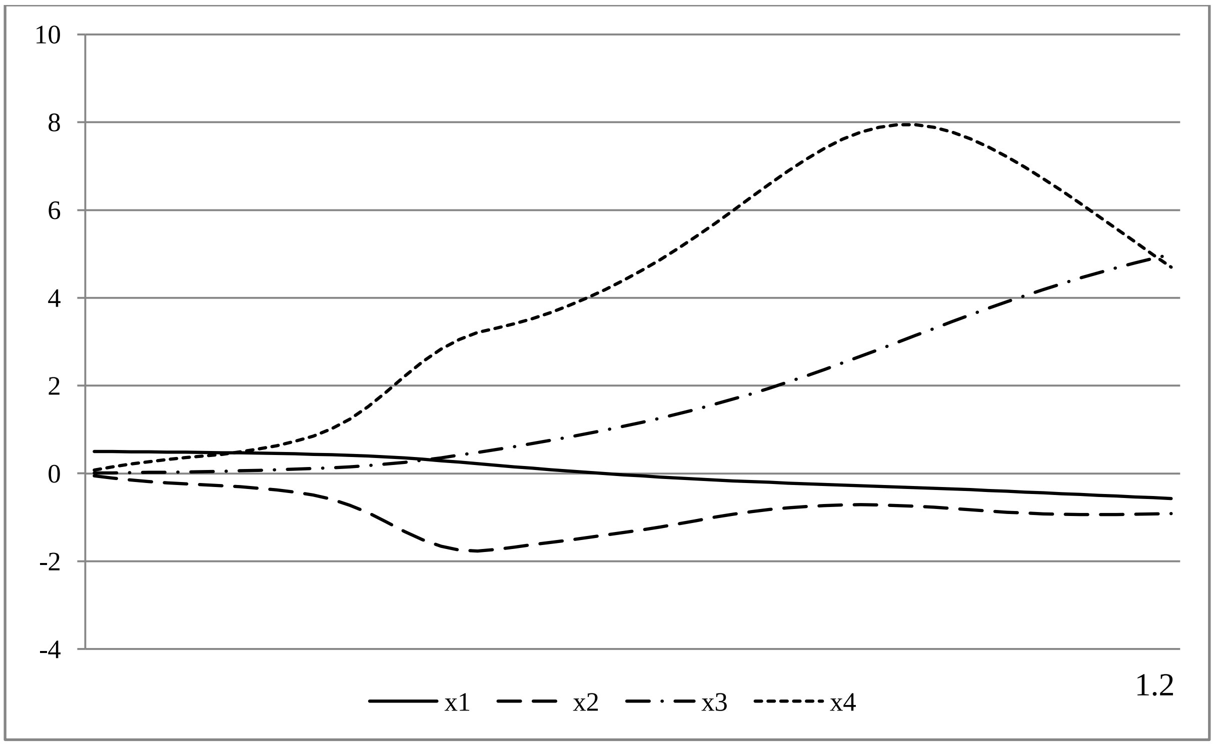

Figure 5.69 State variables convergence L3-23-4.

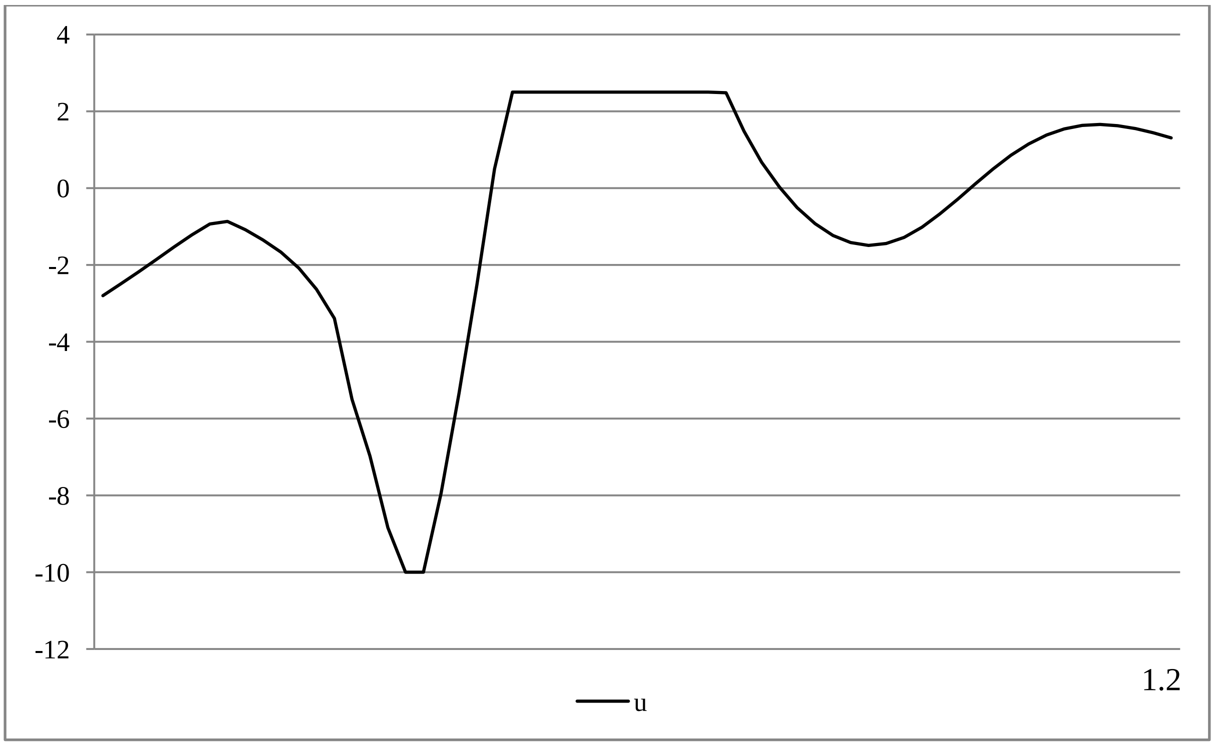

Figure 5.70 Controller L3-23-4.

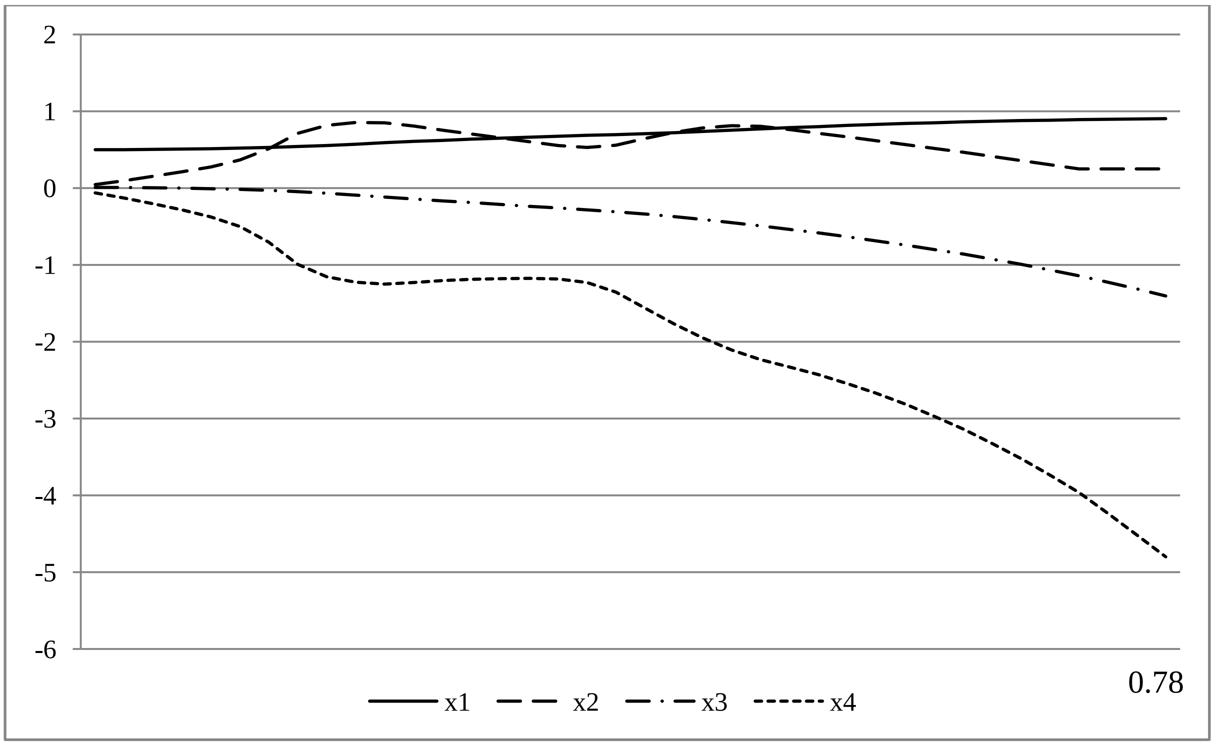

Figure 5.71 State variables convergence L3-23.

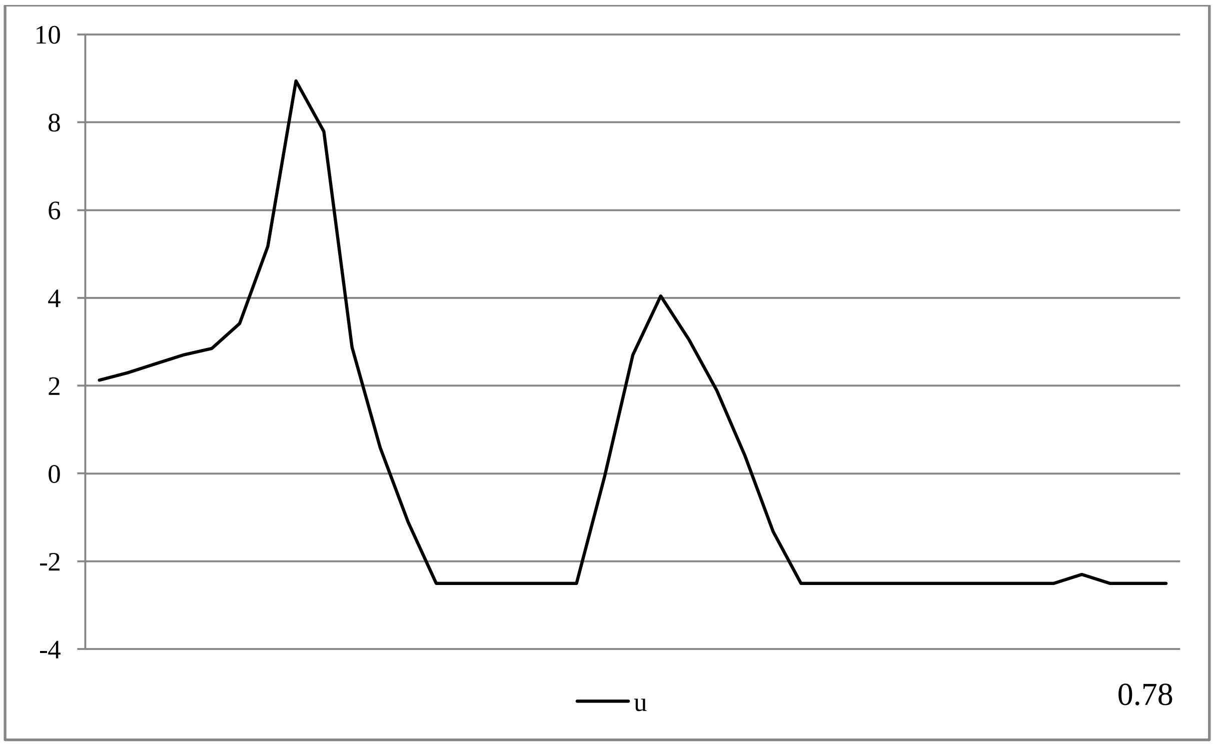

Figure 5.72 Controller L3-23.

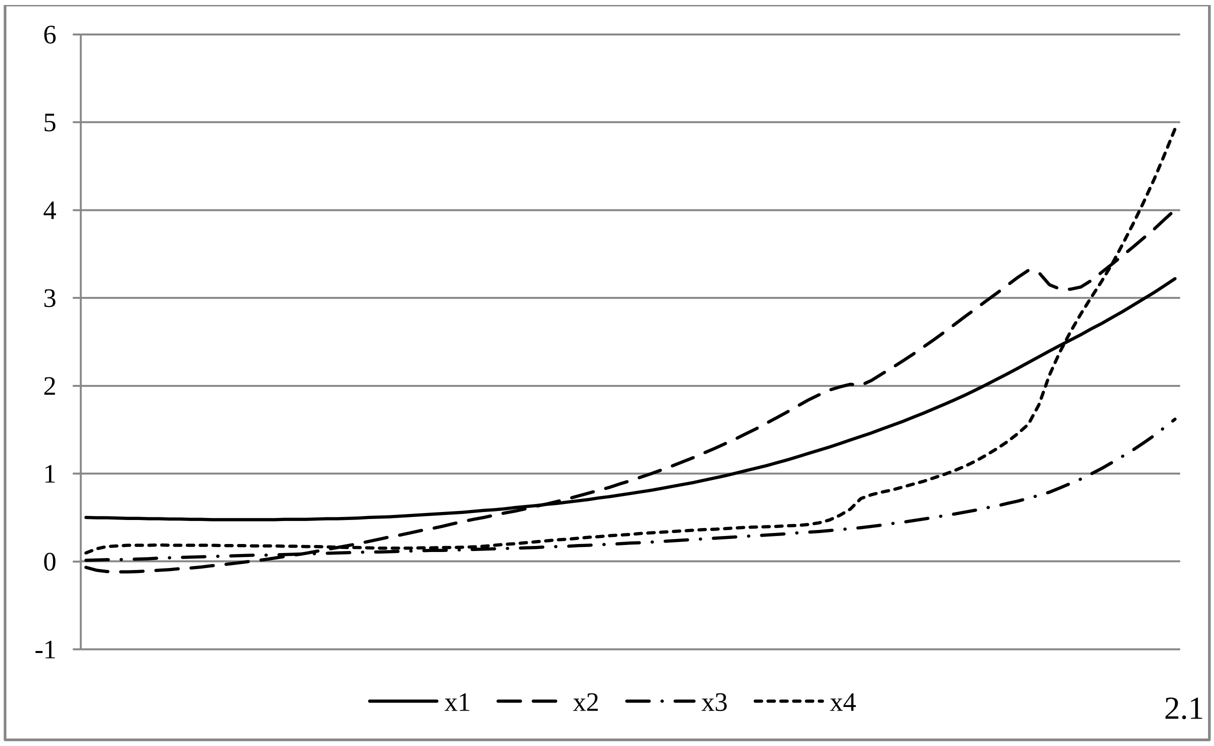

Figure 5.73 State variables convergence L3-14-3.

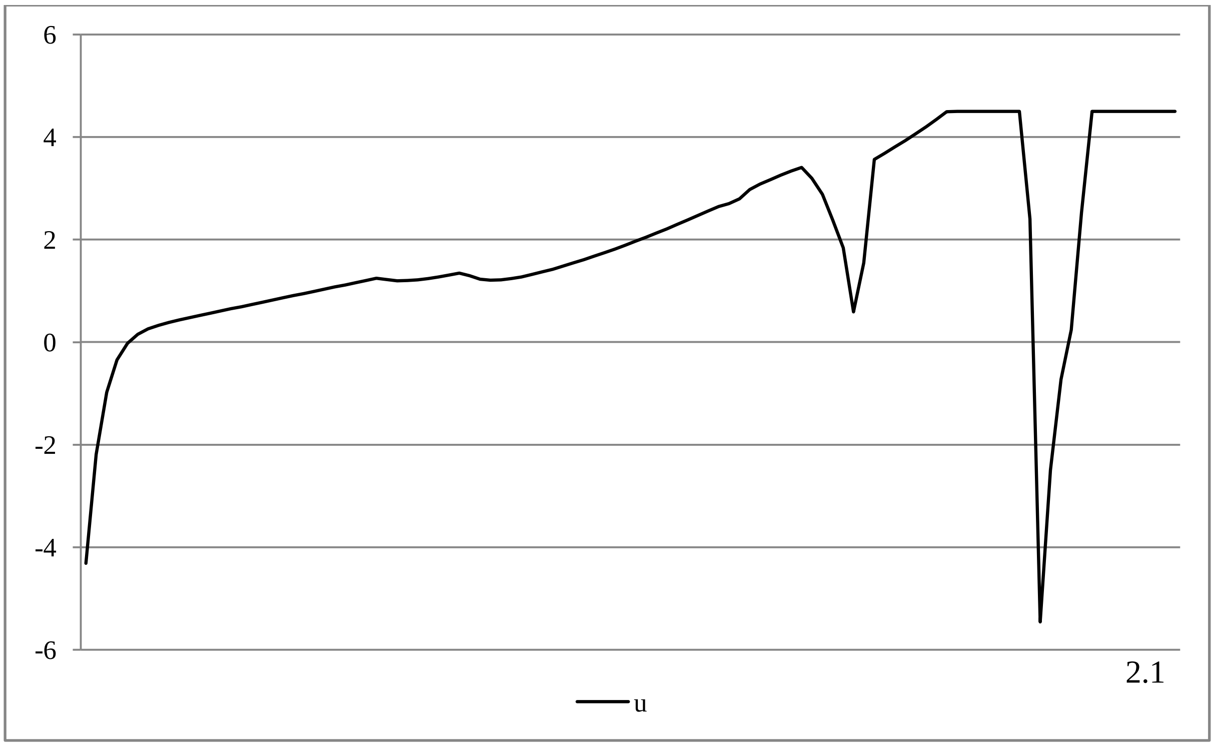

Figure 5.74 Controller L3-14-3.

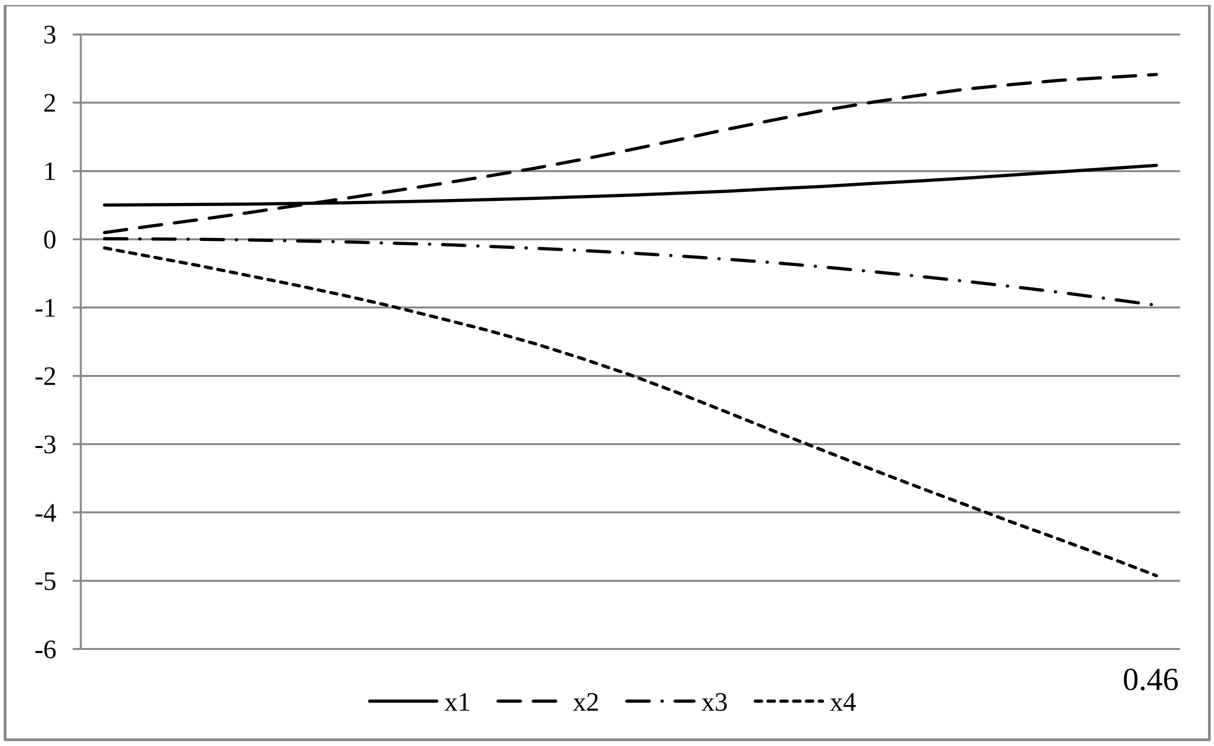

Figure 5.75 State variables convergence L3-14.

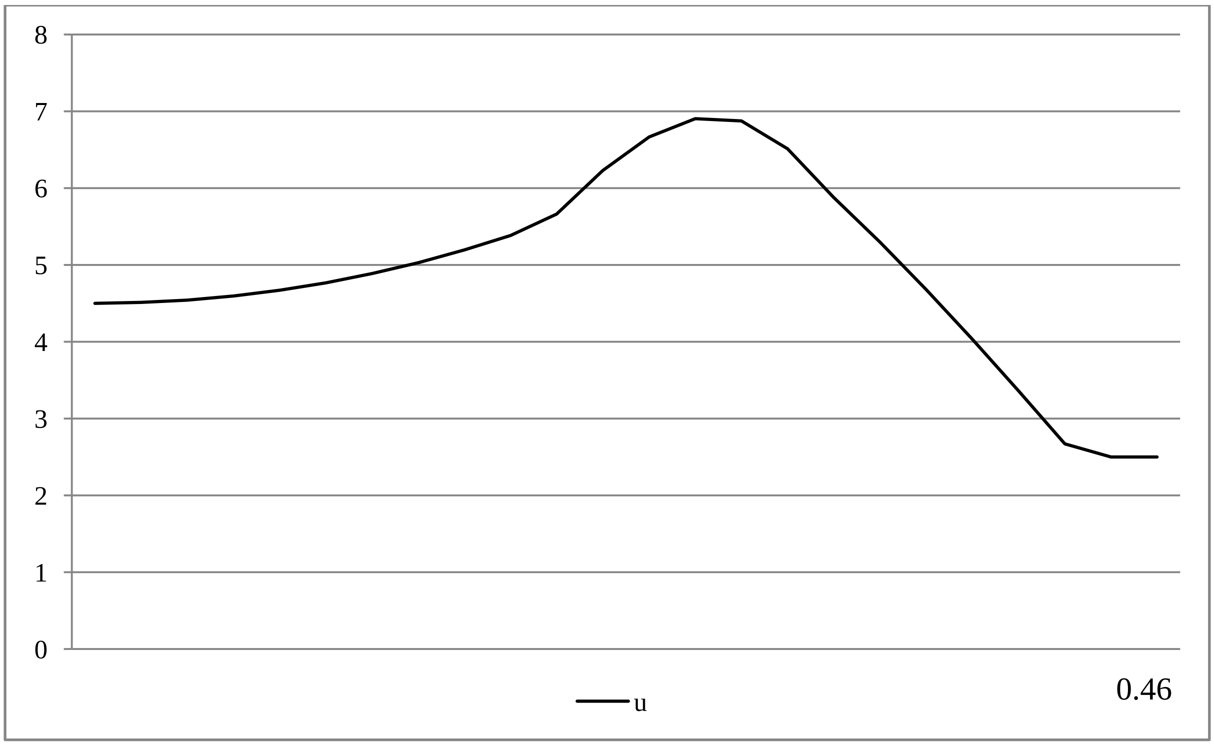

Figure 5.76 Controller L3-14.

5.9.3 Controller test: remarks

The approximate controllers even from the best performing topologies did not exhibit the same quality of control as the final controller u. In some cases, approximate controllers from the ‘worse’ performing topologies, performed reasonably well. Experiments with the HFS with layers removed show that the HFS is not a mere sum of its rule bases in the component layers. Topology of the HFS is a key factor in the performance of the controller. It has been shown that the HFS needs to be considered in its entirety, not as an assembly of the better or worse performing component layers.

5.10 Results

It can be seen that the performance of the fuzzy controller is not related to the speed of learning process. Some very good controllers require a great number of generations to be found, while some poor performers are very ‘fast learners’. In an extreme case, one ‘lucky’ run no 7 for topology *L3-12-4-3* simulation produced a satisfactory controller after only one generation (due to a randomly generated initial population that accidently included individuals that were satisfactory solutions). Simulation for topology *L2-14-2-3* produced satisfactory controller after only four generations in run no 3. The fastest learning process in the 2-layered HFS is achieved in a simulation for topology *L2-23-14* when satisfactory controller was found after seven generations. A single layer FS learning speed was steady at about 200 generations and even though it was slower than all other topologies, it produced a relatively well performing controller.

In the 2-layered HFS, on average, the second ‘fastest’ learner is the best performer: *L2-34-12*. In the 3-layered HFS there is not a clear trend, with *L3-34-2-1* being the third. Furthermore, *L3-34-1-2* is found being an average ‘learner’, with 5 other topologies besting its average learning speed.

Obviously, in all simulations random nature of initial population played a role but if there had been a clear trend that better controllers are found faster, it would have been seen in the simulation data. The tables with learning speed are shown Table 5.5 in and Table 5.6.

Obviously, for one-layer structure the learning speed can be examined for only one topology but for different initial populations. Empty space instead of a numeric value means that the algorithm failed to achieve the desired convergence in 300 generations, which happened four times in the case of topology *L3-23-1-4*. This is a surprising result as one would expect a similar performance as in the case of the 2-layered HFS topology *L2-23-14*. As can be seen, this is not the case and both topologies are worlds apart in terms of performance due to a different architecture of the HFS. Even a seemingly insignificant difference, such as a decomposition which includes an additional layer, can prove critical, especially since physical significance of the intermediate control between layers is unknown. For example, the 2-layered HFS topology *L2-14-23* can be further decomposed into the 3-layered HFS, either *L3-14-2-3* or *L3-14-3-2*, the controller performance for those topologies is shown in Figure 5.29—Figure 5.32; *L3-14-3-2* provides satisfactory control system while *L3-14-2-3* does not.

Table 5.5 Learning speed: 1 and 2-layered HFS

Run No	L1-1234	L2-12-34	L2-13-24	L2-14-23	L2-23-14	L2-24-13	L2-34-12
1	201	150	119	183	150	201	13
2	181	93	174	204	170	188	160
3	204	139	157	179	167	160	114
4	180	54	111	222	150	152	103
5	202	124	151	161	138	176	154
6	217	126	139	199	7	106	169
7	205	127	211	158	150	153	52
8	201	114	159	188	155	158	150
9	189	30	153	167	112	205	177
10	199	44	169	203	151	227	172
Average	198	100	154	186	135	173	126
Total Avg	198	146					

Table 5.6 Learning speed: 3-layered HFS

Run No	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12
1	28	153	210	166	109	192	144	202	154	204	81	55
2	157	93	181	193	150	188	212	165	3	160	178	57
3	156	103	172	115	4	171		179	153	156	161	157
4	151	119	202	211	154	205	162	163	162	162	143	210
5	164	150	201	176	112	236		167	164	244	68	94
6	133	164	205	185	152	191	209	165	150	200	146	81
7	150	1	206	158	117	169	159	208	23	195	152	27
8	151	151	151	49	10	137		36	145	177	151	204
9	152	62	233	174	145	170		184	157	166	156	150
10	152	142	213	153	152	210	215	123	130	155	207	129
Average	139	114	197	158	111	187	184	159	124	182	144	116
Total Avg	151											

where C1 denotes: *L3-12-3-4*, C2: *L3-12-4-3*, C3: *L3-13-2-4*, C4: *L3-13-4-2*, C5: *L3-14-2-3*, C6: *L3-14-3-2*, C7: *L3-23-1-4*, C8: *L3-23-4-1*, C9: *L3-24-1-3*, C10: *L3-24-3-1*, C11: *L3-34-1-2*, C12: *L3-34-2-1*

After analysing ten simulation results for each topology, i.e., *L2-34-12* and *L3-34-1-2*, it seems that the 2-layered HFS provides a better solution to the control problem of the inverted pendulum. This result is consistent with the physical model of the inverted pendulum system as the pole and cart variables can be grouped in two subsystems that are mirrored in the 2-layered HFS.

In general, the results confirm that the decomposition of the hierarchical fuzzy structure should be performed along weak interdependency between input variables.

Summary

In this chapter different topologies associated with hierarchical fuzzy control of the inverted pendulum are examined (with nonlinear dynamics) to gain insight into the decomposition problem of hierarchical fuzzy systems. Investigation of HFS topologies is focused on the influence of the given topology on the controller performance. The fuzzy controllers are learnt by evolutionary algorithm. The results, obtained to examine the efficiencies in learning by the evolutionary algorithms and fuzzy control of the pendulum, are compared.

Furthermore, experiments with intermediate variables u_1 and u_2 (they can be considered as approximations to the controller action) have been conducted to determine the robustness of the controller for different HFS topologies. Tests were performed with one or two layers removed from the HFS to gain insight into controller performance in the hierarchical structure. It has been demonstrated that the HFS needs to be considered in its entirety and not as an assembly of better or worse performing component rule bases.

Chapter 6 CO-EVOLUTIONARY ALGORITHM FOR HIERARCHICAL FUZZY CONTROL OF THE INVERTED PENDULUM

6.1 Introduction

Much research has been conducted on the hierarchical fuzzy control but relatively little on the problems associated with the problem of HFS decomposition to achieve the best controller performance. In (Stonier and Zajaczkowski 2003) and (Stonier and Zajaczkowski 2004) various aspects associated with layers decomposition of a hierarchical fuzzy controller for the control of the inverted pendulum problem are examined, in particular the non-uniqueness associated with such layers decomposition.

A three layered HFS with angular speed and angular position of the pole as input in layer 1, cart's speed as input in layer 2, and cart's position as input in layer 3, provides the 'best' controller under the given performance criteria, see Chapter 5. Very similar performance is achieved when input variables are reversed in the layer 2 and 3. Please note, this might not be true for other input configurations in general.

It was noticed in the computer simulations that the state variables convergence was often oscillatory ('underdamped') and the controller action was not smooth. One way to improve the smoothness of the controller is to add appropriate penalty terms to the fitness function in the evolutionary algorithm but this adds to the complexity of the fitness function in the evolutionary algorithm.

A co-evolutionary algorithm is used to fine tune parameters of the input and output membership sets as well as learn the fuzzy rules in the fuzzy controller, to yield more 'damped' convergence of the state variables and smoother controller function.

One of the problem in designing hierarchical fuzzy system is the choice of membership functions for the input and output variables. In many cases the membership functions are arbitrarily chosen and uniformly spread over the range of the state variables. The purpose of this investigation is to examine how evolutionary algorithm can improve the controller performance by fine-tuning the membership functions associated with the fuzzy sets. Gaussian membership functions are used for the co-evolutionary algorithm.

Co-evolutionary algorithms (CEA) concept was introduced by D.W. Hills (Hills 1990) in a predator-prey model. In many cases, the CEA proved to be more efficient than conventional EA especially in complex multi-variable applications. The increased effectiveness of the CEA can be explained by the fact that the individuals in the population can either cooperate (in cooperative CEAs) or compete (in competitive CEAs) with other individuals in terms of their defined objectives.

CEA differs from the conventional EA, which uses a single population, by employing two or more competing/cooperative populations using the common fitness function. Cooperative CEAs were proposed by M.A. Potter and K.A. De Jong (Potter and De Jong 1994). They presented cooperative co-evolutionary genetic algorithms as a way of managing the increasing complexity in evolutionary problems. The search space increases exponentially with each additional element in the string representing an individual; the search space can be dramatically reduced by using multiple populations that each encodes only part of the complete solution. If there is only weak or no interdependence between the partial strings in different populations the narrower evolutionary focus does not adversely affect cooperative co-evolution, and it benefits from a reduced search space (Potter and De Jong 1994), (Potter and De Jong 1995), (Young and Stonier 2003).

In cooperative CEA the system is decomposed into a number of interacting component subsystems. All component subsystems evolve in parallel. Just like with the hierarchical fuzzy system decomposition, the partial individuals that are to form co-evolutionary populations (subpopulations) are usually grouped to minimise interdependence between individuals from the created subpopulations.

The idea behind cooperative co-evolution is to take a string of elements that completely specifies a solution and break it into smaller strings with each of the smaller strings encoding only part of the complete solution. Those partial strings are stored in different (genetically isolated) populations. Individuals in the same population are strings encoding the same part of the complete solution. Each population is then evolved by its own evolutionary algorithm mechanism, see (Pena-Reyes and Sipper 2001).

Each individual in a co-evolutionary population encodes only part of the complete solution. Therefore, to evaluate the fitness value of any individual (potential solution) a representative from every co-evolutionary population is selected to recombine with the individual that is evaluated. The recombined (complete) individual is then evaluated by a defined fitness function. The fitness value of the representatives is not evaluated at this point.

In case of coevolving the membership functions properties two separate populations are defined that are coupled by the fitness function requiring strings from both populations for the fitness evaluation. In other words, an individual from the controller population needs to be matched with an individual from the membership functions population in order to evaluate fitness function, and vice versa.

There are two different methods of selecting the representatives from the co-evolutionary populations:

- Greedy: selection of the fittest individual.
- Explorative: selection of a random individual.

In Potter and De Jong's method (Potter and De Jong 1995) layers are added dynamically to a cascade neural network as required. Weicker's method (Weicker and Weicker 1999) takes an opposite approach: to start with an extreme decomposition (one variable per population) and merge populations as epistatic links become apparent between them (Potter and De Jong 1994), (Potter and De Jong 1995), (Young and Stonier 2003).

6.2 HFS topologies for co-evolutionary algorithm

The architecture of the 3-layered HFS selected for the simulations is shown in Figure 3.13. This HFS structure has two input variables in the first layer then and one input variable in second and third layer of the 3-layered HFS.

Two input configurations described in Chapter 3 and Chapter 5 are used for experiments with co-evolutionary algorithm: *L-34-1-2* and *L-34-2-1*.

6.3 Co-evolutionary algorithm

A co-evolutionary algorithm is a version of the algorithm used in (Stonier and Zajaczkowski 2004) to learn the fuzzy rules in the HFS with three knowledge bases and at the same time to learn the parameters for membership functions associated with the fuzzy system.

In the evolutionary controller population every string is uniquely representing the hierarchical structure of the fuzzy system as described in Section 4.3. For co-evolutionary algorithm a 3-layered HFS is selected that can be represented as a linear individual string of M = 25 + 35 + 35 = 95 consequents, $\vec{p}_k = (a_{1,\dots,}a_{95})$, where a_j is an integer ∊ [1,7] for j = 1, ... , 95.

The initial controller population $P(0) = \{ \vec{p}_k: k = 1, ... , M_p \}$, is determined by choosing the a_j as a random integer in [1,7], where $M_p = 500$ is the size of the evolutionary population.

As in the EA described in Chapter 4 and Chapter 5, a new controller population $P(t +1)$ is obtained from the old one by the use of genetic operators such as selection, crossover and mutation. Full replacement policy is implemented and requires that the population size remains constant from one generation to the next. Tournament selection with size $n_T = 4$ and a modified mutation operator is used.

A method to evaluate fitness function for individuals in both co-evolutionary populations needs to be determined because it cannot be done directly as in classic evolutionary algorithm with one population only. An individual from the controller population needs to be matched with an individual from the MF population in order to evaluate fitness function, and vice versa.

In the greedy version of the co-evolutionary algorithm each individual from the controller population is matched with the best individual of the MF population to evaluate the fitness function. Obviously, the best MF individual from the current population cannot be used (as their fitness values are not evaluated yet) but the best individual from the previous generation can be used instead.

The same process is repeated by matching the best individual from controller population to evaluate the fitness function of the individual from the MF population.

In the explorative version of the co-evolutionary algorithm a tournament selection is used: for any single individual from the controller population T_N individuals are selected at random (tournament selection) from the MF population. Then fitness function is evaluated for that controller chromosome using each of the selected T_N membership functions. The best fitness value is selected and assigned to the individual from the controller population.

The evaluation of the fitness function of a given string in the controller population is described below. The membership functions for evaluation are chosen from the MF population using either greedy or explorative selection method. The fitness f_k of a given string $\vec{p}_k$ in the controller population is evaluated as follows. Given an initial condition of the system each string $\vec{p}_k$ is decoded into three components defining the fuzzy knowledge base for each layer, then the Mamdani formula (see Equation 3.3) is used to evaluate u_1 and u_2 to find the final control u to be applied at each value of the state $\vec{x}$. Given an initial state the system state equations are integrated by the Runge-Kutta algorithm (RK4) with step size 0.02 over a time interval $[0,T]$. The fitness f_k is then calculated according to Equation 4.1 and 4.2, see Section 4.5.1. A penalty of 1000 is added to the objective function value if the final state leaves the designated TR.

Elitism policy is implemented with four copies of the ten top individuals in controller population passed to the next generation. Four copies of the best four individuals in MF population (sixteen copies altogether) are passed to the next generation.

Random crossover of two parent strings to form two children in the next generation is used. Mutation is undertaken with probability p_m whose value is determined by a mutation schedule that decreases from 0.8 to 0.001 over 300 generations.

```
if ( gen ≥ 0   & gen < 50  ) pm = 0.8
if ( gen ≥ 50  & gen < 100 ) pm = 0.7
if ( gen ≥ 100 & gen < 150 ) pm = 0.6
if ( gen ≥ 150 & gen < 200 ) pm = 0.3
if ( gen ≥ 200 & gen < 250 ) pm = 0.1
if ( gen ≥ 250 & gen < 300 ) pm = 0.01
if ( gen > 300) pm = 0.001
```

where `gen` denotes generation number.

Each individual in the membership functions population represents the possible MFs definition for the control system. For the CEA with random initial population output variable MF centres are included in a chromosome in MF population. As five MFs cover the range of each input variables and seven MFs cover output variables range only three centres are needed to define five input MFs and five centres to define output membership functions. Therefore, a required chromosome length is:

$$chromlMF = 3 + 3 + 3 + 3 + 5 = 17.$$

MF chromosome is a linear array of blocks of three numbers representing centres of MFs for each of four input variables and output variable. The first three numbers describe MFs for the first input variable (here x_3), next three numbers MFs for the second input variable (x_4), etc. For the CEA with uniformly generated initial population only input variables MF centres are included in a chromosome in the MF population:

$$chromlMF = 3 + 3 + 3 + 3 = 12.$$

Test simulations were run with full length of the MF chromosome for random and uniform initial MF population and it was found that the inclusion of output variable does not have a significant effect on results of the CEA with uniform MF population. Therefore, the CEA with randomly generated initial MF population is run with the full length MF chromosome, and the CEA with uniformly generated initial MF population is run with MF chromosome without MF centres for the output variable (fixed MF centres for the output variable).

The lower and upper boundaries for the MFs are defined below. For x_1 and x_2 the lower bound is $LBMFx_1x_2 = -2.0$ and upper bound is $UBMFx_1x_2 = 2.0$. For x_3 the lower bound is $LBMFx_3 = -\pi/2.0$ and upper bound is $UBMFx_3 = \pi/2.0$. For x_4 the lower bound is $LBMFx_4 = -4.0$ and upper bound is $UBMFx_4 = 4.0$. Finally, for output variable y (that can be u_1 , u_2, or u) the lower bound is $LBMFy = -15.0$ and upper bound is $UBMFy = 15.0$.

For random initial MF population membership function centres are generated by random number generator within bounds specified above for each input and output variable. For uniform initial MF population membership function centres are evenly spaced within intervals defined by lower and upper bounds for each variable, see Table 6.1 and Table 6.2. Uniform initial MF population consists of identical individuals (i.e., Gaussian functions evenly spread over the range of each input and output variables) and then diversified in evolutionary process by crossover and mutation in consecutive populations.

The initial MF population $PMF(0) = \{ \overrightarrow{pMF_k} : k = 1, \dots , MMF \}$, where MMF is the number of MF strings (the size of the MF population) and $\overrightarrow{pMF_k}$: $k = 1, \dots , MMF$ are MF chromosomes containing membership function centres.

The same random crossover operator is used for the MF population (adjusted for real numbers that encode MF chromosome) as for the controller population. Mutation for the MF population is similarly undertaken with a mutation schedule that decreases from 0.4 to 0.001 over 300 generations with typical form:

```
if ( gen ≥ 0   & gen < 50  ) pMFm = 0.4
if ( gen ≥ 50 & gen < 100  ) pMFm = 0.3
if ( gen ≥ 100 & gen < 150 ) pMFm = 0.2
if ( gen ≥ 150 & gen < 200 ) pMFm  = 0.1
if ( gen ≥ 200 & gen < 250 ) pMFm  = 0.05
if ( gen ≥ 250 & gen < 300 ) pMFm = 0.01
if ( gen > 300) pMFm  = 0.001
```

Mutation probability p_d for each MF in a chromosome is decided by a flip of the coin to introduce strong mutation rate to reinforce diversity of the MF population (Michalewicz 1994):

```
if (flip(0.5))
   pd = pMF_m;
 else
   pd = 0.995;
```

The MF mutation operator is defined by the following pseudo code:

```
mutate = flip( pmutation ) // Flip the biased coin
  if (mutate)
     nmutation = nmutation + 1
     poww = ((1.0 - pd) * (1.0 - pd))
     fact = 1.0 * (1.0 - power(MyRandom(), poww))
     if (flip(0.5))
         perturbation = fact * (lowerbound - alleleval)
     else
         perturbation = fact * (upperbound - alleleval)
     temp = alleleval + perturbation
   else
     temp = alleleval
```

where `MyRandom()` procedure generates a random number and `power(x,y)` calculates x^y.

The CEA is terminated at a fixed number of generations or allowed to continue until there is minimal change or no change to the string which has the best fitness. The fittest individual from each population is then taken as the best possible solution learnt by this algorithm.

6.4 Experimental setup

Ten simulations are run for each CEA version:

- Explorative with uniform MF initial population.
- Explorative with random MF initial population.
- Greedy with uniform MF initial population.
- Greedy with random MF initial population.

6.4.1 Initial condition

The initial state is: $\vec{x}_0 = (0.5, 0.0, 0.01, 0.0)$.

6.4.2 Initial population

The initial population has significant impact on the evolution of the knowledge base. Therefore a single run of the CEA resulting in a single controller should not be regarded as a sufficient representation of controllers for any particular topology. This is a reason each variant of the CEA is run ten times to produce a representative sample. If the CEA does not produce a relatively uniform population at the end of the algorithm very different controllers are developed for the same topology. One way of dealing with this problem is careful fine-tuning of the CEA parameters and large number of generations.

6.4.3 Population size

Controller population size is set at M_p = 200 and membership functions population at 50.

6.4.4 Termination condition

The algorithm is terminated after 500 generations. Usually, there is little or no change in the minimum value of the objective function in the following generations.

6.4.5 Fitness function

The fitness function positive weights are selected as: $\omega_1 = 3000$, $\omega_2 = 2000$, $\omega_3 = 0$, $\omega_4 = 0$, $\omega_5 = 5000$.

6.4.6 Membership functions

The shape of Gaussian membership functions is varied, see Equation 3.2, by decreasing the value of the stretching coefficient from d = 10.0, gradually to d = 0.01; d = 10.0, d = 5.0, d = 1.0, d = 0.1, to d = 0.01, which has the effect of widening the shape of the Gaussian curve. Simulations for all versions of the co-evolutionary algorithm are run to examine the effect of the wider MFs on the results.

6.5 Computer simulations

To illustrate the controller performance some of the best performing and some typical controllers from ten simulation results for every version of the co-evolutionary algorithm.

Inclusion of output variable MF centres in MF chromosomes does not have any significant effect on the resulting control system. The initial MF population has much more impact on the resultant control system. There is a significant difference in control system and MF centres distribution between the CEA that uses uniform and randomly generated initial MF population. The CEA starting from a uniform population generally produces better performing controllers. However, in most cases, in spite of their differences, the controllers perform in a very similar fashion, i.e., having similar control time history and the character of state variables convergence. The MFs centres resulting from the CEA with uniform MF population change very little from their original definitions and better results from such CEAs suggest that control system performs better with evenly spaced membership functions.

6.5.1 Uniform initial population

For the CEA that starts with uniform initial MF population the results are similar for most simulations. Controller performance is smooth and after initial jump in magnitude of control the control magnitude quickly approaches zero value.

The best explorative algorithm results with uniform initial MF population for topology *L-34-2-1* are shown in Figure 6.1 – Figure 6.4. Best results for controller with topology *L-34-1-2* are shown in Figure 6.5 – Figure 6.8. Best greedy algorithm results with uniform initial MF population for topology *L-34-2-1* are shown in Figure 6.9 – Figure 6.12. The best greedy algorithm results with uniform initial MF population for topology *L-34-1-2* are shown in Figure 6.13 – Figure 6.16.

6.5.2 Random initial population

Generally, for the explorative algorithm with random MF initial population, controller performance with topology *L3-34-2-1* is good. The best explorative results with random initial MF population are shown in: Figure 6.17 – Figure 6.20.

In the second and seventh simulation a smooth and regular convergence of state variables is achieved, which indicates high performance of the control system. Magnitude of control is very low after initial effort to stabilise the system - it settles very quickly to values close to the origin. In the sixth simulation a very quick convergence of state variables is achieved. Magnitude of control is low after initial effort to stabilise the system.

For the explorative algorithm with random MF initial population, the best results for the controller with topology *L3-34-1-2* are shown in Figure 6.21 – Figure 6.24. For the greedy algorithm with random MF initial population, the controller performance with topology *L3-34-2-1* is shown in Figure 6.25 – Figure 6.28. In general there is not much difference between results from the explorative and greedy versions of the CEA with random initial population algorithms. For the greedy algorithm with random MF initial population, topology *L3-34-1-2*, good controller performance is in Figure 6.29 – Figure 6.32.

6.5.3 Widening the shape of Gaussian membership functions

The shape of the membership functions has significant effect on the CEA performance. Wider Gaussian functions result in deterioration in the CEA performance with most simulations failing to converge for d = 0.1 and d = 0.01. Narrower MFs produce better results. A possible explanation can be found in (Wang 1997) where the author remarks that σ parameter (standard deviation) usually is found by trial-and-error method and that large σ can smooth out noisy data, while small σ can make the system as nonlinear as is required to approximate closely the training data. Indeed, by taking a bigger value of stretching coefficient d the value of standard deviation σ decreases.

6.5.4 Non-overlapping Gaussian membership functions

The effect of non-overlapping MFs on the controller performance is examined in a series of experiments. Each MF is cut at the ends of the interval it is defined on and has zero values outside that interval (no overlap between membership functions). For MFs defined in such a way the simulations results with initial uniform MF population are very poor.

Apparently a combination of this kind of initial population and cut-off Gaussian membership functions had a dramatic effect on the controller performance. In many cases the CEA fails to converge to a desired solution, i.e., effective controller.

The simulations with randomly generated initial populations produce much better results indicating that the cut-off MFs do not have much effect on the CEA performance. The results are similar to the ones achieved with overlapping membership functions.

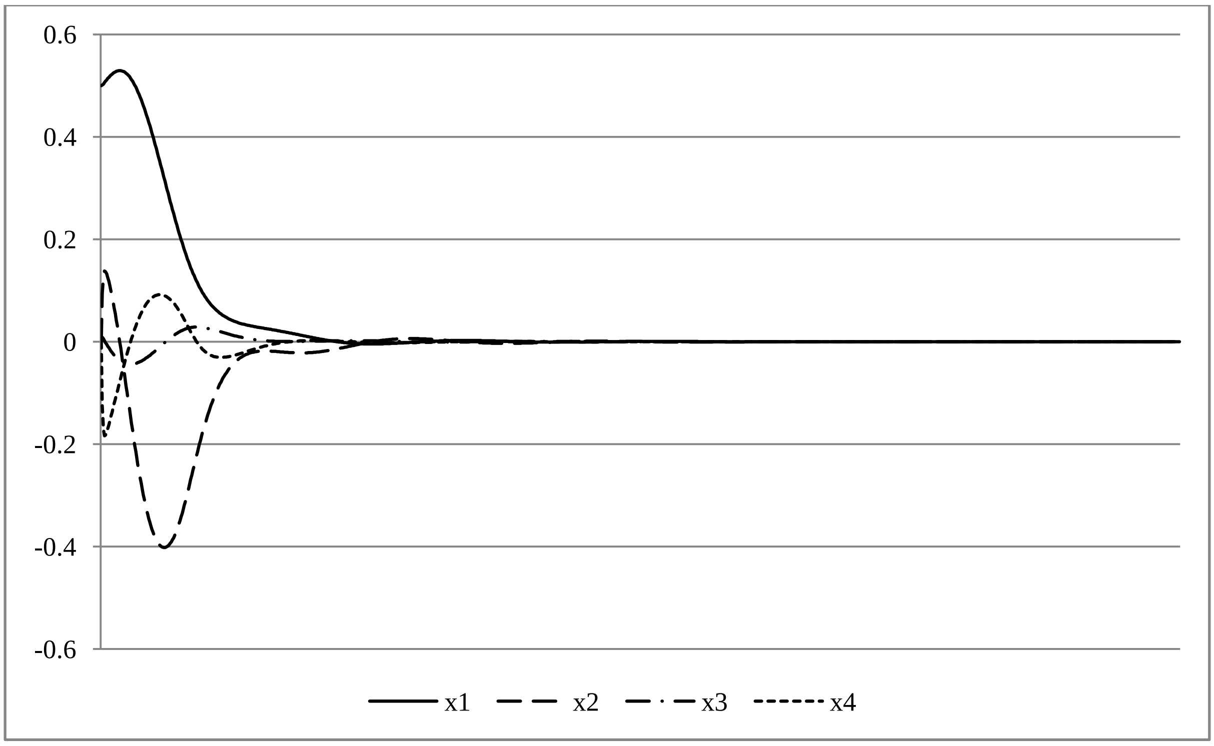

Figure 6.1 Explorative, uniform population, topology L3-34-2-1, simulation 4, state variables x_k, $k = 1, \ldots, 4$.

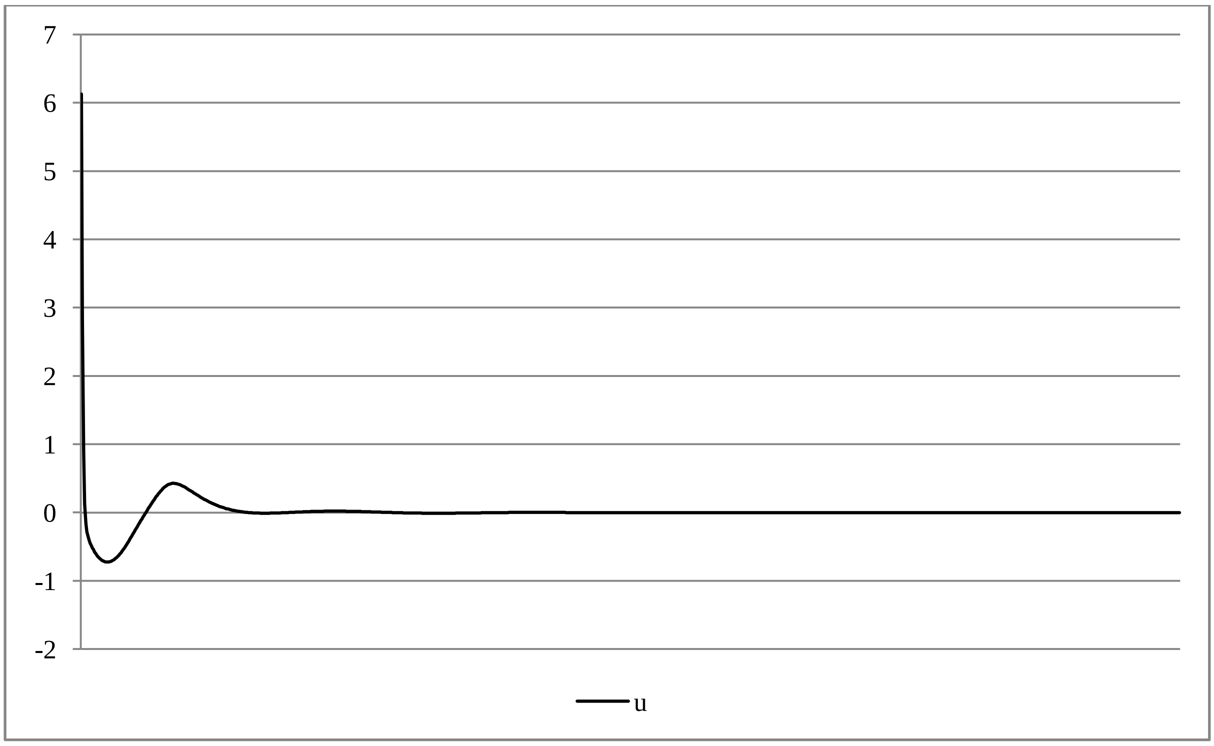

Figure 6.2 Explorative, uniform population, topology L3-34-2-1, simulation 4, control u.

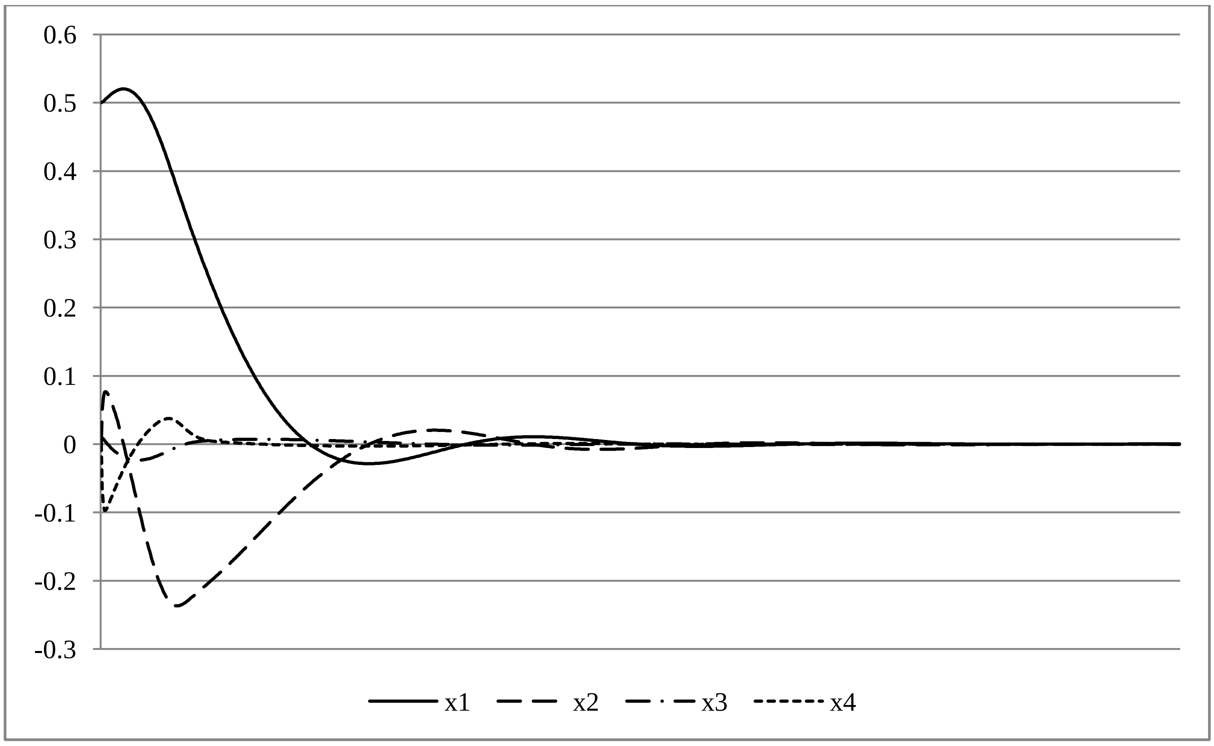

Figure 6.3 Explorative, uniform population, topology L3-34-2-1, simulation 8, state variables x_k, k = 1, … , 4.

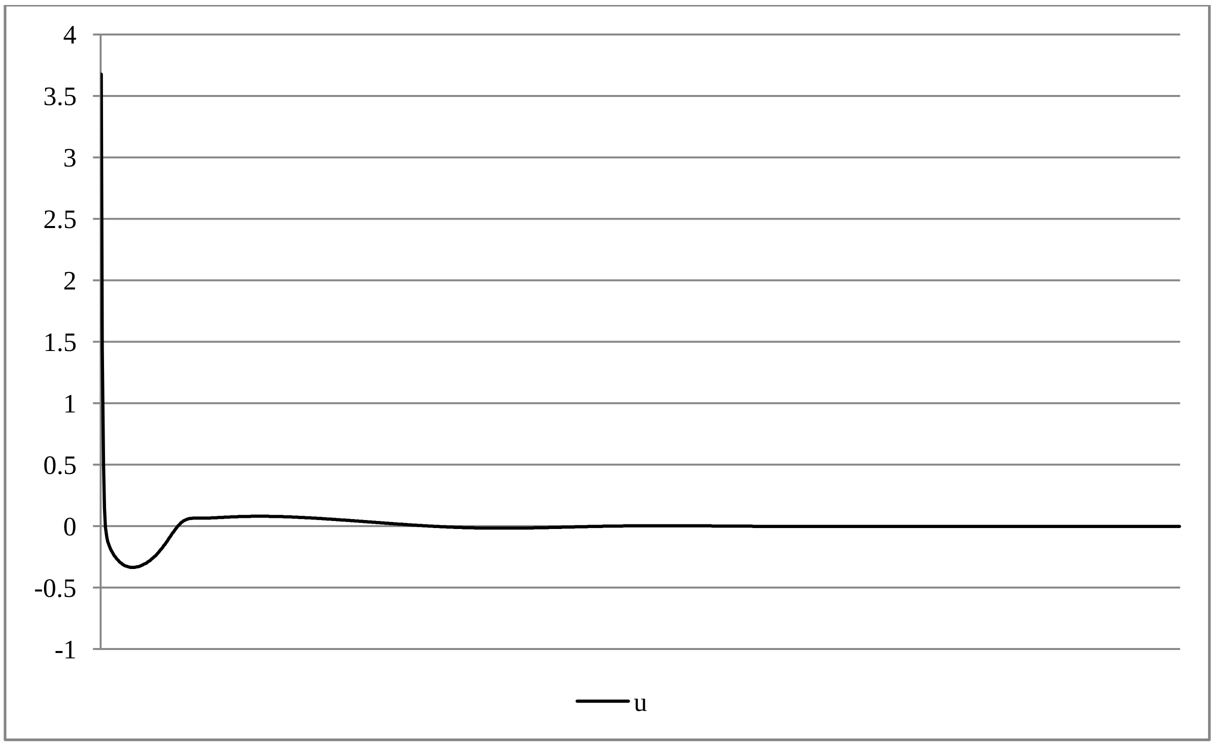

Figure 6.4 Explorative, uniform population, topology L3-34-2-1, simulation 8, control u.

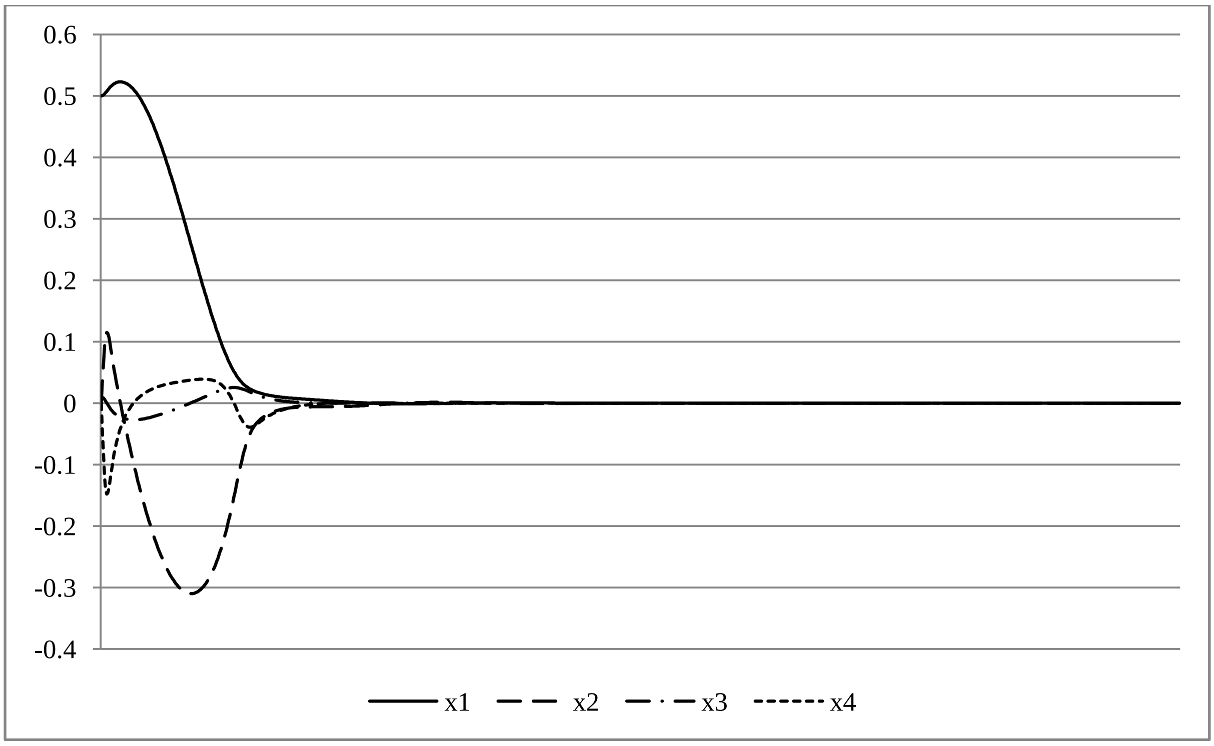

Figure 6.5 Explorative, uniform population, topology L3-34-1-2, simulation 5, state variables x_k, $k = 1, \ldots , 4$.

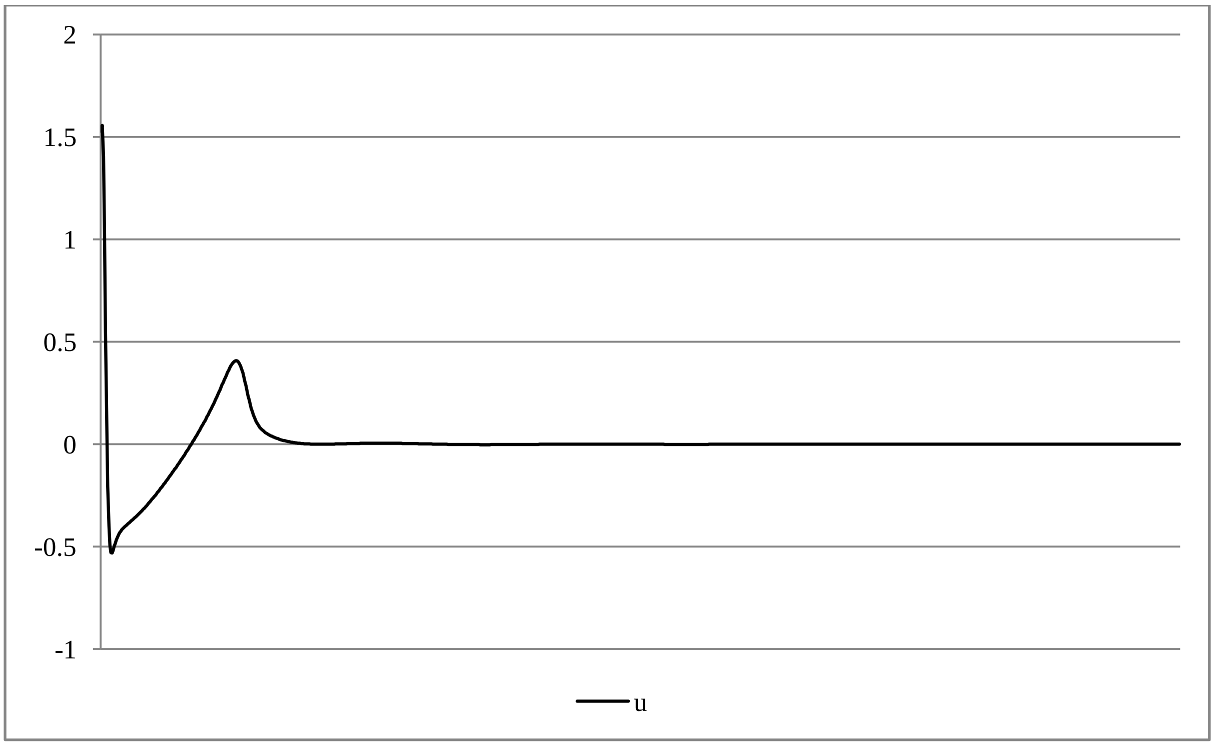

Figure 6.6 Explorative, uniform population, topology L3-34-1-2, simulation 5, control u.

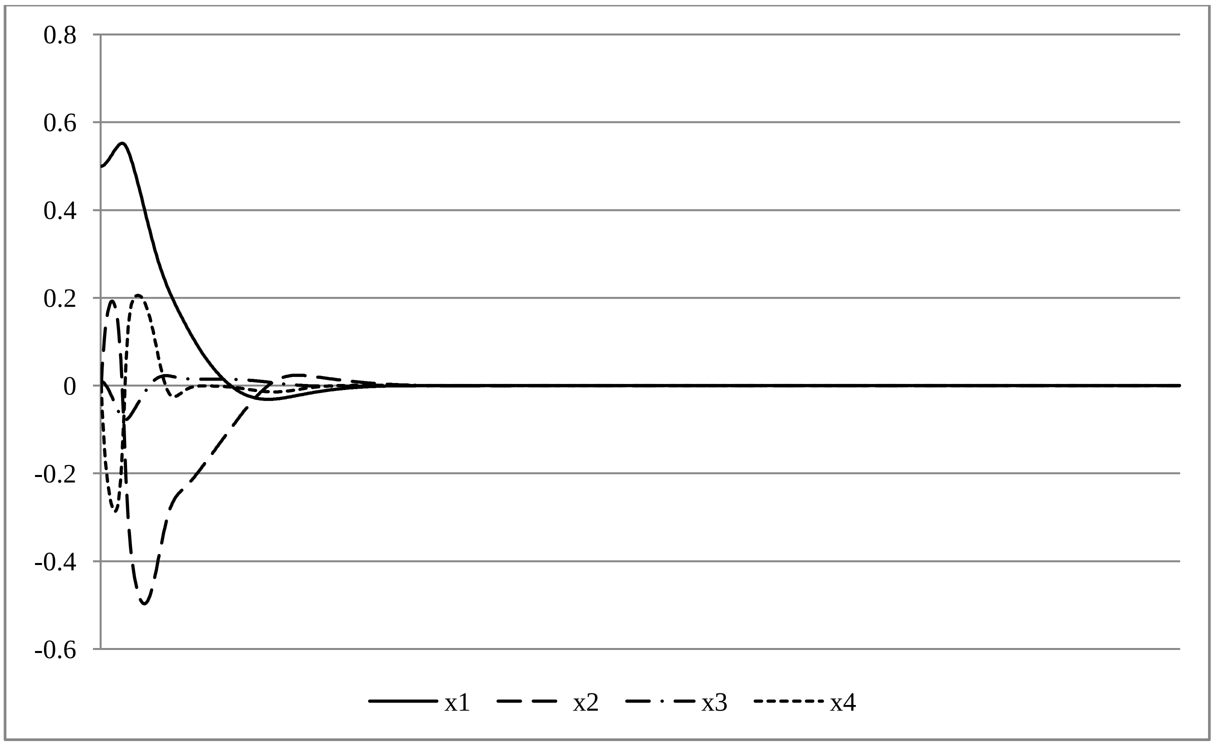

Figure 6.7 Explorative, uniform population, topology L3-34-1-2, simulation 10, state variables x_k, $k = 1, \dots, 4$.

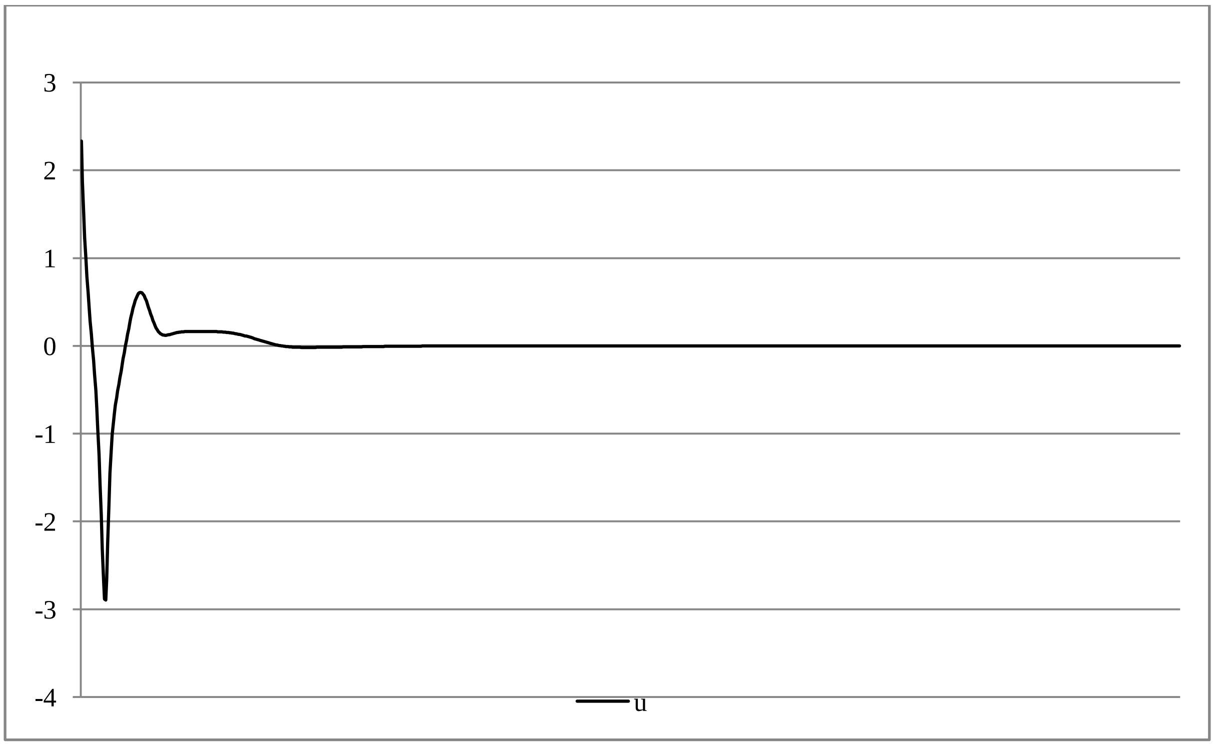

Figure 6.8 Explorative, uniform population, topology L3-34-1-2, simulation 10, control u.

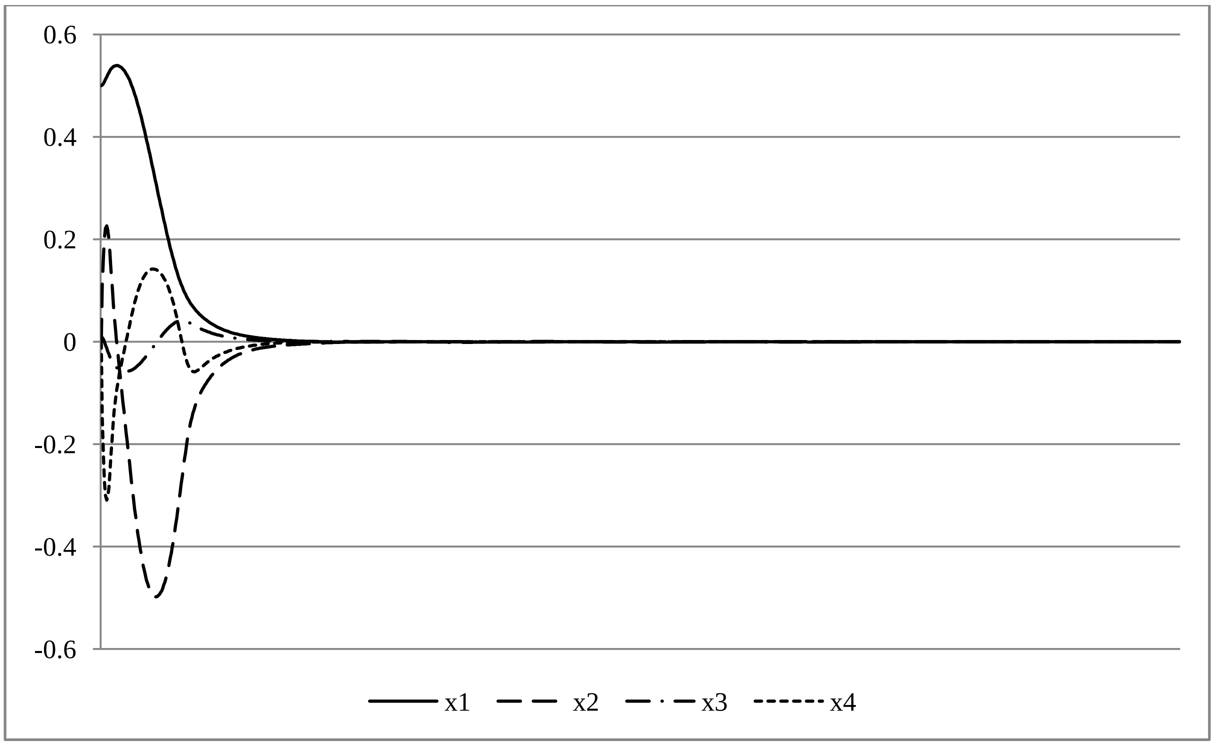

Figure 6.9 Greedy, uniform population, topology L3-34-2-1, simulation 4, state variables x_k, $k = 1, \ldots, 4$.

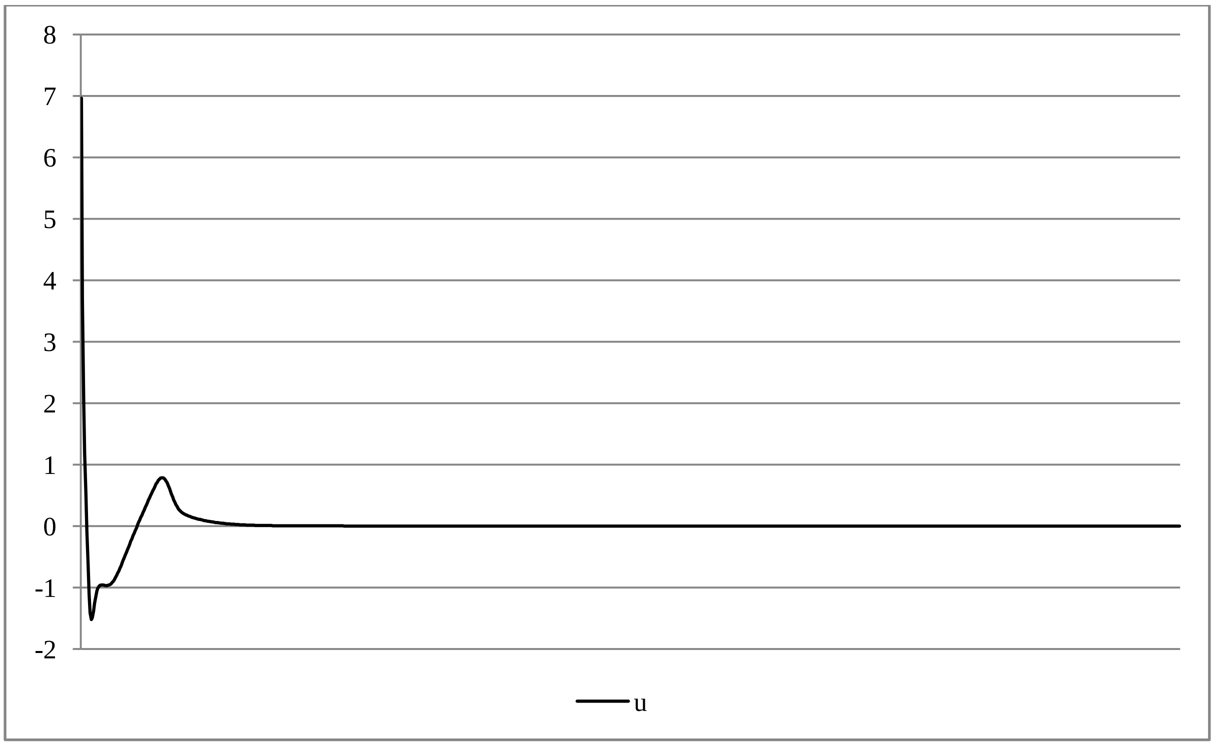

Figure 6.10 Greedy, uniform population, topology L3-34-2-1, simulation 4, control u.

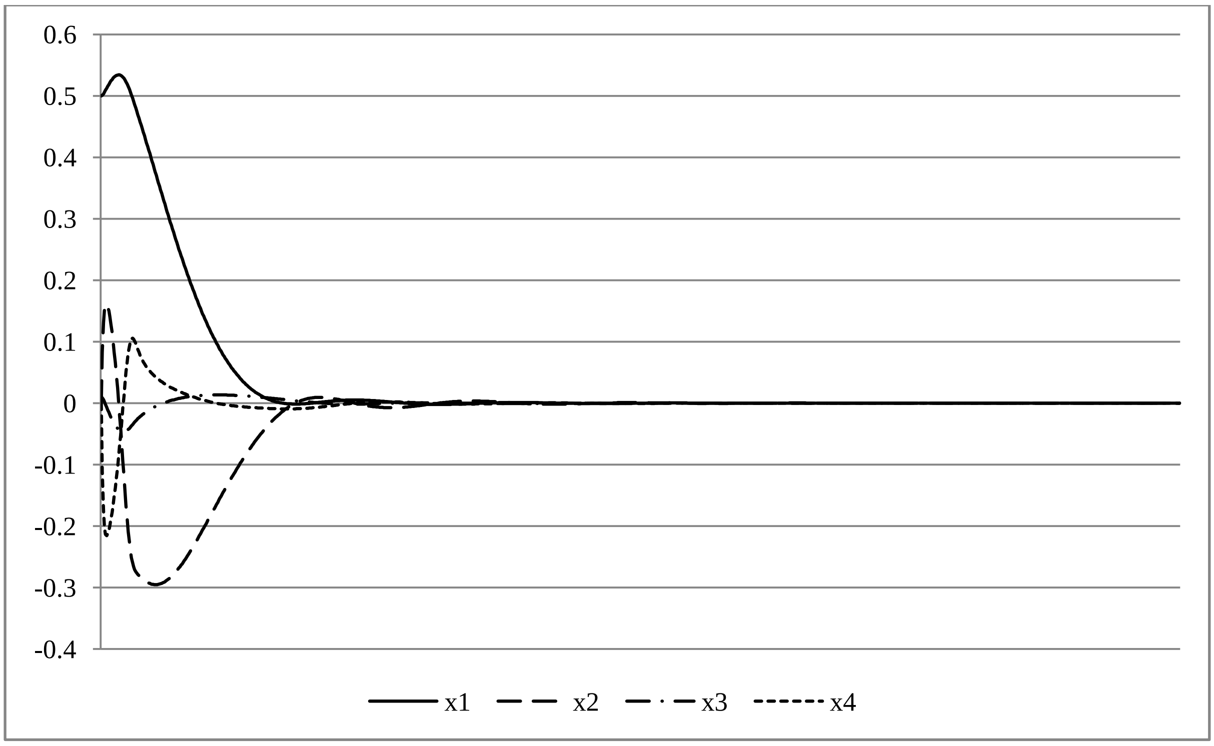

Figure 6.11 Greedy, uniform population, topology L3-34-2-1, simulation 8, state variables x_k, $k = 1, \ldots, 4$.

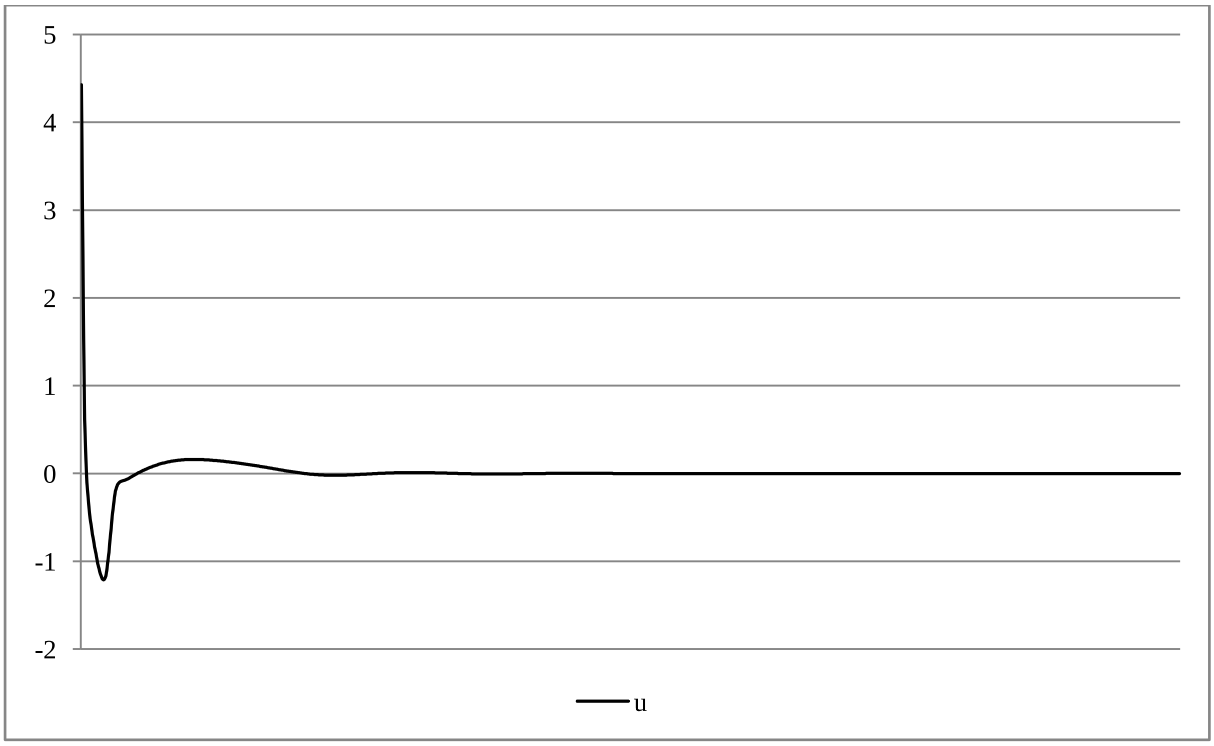

Figure 6.12 Greedy, uniform population, topology L3-34-2-1, simulation 8, control u.

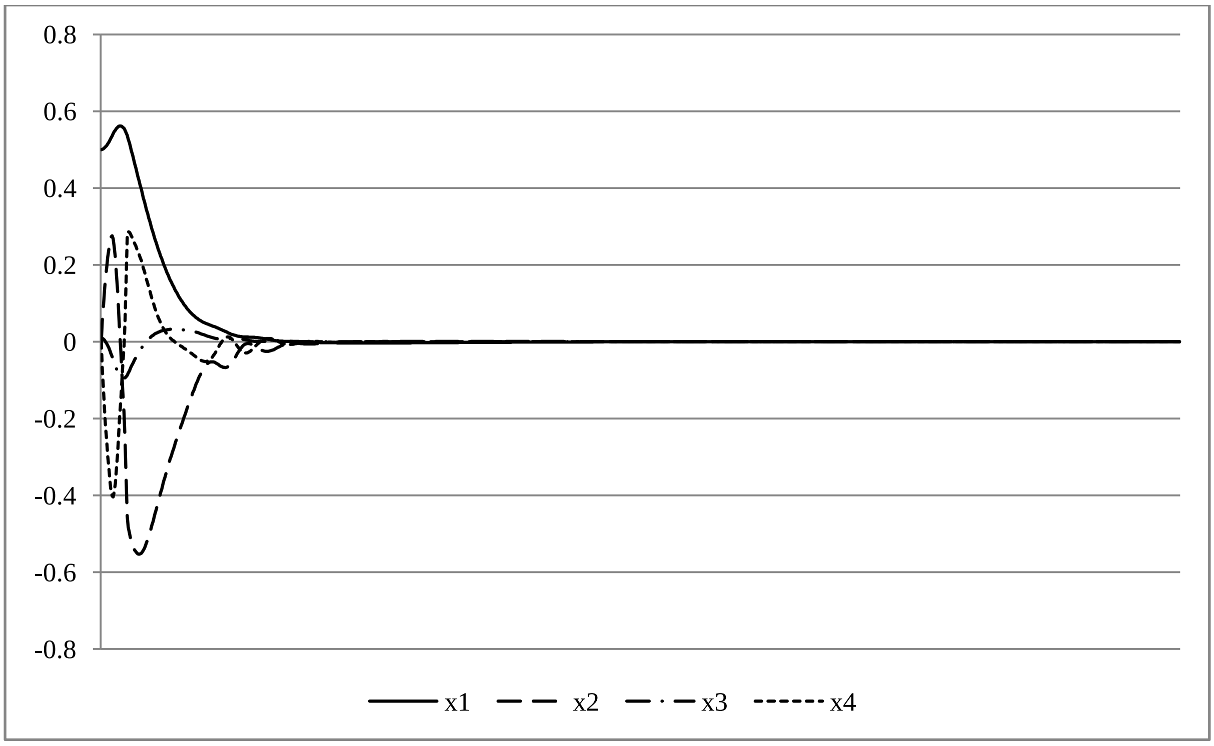

Figure 6.13 Greedy, uniform population, topology L3-34-1-2, simulation 3, state variables x_k, $k = 1, \dots , 4$.

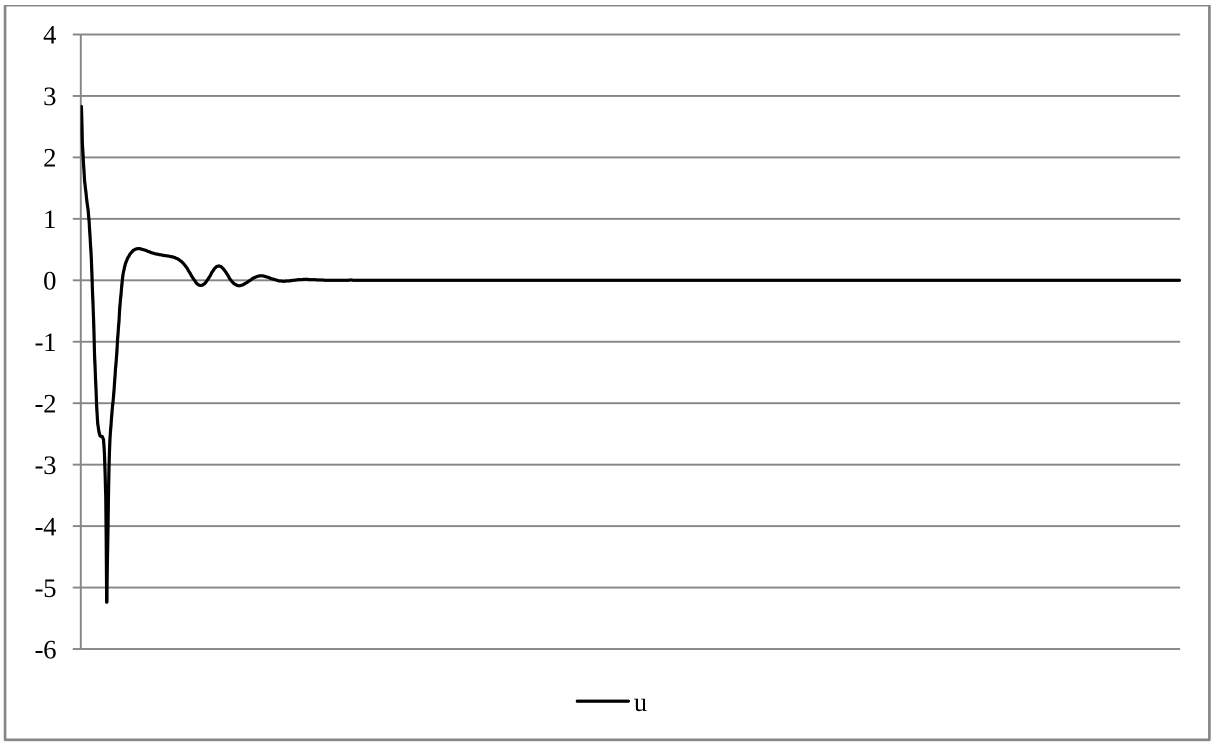

Figure 6.14 Greedy, uniform population, topology L3-34-1-2, simulation 3, control u.

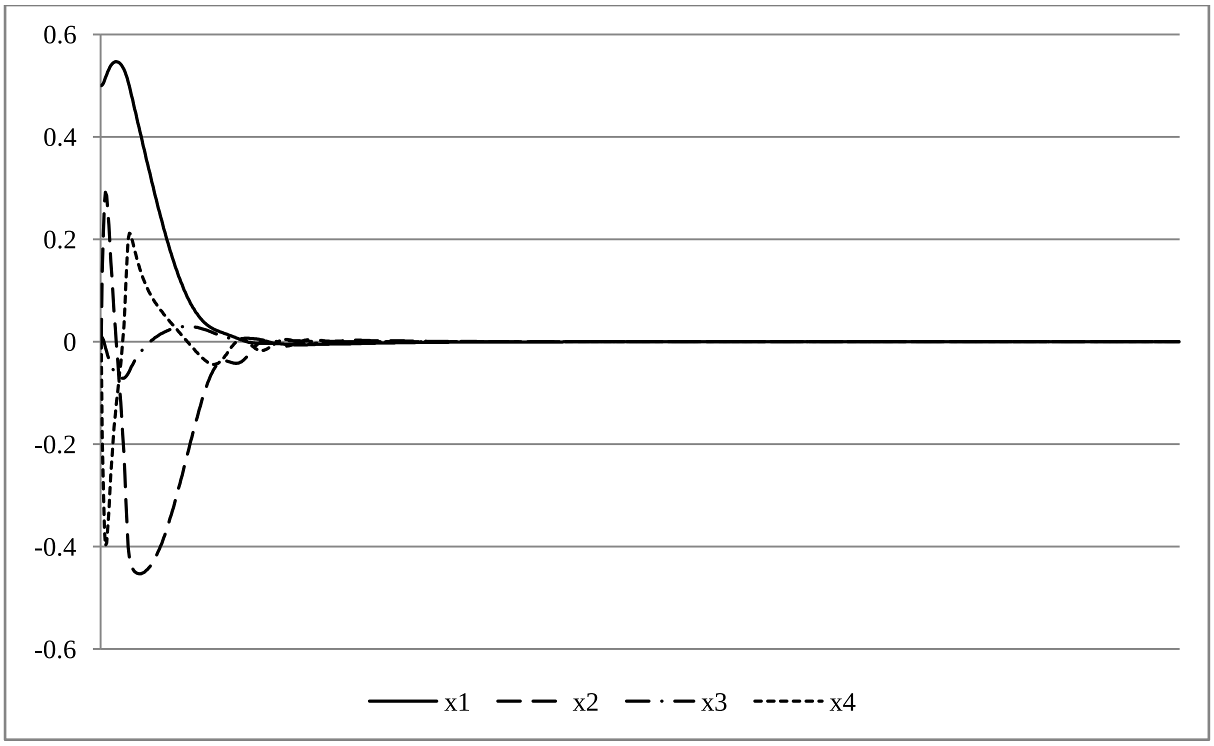

Figure 6.15 Greedy, uniform population, topology L3-34-1-2, simulation 10, state variables x_k, $k = 1, \ldots, 4$.

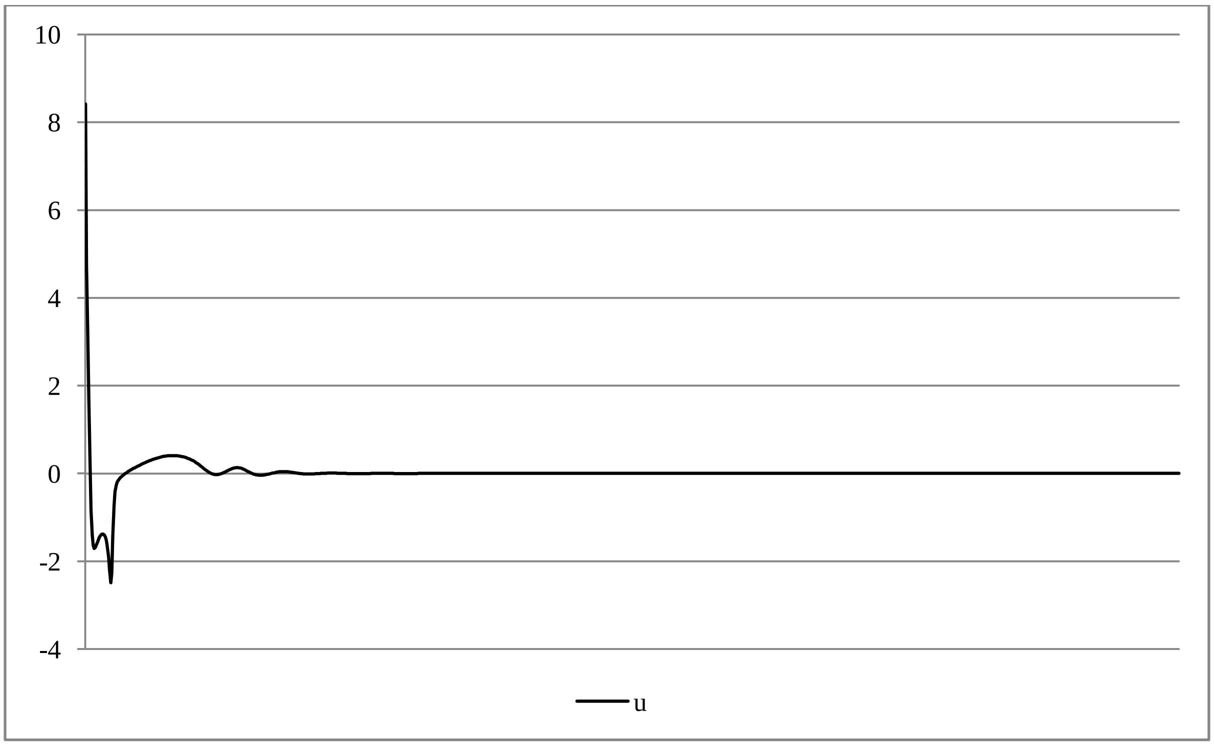

Figure 6.16 Greedy, uniform population, topology L3-34-1-2, simulation 10, control u.

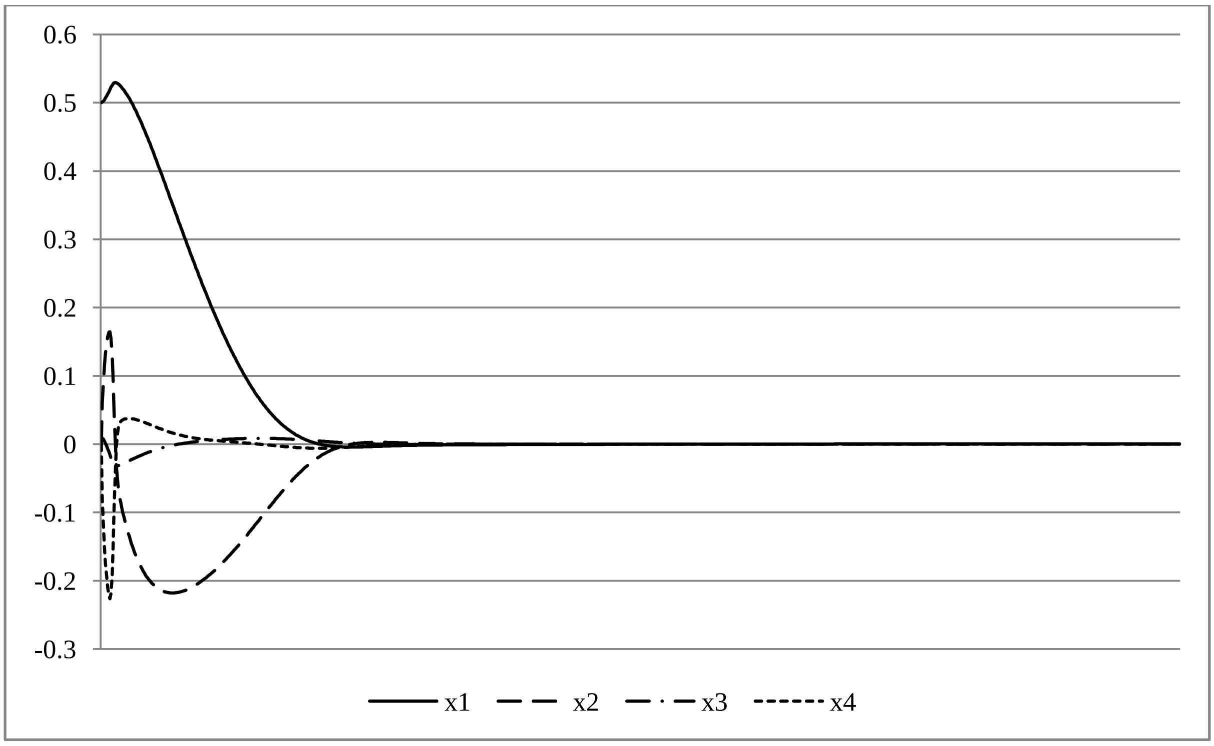

Figure 6.17 Explorative, random population, topology L3-34-2-1, simulation 2, state variables x_k, $k = 1, \dots , 4$.

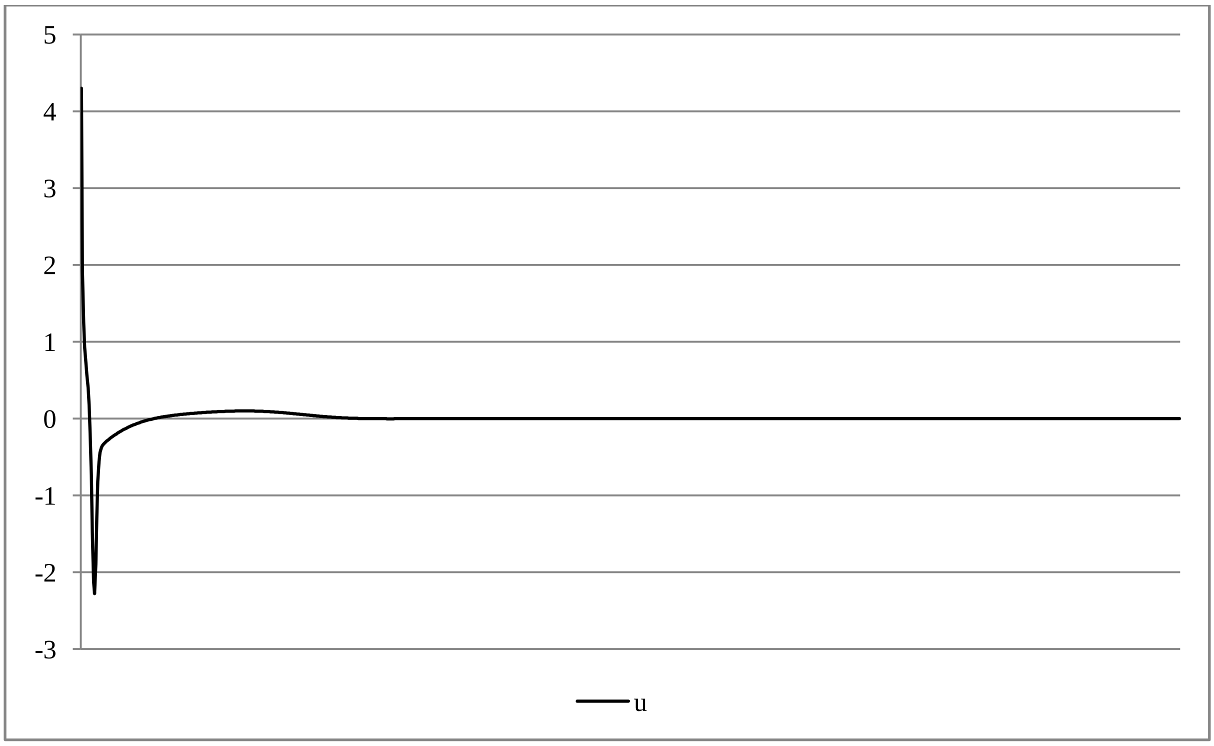

Figure 6.18 Explorative, random population, topology L3-34-2-1, simulation 2, control u.

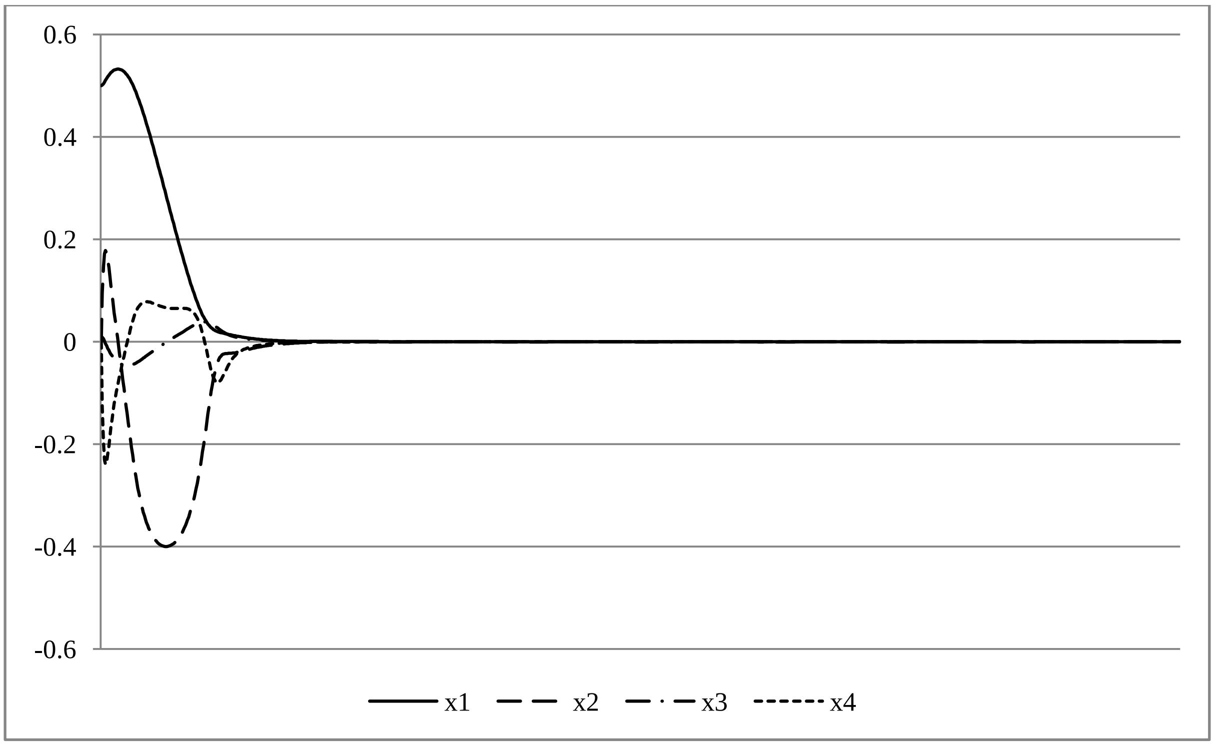

Figure 6.19 Explorative, random population, topology L3-34-2-1, simulation 7, state variables x_k*, k = 1, … , 4.*

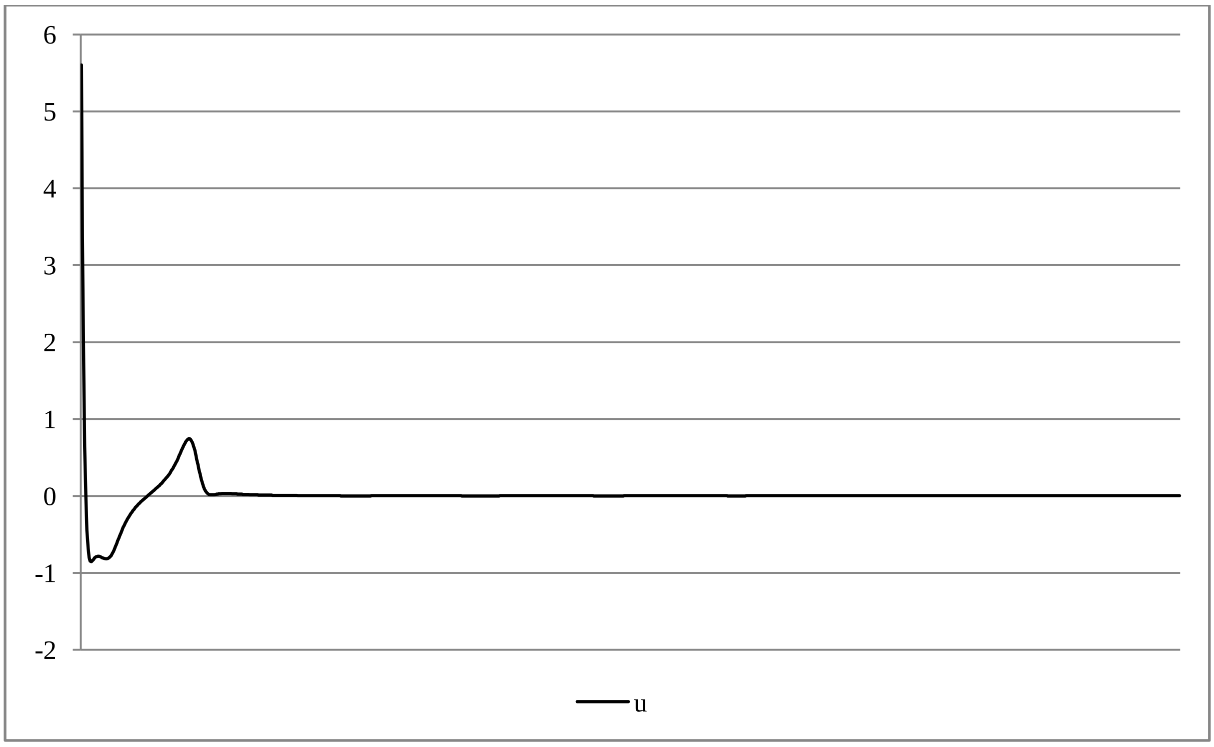

Figure 6.20 Explorative, random population, topology L3-34-2-1, simulation 7, control u.

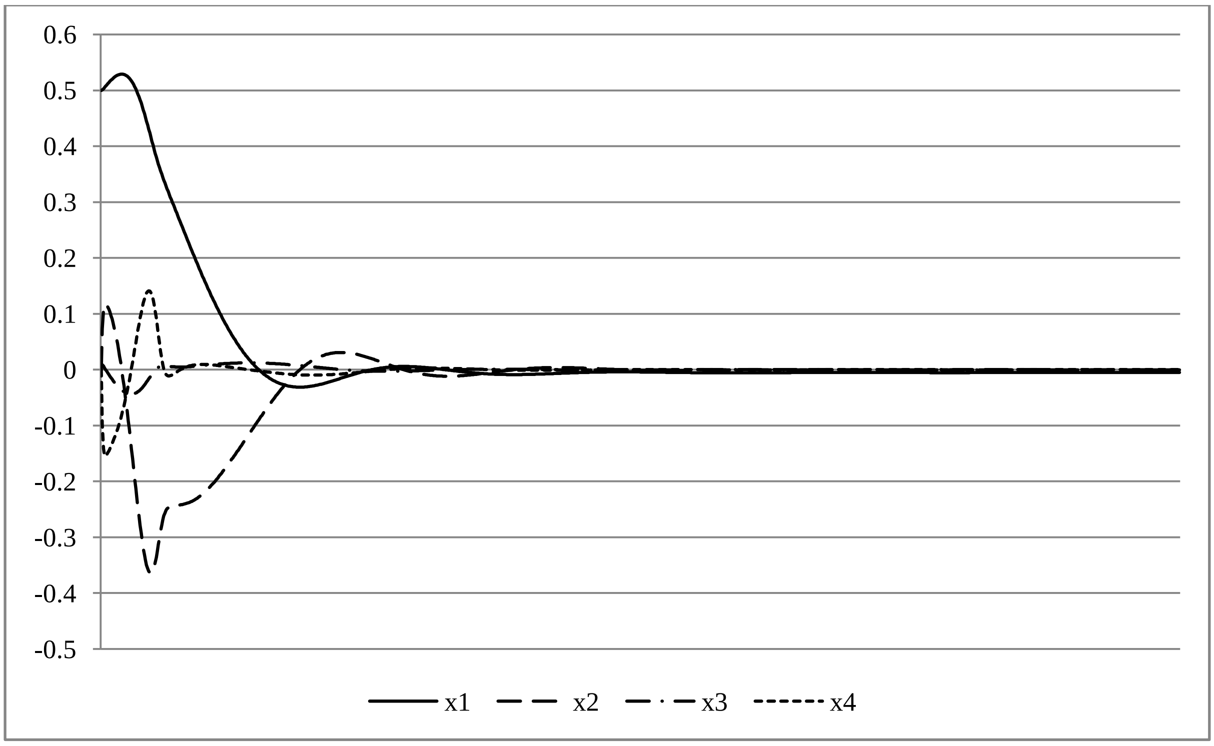

Figure 6.21 Explorative, random population, topology L3-34-1-2, simulation 5, state variables x_k, $k = 1, \ldots , 4$.

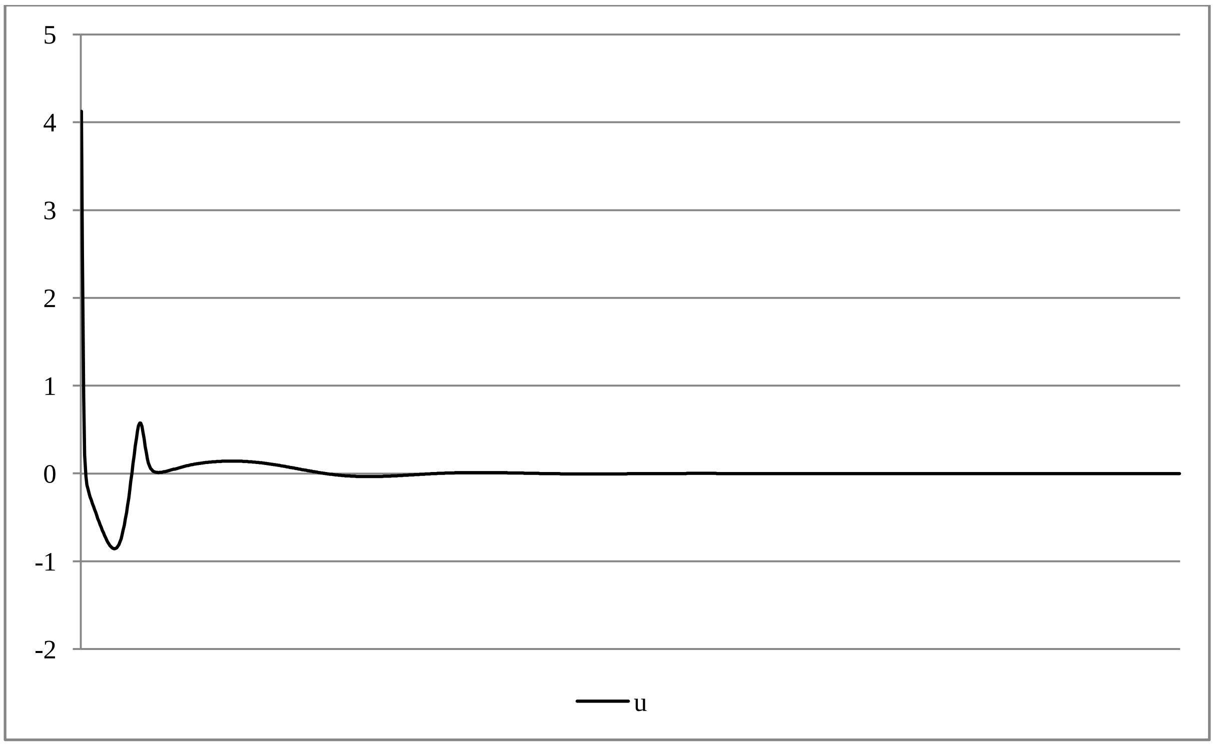

Figure 6.22 Explorative, random population, topology L3-34-1-2, simulation 5, control u.

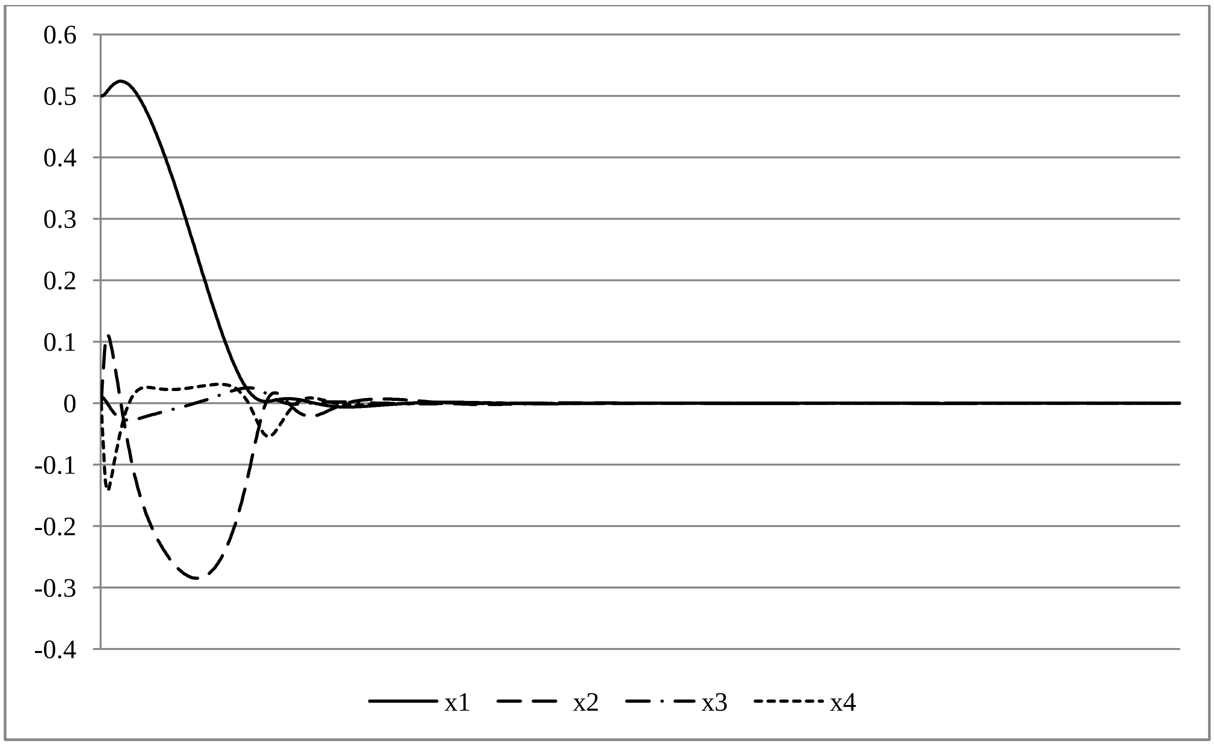

Figure 6.23 Explorative, random population, topology L3-34-1-2, simulation 8, state variables x_k, $k = 1, \ldots, 4$.

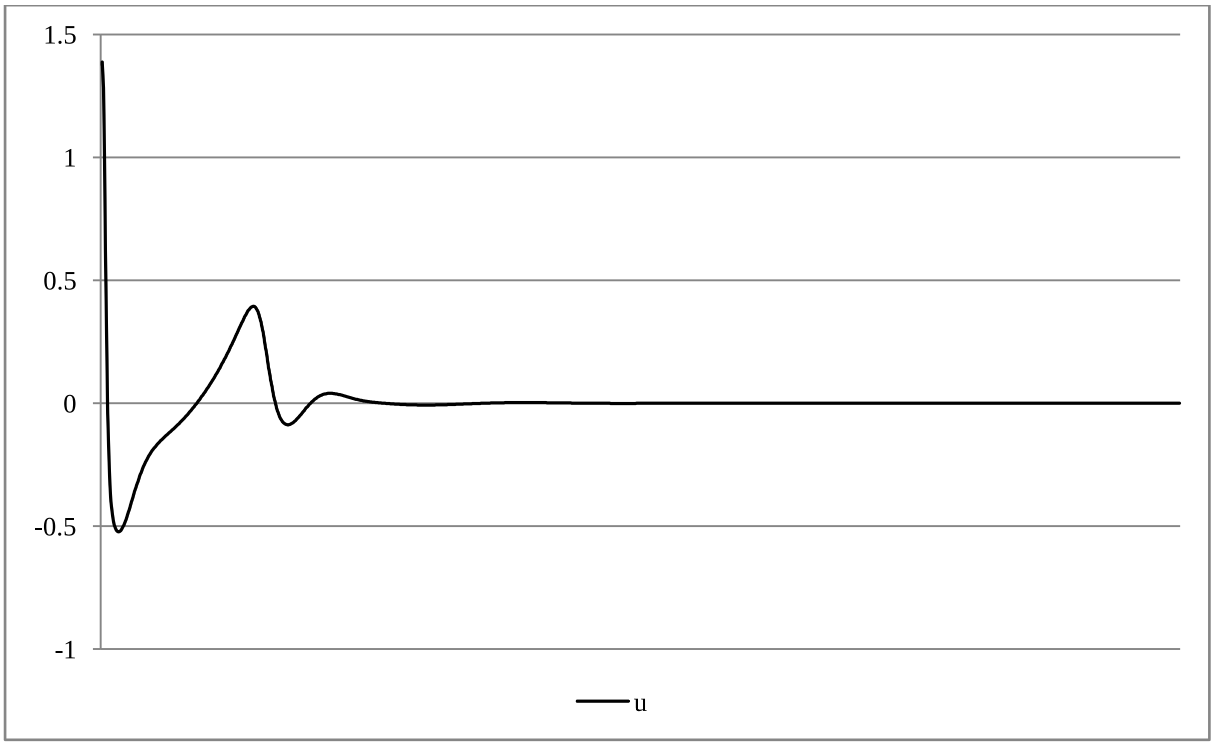

Figure 6.24 Explorative, random population, topology L3-34-1-2, simulation 8, control u.

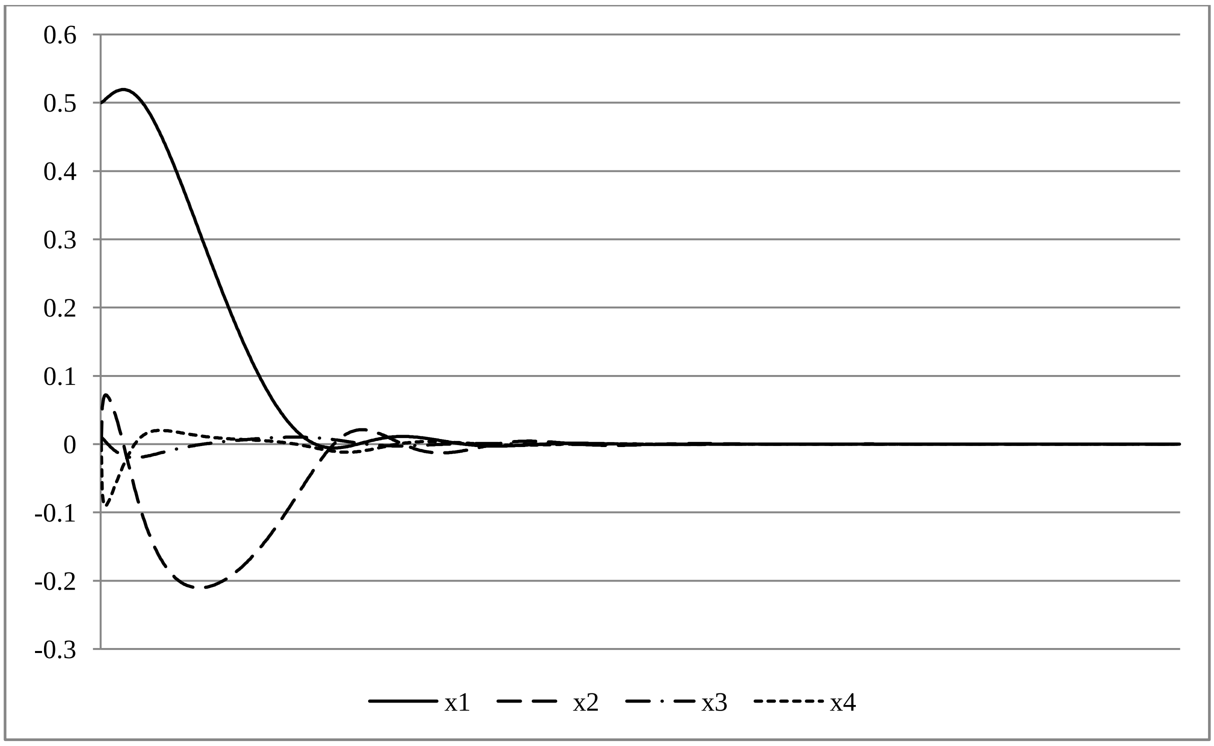

Figure 6.25 Greedy, random population, topology L3-34-2-1, simulation 2, state variables x_k, $k = 1, \dots , 4$.

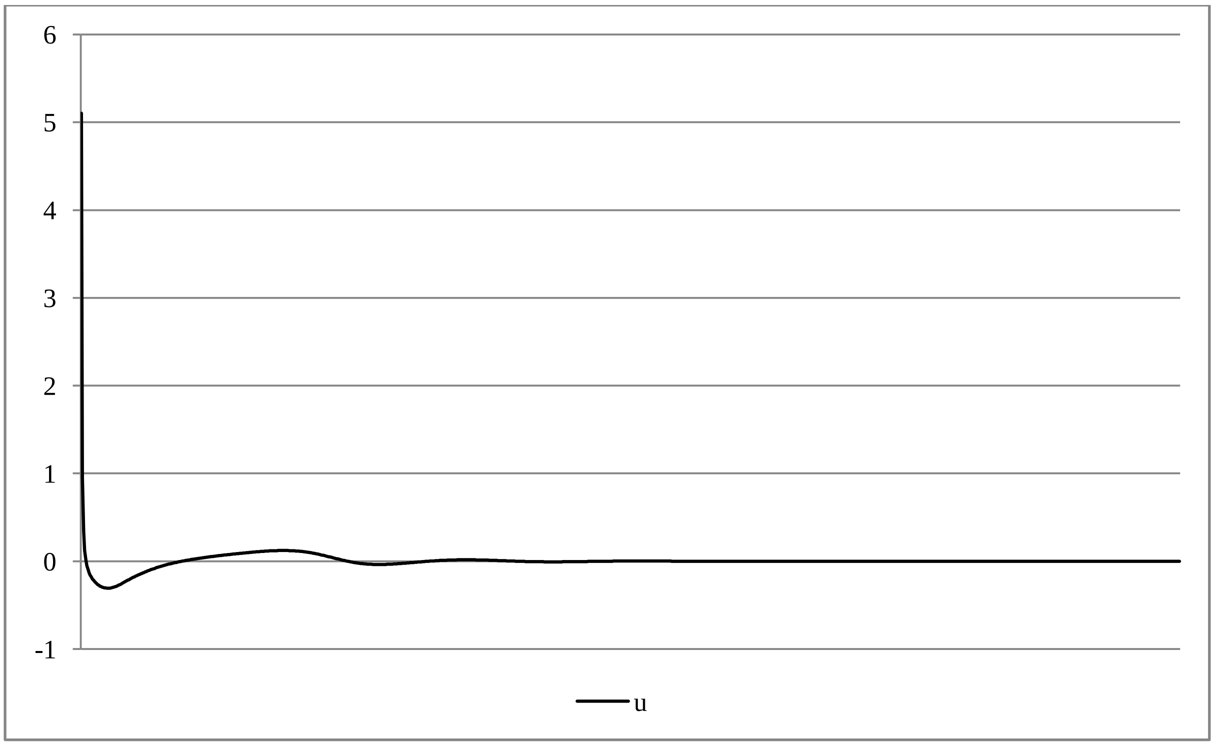

Figure 6.26 Greedy, random population, topology L3-34-2-1, simulation 2, control u.

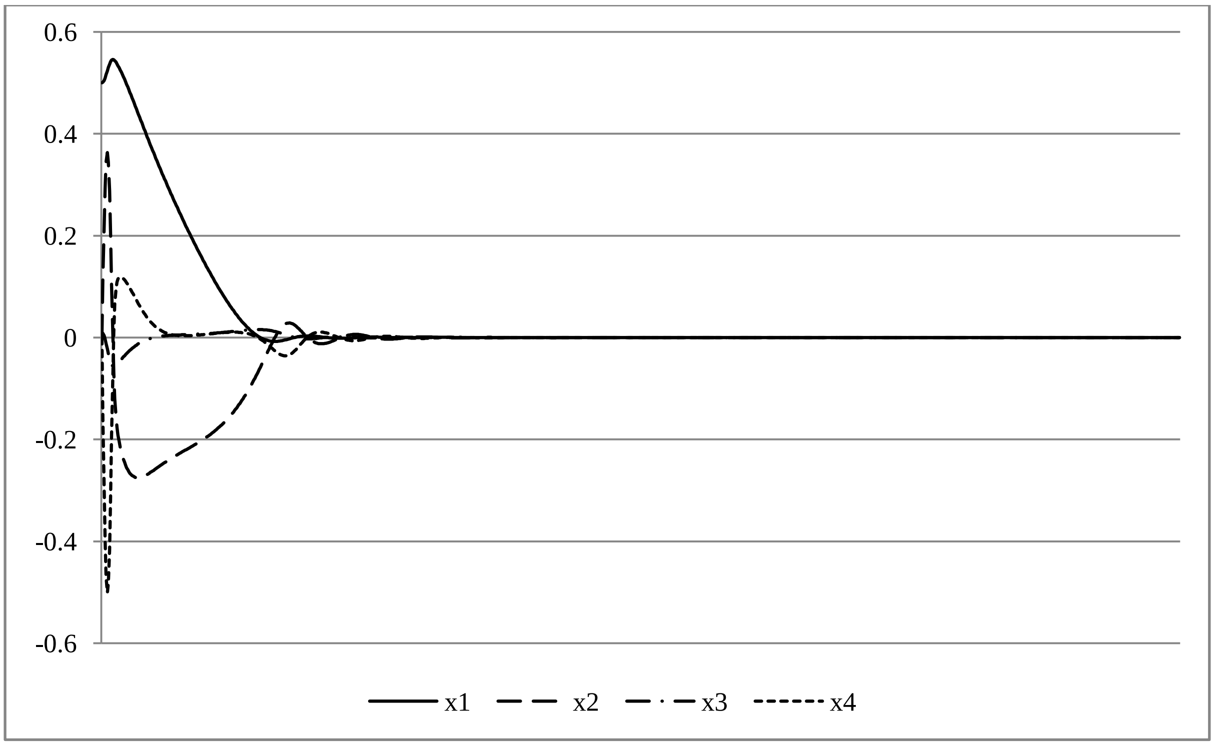

Figure 6.27 Greedy, random population, topology L3-34-2-1, simulation 3, state variables x_k, $k = 1, \ldots, 4$.

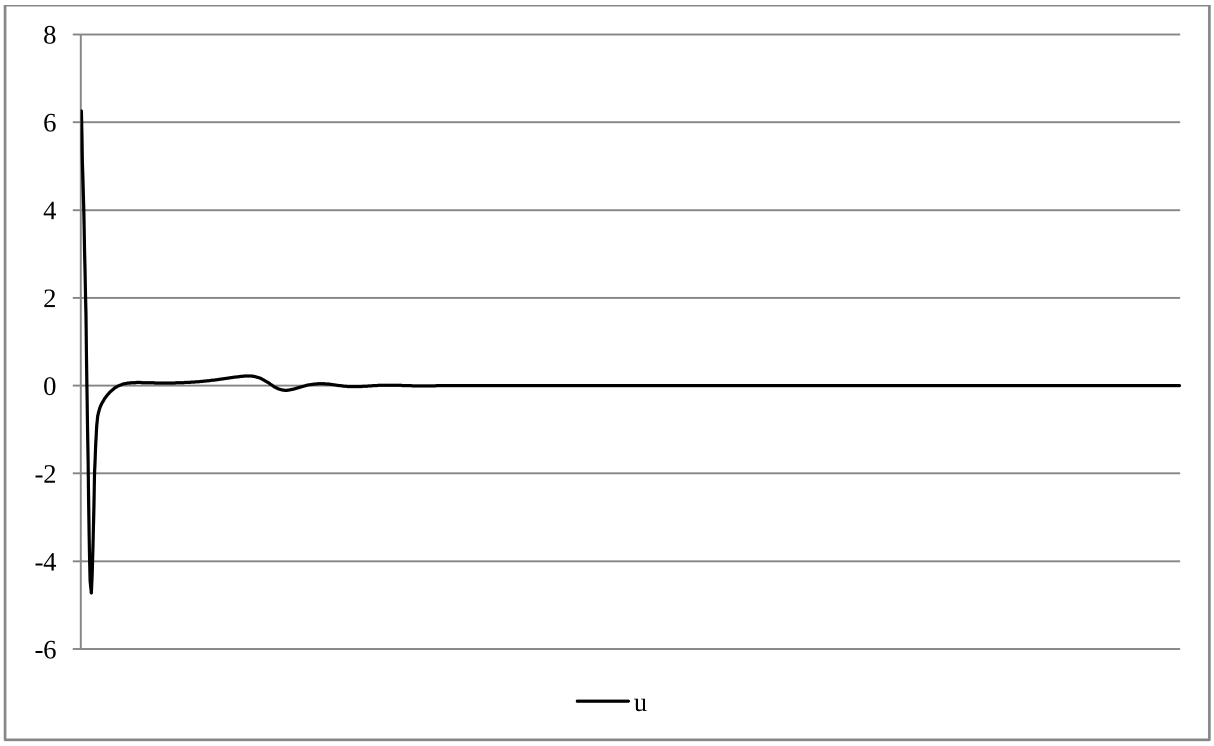

Figure 6.28 Greedy, random population, topology L3-34-2-1, simulation 3, control u.

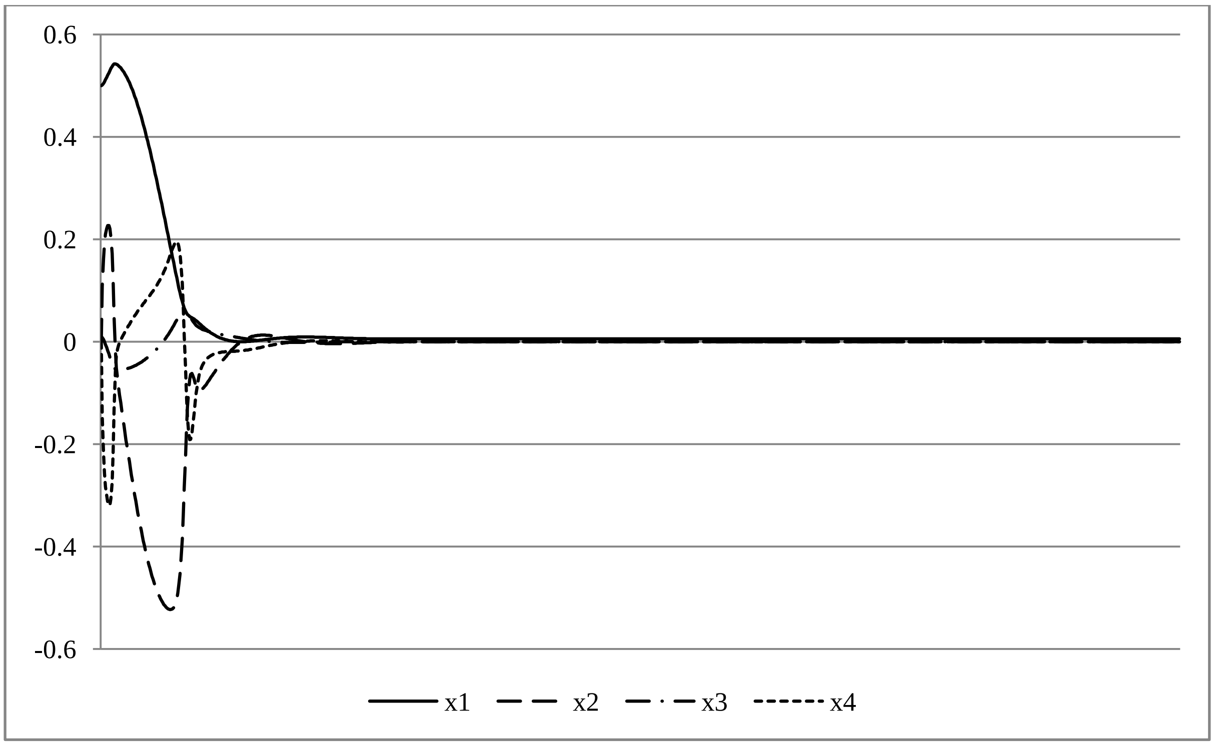

Figure 6.29 Greedy, random population, topology L3-34-1-2, simulation 3, state variables x_k, $k = 1, \dots , 4$.

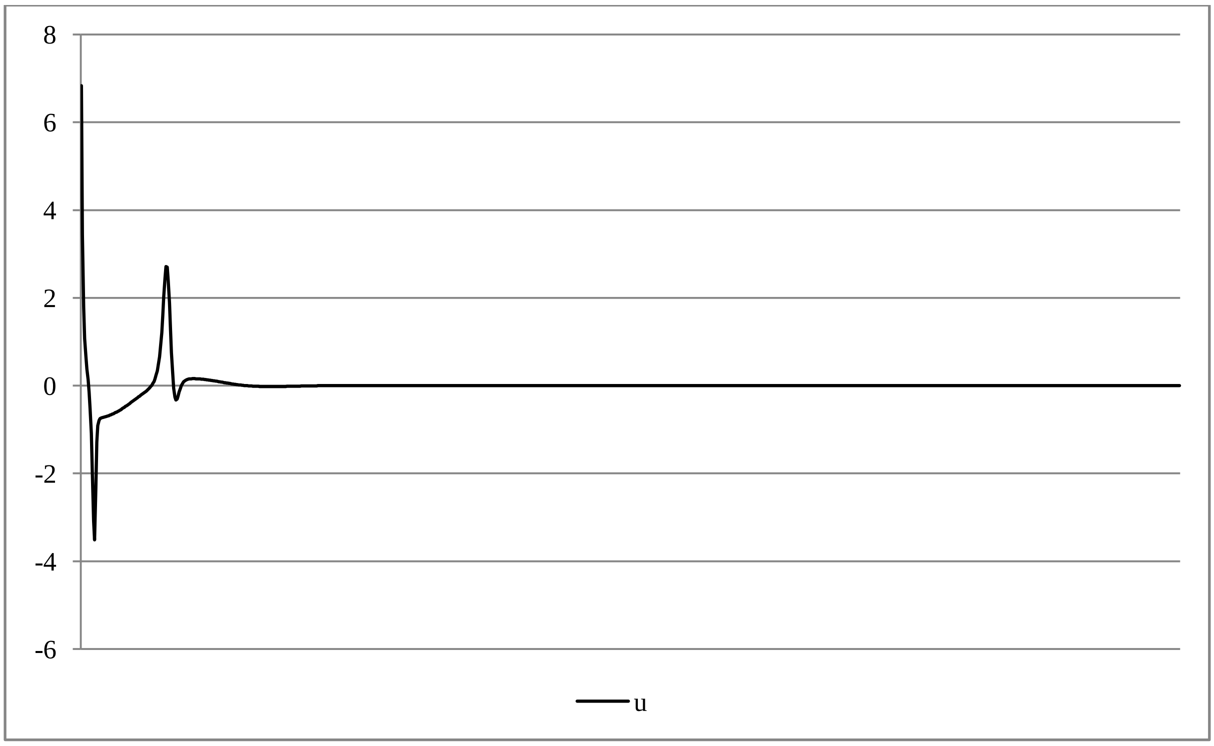

Figure 6.30 Greedy, random population, topology L3-34-1-2, simulation 3, control u.

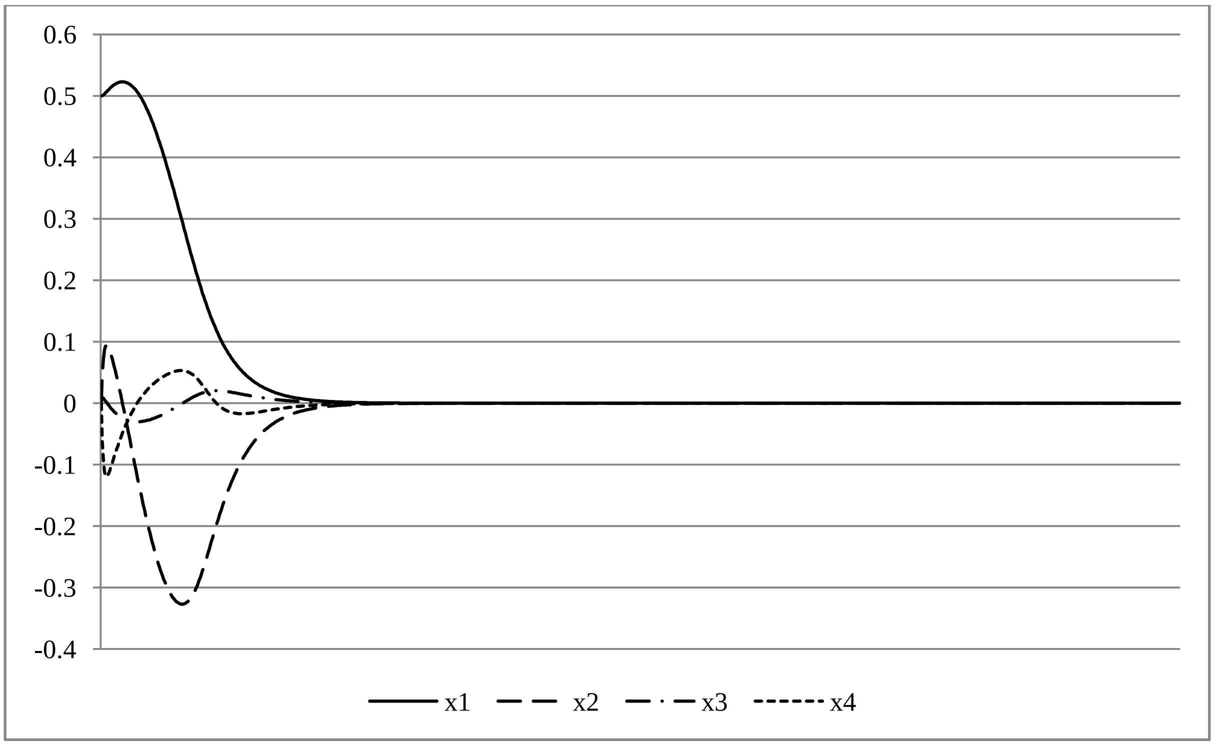

Figure 6.31 Greedy, random population, topology L3-34-1-2, simulation 5, state variables x_k, $k = 1, \ldots , 4$.

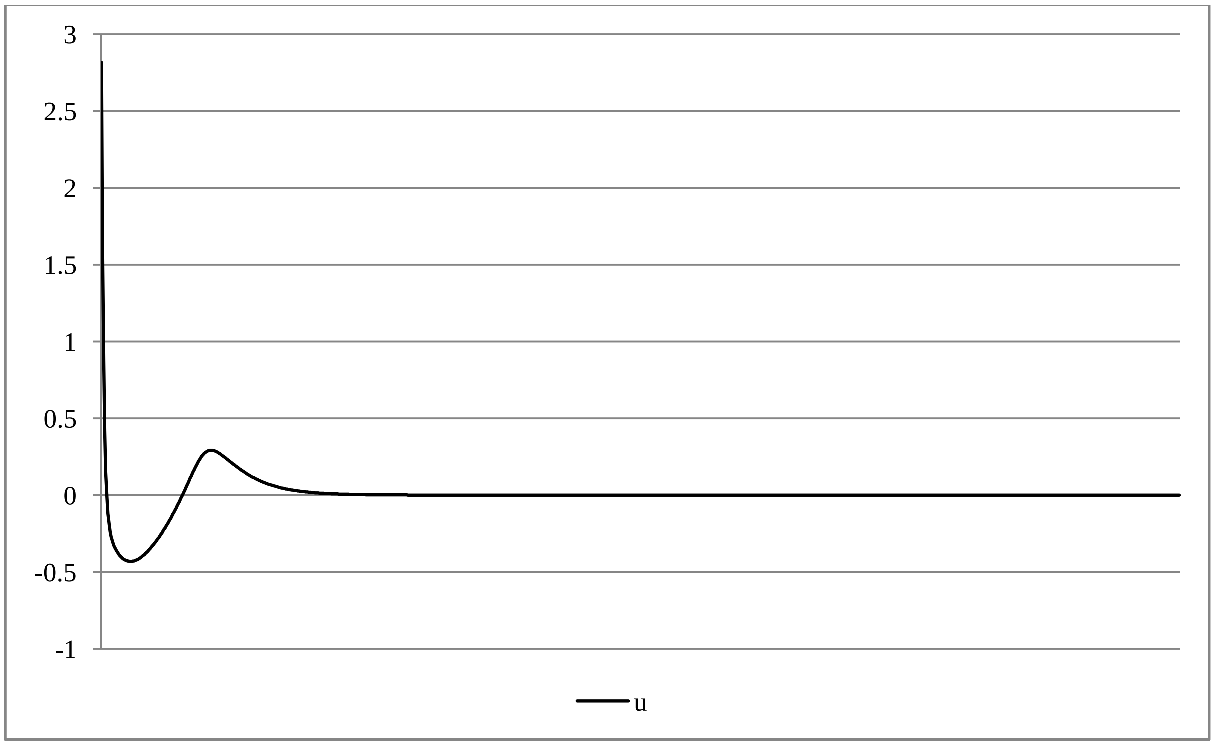

Figure 6.32 Greedy, random population, topology L3-34-1-2, simulation 5, control u.

6.5.5 Gaussian membership functions: computer simulations

Membership functions centres for uniform initial MF population are given in Table 6.1 and Table 6.2. These values are chosen arbitrarily and can be compared to the MF centres generated by the CEA in selected simulations.

Table 6.1 MF centres for uniform initial MF population: input variables.

MF	c1	c2	c3	c4	c5
x_1	-2.0	-1.0	0.0	1.0	2.0
x_2	-2.0	-1.0	0.0	1.0	2.0
x_3	-1.5707963	-0.78539815	0.0	0.78539815	1.5707963
x_4	-4.0	-2.0	0.0	2.0	4.0

where c1,c2,..., c5 denote centres of the membership functions.

Table 6.2 MF centres for uniform initial MF population: output variable.

MF	c1	c2	c3	c4	c5	c6	c7
y	-15.0	-10.0	-5.0	0.0	5.0	10.0	15.0

In the following tables membership functions centres are shown as found by the CEA. The major differences are found between results of CEA with uniform and CEA with randomly generated initial MF population.

Table 6.3 MF centres for L-34-2-1 explorative, uniform initial MF population: sim. no 4.

MF	c1	c2	c3	c4	c5
x_1	-2.0	-0.999987	-0.349197	0.999976	2.0
x_2	-2.0	-0.755659	0.000054	0.999914	2.0
x_3	-1.5707963	-0.785355	-0.000025	0.785264	1.5707963
x_4	-4.0	-2.000062	-0.000284	1.965566	4.0

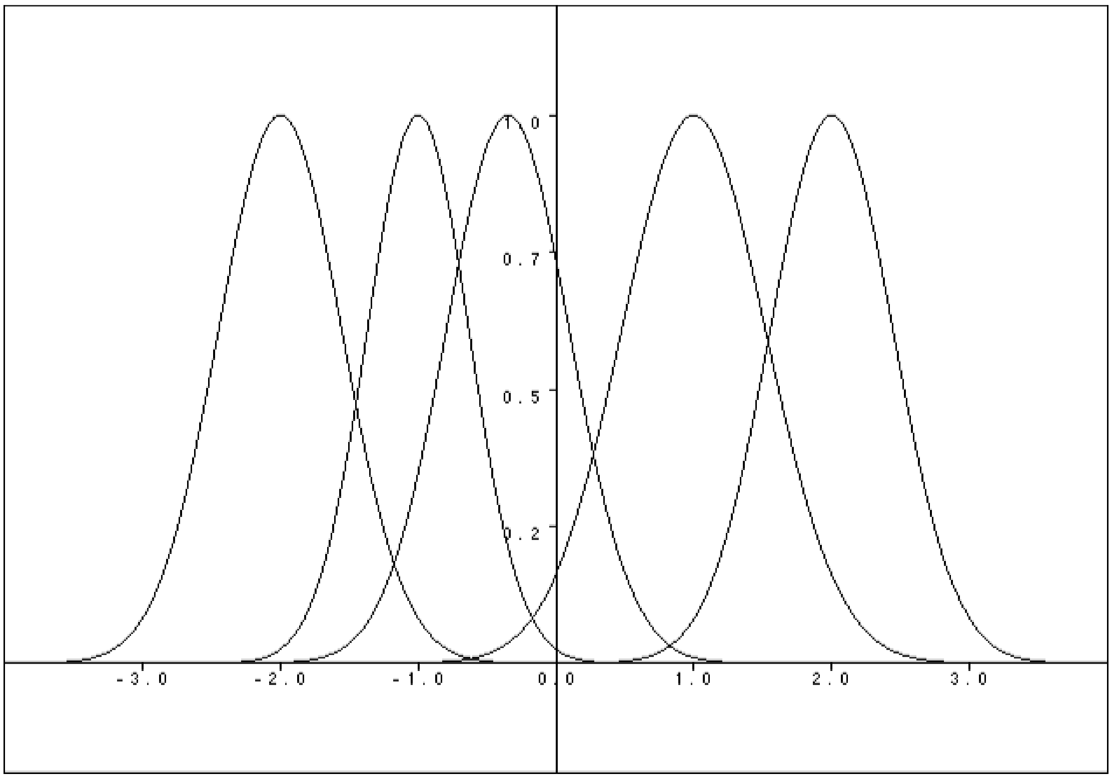

Figure 6.33 Membership functions for explorative, uniform population, topology L3-34-2-1, simulation 4, x_1.

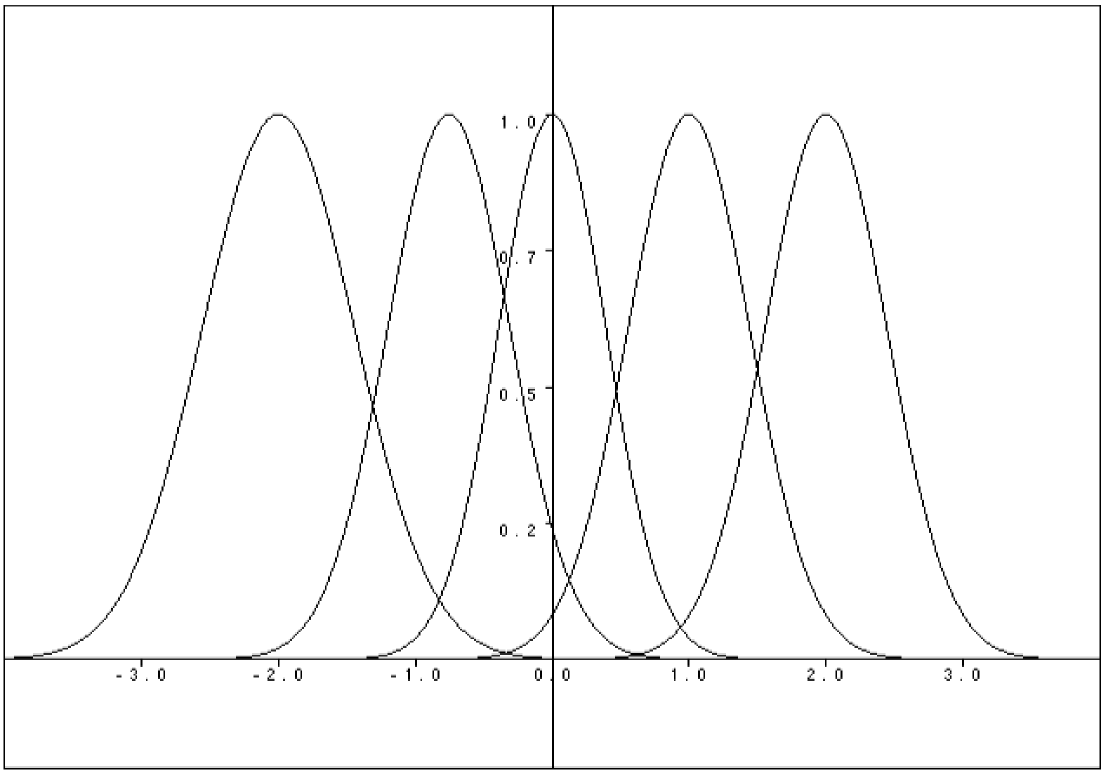

Figure 6.34 Membership functions for explorative, uniform population, topology L3-34-2-1, simulation 4, x_2.

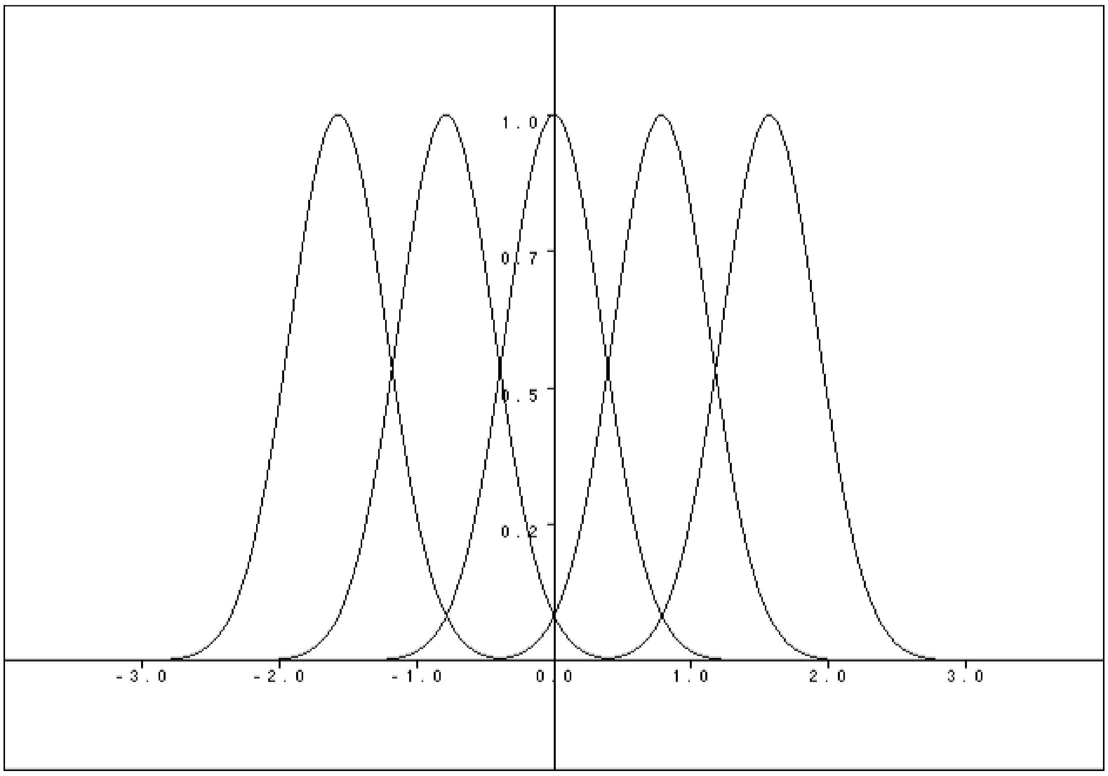

Figure 6.35 Membership functions for explorative, uniform population, topology L3-34-2-1, simulation 4, x_3.

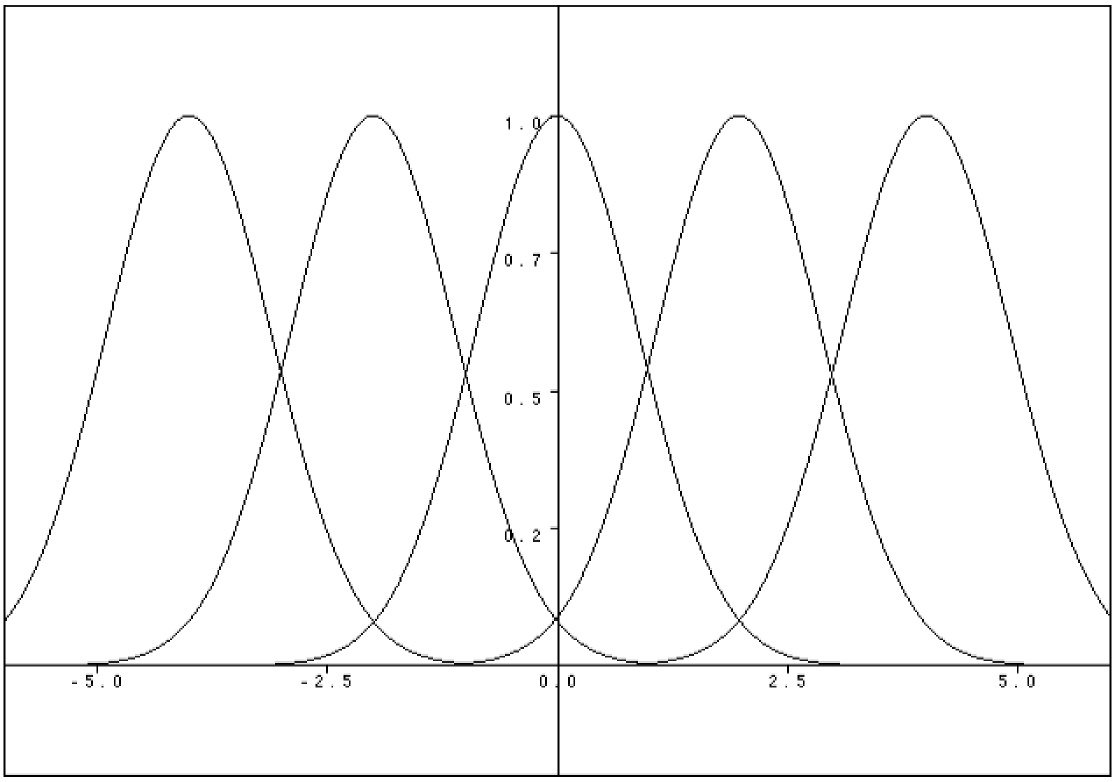

Figure 6.36 Membership functions for explorative, uniform population, topology L3-34-2-1, simulation 4, x_4.

Table 6.4 MF centres for L-34-2-1 explorative, uniform initial MF population, sim. no 8.

MF	c1	c2	c3	c4	c5
x_1	-2.0	-0.999761	-0.476308	0.966232	2.0
x_2	-2.0	-0.999745	0.000074	0.999943	2.0
x_3	-1.5707963	-0.785398	-0.000015	0.649385	1.5707963
x_4	-4.0	-2.248928	-0.000061	1.999595	4.0

Table 6.5 MF centres for L-34-1-2 explorative, uniform initial MF population, sim. no 5.

MF	c1	c2	c3	c4	c5
x_1	-2.0	-1.000000	0.191510	0.191510	2.0
x_2	-2.0	-1.000000	-0.190039	0.999810	2.0
x_3	-1.5707963	-0.727477	-0.000005	0.785352	1.5707963
x_4	-4.0	-2.911787	-0.000162	1.999668	4.0

Table 6.6 MF centres for L-34-1-2 explorative, uniform initial MF population, sim. no 10.

MF	c1	c2	c3	c4	c5
x_1	-2.0	-0.951743	-0.000083	0.468950	2.0
x_2	-2.0	-0.999813	-0.112785	0.999956	2.0
x_3	-1.5707963	-0.785278	-0.004262	0.785277	1.5707963
x_4	-4.0	-2.863850	-0.004846	1.999147	4.0

Table 6.7 MF centres for L-34-2-1 greedy, uniform initial MF population, sim. no 4.

MF	c1	c2	c3	c4	c5
x_1	-2.0	-0.242209	0.330845	0.999604	2.0
x_2	-2.0	-0.524521	0.118957	0.806073	2.0
x_3	-1.5707963	-0.785256	0.000162	0.785324	1.5707963
x_4	-4.0	-2.325819	0.000509	2.000036	4.0

Table 6.8 MF centres for L-34-2-1 greedy, uniform initial MF population, sim. no 8.

MF	c1	c2	c3	c4	c5
x_1	-2.0	-0.741330	-0.238746	0.687604	2.0
x_2	-2.0	-0.999952	0.419326	0.999608	2.0
x_3	-1.5707963	-0.785020	0.000034	0.785167	1.5707963
x_4	-4.0	-1.992406	-0.248836	2.677708	4.0

Table 6.9 MF centres for L-34-1-2 greedy, uniform initial MF population, sim. no 3.

MF	c1	c2	c3	c4	c5
x_1	-2.0	-0.986411	-0.865216	-0.206337	2.0
x_2	-2.0	-0.526121	0.190813	0.228068	2.0
x_3	-1.5707963	-0.643332	-0.042579	0.709030	1.5707963
x_4	-4.0	-2.782232	-0.463713	2.544888	4.0

Table 6.10 MF centres for L-34-1-2 greedy, uniform initial MF population, sim. no 10.

MF	c1	c2	c3	c4	c5
x_1	-2.0	-0.929495	0.179905	0.999581	2.0
x_2	-2.0	-0.821071	-0.471415	0.999901	2.0
x_3	-1.5707963	-0.617001	-0.251729	0.402342	1.5707963
x_4	-4.0	0.302217	1.524897	2.949449	4.0

Table 6.11 MF centres for L-34-2-1 explorative, random initial MF population, input variables, sim. no 2.

MF	c1	c2	c3	c4	c5
x_1	-2.0	-0.4746	0.49005	0.67679	2.0
x_2	-2.0	-0.790297	0.995587	0.99977	2.0
x_3	-1.5707963	0.121182	0.273279	0.52306	1.5707963
x_4	-4.0	-2.105933	-1.726494	2.996628	4.0

Table 6.12 MF centres for L-34-2-1 explorative, random initial MF population, sim. no 2.

MF	c1	c2	c3	c4	c5	c6	c7
y	-15.0	-10.894	-0.018	7.896	10.848	12.226	15.0

Table 6.13 MF centres for L-34-2-1 explorative, random initial MF population, input variables, sim. no 7.

MF	c1	c2	c3	c4	c5
x_1	-2.0	0.770300	0.770300	0.972722	2.0
x_2	-2.0	-0.986932	-0.979073	0.999532	2.0
x_3	-1.5707963	-0.283689	0.512616	0.519055	1.5707963
x_4	-4.0	-1.872221	2.459921	2.967251	4.0

Table 6.14 MF centres for L-34-2-1 explorative, random initial MF population, output variable, sim. no 7.

MF	c1	c2	c3	c4	c5	c6	c7
y	-15.0	-12.986	-5.961	-0.169	11.411	11.503	15.0

Table 6.15 MF centres for L-34-1-2 explorative, random initial MF population, input variables, sim. no 5.

MF	c1	c2	c3	c4	c5
x_1	-2.0	0.467513	0.967171	0.984325	2.0
x_2	-2.0	0.903821	0.999520	0.999616	2.0
x_3	-1.5707963	-0.499339	-0.173375	0.222275	1.5707963
x_4	-4.0	-1.851142	-1.200847	0.247133	4.0

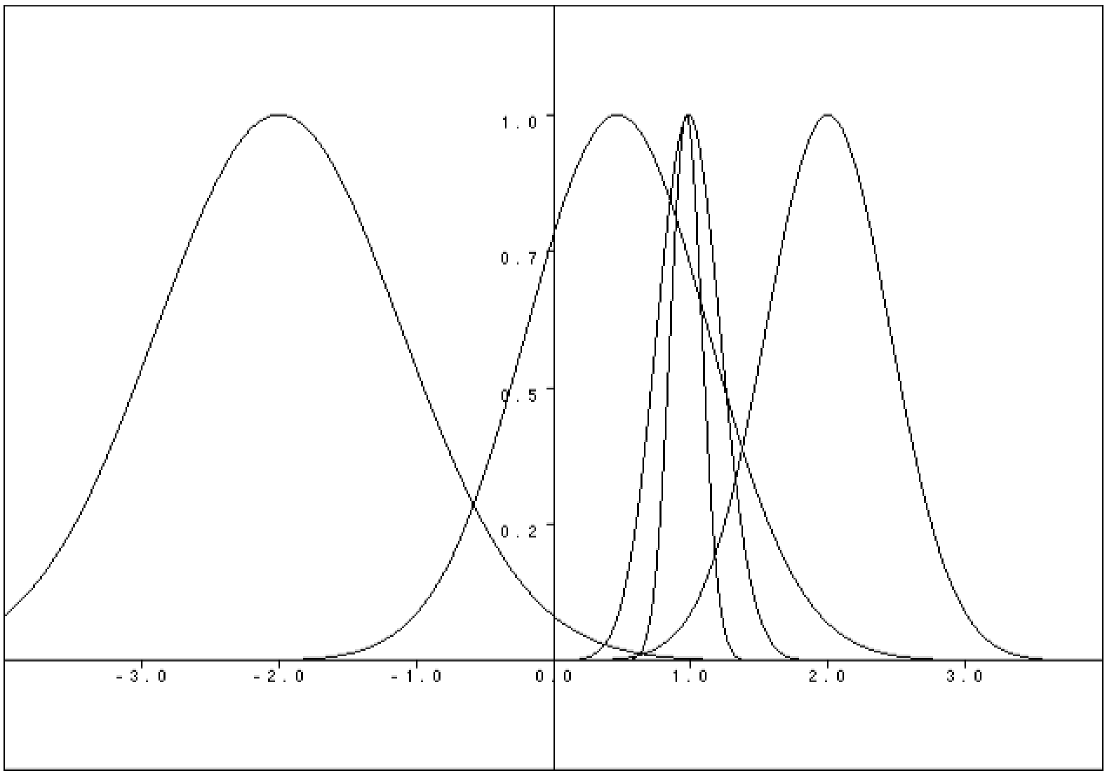

Figure 6.37 Membership functions for explorative, random population, topology L3-34-1-2, simulation 5, x_1.

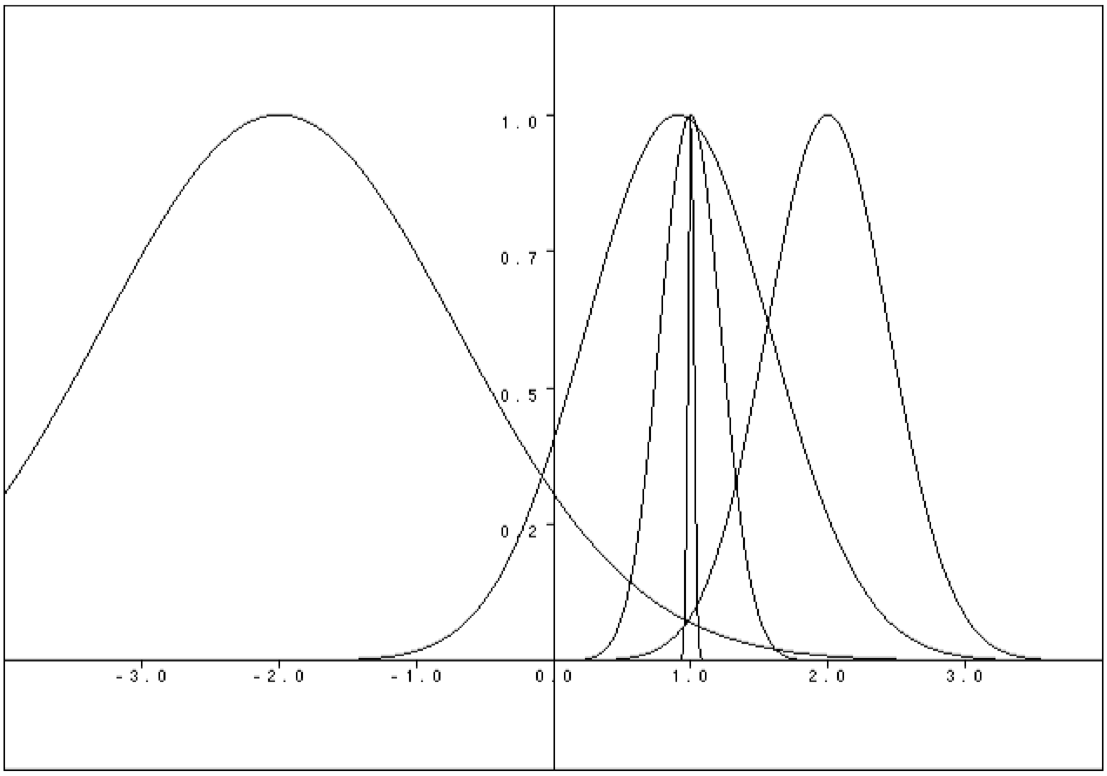

Figure 6.38 Membership functions for explorative, random population, topology L3-34-1-2, simulation 5, x_2.

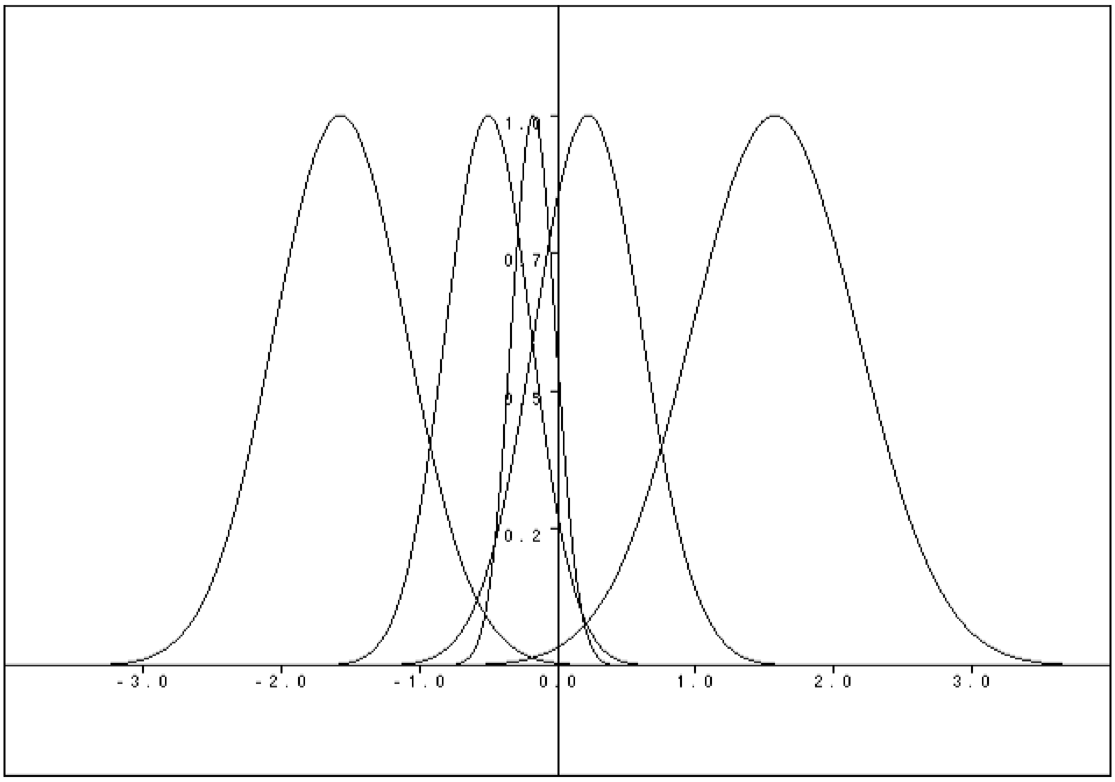

Figure 6.39 Membership functions for explorative, random population, topology L3-34-1-2, simulation 5, x_3.

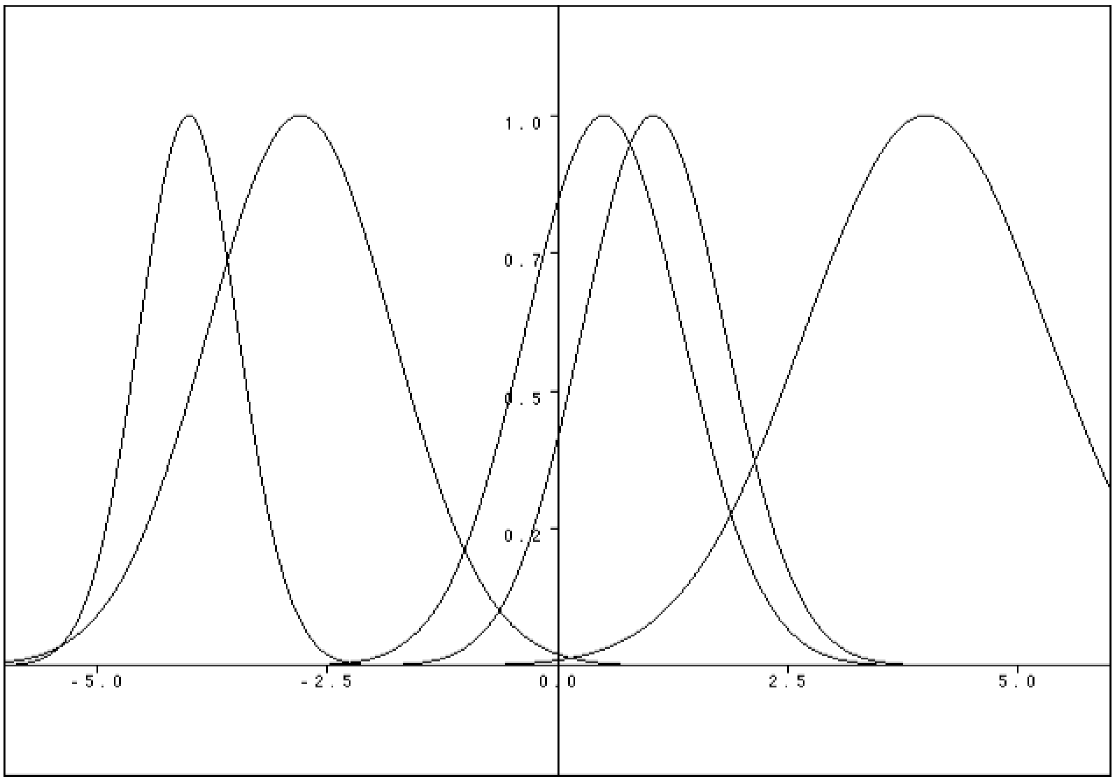

Figure 6.40 Membership functions for explorative, random population, topology L3-34-1-2, simulation 5, x_4.

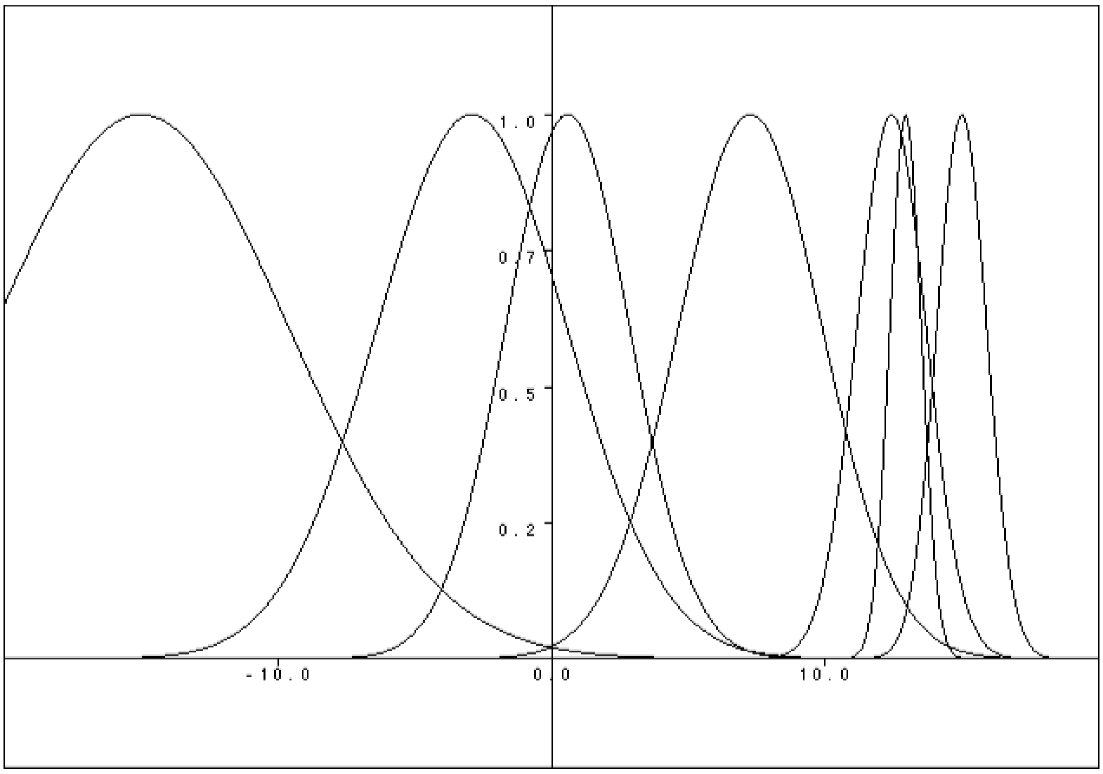

Figure 6.41 Membership functions for explorative, random population, topology L3-34-1-2, simulation 5, output variable u.

Table 6.16 MF centres for L-34-1-2 explorative, random initial MF population, output variable, sim. no 5.

MF	c1	c2	c3	c4	c5	c6	c7
y	-15.0	-2.912	0.590	7.270	12.429	12.944	15.0

Table 6.17 MF centres for L-34-1-2 explorative, random initial MF population, input variables, sim. no 8.

MF	c1	c2	c3	c4	c5
x_1	-2.0	-0.779265	-0.340181	0.939943	2.0
x_2	-2.0	-0.256899	-0.024240	0.650496	2.0
x_3	-1.5707963	0.254259	0.522811	0.522922	1.5707963
x_4	-4.0	-2.806648	-1.628743	0.470980	4.0

Table 6.18 MF centres for L-34-1-2 explorative, random initial MF population, output variable, sim. no 8.

MF	c1	c2	c3	c4	c5	c6	c7
y	-15.0	-3.212	-1.046	-0.457	4.844	9.916	15.0

Table 6.19 MF centres for L-34-2-1 greedy, random initial MF population, input variables, sim. no 2.

MF	c1	c2	c3	c4	c5
x_1	-2.0	-0.999944	-0.963702	0.999855	2.0
x_2	-2.0	-0.786511	-0.568970	0.995108	2.0
x_3	-1.5707963	0.069727	0.367798	0.523443	1.5707963
x_4	-4.0	-1.199057	-0.287595	2.999328	4.0

Table 6.20 MF centres for L-34-2-1 greedy, random initial MF population, output variable, sim. no 2.

MF	c1	c2	c3	c4	c5	c6	c7
y	-15.0	-8.006	-5.253	7.034	8.571	9.943	15.0

Table 6.21 MF centres for L-34-2-1 greedy, random initial MF population, input.

MF	c1	c2	c3	c4	c5
x_1	-2.0	-0.992531	-0.930294	0.999251	2.0
x_2	-2.0	-0.885832	0.499797	0.833121	2.0
x_3	-1.5707963	0.331195	0.450526	0.522577	1.5707963
x_4	-4.0	-1.699458	1.593186	2.899929	4.0

Table 6.22 MF centres for L-34-2-1 greedy, random initial MF population, output variable, sim. no 3.

MF	c1	c2	c3	c4	c5	c6	c7
y	-15.0	-12.831	-1.955	1.770	12.144	12.555	15.0

Table 6.23 MF centres for L-34-1-2 greedy, random initial MF population, input variables, sim. no 3.

MF	c1	c2	c3	c4	c5
x_1	-2.0	-0.537982	0.887288	0.997506	2.0
x_2	-2.0	-0.955511	-0.430739	0.987376	2.0
x_3	-1.5707963	-0.478354	0.458689	0.484029	1.5707963
x_4	-4.0	-1.456939	1.977950	2.923114	4.0

Table 6.24 MF centres for L-34-1-2 greedy, random initial MF population, output variable, sim. no 3.

MF	c1	c2	c3	c4	c5	c6	c7
y	-15.0	-12.494	-9.998	1.270	7.952	12.219	15.0

Table 6.25 Memb. Funct. centres for L-34-1-2 greedy, random initial MF population, input variables, sim. no 5.

MF	c1	c2	c3	c4	c5
x_1	-2.0	-0.972411	-0.033781	0.998627	2.0
x_2	-2.0	0.840976	0.895264	0.986912	2.0
x_3	-1.5707963	0.522430	0.523169	0.523427	1.5707963
x_4	-4.0	-2.791375	0.503425	1.041765	4.0

Table 6.26 MF centres for L-34-1-2 greedy, random initial MF population, output variable, sim. no 5.

MF	c1	c2	c3	c4	c5	c6	c7
y	-15.0	-12.999	-10.974	-0.822	1.327	1.981	15.0

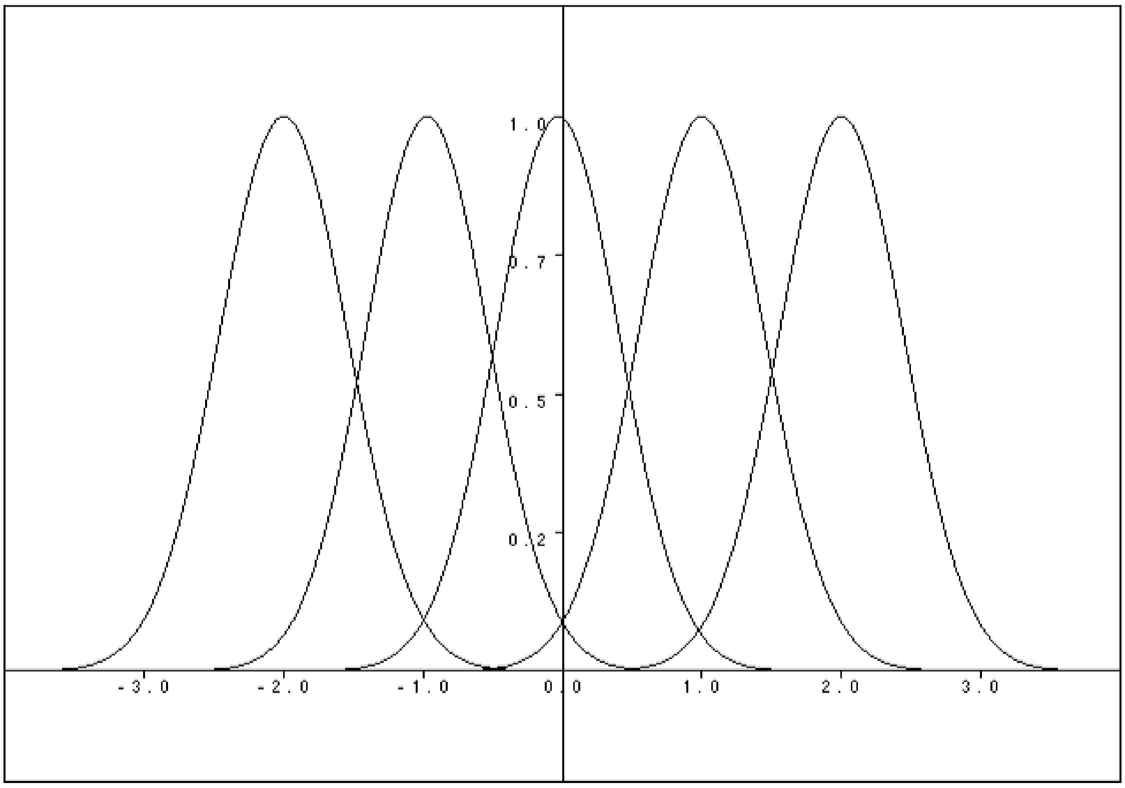

Figure 6.42 Membership functions for greedy, random population, topology L3-34-1-2, simulation 5, x_1.

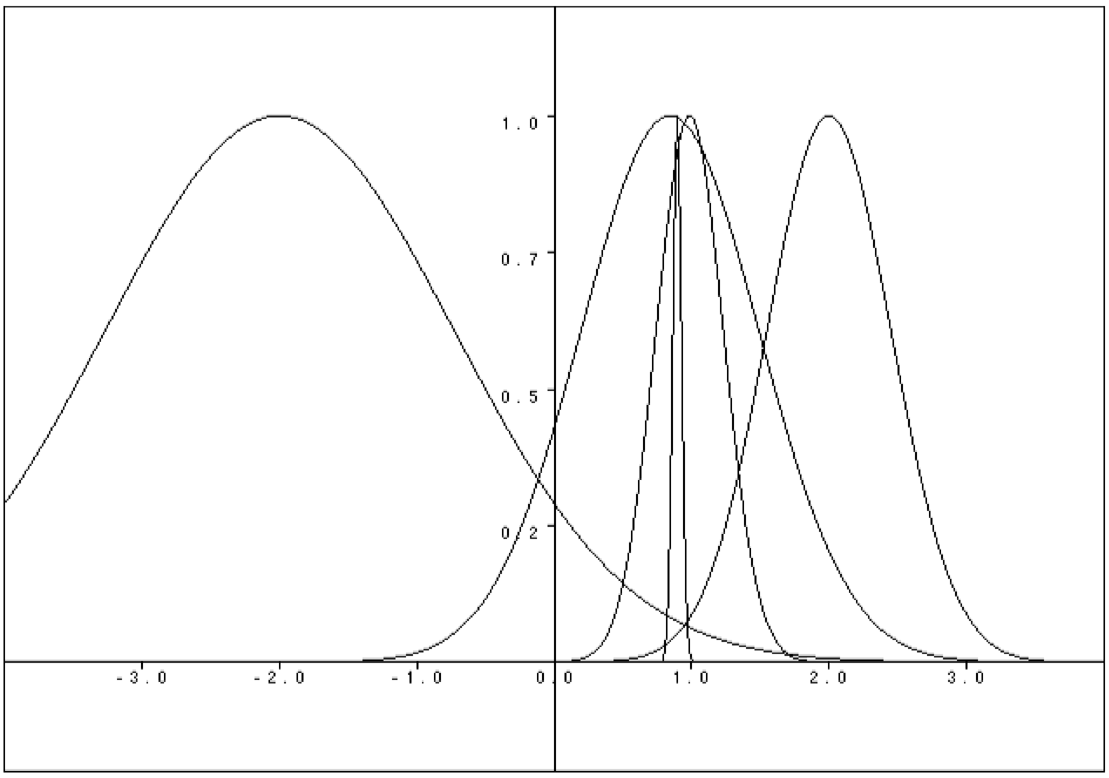

Figure 6.43 Membership functions for greedy, random population, topology L3-34-1-2, simulation 5, x_2.

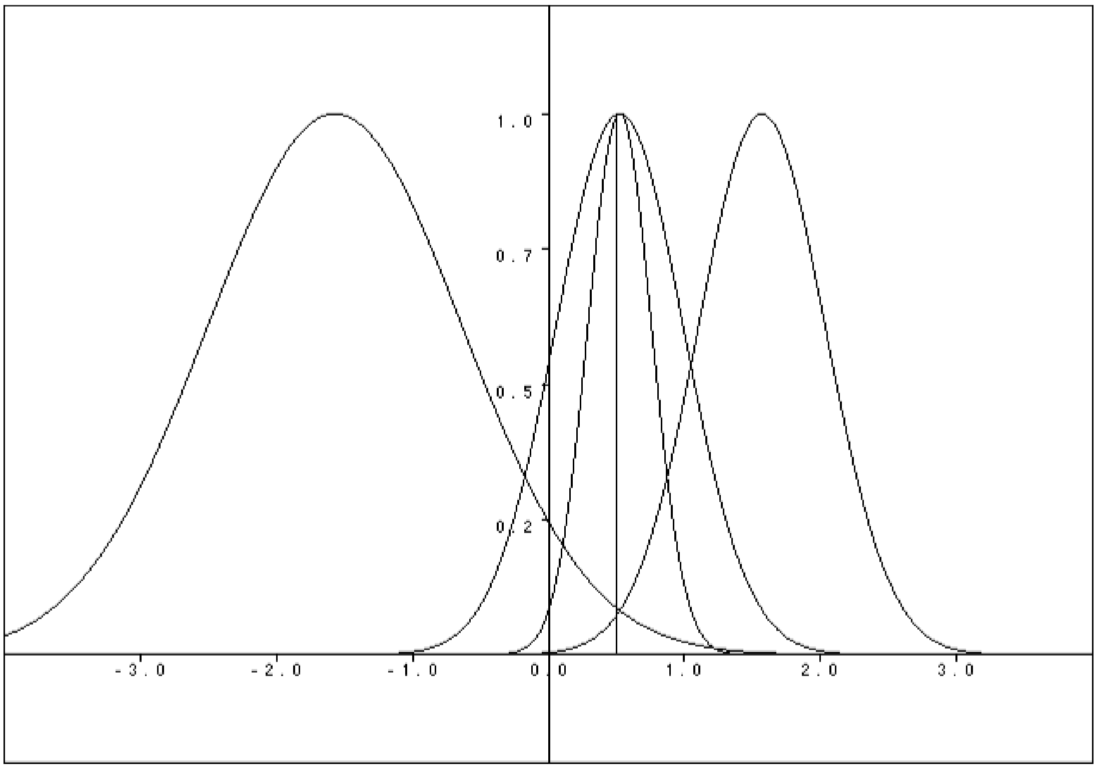

Figure 6.44 Membership functions for greedy, random population, topology L3-34-1-2, simulation 5, x_3.

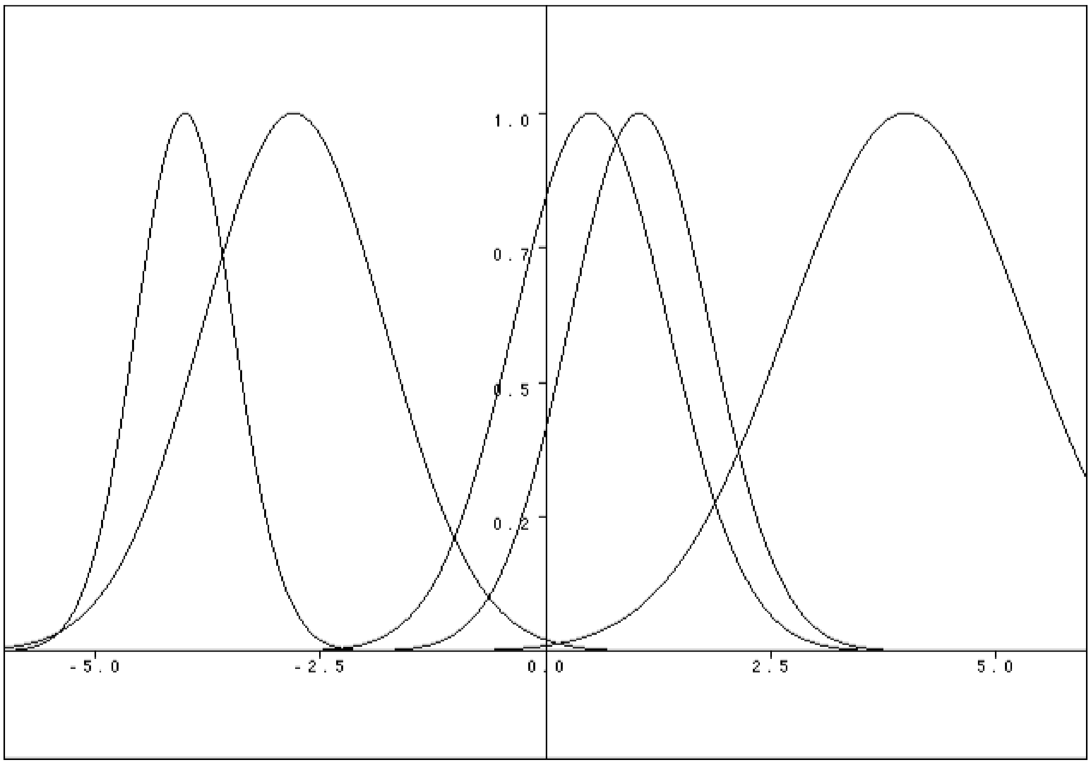

Figure 6.45 Membership functions for greedy, random population, topology L3-34-1-2, simulation 5, x_4.

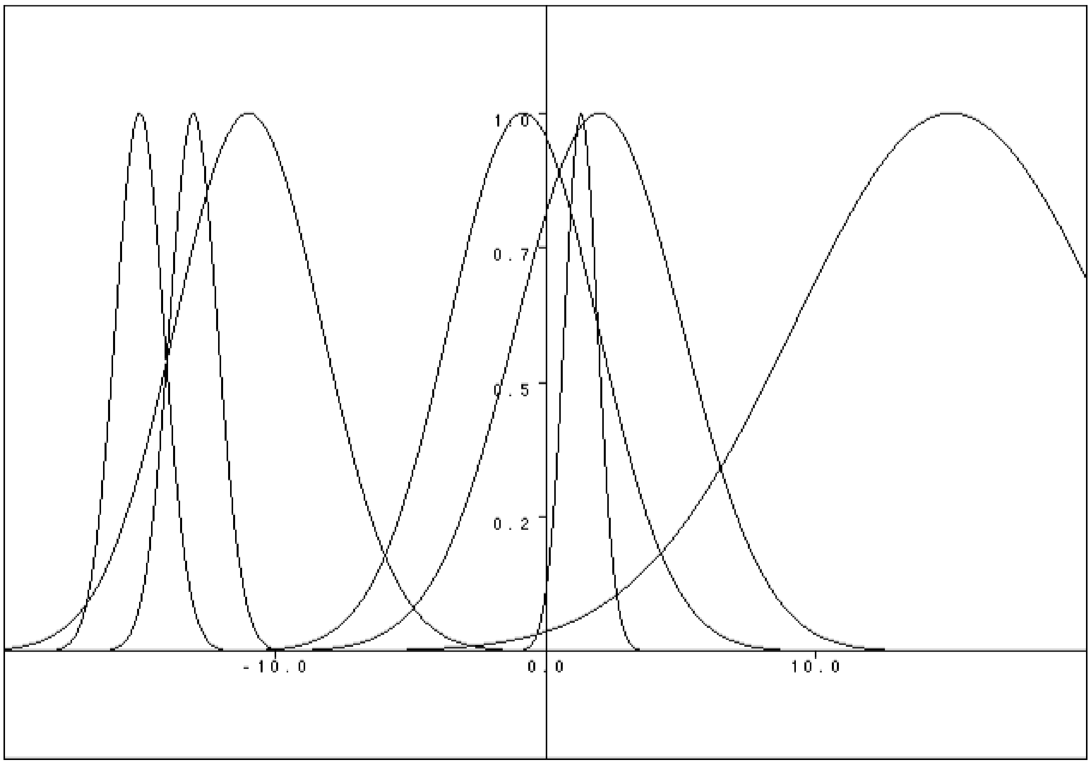

Figure 6.46 Membership functions for greedy, random population, topology L3-34-1-2, simulation 5, output variable u.

As can be seen from the tables and diagrams illustrating MF centres distribution, the membership functions that are generated by the CEA with random initial MF population show tendency to cluster within their pre-defined intervals. Occasionally, membership function centres cluster very closely, for example MFs for x_1 and x_2, see Table 6.15 and Table 6.25. One possible explanation for membership functions clustering effect is that they tend to cluster in the regions of state space where the magnitude of control is changing rapidly to stabilise the system. Detailed investigation of state trajectories should give more clues to explain this phenomenon.

The membership function centres found by the CEA with the uniform initial MF population are usually similar to their original definitions in the initial population, i.e., they change little in the evolutionary process.

Summary

In this chapter a hierarchical fuzzy controller is designed, which gives improved performance for the control of the inverted pendulum using a co-evolutionary algorithm to fine tune parameters of the input and output sets membership sets in each layer as well as learn a complete set of fuzzy rules in the fuzzy knowledge base.

Different versions of the co-evolutionary algorithm are investigated: explorative and greedy. Furthermore, the impact of initial membership functions population on the controller performance is investigated. The membership functions population is either randomly generated or initial membership functions are evenly spaced across input and output variables domains. The effect of different Gaussian function shapes on the controller performance is briefly discussed.

Chapter 7 EVOLUTIONARY ALGORITHM BASED COMPOSITIONAL METHOD

7.1 Introduction

In previous chapters, the EA has been used to develop the rule base to control the system to the target region from a single initial condition. Obviously, this procedure can be repeated for any other initial condition but that would be impractical approach to the real-life applications. The controller developed for a single initial condition may or may not control the system from a different initial condition. Therefore, a different approach is required to address the problem of controllability region.

A new evolutionary algorithm based compositional method is proposed to control system over the set of user-defined initial conditions. The method addresses directly the problem of controlling the system from specific, user-defined initial conditions. In many practical applications there is no necessity to secure controllability over the large region in the state space, which is often difficult to achieve. Instead, a selected region of the state space, or even specific initial conditions can be considered.

The problem of designing hierarchical fuzzy control over a set of initial conditions can be investigated by two methods: amalgamation or compositional. The amalgamation method develops a fuzzy rule base for every initial condition in the user-defined region of state space separately and then amalgamates them into a single knowledge base. Usually a regular grid of initial conditions is used to cover the region in the state space for which a control system is to be developed. Different initial conditions create different dynamical conditions for the system. In the inverted pendulum case that means different initial pole angles and angular velocities, different cart position and cart’s velocity. Therefore, corresponding linguistic variable values are also diametrically different. In amalgamation method every initial condition is considered separately and the EA searches for a controller just for this particular condition. Every initial condition reflects a specific initial dynamics of the system and therefore the resulting control system can be diametrically different from other controllers developed for different initial conditions. For the inverted pendulum, the output linguistic variable takes integer values in [1,7]. After amalgamation the resulting control values are usually meaningless arithmetic average: either 3 or 4. The experiments with the amalgamation method did not produce a successful result. Amalgamation method based on averaging the fuzzy rules for better performance and larger control region does not seem feasible. This fact is examined in a different series of experiments. It is possible that specific tuning of the evolutionary algorithm would produced more homogenous knowledge bases but at this point it is a mere speculation. Therefore amalgamation method to find an 'averaged' controller (that would cover different initial populations and different initial conditions) fails, giving in result just an arithmetical average of the linguistic variables encoded as integer numbers $\in [1,7]$ that does not stabilise the system. Amalgamation method might give good results for a set of initial conditions that do not generate diametrically different dynamics in the system, i.e., for a set of initial conditions representing similar dynamical conditions of the physical system.

An alternative to amalgamation method approach is to let the evolutionary algorithm learn the 'final' knowledge base by itself. This approach is called the compositional method. The learning of the knowledge base is achieved by evaluating the fitness of an individual at an arbitrary configuration within the pre-defined set of initial configurations in each generation. This local knowledge is then inherently learned from generation to generation in the evolution of the EA (Mohammadian and Stonier 1996a), (Stonier 1999).

A set of 255 initial conditions in the state space (that is viable in terms of the inverted pendulum dynamics) is defined for experiments with the compositional method, see Appendix for the explicit values of initial conditions. Usually, only static initial conditions are considered for the inverted pendulum: the cart's position and the angle of the pole. In this investigation also dynamic initial conditions of the system are considered: the initial cart's velocity and the angular velocity of the pole. Thus, this constitutes a more comprehensive investigation of the initial conditions for the inverted pendulum system. The goal in this investigation is to achieve control to the target region from as many initial conditions as possible.

7.2 Evolutionary algorithm based compositional method

Basic features of the compositional method can be summarised in the following points:

- Evolutionary algorithm based compositional method searches for a controller over the whole set of initial conditions at every generation.
- Fitness of each individual reflects the controller performance for every initial condition in the set.

- Every string in the population is assigned the fitness value which is a composite value representing string's performance for every single initial condition in the user-defined set.

7.2.1 Hierarchical fuzzy system for compositional method

The development of the compositional method is based on the example of the inverted pendulum system (as the test system). However, the algorithm can be applied to a wide range of the HFS with suitable replacements (system dynamics, hierarchical structure, population encoding, objective functions definitions, etc). The HFS selected for the inverted pendulum system is the 3-layered topology *L3-34-2-1* that is a good compromise between controller performance and the size of the knowledge base. Mamdani product and minimum inference engine are used in computer simulations, see Equation 3.3 and 3.4 respectively.

7.2.2 Defining evolutionary population

Each fuzzy rule in any of the HFS knowledge bases is uniquely defined by the consequent part that is associated with a particular output fuzzy set, for example B^k, identified by the integer $k \in [1,7]$, see Section 4.3 for details. The three fuzzy rule base structure is represented as a linear individual string of $M = 95$ elements and the evolutionary population is defined as the set of M_p individuals: $P = \{ \vec{p}_k\colon\ \vec{p}_k = (a_{1, \dots ,} a_{95}), k = 1, \dots , M_p,\ \ a_j \in \{1, \dots , 7\}\ \}$.

7.2.3 Fitness function

Assume that there are N_c initial conditions in the user-defined set of initial conditions. To evaluate the objective function value for a given string $\vec{p}_k$ from the controller population first the objective function values f_i , $i = 1, \dots , N_c$, are evaluated for every single initial condition. Then overall objective function value f is determined from the values f_i, $i = 1, \dots , N_c$, calculated for every single initial condition.

The fitness for a single initial condition is evaluated as follows: each string $\vec{p}_k$ is decoded into three components defining the fuzzy knowledge base for each layer, then the Mamdani or minimum inference formula is used to evaluate u_1, u_2, and u to find the final control to be applied at each value of the state $\vec{x}$. The system state equations are integrated by the Runge-Kutta algorithm (RK4) with step size 0.02 over time interval $[0,T]$. The fitness f_i is then calculated based on the behaviour of the system over the time interval, see Equations 4.1 and 4.2.

Then overall objective function value is determined from the f_i values (calculated for every single initial condition) and assigned to the string $\vec{p}_k$ as its objective value. The choice of fitness function evaluation method based on fitness values for every initial condition decides of effectiveness of the compositional method and therefore plays a crucial role. For this experiment with the EA based compositional method, a simple evaluation method is defined: the fitness function is evaluated as arithmetic average over all fitness values f_i , $i = 1, \dots , N_c$, calculated for every single initial condition and then assigned to a particular string in controller population. This is not an ideal choice – it may happen that only for a few initial conditions fitness function value is low and those initial conditions distort the average fitness value even though the rest of the initial conditions may give high fitness function values which is undesirable (minimisation problem).

$$f = \frac{1}{Nc}\sum_{i=1}^{Nc} f_i \tag{7.1}$$

A penalty is added to the objective if the final state breaks the following bounds: $|x_1| \leq 0.1$, $|x_2| \leq 0.1$, $|x_3| \leq \pi/24$, $|x_4| \leq 3.0$, i.e., leaves the designated TR. Two penalties schedules are tested, penalty schedule-A and penalty schedule-B, see Section 4.5.2. To test the influence of penalty schedules a number of simulations are run with and without any penalty schedule.

7.2.4 Membership functions

Another method to increase accuracy and achieve better performance from control system is to increase the number of membership functions covering input and output variables. This, however, increases the size of the knowledge base and therefore with larger number of rules to be learned the computation time is longer.

7.2.5 Crossover and mutation

The crossover operation maintains the diversity in the population. For this reason the choice of the crossover operator often plays a crucial role in the successful application of the EA. Too disruptive crossover procedure may cause extended number of generations before achieving convergence or can cause the EA to fail to find a solution at all. The right selection of the crossover operator is case dependant and crossover operator that performed successfully in one application may fail in another. For testing the compositional method random, arithmetic, and uniform crossover of two parent strings to form two children in the next generation are used.

Another mechanism of maintaining population diversity is mutation operator. It has also a role in preventing pre-mature convergence to a local minimum (or maximum, depending on the problem formulation). With a given probability, the mutation operator mutates elements of the strings in the population. Mutation is undertaken with probability p_m whose value is determined by a mutation schedule that decreases typically from 0.8 to 0.001 over 1000 generations. Below is the typical mutation schedule used in the simulations:

```
if ( gen ≥ 0   & gen < 100 ) pm = 0.8
if ( gen ≥ 100 & gen < 200 ) pm = 0.7
if ( gen ≥ 200 & gen < 300 ) pm = 0.6
if ( gen ≥ 300 & gen < 400 ) pm  = 0.4
if ( gen ≥ 400 & gen < 500 ) pm  = 0.2
if ( gen ≥ 500 & gen < 600 ) pm = 0.1
if ( gen ≥ 600 & gen < 800 ) pm  = 0.01
if ( gen > 800) pm = 0.001
```

where `gen` denotes the generation number. The mutation operator is defined by the pseudo-code given in Section 4.4.

7.2.6 Overview of the algorithm

The EA is used to learn fuzzy rules in the HFS that constitutes a control system for the inverted pendulum. A schematic EA algorithm in its general form, applicable to a wide range of dynamical systems, is given below:

1. EA parameters are selected: type of inference engine, crossover, mutation schedule, selection method, elitism, fitness function (with or without a penalty schedule), and termination condition (i.e. number of generations or lack of significant change in the state vector).
2. Population $P(t)$, $t = 0$, is randomly initialised: every component of individual string is given a randomly selected value from a predefined interval. Objective functions are evaluated for the first generation $P(0)$.
3. $t = t + 1$: next generation is created using EA operators: selection, crossover, mutation.
4. Individual from the population is selected.
5. Initial condition is selected from the predefined list.
6. Dynamical system is simulated from a given initial condition.
7. Final state of state variables and survival time are determined.

8. Based on values from the previous step temporary fitness function value is evaluated for an individual. Penalties are added to the fitness value (if penalty schedule is defined).
9. Steps 5— 8 are repeated until all system simulations for every initial condition in the list are performed.
10. Average of all temporary fitness values is calculated and assigned to the individual as its fitness.
11. Steps 4—10 are repeated until all individuals in the population have their fitness evaluated.
12. Steps 3—11 repeated until the termination condition is satisfied.
13. Final control system is determined by either selecting the top individual or by averaging a pre-defined number of N_{top} best (with regard to the objective function) individuals from the final population. Its performance is evaluated by running a simulation of the dynamical system for all initial conditions and counting initial conditions for which the final state variables are within the target region.

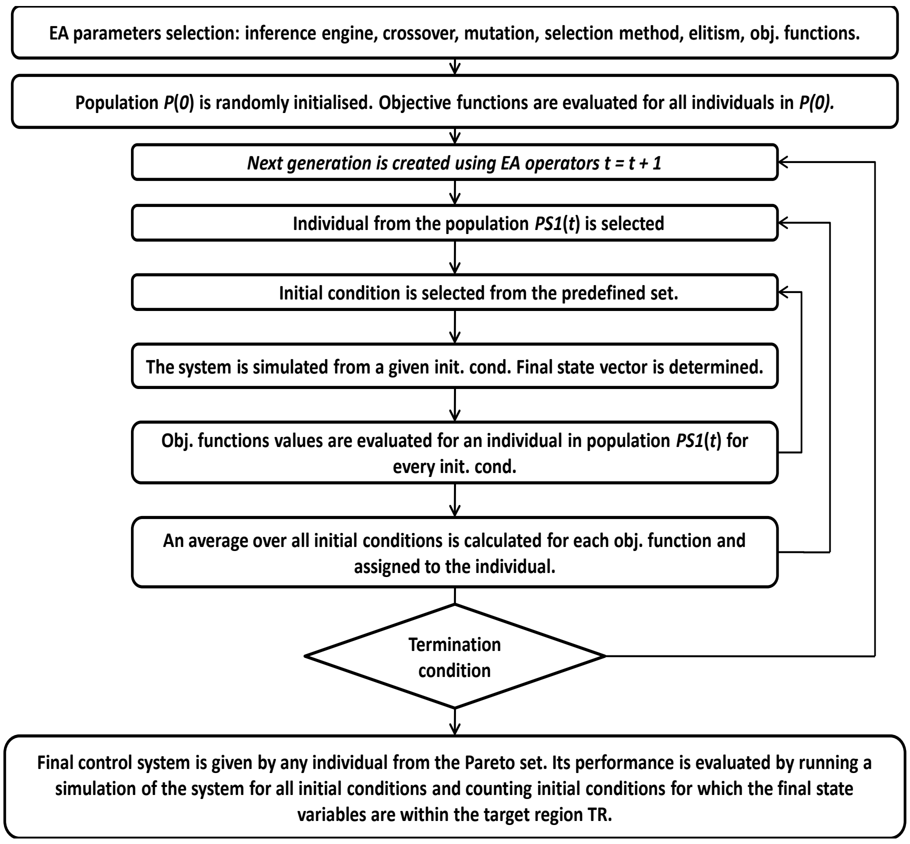
EA parameters selection: inference engine, crossover, mutation, selection method, elitism, obj. functions.
Population P(0) is randomly initialised. Objective functions are evaluated for all individuals in P(0).
Next generation is created using EA operators t = t + 1
Individual from the population PS1(t) is selected
Initial condition is selected from the predefined set.
The system is simulated from a given init. cond. Final state vector is determined.
Obj. functions values are evaluated for an individual in population PS1(t) for every init. cond.
An average over all initial conditions is calculated for each obj. function and assigned to the individual.
Termination condition
Final control system is given by any individual from the Pareto set. Its performance is evaluated by running a simulation of the system for all initial conditions and counting initial conditions for which the final state variables are within the target region TR.

Figure 7.1 EA based compositional method block diagram.

7.3. Experimental setup

The proposed method is implemented and the experiments are conducted. The goal of experiments is to find the optimal combination of EA parameters that would result in finding fuzzy rules capable of successfully controlling the inverted pendulum system to the target area from a large number of user-defined initial conditions.

In all simulations the uniform crossover (some simulations were run with the random and arithmetic crossover) and mutation schedule described in Section 7.2.5 are used.

7.3.1 Initial conditions

A set of regularly interspaced 255 initial conditions is defined within the region given by: $|x_1| \leq 0.75$, $|x_2| \leq 1.0$, $|x_3| \leq \pi/12$, $|x_4| \leq 1.0$. The table of initial conditions is shown in the Appendix.

7.3.2 Initial population

The initial population $P(0) = \{ \vec{p}_k\text{:}\ \ k = 1, \ldots , M_p \}$, is determined by choosing the a_j as a random integer $\in [1,7]$ where M_p is the size of the evolutionary population. Full replacement policy is used and for selection process tournament selection with size $n_T = 4$ is used.

7.3.3 Population size

The population size is set at $M_p = 500$. Smaller population size is often sufficient but with the decreasing population size it is difficult for the EA to maintain the required diversity in population to avoid pre-mature convergence to the local minimum.

7.3.4 Termination condition

The EA is terminated after 1000 generations as it was found that the algorithm either finds solution in about 300—800 generations or fails regardless of how many generations follow.

7.3.5 Elitism

An elitism strategy is typically used to pass the fittest individuals or copies of the fittest individual to the new population, so that the information encapsulated in the best individual is not lost and the fittest individuals are passed into the next generation. In most simulations a strong elitism policy is implemented with four copies of the ten top individuals (forty copies altogether) passed to the next generation.

7.3.6 Fitness function

A fitness function given in Section 7.2.3, and by Equations 4.1, 4.2, and 4.3 is used in all experiments. The fitness function is adjusted to allow the EA a better selection of possible solutions. By selecting some of ω weights to zero the fitness function can be manipulated to achieve a better performance for the control system. The weights in the fitness function are adjusted in several simulations. The most commonly used sets of ω parameters are:

$\omega_1 = 1000, \omega_2 = 0, \omega_3 = 1000, \omega_4 = 0, \omega_5 = 3000;$

$\omega_1 = 0, \omega_2 = 0, \omega_3 = 0, \omega_4 = 0, \omega_5 = 1;$

$\omega_1 = 1000, \omega_2 = 100, \omega_3 = 1000, \omega_4 = 100, \omega_5 = 3000.$

Changing the weights ω in fitness function had a significant impact on the EA performance. In fact, the fifth component of the fitness function ω_5 (survival time) implicitly contains all other components but by specifying them separately the EA process is influenced, i.e., smaller or bigger bias towards one or another component is introduced.

7.4 Computer simulations

More than 130 simulations were run to find the right combination of EA parameters that would result in finding fuzzy rules capable of successfully controlling the inverted pendulum system to the target area from a large user-defined of initial conditions. The controller acted on the inverted pendulum for $T_f = 20.0$s. The results are illustrated on a few typical examples.

7.4.1 Amalgamated controller

Amalgamated controller is tested: top ten control strings from the final population is amalgamated into one final controller. The amalgamation is achieved by taking arithmetic average of corresponding components in ten controller strings; every element of the controller string represents values of the linguistic output variable. Because top ten strings from the controller population do not differ significantly (and often there is little difference if any) the resulting controller string is not much different from the single top controller string. Had the differences been more pronounced (corresponding string elements differing in value significantly) the amalgamated controller would not succeed in controlling the system to the TR from a large number of initial conditions. This can be seen as a trade-off between controllability region and quality of control; for an increase in controllability region the quality of control for the remaining region is much lower. With the controller defined as a single top string from the final population control is always maintained, i.e, the trajectories converge if not to the target area than relatively close to it. Furthermore, as a side-effect of controller amalgamation, the convergence of the state variables is much slower, especially for the cart's position x_1 and its velocity x_2. Example of amalgamated controller performance is shown in Figure 7.2 and Figure 7.3 for randomly selected initial condition no 4: $\vec{x}_0 = (-0.75, -1.0, -0.2617, 1.0)$ and: $\omega_1 = 0, \omega_2 = 0, \omega_3 = 0, \omega_4 = 0, \omega_5 = 1$. Minimum inference engine, uniform crossover, and no penalty schedule is used. The inverted pendulum system is controlled successfully to the target area from 195 initial conditions out of 255, which constitutes 76% success rate.

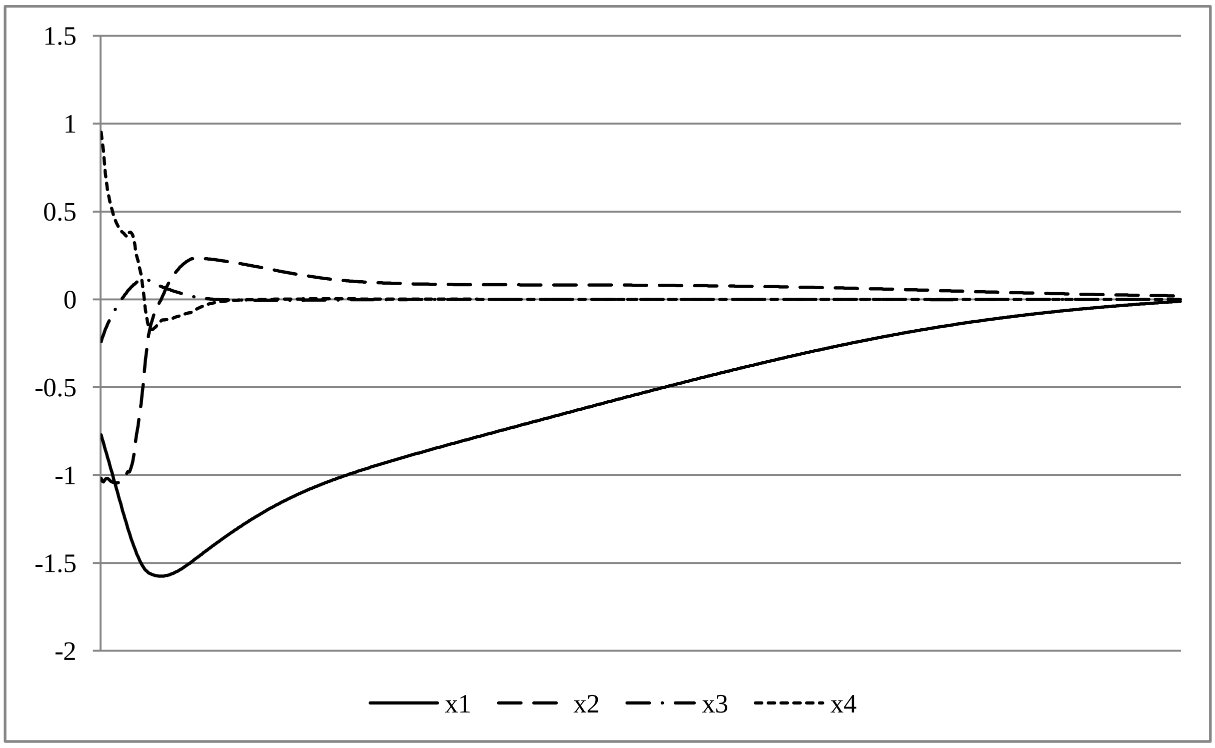

Figure 7.2 Amalgamated controller: State variables convergence, minimum inference engine, uniform crossover, no penalty schedule, init. cond. 4.

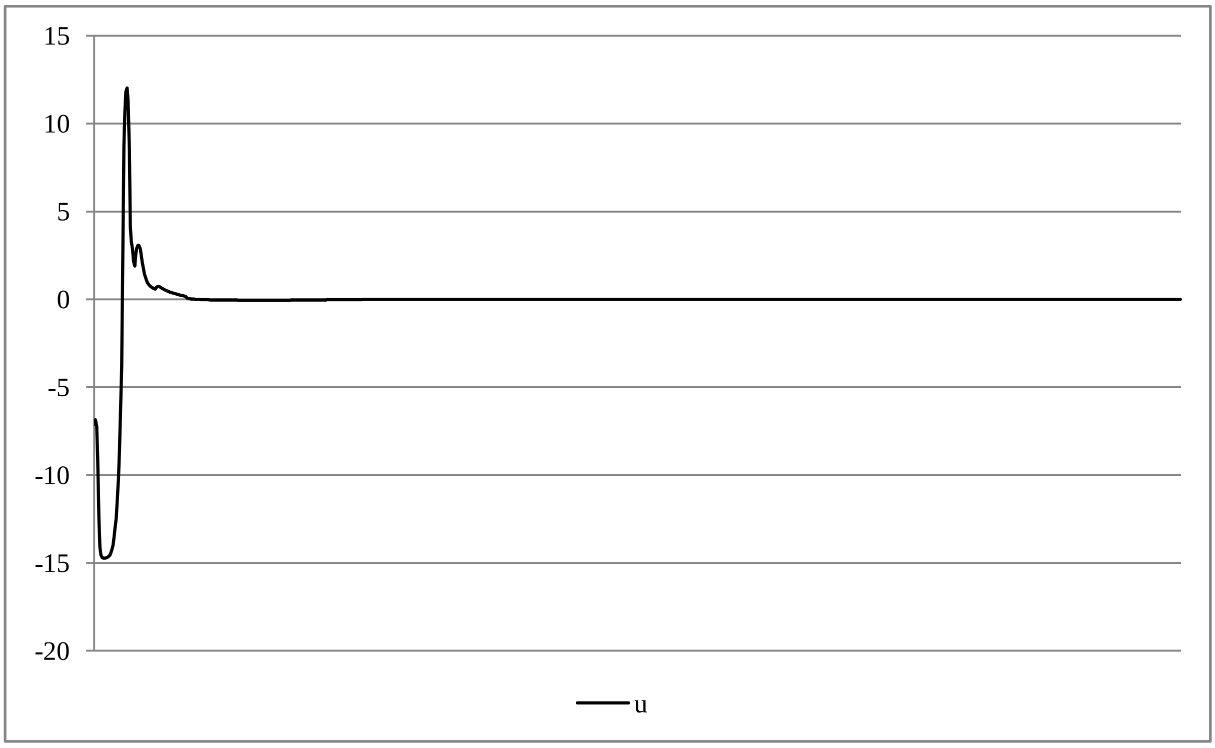

Figure 7.3 Amalgamated controller, init. cond. 4.

7.4.2 Typical results

Regardless of what combination of the EA parameters is used, in most simulations the percentage of initial conditions for which controller successfully controlled the system to the TR varied from about 40% to 60%, with a bulk of simulations achieving below 50% success rate. In many simulations the number of initial conditions from which the controller performance is satisfactory oscillated around 100 (out of 255). This trend might reflect the nature of the inverted pendulum dynamics. It was observed that even though for some initial conditions the state variables did not converge to the TR the final state variables values were very close to the TR. Typical result is illustrated in Figure 7.4 and Figure 7.5 for randomly selected initial condition no 88: $\vec{x}_0 = (-0.35, -0.5, -0.1308, 1.0)$ and: $\omega_1 = 1000$, $\omega_2 = 0$, $\omega_3 = 1000$, $\omega_4 = 0$, $\omega_5 = 3000$. In this simulation minimum inference engine, random crossover, and penalty schedule-A in fitness function are used.

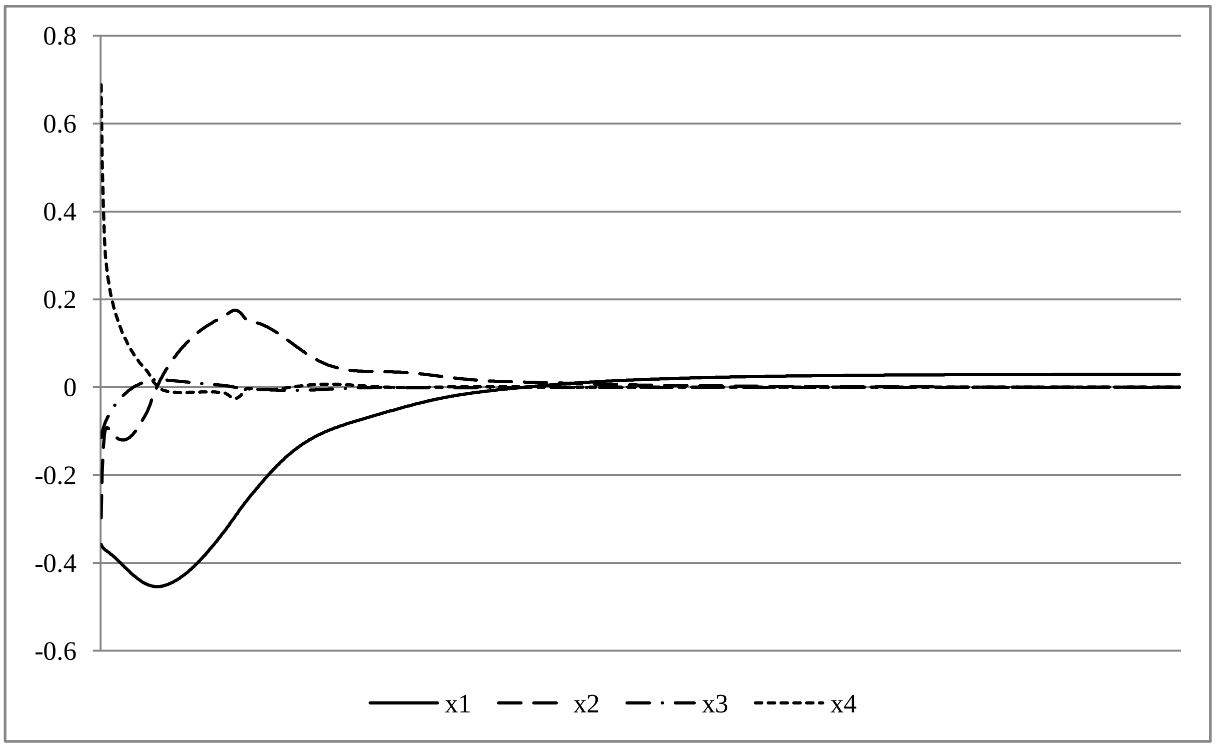

Figure 7.4 Typical result: State variables convergence, minimum inference engine, random crossover, penalty schedule-A, init. cond. 88.

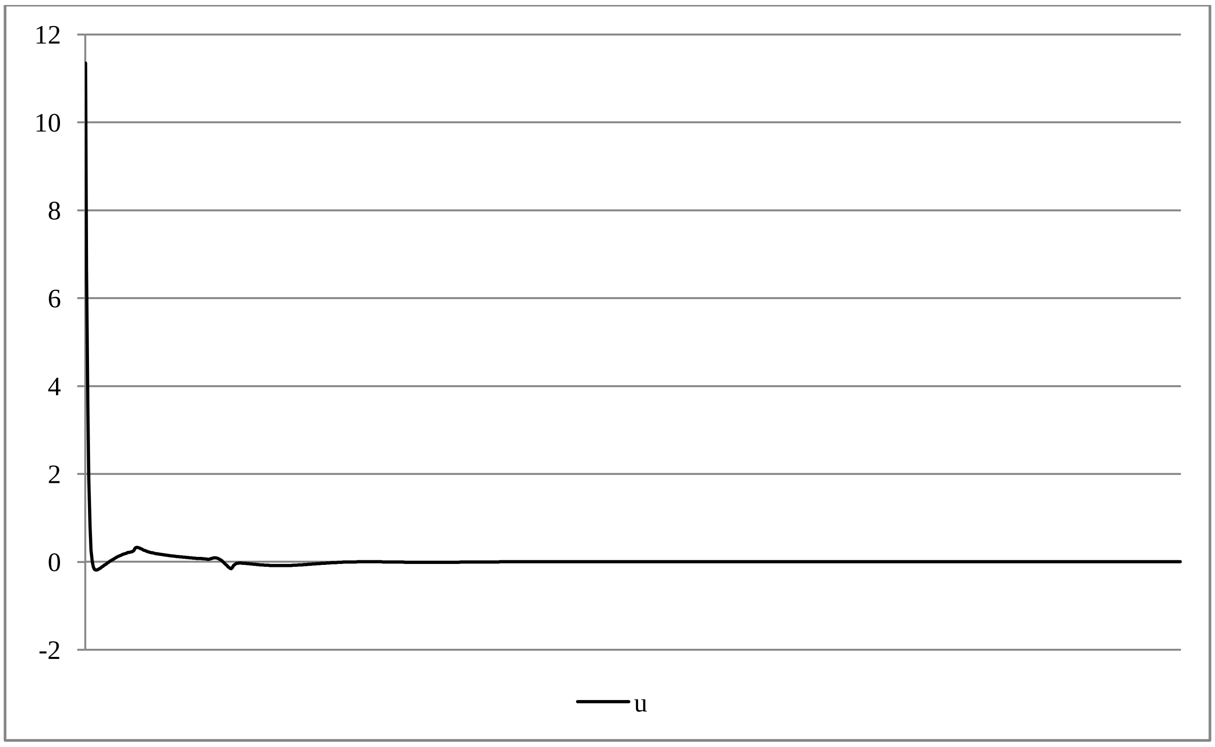

Figure 7.5 Typical result: controller, init. cond. 88.

7.4.3 Good convergence of state variables

In series of experiments the performance of the product inference engine in the fuzzy control system is tested. It was observed that, on average, product inference engine provided fast and smooth convergence of the state variables to the TR but at the cost of the size of controllability region. Minimum inference engine produced better results in terms of larger number of initial conditions from which the controller successfully controlled the system to the TR. Very good state variables convergence is achieved for a simulation with Mamdani inference engine (see Equation 3.3), uniform crossover, and no penalty schedule, but only for 99 out of 255 initial conditions. All state variables converged quickly to zero, except x_1 with values remaining about 0.04 from the origin. A typical result for this simulation is shown in Figure 7.6 and Figure 7.7 for randomly selected initial condition no 122: $\vec{x}_0 = (-0.75, -1.0, -0.2617, -1.0)$ and: $\omega_1 = 3000$, $\omega_2 = 100$, $\omega_3 = 100$, $\omega_4 = 0$, $\omega_5 = 2000$.

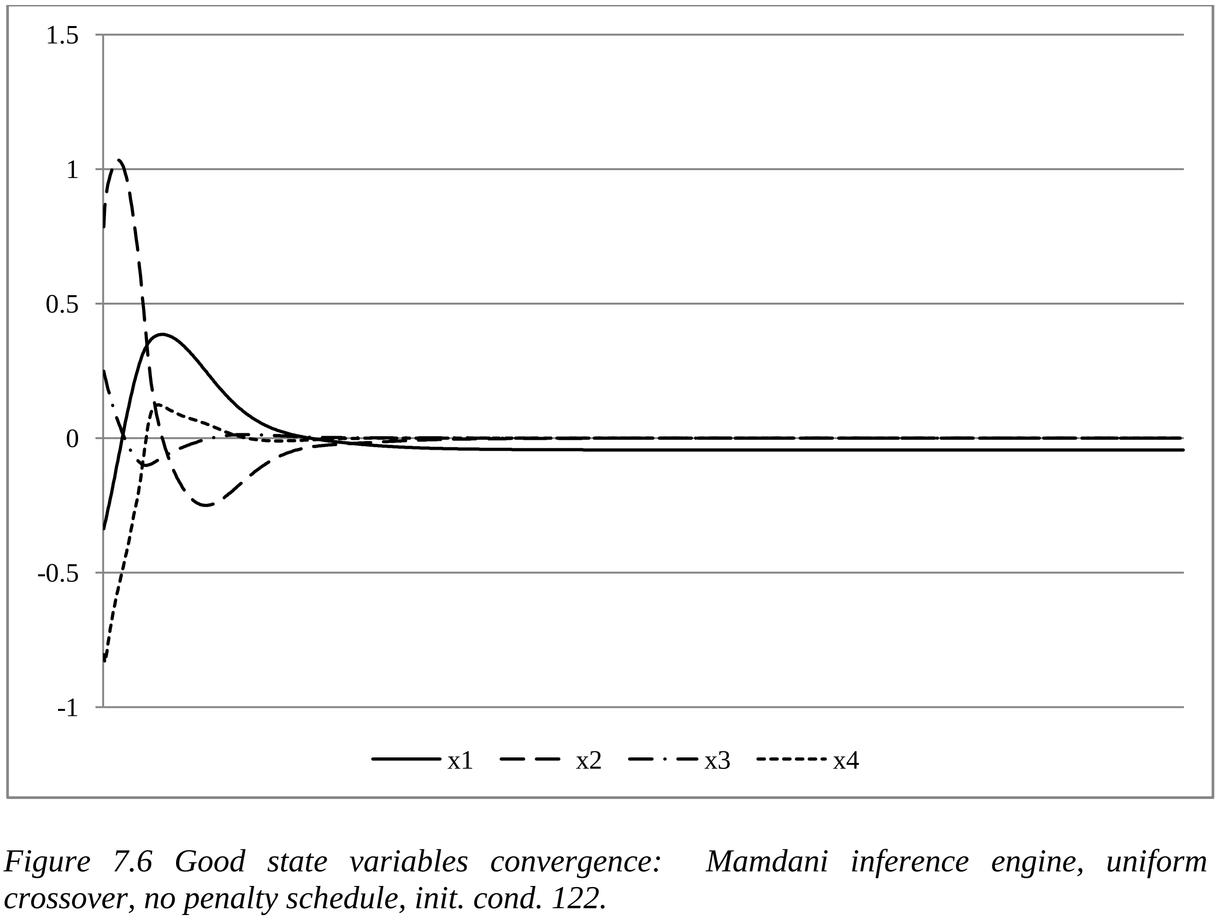

Figure 7.6 Good state variables convergence: Mamdani inference engine, uniform crossover, no penalty schedule, init. cond. 122.

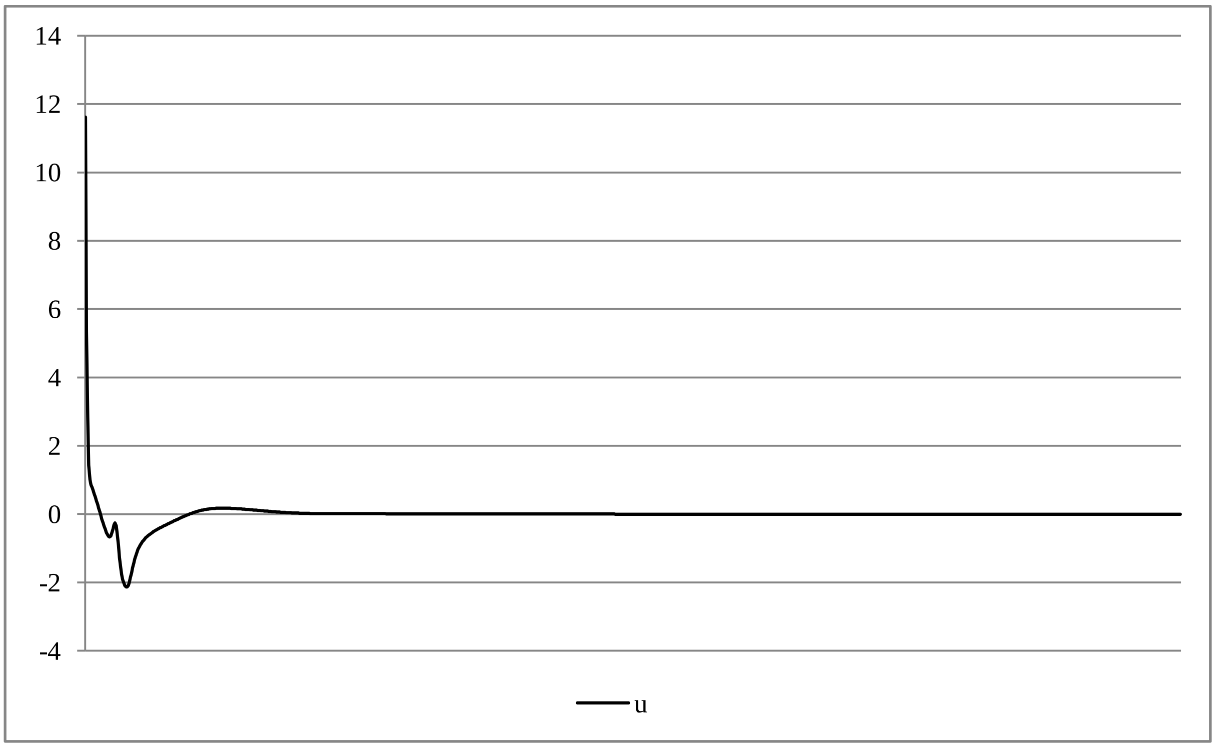

Figure 7.7 Good convergence: controller, init. cond. 122.

7.4.4 Discussion

7.4.4.1 Variations in fitness function parameters

Changing the weights ω in fitness function had a significant impact on the EA performance. In fact, the fifth component of the fitness function ω_5 (survival time) implicitly contains all other components but by specifying them separately the EA process can be influenced, i.e., smaller or bigger bias towards one or another component is introduced. This is especially true for the first component ω_1 that corresponds to the cart's position x_1. As can be seen from some simulation results, see for example Figure 7.2, the EA 'struggled' to drive the cart towards the target area (near the origin). Setting ω_1 to a non-zero value introduces bias in the EA towards x_1, or in plain language: makes it pay extra attention to the cart's position. Effect of penalty schedule depends on other EA parameters. In some cases it improved the results and in some others it had adverse effect. This shows that EA parameters need to be fine-tuned to achieve desired results and that finding the right balance between their values requires extensive experimentation. Alternatively, some of them can be co-evolved with the original population.

7.4.4.2 Effect of penalty schedule

Simulations were run both with penalty schedule-A and schedule-B. To test the impact of above penalty schemes the same simulations were run without any penalties with mixed results that proved that usually not one factor decides on the EA performance but a combination of EA parameters. As mentioned before, applying a penalty schedule can have an adverse effect on the EA. Penalty schedule can be seen as a temporary measure in the absence of a more fitting definition of the fitness function.

7. 4.4.3 Effect of elitism

By introducing strong elitist strategy the convergence of the average value of the objective function across population close to the minimum value of the objective function is achieved. Such a convergence of the population average to the minimum population fitness value is desired as an indication of good EA performance resulting in majority of population being valid control systems. In several simulations the average population fitness is on par with the minimal fitness indicating that almost all individuals in the last population represented the control system of the same or very similar quality.

7. 4.4.4 Product inference engine vs minimum inference engine

It was observed that on average, product inference engine provided faster and smooth convergence of the state variables to the TR but at the cost of the size of controllability region. Minimum inference engine produced better results in terms of larger number of initial conditions from which the controller successfully controlled the system to the TR.

Summary

This chapter presents a novel evolutionary algorithm based compositional method for hierarchical fuzzy control over a large set of initial conditions, including dynamical conditions of the system under investigation. Control system is designed as a three-layered hierarchical fuzzy logic structure. The inverted pendulum system is selected as an example of a dynamical system and used to test the proposed method. The proposed method can be applied to a wide range of dynamical systems with appropriate modifications. Evolutionary population encoding, objective functions, number and range of membership functions, and the hierarchical fuzzy logic structure are case dependant but the overall algorithm covers a large number of control systems.

Chapter 8 MULTIOBJECTIVE EVOLUTIONARY ALGORITHM BASED COMPOSITIONAL METHOD

8.1 Introduction

8.1.1 Motivation for MOEA approach

After development of the single objective EA compositional method there is a question of improving the EA performance. One approach to this task is to investigate how modularising the objective function, i.e., splitting into two or more components, might improve the controller performance. This concerns especially the number of initial conditions from which the controller successfully controls the system to the target region. The single objective EA performance in this respect is not satisfactory. MOEA performance as a multi-objective optimisation method is secondary concern in this investigation which is primarly focused on the effect of modularisation of the objective function.

8.1.2 Basic concepts and terminology

The multiobjective optimisation definitions and terminology are based on the following publications: (Zitzler 1999), (Zitzler *et al.* 2000), (Deb 2001), (Coello Coello *et al.* 2002). Multiobjective optimisation problem can be formulated as follows:

Definition 8.1. *Multiobjective Optimisation (MOP).* Find vector $\vec{x}^* = (x_1^*, \dots, x_m^*)$ in decision space X that minimises/maximises objective vector function:

$$\vec{f}(\vec{x}) = (f_1(\vec{x}), \dots, f_n(\vec{x})) \in Y$$

subject to: $\vec{x} \in X_c \subset X, \quad \vec{f}(\vec{x}) \in Y_c \subset Y$

Usually X_c takes form of inequality and equality constraints:

$$g_i(\vec{x}) \geq 0, \; i = 1, \dots, m$$

$$h_i(\vec{x}) = 0, \; i = 1, \dots, p, \quad p < n$$

Without loss of generality a minimisation problem can be assumed. Maximisation problem can be converted into minimisation problem by the following formula:

$$\max f_i(\vec{x}) = -\min(-f_i(\vec{x}))$$

Definition 8.2. *Pareto Dominance.* Consider two solutions $\vec{x}$ and $\vec{y}$, $\vec{x}, \vec{y} \in X$. Solution $\vec{x}$ dominates $\vec{y}$, denoted as $\vec{x} \prec \vec{y}$, if and only if:

$$\forall_i\, f_i(\vec{x}) \leq f_i(\vec{y}) \text{ and } \exists_j : \; f_j(\vec{x}) < f_j(\vec{y}), \quad i,j \in \{1, \dots, n\}$$

All solutions that are not dominated by any other solution are called non-dominated. The set of all non-dominated solutions is called *Pareto-optimal set*, see the definition below.

Let say, that for any two objective vectors $\vec{u}$ and $\vec{v}$:

$\vec{u} = \vec{v}$ if and only if $\forall_i\; u_i = v_i \quad i \in \{1, \dots, n\}$

$\vec{u} \leq \vec{v}$ if and only if $\forall_i\; u_i \leq v_i \quad i \in \{1, \dots, n\}$

$\vec{u} < \vec{v}$ if and only if $\forall_i\; u_i < v_i \quad i \in \{1, \dots, n\}$

Similarly, relations $>$ and $\geq$ can be defined accordingly.

Definition 8.3. *Pareto-optimal set*. For a given MOP $\vec{f}(\vec{x})$, the Pareto optimal set P is defined as:

$$P = \{ \vec{x} \in X : \sim \exists \vec{x}' \in X \;\; \vec{f}(\vec{x}') < \vec{f}(\vec{x}) \}$$

Definition 8.4. *Pareto Front.* For a given multiobjective optimisation problem defined by the vector function $\vec{f}(\vec{x})$ and Pareto optimal set P, the Pareto Front (*PF*) is defined as:

$$PF = \{ \vec{u} = \vec{f} = (f_1(\vec{x}), \dots, f_n(\vec{x})) : \vec{x} \in P \}.$$

As in the single objective EA described in Chapter 7, the proposed multiobjective evolutionary algorithm based compositional method searches for a controller over the whole set of initial conditions at every generation. Objective function value of each individual reflects the controller performance over the whole set of initial conditions in the set. In other words, every string in the population is assigned the objective function value which is a composite value representing string's performance for every single initial condition in the user-defined set. Each individual represents a potential solution to a given problem. Depending on whether the problem is defined as a maximisation or minimisation problem, the best solution may be the string with the highest or lowest objective function value (the inverted pendulum problem is defined as minimisation problem).

In applying an evolutionary algorithm to multiobjective optimisation three most important issues need to be addressed (Zitzler *et al*. 2000):

- Objective functions evaluation method.
- Selection process to guide the EA evolution towards the Pareto set.
- Maintaining sufficient population diversity to prevent premature convergence and to generate well distributed Pareto Front.

8.2 Multiobjective evolutionary algorithm based compositional method

In the following subsections the multiobjective evolutionary algorithm for compositional method is described on the example of the inverted pendulum. Using case study is required as the MOEA and fuzzy system are strongly coupled and developing the method in general form would be meaningless, apart from the algorithm description.

8.2.1 Defining evolutionary population

The three fuzzy rule base structure, described in Chapter 3, is represented as a linear individual string of M = 95 elements. The population is defined as a set of M_p individuals:

$P = \{ \vec{p}_k\colon\ \vec{p}_k = (a_1, \dots, a_{95}),\ \ k = 1, \dots, M_p,\ \ \ a_j \in \{1, \dots, 7\}\ \}$.

A simple method is used to handle MOEA population size in the algorithm:

1. Initial population $P(0)$ is generated.
2. Pareto set PS is selected from the non-dominated individuals of $P(0)$.
3. If population size of PS smaller than a pre-defined threshold then PS is refilled with either randomly generated individuals or by the use of elitist strategy: filling the reminder of PS with the best (with regard to each and every objective function) individuals from the previous population.
4. MOEA operators are used on PS to generate the next generation $PS1$. Population $PS1$ is copied to P.
5. Steps 2—4 are repeated until termination condition is satisfied.

8.2.2 Objective functions

Assume that there are N_c initial conditions in the user-defined set of initial conditions. To evaluate the objective function value for a given string $\vec{p}_k$ from the controller population first the objective function values f_{ij} ,i = 1, ... , N_c, j = 1,2, are evaluated for every single initial condition. Then overall objective function value is determined from the values calculated for every single initial condition.

The fitness for a single initial condition is evaluated as follows: each string $\vec{p}_k$ is decoded into three components defining the fuzzy knowledge base for each layer, then the Mamdani or minimum inference formula is used to evaluate u_1, u_2, and u to find the final control to be applied at each value of the state $\vec{x}$. The system state equations are integrated by the Runge-

Kutta algorithm (RK4) with step size 0.02 over time interval $[0,T]$. The fitness f_{ij} is then calculated based on the behaviour of the system over the time interval. The objective function f_{i1} has the general form:

$$f_{i1} = \omega_1 F_1 + \omega_2 F_2 + \omega_3 F_3 + \omega_4 F_4 \tag{8.1}$$

with: $F_1 = \frac{1}{N} \sum_1^N \frac{|x_1|}{x_{max}}$, $F_2 = \frac{1}{N} \sum_1^N \frac{|x_2|}{\dot{x}_{max}}$, $F_3 = \frac{1}{N} \sum_1^N \frac{|x_3|}{\theta_{max}}$, $F_4 = \frac{1}{N} \sum_1^N \frac{|x_4|}{\dot{\theta}_{max}}$.

The objective function f_{i2} has the general form:

$$f_{i2} = \omega_5 F_5 \text{ with: } F_5 = \frac{1}{N}(T - T_S), \tag{8.2}$$

where $x_{max} = 1.0$, $\theta_{max} = \pi/6$, $\dot{x}_{max} = 1.0$, $\dot{\theta}_{max} = 3.0$, N is the number of iteration steps, survival time $T_S = 0.02 \cdot N$, $T = 0.02 \cdot N_{max}$ with the maximum number of iterations $N_{max} = 1000$, and ω_k are selected positive weights.

A simple evaluation method is defined, similar to the method described in Chapter 7, the fitness function value (either f_1 or f_2) is evaluated as arithmetic average over all fitness values f_{ij}, $i = 1, \ldots, N_c$, $j = 1,2$, calculated for every single initial condition, and then assigned to a particular string in controller population:

$$f_1 = \frac{1}{Nc} \sum_{i=1}^{Nc} f_{i1} \tag{8.3}$$

$$f_2 = \frac{1}{Nc} \sum_{i=1}^{Nc} f_{i2} \tag{8.4}$$

According to the Definition 8.2, Pareto dominance condition with two objective functions f_1 and f_2 can be expressed as: individual $\vec{a}$ from the population dominates another individual $\vec{b}$ if $\forall i \; f_i(\vec{a}) \leq f_i(\vec{b})$ and $\exists j: \quad f_j(\vec{a}) < f_j(\vec{b}), \quad i,j \in \{1,2\}$.

The restrictions are given by: $-1.0 \leq x_1 \leq 1.0$ and $-\pi/6 \leq x_3 \leq \pi/6$.

Objective functions can be modified in order to reward strings which successfully control the system from a large number of initial conditions. One of the simplest methods is to establish threshold values for the objective function and penalise strings that exceed those threshold values (for each initial condition), see penalty schedules-A and B, Chapter 4. In MOEA experiments penalty schedule-A is used:

```
if ObjFun ≥ 0.3·avg and ObjFun < 0.5·avg  then ObjFun = ObjFun + 500.0
if ObjFun ≥ 0.5·avg and ObjFun < 0.8·avg  then ObjFun = ObjFun + 1000.0
if ObjFun ≥ 0.8·avg then  ObjFun = ObjFun + 2000.0
```

where `avg` is a variable representing average objective function value (either f_1 or f_2) of the previous population (in a preceding generation). Penalties need to be fine-tuned to focus the MOEA on selecting strings that perform well for the large number of initial conditions.

8.2.3 Membership functions

Gaussian functions are used as membership functions in all experiments. To increase accuracy and achieve better performance from control system one can increase the number of membership functions covering input and output variables. This, however, increases the size of the knowledge base and therefore with larger number of rules to be learned the computation time is longer.

8.2.4 Crossover and mutation

For MOEA compositional method four crossover operators were tested: random, uniform, arithmetic, and one-point crossover. The use of one-point crossover was abandoned after several simulations that provided inferior results to results from simulations run with other crossover operators. It proves only that in this particular application one point-crossover is not suitable (it was used in initial experiments, see Chapter 5).

The mutation operator, described in Chapter 4, mutates elements of the strings in the population to ensure satisfactory diversity within the population which is required for the MOEA to find better approximate solutions to the problem. Mutation is undertaken with probability p_m determined by a mutation schedule that decreases typically from 0.8 to 0.001 over the fixed number of generations. The same mutation schedule is used as described in Chapter 7.

8.2.5 Overview of the multiobjective evolutionary algorithm

The MOEA is used to learn fuzzy rules in the HFS that constitutes a control system for the inverted pendulum. A schematic MOEA algorithm, in a very general form, is given below:

1. MOEA parameters are selected: type of inference engine, crossover, mutation schedule, selection method, elitism, objective functions (with or without a penalty schedule), and termination condition (number of generations or lack of significant change in the state vector).
2. Population $PS(t)$, $t = 0$, is randomly initialised: every component of individual string is given a randomly selected value from a predefined interval. Objective functions are evaluated for the first generation $PS(0)$.
3. $t = t + 1$: next generation is created using MOEA operators: selection, crossover, and mutation.
4. Pareto set $PS1(t)$ is created from the non-dominated solutions of $PS(t)$.

5. If *size*(*PS1*(*t*)) < *Threshold* then the population is filled to its maximum size (maintaining the constant population size) with randomly generated individuals or by use of the elitist strategy: copying the best individuals (with respect to each objective function) from *PS*(*t*).

6. Individual from the population *PS1*(*t*) is selected.

7. Initial condition is selected from the predefined set.

8. Dynamical system is simulated from a given initial condition.

9. Final state of state vector is determined.

10. Based on values from Step 9 objective functions values are evaluated for an individual in population *PS1*(*t*). Penalties are added to the objective functions values (if penalty schedule is defined).

11. Steps 7—10 are repeated until all system simulations for every initial condition in the set are performed.

12. An average over all initial conditions is calculated for each objective function and assigned to the individual. Optionally, fitness function value is evaluated as a function of objective functions values (usually as a linear combination of component objective functions).

13. Steps 6—12 are repeated until all individuals in the population have their objective functions values evaluated.

14. Pareto set *PS2*(*t*) is selected from the population *PS1*(*t*).

15. Pareto set *PS2*(*t*) is copied to *PS*(*t*).

16. Steps 3— 5 are repeated until the termination condition is satisfied.

17. Final control system is given by any individual from the Pareto set. Its performance is evaluated by running a simulation of the dynamical system for all initial conditions and counting initial conditions for which the final state variables are within the target region.

A simplified MOEA based compositional method block diagram is shown in Figure 8.1.

8.3 Experimental setup

The experiments were conducted to test the proposed method. The experiments aimed at finding the combination of MOEA parameters that would result in finding fuzzy rules capable of successfully controlling the inverted pendulum system to the target area from the largest possible number of initial conditions. In the following subsections the computer simulations setup is described and then typical results are illustrated on the successful versions of MOEA algorithm.

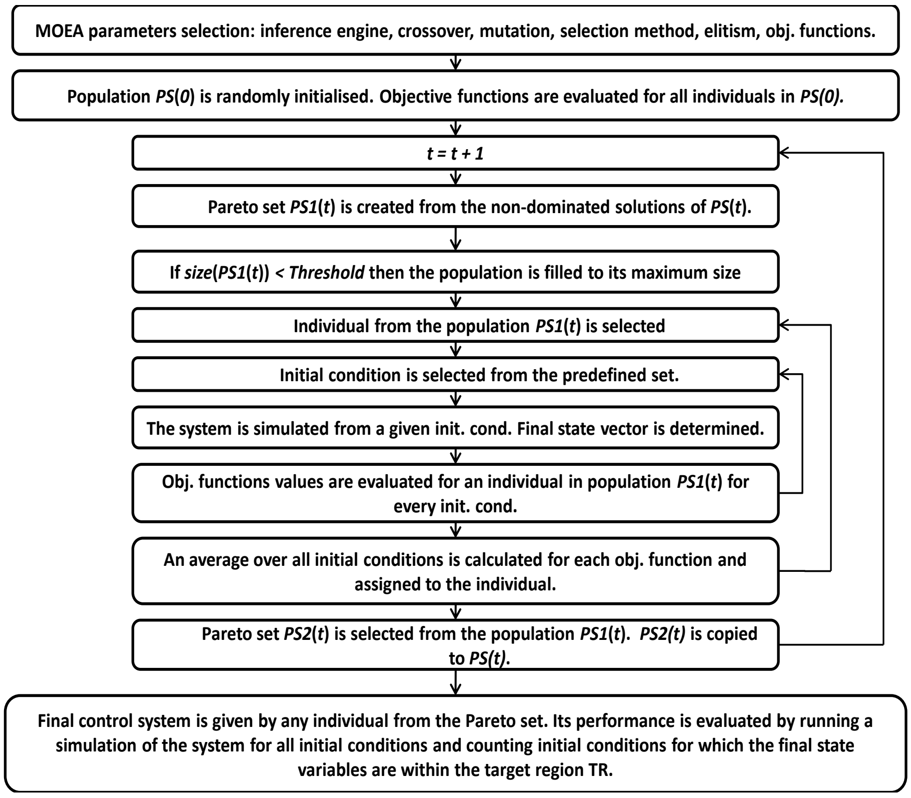
MOEA parameters selection: inference engine, crossover, mutation, selection method, elitism, obj. functions.
Population PS(0) is randomly initialised. Objective functions are evaluated for all individuals in PS(0).
t = t + 1
Pareto set PS1(t) is created from the non-dominated solutions of PS(t).
If size(PS1(t)) < Threshold then the population is filled to its maximum size
Individual from the population PS1(t) is selected
Initial condition is selected from the predefined set.
The system is simulated from a given init. cond. Final state vector is determined.
Obj. functions values are evaluated for an individual in population PS1(t) for every init. cond.
An average over all initial conditions is calculated for each obj. function and assigned to the individual.
Pareto set PS2(t) is selected from the population PS1(t). PS2(t) is copied to PS(t).
Final control system is given by any individual from the Pareto set. Its performance is evaluated by running a simulation of the system for all initial conditions and counting initial conditions for which the final state variables are within the target region TR.

Figure 8.1 MOEA based compositional method block diagram.

8.3.1 Initial conditions

A set of regularly interspaced 255 initial conditions (N_c = 255) is defined within the region defined by: $|x_1| \leq 0.75$, $|x_2| \leq 1.0$, $|x_3| \leq \pi/12$, $|x_4| \leq 1.0$, see Appendix.

8.3.2 Initial population

The initial population $P(0) = \{ \vec{p}_k\colon\ k = 1, \dots , M_p \}$, is determined by choosing the a_j as a random integer $\in [1,7]$ where M_p is the size of the evolutionary population. Full replacement policy is used and for selection process tournament selection with size $n_T = 4$.

8.3.3 Population size

The initial population size is set at $M_p = 500$. Smaller population size is possible but then it is more difficult to maintain the required population diversity to avoid pre-mature convergence to the local minimum.

8.3.4 Termination condition

Similarly to single objective EA simulations, the MOEA is terminated after 1000 generations as it is found that the algorithm either finds solution in about 300—500 generations or fails regardless of how many generations follow.

8.3.5 Elitism

An elitism strategy is typically used to pass the fittest individuals or copies of the fittest individual to the new population, so that the information encapsulated in the best individual is not lost and the fittest individuals are passed into the next generation. A variable number of copies of best individuals in terms of f_1 and f_2 were passed to the next generation to maintain the fixed population size.

8.3.6 Objective functions

Objective functions given by Equation 8.3 and 8.4 are used in all experiments. The weights in the objective functions, see Equation 8.1 and 8.2, are adjusted in several simulations with most commonly used sets of ω parameters: $\omega_1 = 1000$, $\omega_2 = 1$, $\omega_3 = 1000$, $\omega_4 = 1$, $\omega_5 = 3000$, and also $\omega_1 = 1$, $\omega_2 = 1$, $\omega_3 = 1$, $\omega_4 = 1$, $\omega_5 = 1$. Changing the weights ω in the objective functions has a significant impact on the MOEA performance.

8.4 Computer simulations

Simulations are run to fine-tune the EA parameters that in turn result in finding fuzzy rules capable of successfully controlling the inverted pendulum system to the target area from the largest number from a pre-defined set of 255 initial conditions. In a computer simulation the controller acted on the inverted pendulum for $T_f = 20.0$s.

The successful, in terms of convergence, modification of MOEA (test-1) has the following parameters: minimum inference engine, uniform crossover, mutation schedule, and tournament selection. Termination condition: exceeding 1000 generations. Positive weights in the objective functions were defined as: $\omega_1 = 1000$, $\omega_2 = 0$, $\omega_3 = 1000$, $\omega_4 = 0$, $\omega_5 = 3000$. Population $P(t)$, $t = 0$, is randomly initialised, with every component of individual string given by a randomly selected value from [1,7]. If $size(PS1(t)) < 0.3 \cdot MAXPOP$ (where $MAXPOP$ is maximum population size) then the best individuals with regard to f_1 and best individuals with regard to f_2 are copied to the 2/3 of the reminder of the population $P(t+1)$. The remaining reminder is filled with individuals from the previous population $P(t)$. The algorithm follows the methodology described in Section 4.5. This version of MOEA found 43 non-dominated solutions that are the approximation of the true Pareto Front.

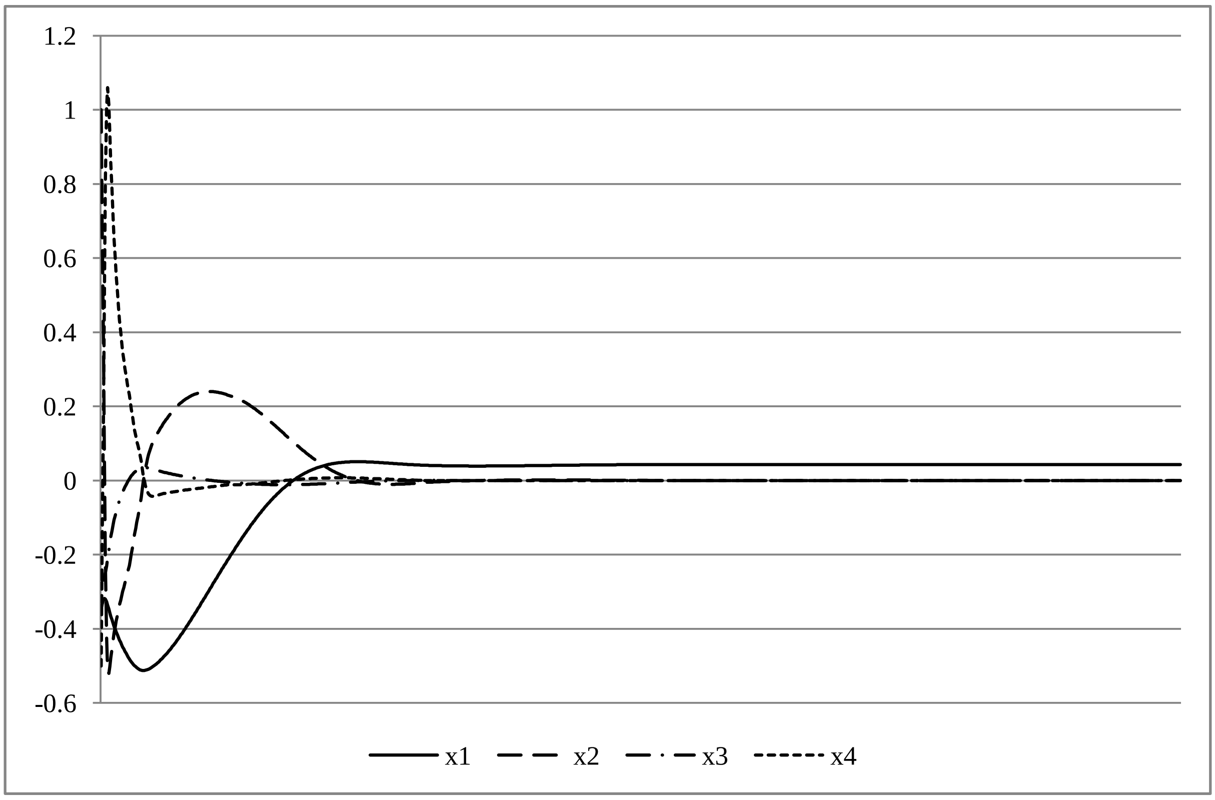

Figure 8.2 State variables convergence for init.cond. 78 (test-1) – controller no 1.

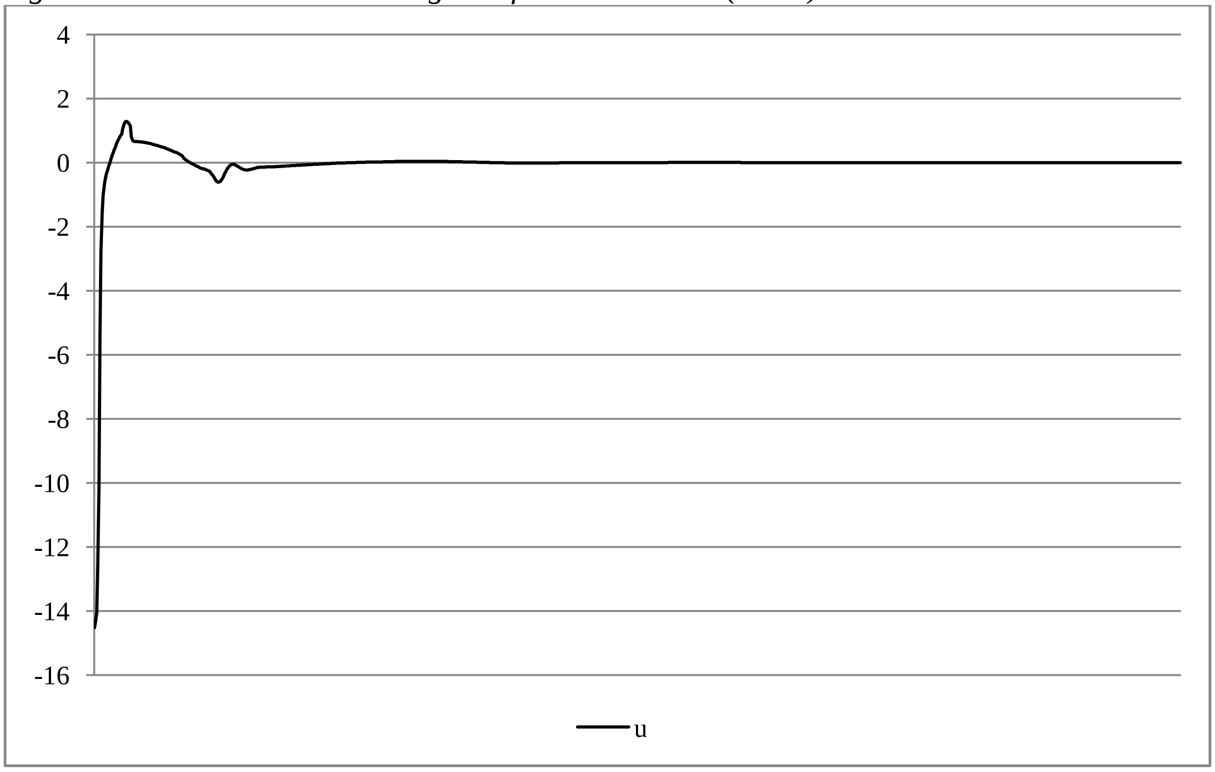

Figure 8.3 Controller no 1, init. cond. 78 (test-1).

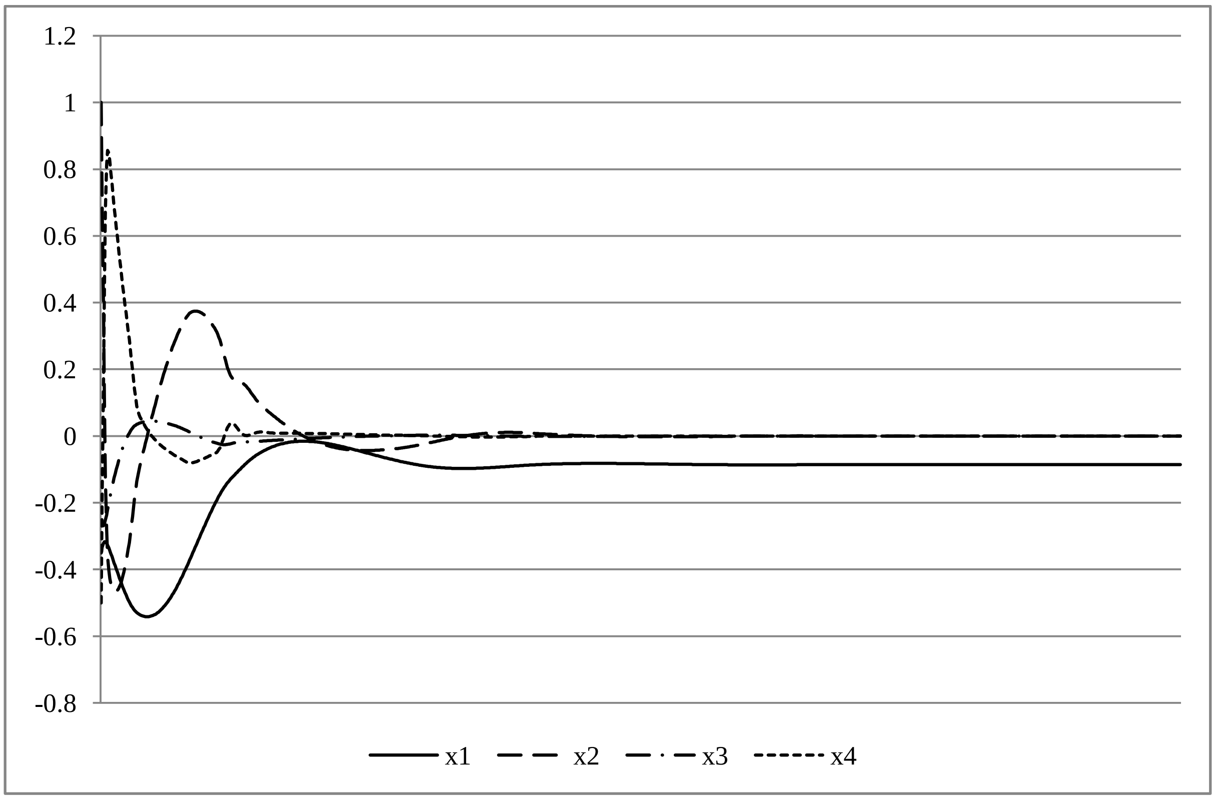

Figure 8.4 State variables convergence for init.cond. 78 (test-1) – the best controller no 8.

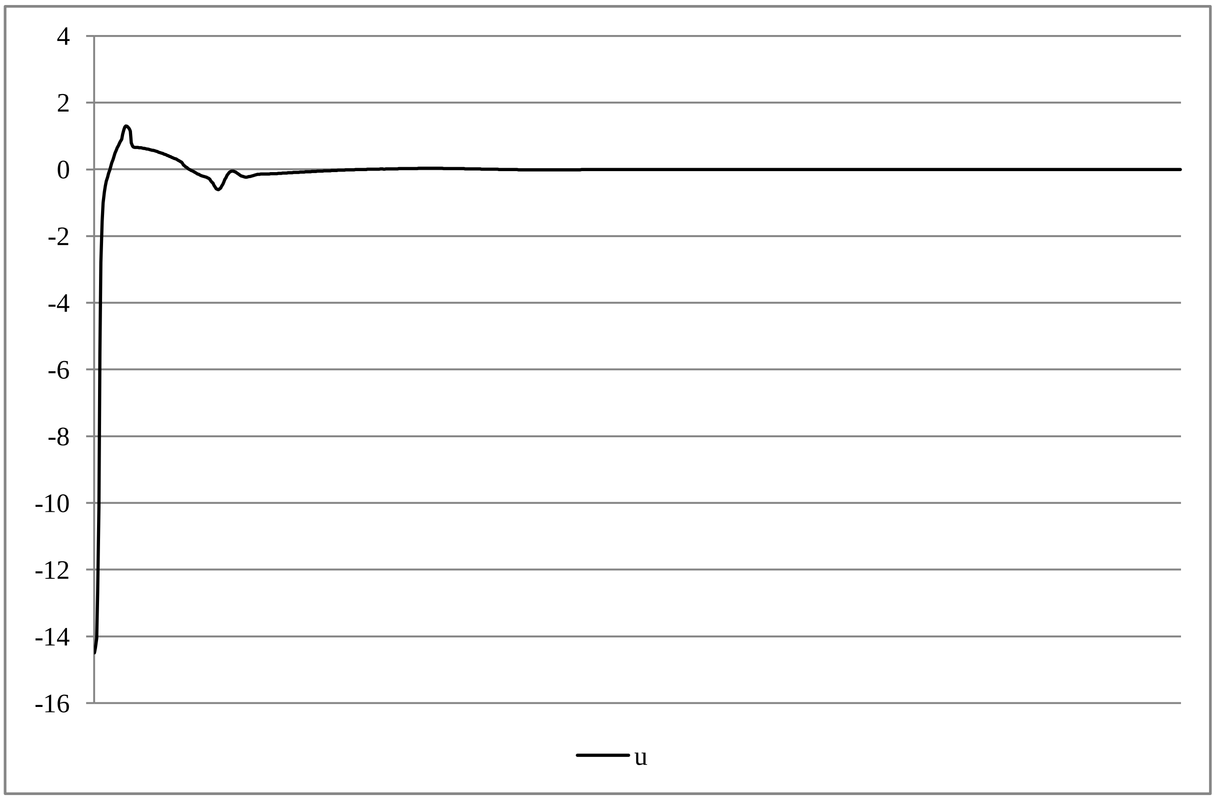

Figure 8.5 Controller no 8, init. cond. 78 (test-1).

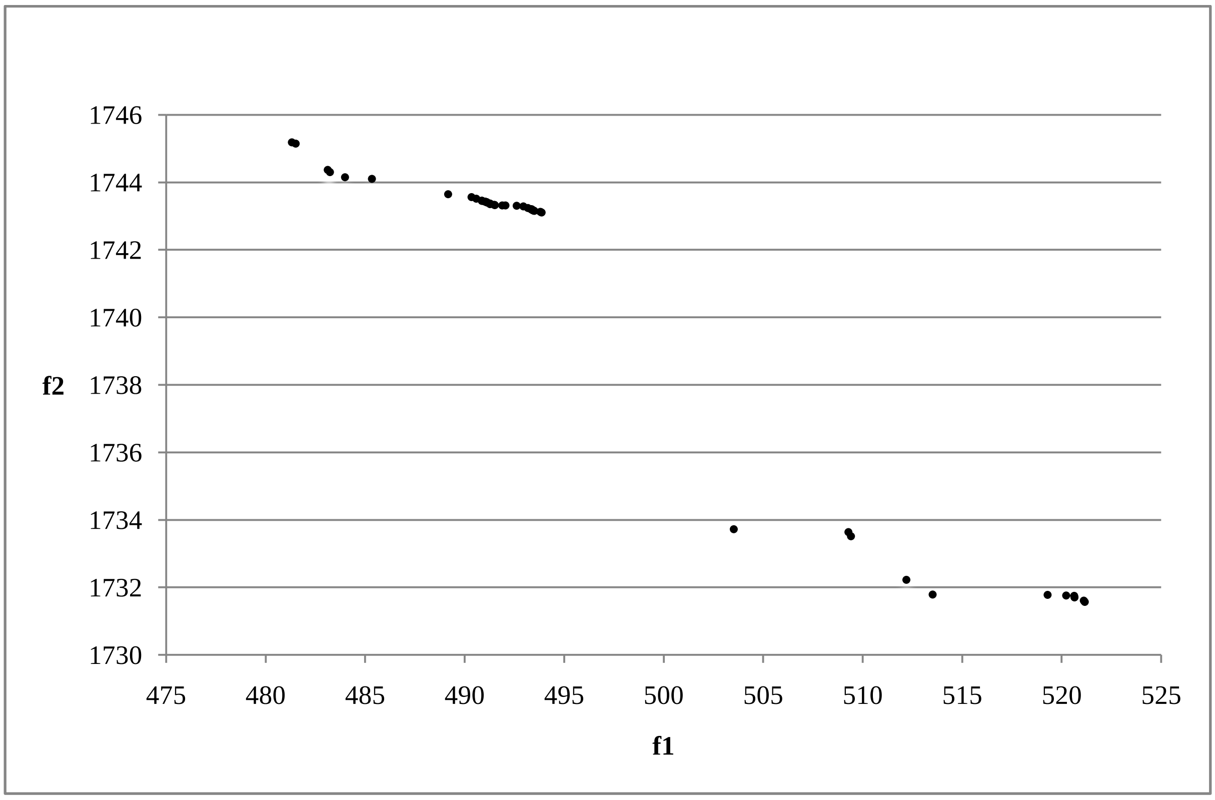

Figure 8.6 Pareto Front approximation for the MOEA (test-1) simulation: 43 solutions.

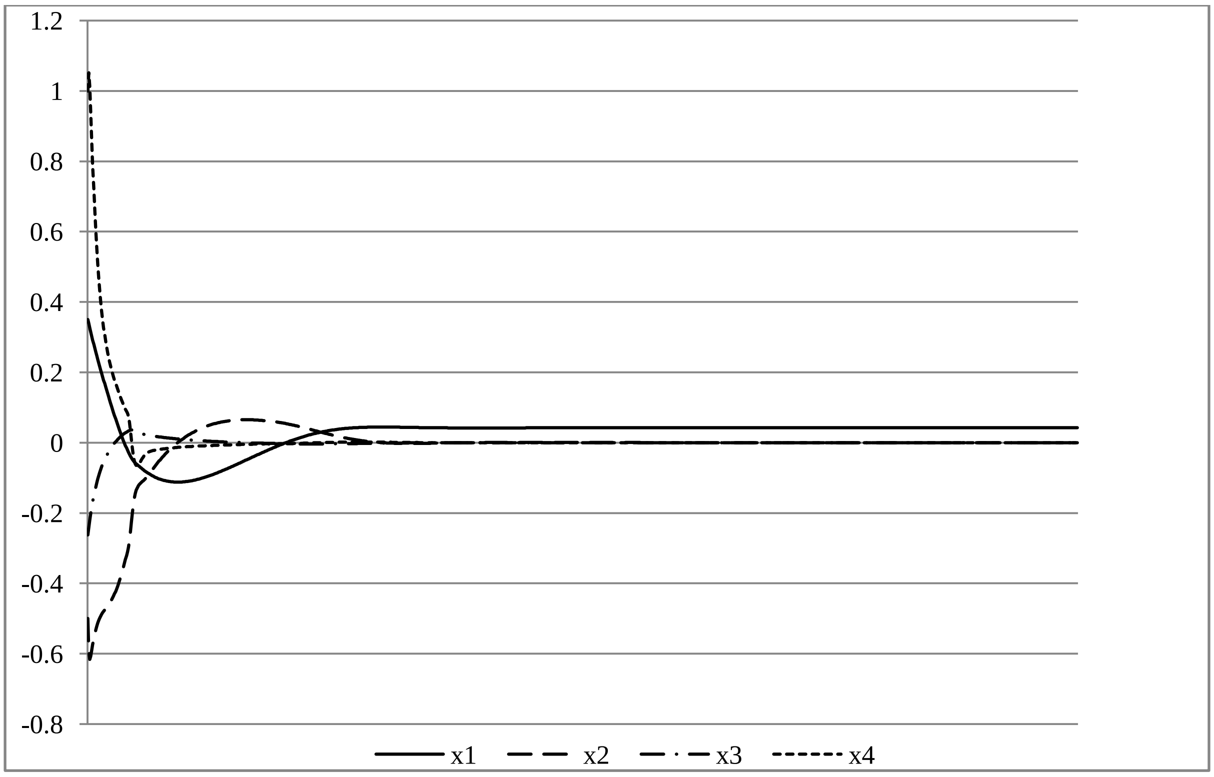

Figure 8.7 State variables convergence for init.cond. 136 (test-2).

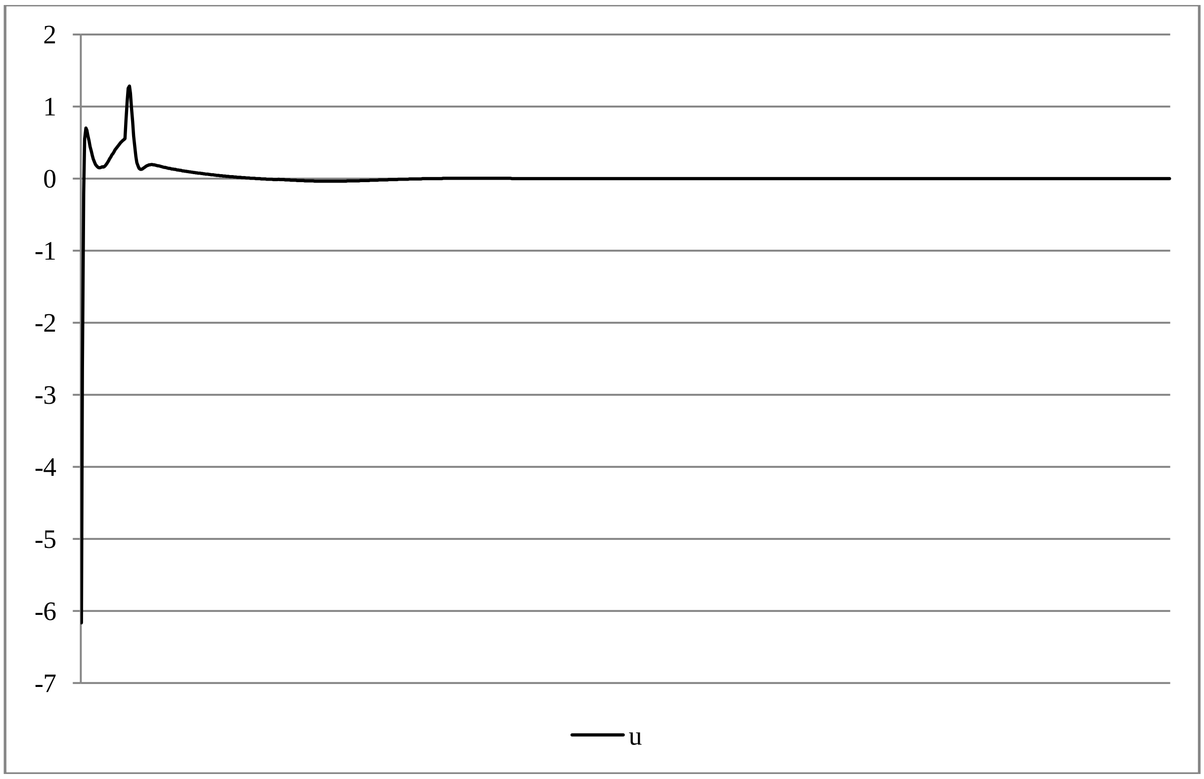

Figure 8.8 Controller no 1, init. cond. 136 (test-2).

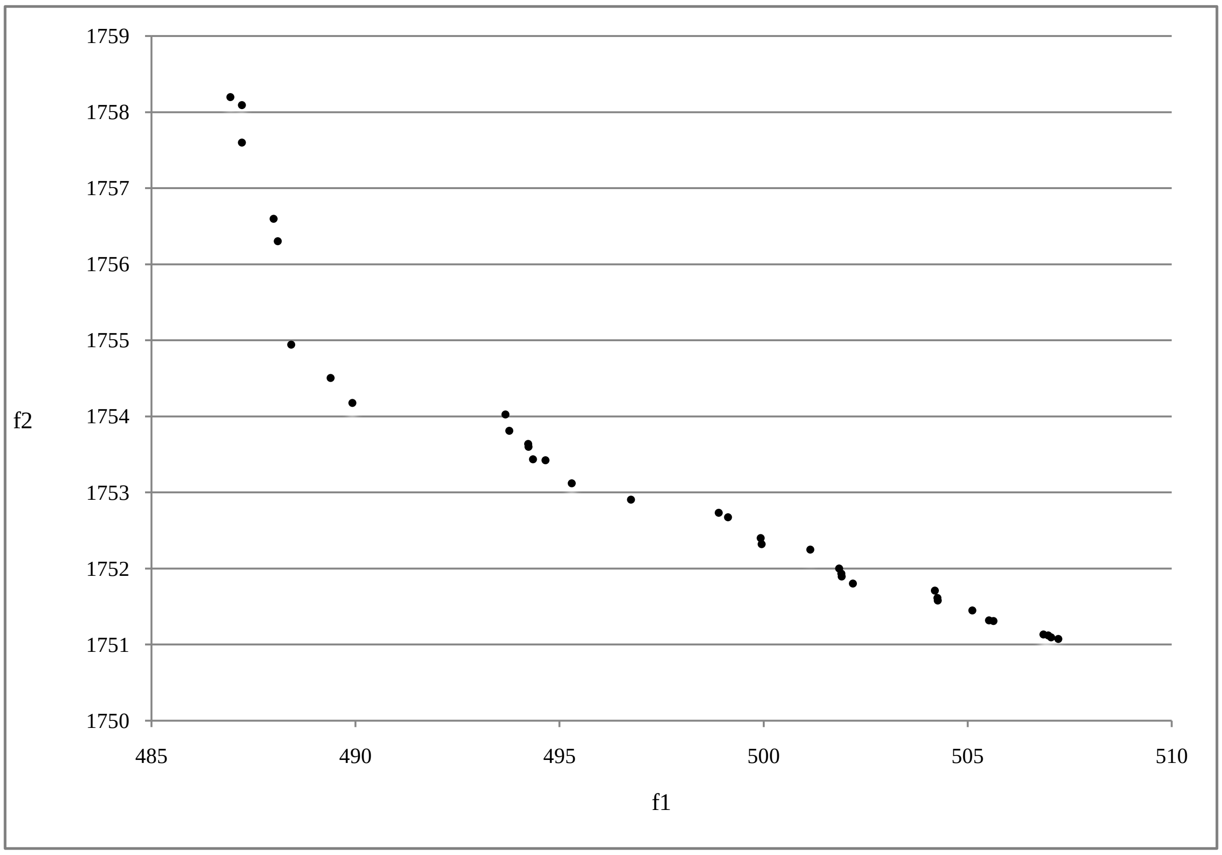

Figure 8.9 Pareto Front approximation for the MOEA (test-2) simulation: 35 solutions.

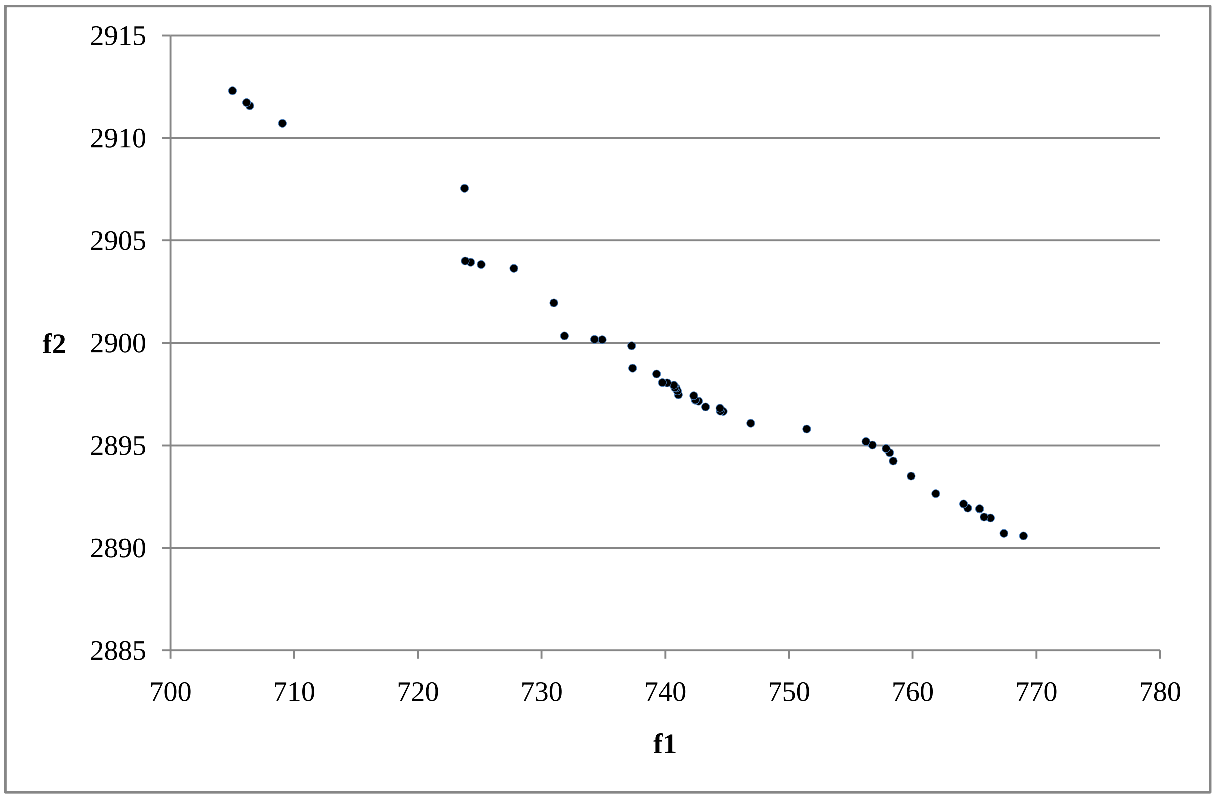

Figure 8.10 Pareto Front approximation for MOEA (test-3) simulation: 48 solutions.

The reason for a very low percentage of success in 10-th controller is that it narrowly failed to keep the cart (represented by x_1) within the target region. In majority of failed control actions the margin was about 0.001. However close the x_1 was to the target region it was still outside it in the final state at T = 20.0s and therefore was considered a failure. Average success rate in test-1 for all 43 controllers in the Pareto set is 81.4% (208 init. cond.) including anomalous controller no 10, and 83.3% (212 init. cond.) without anomalous controller no 10.

Results are illustrated on randomly selected initial condition 78: $\vec{x}_0$ = (0.35, 1.0,−0.261799, −0.5), for the first and 8-th controller in the Pareto set, see Figure 8.2 and Figure 8.3 (first controller) and Figure 8.4 and Figure 8.5 (8-th controller). This first controller achieved 81.4% success rate, controlling the system from 207 (out of 255) initial conditions to the target region. The best controller, see Table 8.1, is controller no 8 with 94.5% success rate (241 convergences to the TR out of total 255 initial conditions).

Another successful MOEA version (test-2) used product inference engine and $\omega_1 = 1$, $\omega_2 = 1$, $\omega_3 = 1$, $\omega_4 = 1$, $\omega_5 = 1$ and found 35 non-dominated solutions. This result is illustrated by the state variables convergence from a selected initial condition 136: $\vec{x}_0$ = (0.3,−0.5,−0.261799, 1.0). The controller is selected as the first solution in the Pareto set. Out of 255 initial conditions, the controller successfully controlled the system from 184 initial conditions (about 72% success rate) to the TR, see Figure 8.7 and Figure 8.8.

The same MOEA (test-3) but with different set of weights in the objective functions: ω_1 = 1000, ω_2 = 0, ω_3 = 1000, ω_4 = 0, ω_5 = 3000, found 48 non-dominated solutions, see Figure 8.10. Please note the different objective functions values resulting from different weights values ω used in definition of objective functions. Better approximation of Pareto Front can be generated with the increase of population size but it would also significantly increase the computation time.

In experiments with a single objective EA, as described in Chapter 7, similar EA parameters were used, which enables an adequate comparison with the MOEA results. The objective function is defined as: $f = \omega_1 F_1 + \omega_2 F_2 + \omega_3 F_3 + \omega_4 F_4 + \omega_5 F_5$, which is simply $f = f_1 + f_2$. MOEA solutions consistently approached or, as in most cases, exceeded 80% success rate while single objective EA averaged 50%, see (Zajaczkowski and Verma 2008). The best MOEA based method result is 94.5% success rate.

Better MOEA results seem to indicate that splitting the objective function into its composite parts might improve the controller performance. In case of the inverted pendulum example one objective function is defined as a measure of state variables ‘distance’ to the target region and second one as survival time (the total time in which the pole and cart remain within specified bounds). This split represented two different aspects of the inverted pendulum problem even though they are strongly coupled. The state variables convergence for the same initial condition no 78 (as shown in Figure 8.2) but for the single objective EA is shown in Figure 8.13 and Figure 8.14.

The MOEA (test-1, see Figure 8.2) final state for initial condition no 78 is: $\vec{x}_f$ = (0.043230, 0.0, 0.0, 0.0). The single objective EA (see Figure 8.13) final state for initial condition no 78 is: $\vec{x}_f$ = (0.00145, 0.0, 0.0, 0.0). In terms of state variables convergence both method performed on par, which is not surprising as they share the same EA parameters (with different variations in values), same fuzzy system and membership functions. What distinguishes them is the definition of the objective function and algorithm that accommodates such a modification (MOEA).

Table 8.1 Number of convergences to TR and success rates, test-1.

Controller	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
No of Init.Cond.	207	209	209	206	203	202	215	241	205	3
Percentage	81.2	82.0	82.0	81.0	79.6	79.2	84.3	94.5	80.4	1.2
Controller	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
No of Init.Cond.	212	220	203	213	209	210	221	212	214	206
Percentage	83.1	86.3	79.6	83.5	82.0	82.6	86.7	83.1	83.9	81.0
Controller	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**
No of init. cond.	203	216	218	226	198	211	225	220	217	208
Percentage	79.6	84.7	85.5	88.6	77.6	82.7	88.2	86.3	85.1	81.6
Controller	**31**	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**
No of init. cond.	220	216	224	211	202	215	206	216	211	205
Percentage	86.3	84.7	87.8	82.7	79.2	84.3	81.0	84.7	82.7	80.4
Controller	**41**	**42**	**43**							
No of init. cond.	206	219	212							
Percentage	81.0	85.9	3.1							

MOEA also allows greater flexibility in algorithm design. Considering arbitrary dynamical system to be controlled, the objective functions can reflect various aspects of the system. They can be adjusted to achieve better control system performance, as proven by comparison of the single objective EA and MOEA solutions.

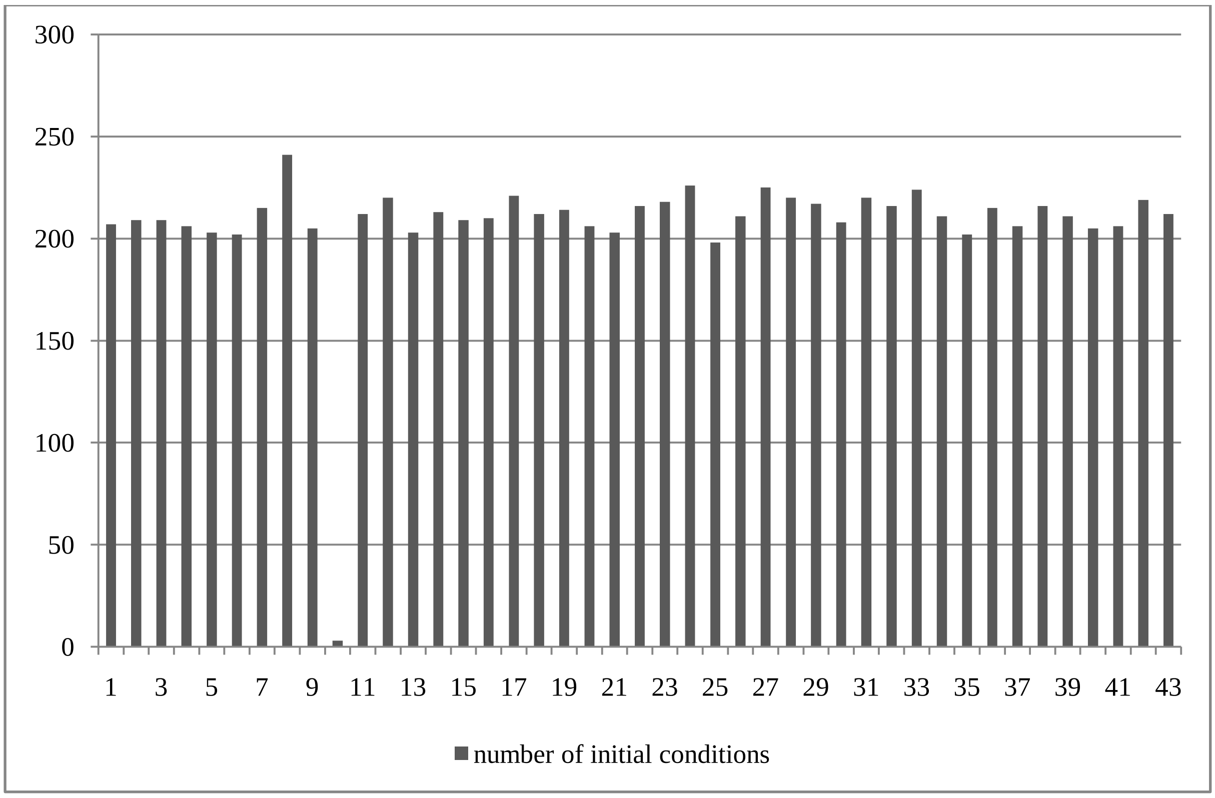

Figure 8.11 Number of convergences for every controller (43 controllers) in the Pareto set (test-1).

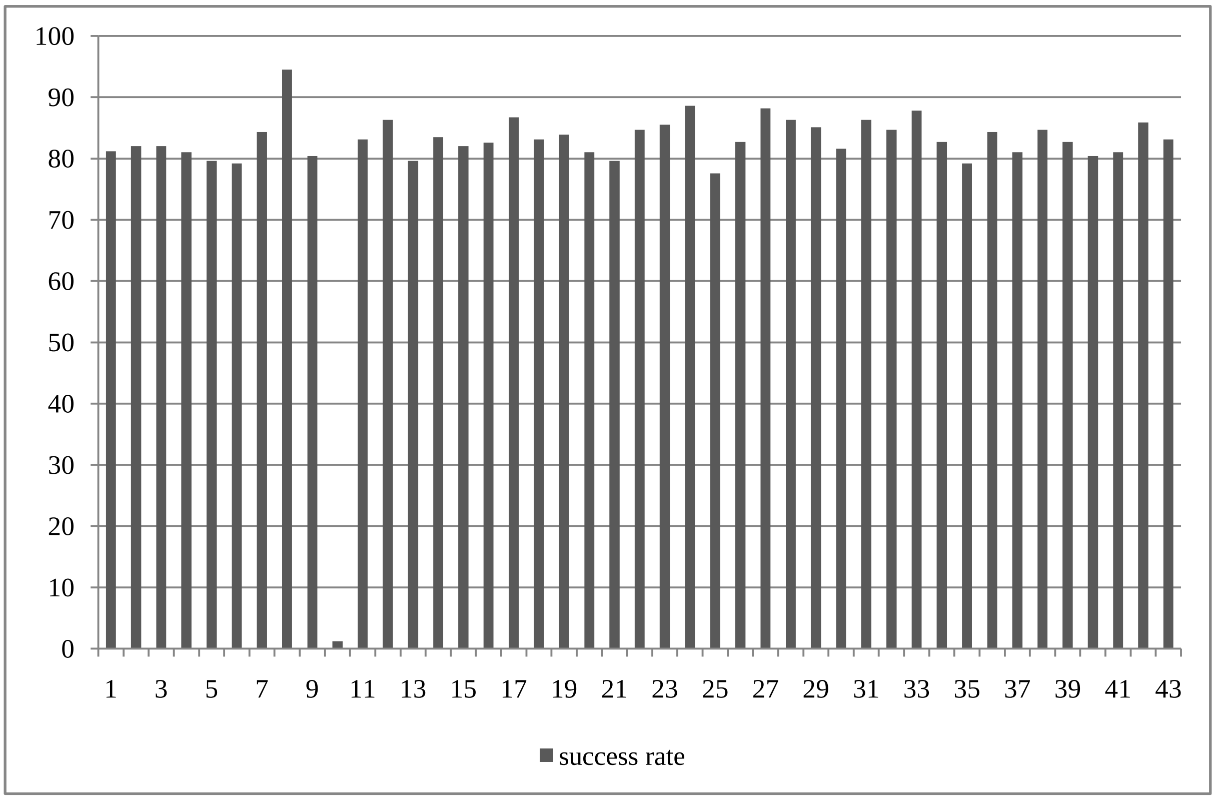

Figure 8.12 Success rate for every controller (43 controllers) in the Pareto set (test-1).

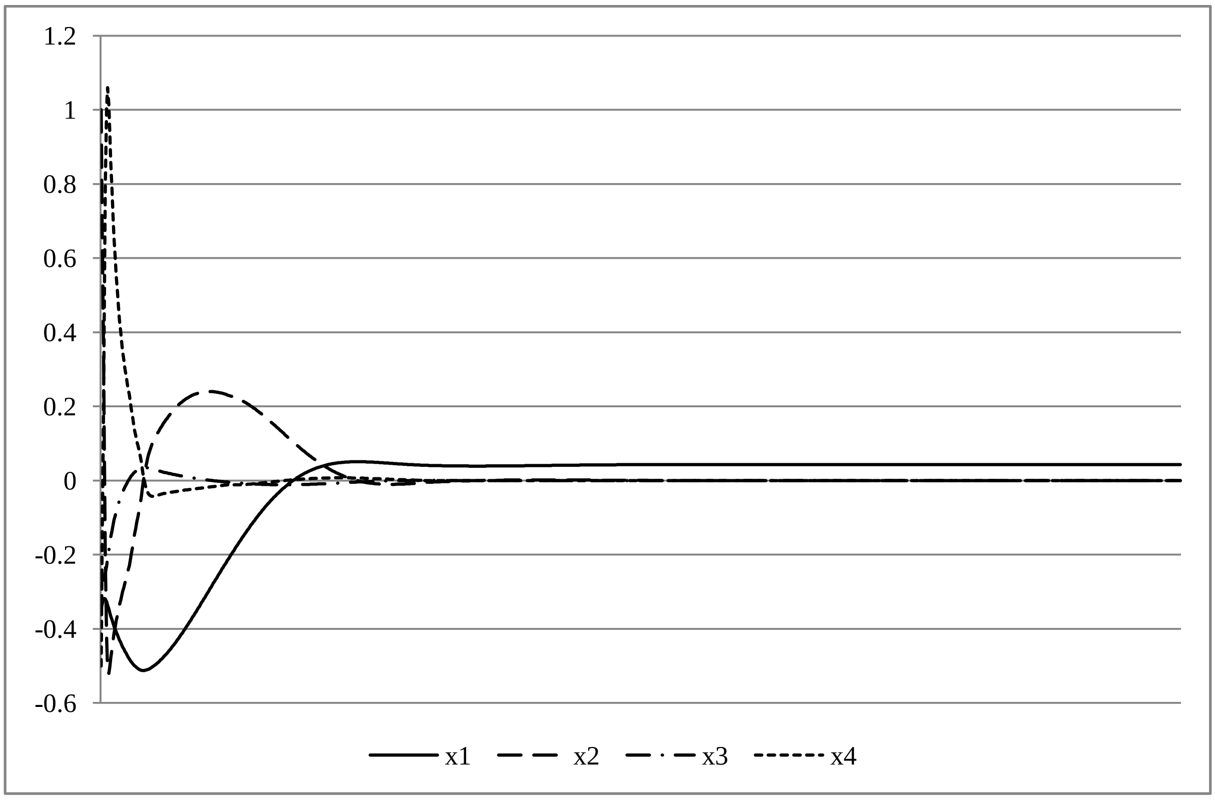

Figure 8.13 Single objective EA state variables convergence for init. cond. 78.

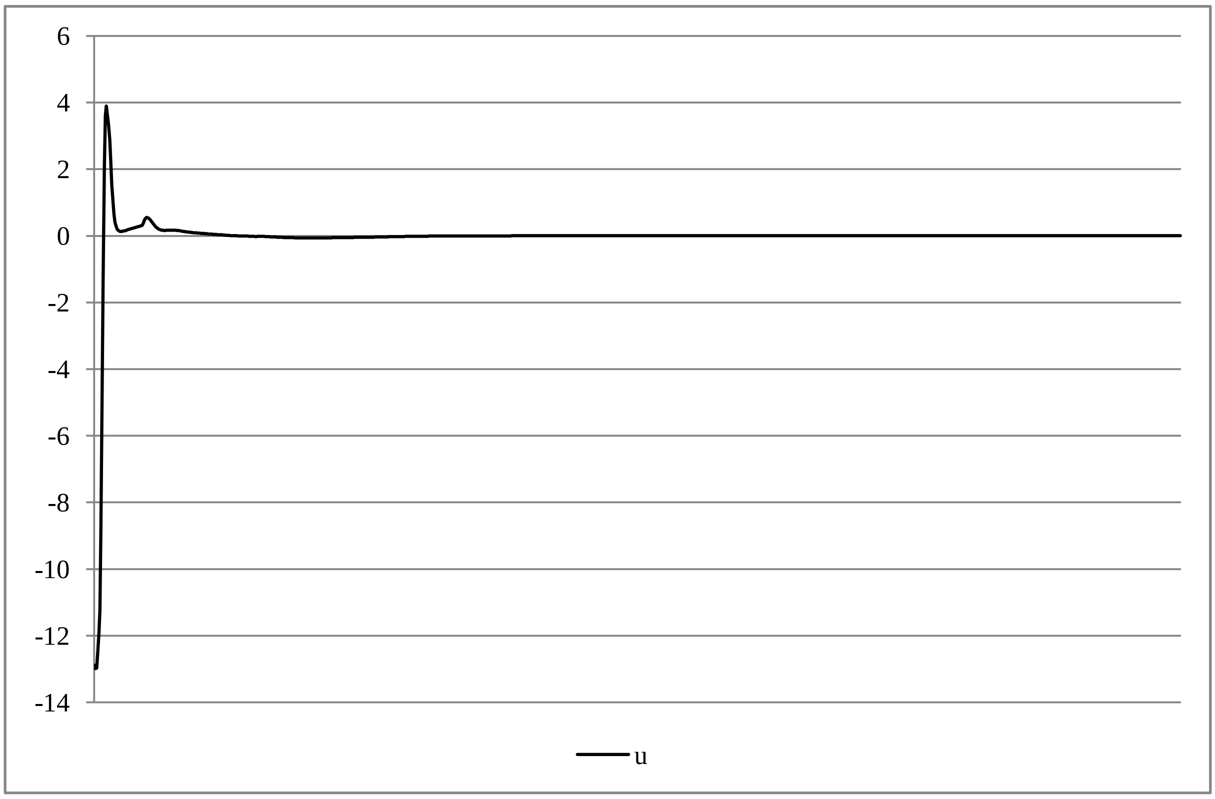

Figure 8.14 Single objective EA controller, init. cond. 78.

Summary

In this chapter the multiobjective evolutionary algorithm based compositional method is introduced as an extension of the method presented in Chapter 7 for a single objective function. Detailed algorithm description is provided followed by simulation results on the example of the inverted pendulum system.

Two objective functions are defined for multiobjective evolutionary algorithm by splitting single objective EA fitness function into its components: sum of deviations of state variables from the origin as the first objective function and survival time (the total time in which the pole and cart remain within specified bounds) as a second objective function. A constant population size is maintained after selection of the Pareto set by the use of elitist strategy.

The multiobjective modification of the EA (modularising the objective function) provides better and more consistent results than single objective compositional method described in Chapter 7. The controller performance from the Pareto set for one particular MOEA version is presented in terms of the number of initial conditions from which the system was controlled to the target region. A satisfactory controller is developed for the set of pre-defined initial conditions with the controlled system controlled to the target region from 94.5% of all initial conditions.

Chapter 9 CONCLUSIONS AND FUTURE DIRECTIONS

9.1 Conclusions

In the following subsections the final conclusions of the research carried out in this thesis are presented. Future directions of the current research are discussed briefly afterwards.

9.2 Topologies for hierarchical fuzzy structures

The hierarchical fuzzy control of the simple inverted pendulum was examined and evolutionary algorithm was used to learn a fuzzy controller for all possible hierarchical topologies: single layer, two-layered HFS, three-layered HFS, and four-layered HFS with different input configurations. It has been shown that it is important to select the correct input variables into the first layer to achieve effective and accurate control. Furthermore, structure of the second and third layer in the 3 layered HFS plays a significant role as reversed order in input in those layers produced dramatically different results, as shown for example in case of *L3-13-2-4* (poor results) and *L3-13-4-2* (good results), see Figure 5.21 and Figure 5.23 respectively. Similarly, controller *L3-14-2-3* shows poor results and *L3-14-3-2* good results, see Figure 5.29 and Figure 5.31 respectively. Both cases illustrate how intricate interdependencies between input variables can be.

It was shown that the inverted pendulum system should be decomposed into two input variables groupings:

- cart variables: x_1 and x_2 (cart's position and its velocity).
- pole variables: x_3 and x_4 (pole angle and pole's angular velocity).

Results from the 4-layered HFS simulations established that the most influential variable is x_4 (angular velocity), then x_3 (pole angle) in the first grouping followed by x_2 (cart's velocity) and x_1 (cart's position) in the second grouping. Simulation results obtained for the two, three and four layered HFS confirm that it is important to control the inverted pendulum, by examining first its angular speed and angular position then the cart's speed and position displacement, as the best overall results (in terms of state variables convergence and control magnitude) are achieved in both cases by topologies *L2-34-12* and *L3-34-2-1*, *L3-34-1-2*, and *L4-4-3-2-1*, see Figure 5.5, Figure 5.17, Figure 5.19, and Figure 5.41.

Alternatively, a good result is achieved when cart velocity and pole angle are examined first and then the cart velocity and angular velocity of the pole, which is represented by topology *L3-23-4-1*. A small change in the input configuration, as in *L3-23-1-4*, renders this topology one of the worst performers and proves how important is to find the right topology for the control system.

Three-layered topology breaks strong interdependence between state variables in layers 2 and 3 but it does not have adverse effect on the controller performance for topologies *L3-34-1-2* and *L3-34-2-1* as this decomposition reflects physical properties of the system (ranking of the most influential variables). For topologies *L3-14-2-3* or *L3-14-3-2* the difference in decomposition has a profound effect as can be seen in Figure 5.29 and Figure 5.31. Decomposition needs to reflect the physical properties of the system under consideration and it requires grouping of the input variables along weak interdependences between state variables. The inverted pendulum can be decomposed into two subsystems: the cart represented by x_1 and x_2, and the pole represented by x_3 and x_4. Swapping the input variables between the layers but preserving to some extent abovementioned groupings has little effect on the controller performance. When this grouping principle is broken, the results are often detrimental (depending which variables are more influential in the dynamical system). The simulation for 4-layered topologies show that the topology *L4-4-3-2-1* is the most consistent controller in ten different simulations indicating the ranking of the most influential input variables: first - x_4, second - x_3, third - x_2, and finally x_1.

The initial population (randomly generated in the simulations) has significant impact on the evolution of the knowledge base. Some controllers, from ten control systems developed for each topology, differ considerably in their performance. Therefore a simulation resulting in a single controller should not be regarded as a sufficient representation of controllers developed for any particular topology. Especially, if the EA does not produce a relatively uniform population at the end of the algorithm. Developing a relatively homogenous set of controllers requires careful fine-tuning of the EA parameters and usually a large number of generations.

In most cases, in spite of their differences, the controllers perform in a very similar fashion, i.e., having similar control time history and the character of state variables convergence.

In the case of a single layer topology *L1-1234*, the controller stabilises the system relatively well, with no preference given to any input variable, and interdependence between input variables (being locked in the fuzzy rules) remains hidden. Only by decomposition of the

HFS (by breaking a single knowledge base into a hierarchically structured knowledge base) this interdependence comes into play with dramatic effect.

It was observed that the performance of the fuzzy controller is not related to the speed of learning process.

The analysis performed in Chapter 5 shows that the 2-layered HFS provides a slightly better solution to the control problem of the inverted pendulum than the 3-layered HFS. This result reflects the physical nature of the inverted pendulum system with pole and cart variables grouped in two separate 'subsystems' that are mirrored in the 2-layered HFS. However, the 3-layered HFS significantly reduces the size of the knowledge bases while providing control system of similar performance.

By introducing the hierarchical structure the control system is greatly simplified (with the exception of 'different topologies'). One would argue that a complex fuzzy rule base should stabilise the system better than a simple one, but on the other hand the complex knowledge base is usually more vulnerable to external disturbances and uncertainty. The investigation into the HFS topologies suggests that the size of the knowledge base, i.e. number of rules, is not a decisive factor in controller performance. On one side of the 'spectrum' there is topology *L2-3-412* with 880 fuzzy rules or single layer topology *L1-1234* with 625 fuzzy rules, and on the other side there is 3-layered topology with 95 fuzzy rules in its knowledge base (except those 3 layered topologies investigated as 'different topologies'), with some of them providing similar controller performance. However, the topology of the HFS seems to be the decisive factor in controller performance.

Similarly, the HFS topologies investigation shows that the number of layers is not an important factor in performance of the controller (in terms of control magnitude and stabilisation rate of the state variables). This fact allows large knowledge base of simple structure (for example: single layer) to be replaced with the HFS without loss of controller performance. In fact, the HFS produced more efficient controllers (in terms of system stabilisation) than single layer controller except for magnitude of control which for single layer control system is the lowest.

The tests performed on topologies with one or two layers removed have demonstrated that the hierarchical fuzzy system needs to be considered in its entirety and not as an assembly of its better or worse performing component rule bases.

The simulation results indicate that a particular input configuration in the HFS layers is more important than the number of layers as good controller performance was achieved for both: 2 layered *L2-34-12* and 3 layered *L3-34-2-1* and *L3-34-1-2*, see Figure 5.5, Figure 5.17, and Figure 5.19 respectively. This indicates that interdependence of variables plays a crucial role in finding the 'optimal' HFS for a particular problem. Examining the nature of variables interdependence is a key to an automated determination of the decomposition of the fuzzy model of control. The decomposition of the hierarchical fuzzy structure should be performed along weak interdependency between input variables. However, with more complex dynamical systems there might be multiple weak interdependencies in input configuration. In such cases either expert knowledge is required to resolve the decomposition problem or an automated process that finds optimal or near optimal hierarchical fuzzy topology.

9.3 Co-evolutionary algorithm

In Chapter 6 the hierarchical fuzzy control of the simple inverted pendulum was examined and co-evolutionary algorithm was used to learn a fuzzy controller and its membership functions (within a class of Gaussian membership functions).

It was observed that for every simulation a different knowledge base was developed. The controllers from different simulations are different partly because the EA starts from different initial controller population (randomly initialized). With the increased number of generations (5000—10000) co-evolutionary algorithm still converges to different solutions but very similar in performance. The state variables convergence and controller time history look very similar for all simulations. The knowledge bases are thus different but control system performance is very similar. One possible explanation is that there are several or more suboptimal solutions in search space and co-evolutionary algorithm converges to one of them.

The algorithm starting from a uniform MF population generally produces better performing controllers. However, in most cases, in spite of their differences, the controllers perform in a very similar fashion, i.e., having similar control time history and similar character of state variables convergence. The MFs centres resulting from the EA with uniform MF population change very little from their original definitions and better results from such EAs suggest that control systems perform better with evenly spaced membership functions.

The results of the co-evolutionary algorithm are non-trivial especially on the part of evolved membership functions. In case of the EA with randomly generated MF initial population, MF centres are not uniformly spread over system variables range but tend to congregate in certain regions of the input or output variable range. Such ‘clustering’ of the membership functions requires additional investigation. The working hypothesis would be that MFs tend to ‘gravitate’ towards region where the final states of the state variables values lie after simulation of the dynamical system.

9.4 Compositional method

In Chapter 7 an evolutionary algorithm based compositional method has been examined and applied to a simple dynamical system (inverted pendulum). Evolutionary algorithm designed for the compositional method is used to learn fuzzy rules for a 3-layered hierarchical fuzzy control system over a user-defined set of initial conditions. The initial conditions include both static and dynamic conditions of the system. For the inverted pendulum problem the static initial conditions include the cart’s position and the angle of the pole, and dynamic initial conditions: the initial cart’s velocity and the angular velocity of the pole.

The experiments with the compositional method for the inverted pendulum system prove that with the right combination of the EA parameters the resulting fuzzy control system is capable of controlling the system from the wide range of initial conditions, the best result being 76% success rate (the system was controlled to the TR from 195 out of 255 of initial conditions), while most simulations averaged slightly above 50% success rate.

For the system that starts from diametrically different initial conditions it is unlikely to find reasonably small fuzzy rule base capable of handling every possible dynamics of the system. This fact reflects physical reality of complex non-linear dynamical systems, including even relatively simple inverted pendulum dynamics.

9.5 MOEA based compositional method

A multiobjective evolutionary algorithm based compositional method is used to learn fuzzy rules for a three-layered hierarchical fuzzy control system over the large set of initial conditions. The proposed method has a relatively high success rate in terms of the number of initial conditions from which the system is controlled to the TR.

In experiments with a single objective EA: $f = f_1 + f_2$, with similar EA parameters, success rate averaged 50% while the MOEA based method consistently approached or exceeded 80% success rate. The best achieved result was 94.5% success rate (the system controlled to the TR from 241 out of total 255 initial conditions).

Better MOEA results indicate that splitting the objective function into its composite parts improves the controller performance. In case of the inverted pendulum one objective function is defined as a measure of state variables 'distance' to the target region and second one as survival time (the total time in which the pole and cart remain within specified bounds). This split represents two different aspects of the inverted pendulum problem but they are strongly coupled.

The proposed MOEA based compositional method shows significant improvement over single objective EA especially in terms of consistency of results from different controllers, see Section 8.4. The MOEA used in the investigation is a simple modification of the EA used in all other experiments. This was done in order to examine the effect of modularising the objective function definition. Further improvements in the MOEA performance are expected with refinement of objective functions definitions and more efficient MOEA scheme, such as NSGA II or SPEA-2.

Just as the single objective EA based compositional method, the MOEA based compositional method can also be applied to a wide range of dynamical systems. This require a re-definition of system dynamics, MOEA encoding, objective functions, and determining a set of initial conditions. The hierarchical structure is also case-dependent and reflects the physical properties of the dynamical system. Therefore, the number of membership functions and their range needs to be adjusted. However, the overall algorithm structure remains the same, see Section 8.2.5. The proposed MOEA based compositional method is specifically tailored to control the system from a user defined set of initial conditions and this is its greatest advantage.

9.6 Comparison of controller performance for proposed methods

The developed control methods for the inverted pendulum system allow the comparison of certain aspects of the controller performance. Two aspects are selected for comparison:

- System stabilisation times.
- State variables convergence and control magnitude.

For clarity, only the best performing controllers developed in the course of this investigation are compared.

9.6.1 Stabilisation times comparison

The stabilisation time is compared for the selected best performing topologies for different methods: single initial condition EA (Chapter 5), co-evolutionary algorithm (Chapter 6) and MOEA based compositional method.

Table 9.1 Stabilisation times for the examples of the best performing topologies, single initial condition.

Topology	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12
Stabilisation time (sec)	2.74	2.9	2.06	3.22	1.68	2.64	1.82	2.4	4.32	1.4	1.74	3.92

where C1 denotes: *L2-34-12*, C2: *L3-34-1-2*, C3: *L3-34-2-1*, C4: *L4-3-4-1-2*, C5: *L4-3-4-2-1*, C6: *L4-4-3-1-2*, C7: *L4-4-3-2-1*, C8: *L2-3-412*, C9: *L2-342-1*, C10: *L3-3-4-12*, C11: *L3-3-41-2*, C12: *L1-1234*.

Alternative topologies *L3-3-4-12* and *L3-3-41-2* are characterised by very fast stabilisation times but relatively large knowledge base of 215 rules compared to 95 rules in any *L3-mn-k-l* topology, $m,n,k,l \in \{1,2,3,4\}$.

Stabilisation times for controllers developed by the co-evolutionary algorithm (see Chapter 6) are shown in Table 9.2. On average, the co-evolutionary algorithm produces faster acting controllers than the conventional EA investigated in Chapter 5.

Table 9.2 Stabilisation times for co-evolutionary algorithm examples.

Topology	C1	C2	C3	C4	C5	C6	C7	C8
Stabilisation time (sec)	2.6	2.78	1.8	1.63	2.14	2.94	1.84	2.74

where C1 denotes: *L3-34-1-2* explorative, uniform, *C2: L3-34-1-2* explorative, random, *C3: L3-34-1-2* greedy, uniform, *C4: L3-34-1-2* greedy, random, *C5: L3-34-2-1* explorative, uniform, *C6: L3-34-2-1* explorative, random, *C7: L3-34-2-1* greedy, uniform, *C8: L3-34-2-1* greedy, random.

Table 9.3 Stabilisation time for randomly selected init. conditions for single objective EA and MOEA based compositional method.

Topology *L3-34-2-1*	EA init. 4 best controller	EA init. 88	MOEA init. 78 controller no 8	MOEA init. 78 controller no 1	MOEA init. 136
Stabilisation time (sec)	16.84	4.22	2.94	2.90	2.32

As can be seen from stabilisation time comparison, the co-evolutionary algorithm improves the controller performance compared to the earlier EA version investigated in Chapter 5.

MOEA based compositional method is a significant improvement on the single objective EA based method in terms of stabilisation times. However, in case of the compositional method this is not a decisive factor in evaluating the controller performance. As was mentioned before, the MOEA based method is far superior to single objective EA in terms of the total number of initial conditions from which it controls the system to the target region.

9.6.2 State variables convergence and control magnitude

One of the critical controller performance criteria was state variable convergence to the target region and magnitude of control. Additionally smoothness of both state variable convergence and controller time history were taken into account. The examples of the best performing controllers are compared in this section.

Comparison of state variables convergence for selected examples of the best performing topologies is shown in Figure 9.1: *L2-34-12* and *L3-34-1-2* (top) and *L4-3-4-2-1* and *L3-34-2-1* (bottom). Comparison of control magnitude and smoothness of the control action over [0,*T*] interval is shown in Figure 9.2: *L2-34-12* and *L3-34-1-2* (top) *L4-3-4-2-1* and *L3-34-2-1* (bottom). State variables convergence for explorative co-evolutionary examples is shown in Figure 9.3: *L3-34-1-2* uniform, *L3-34-1-2* random (top), *L3-34-2-1* uniform, *L3-34-2-1* random (bottom).

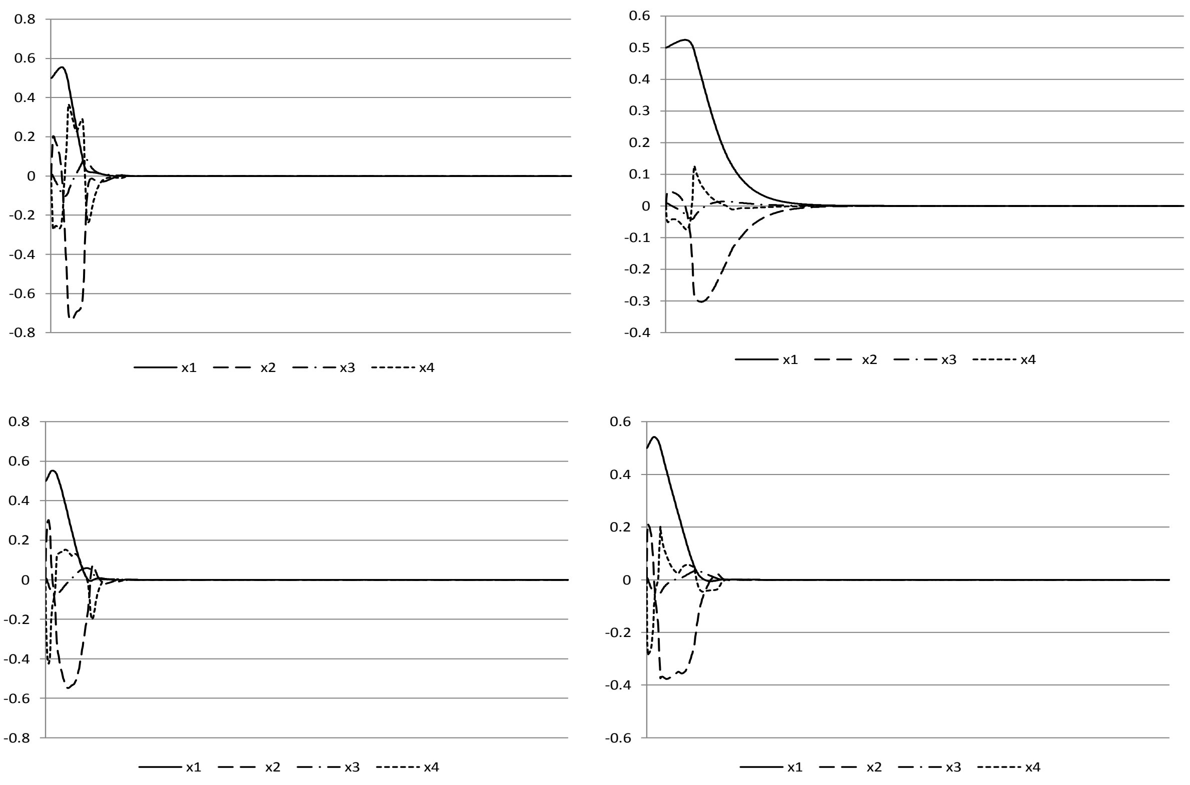

Figure 9.1 State variables convergence for L2-34-12 and L3-34-1-2 (top) and L4-3-4-2-1 and L3-34-2-1(bottom).

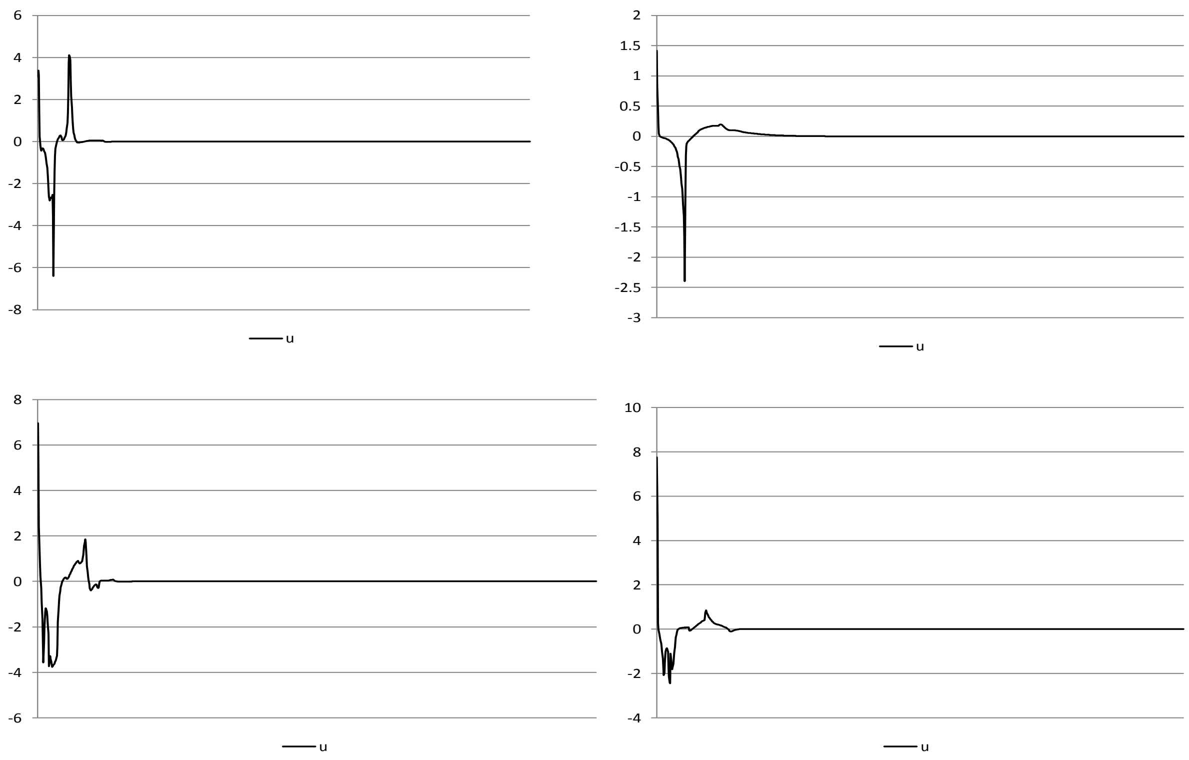

Figure 9.2 Control magnitude and smoothness of the control action for L2-34-12 and L3-34-1-2 (top) and for L4-3-4-2-1 and L3-34-2-1 (bottom).

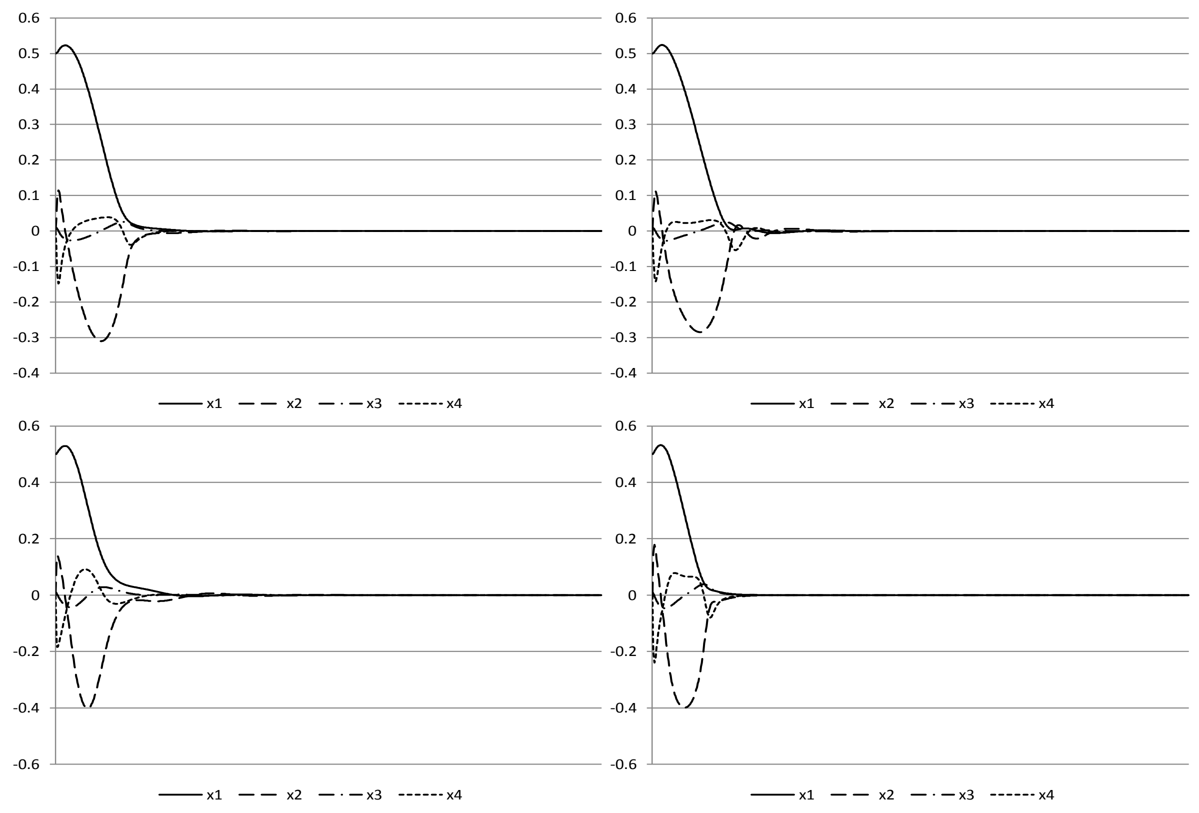

Figure 9.3 State variables convergence explorative L3-34-1-2 uniform, L3-34-1-2 random (top), L3-34-2-1 uniform, L3-34-2-1 random (bottom).

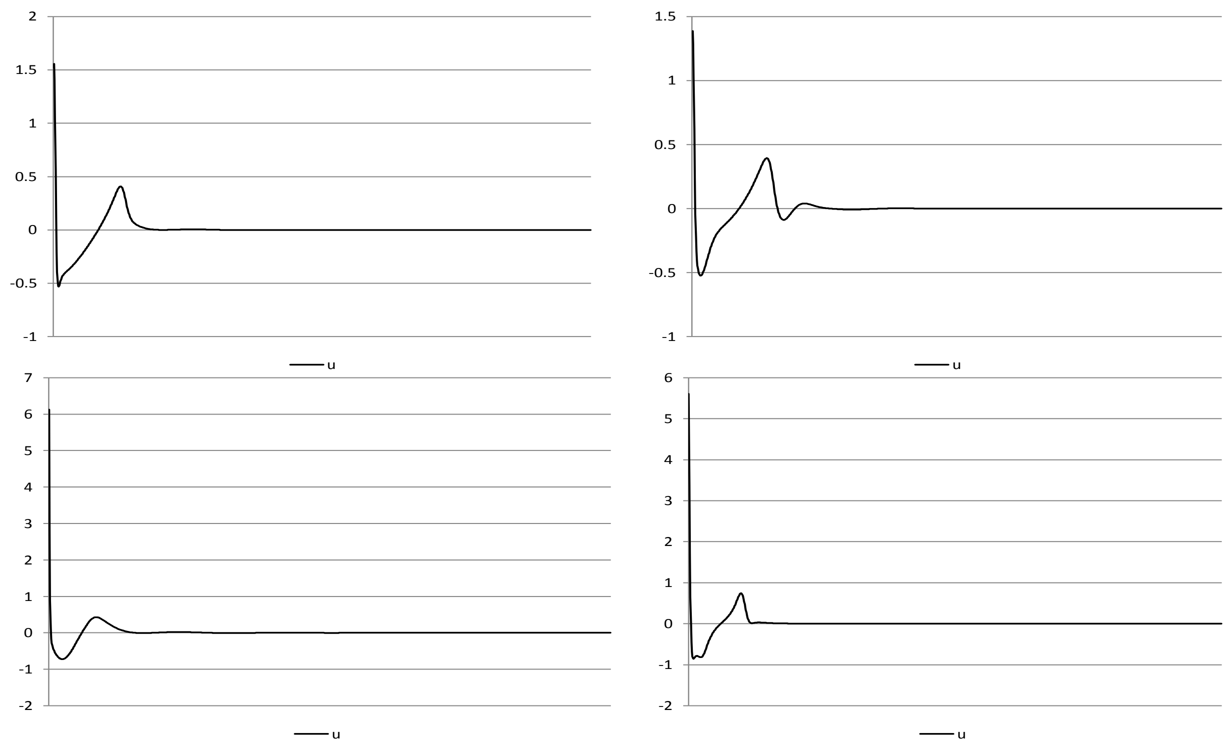

Figure 9.4 Control magnitude and smoothness of the control action for explorative L3-34-1-2 uniform, L3-34-1-2 random (top), L3-34-2-1 uniform, L3-34-2-1 random (bottom).

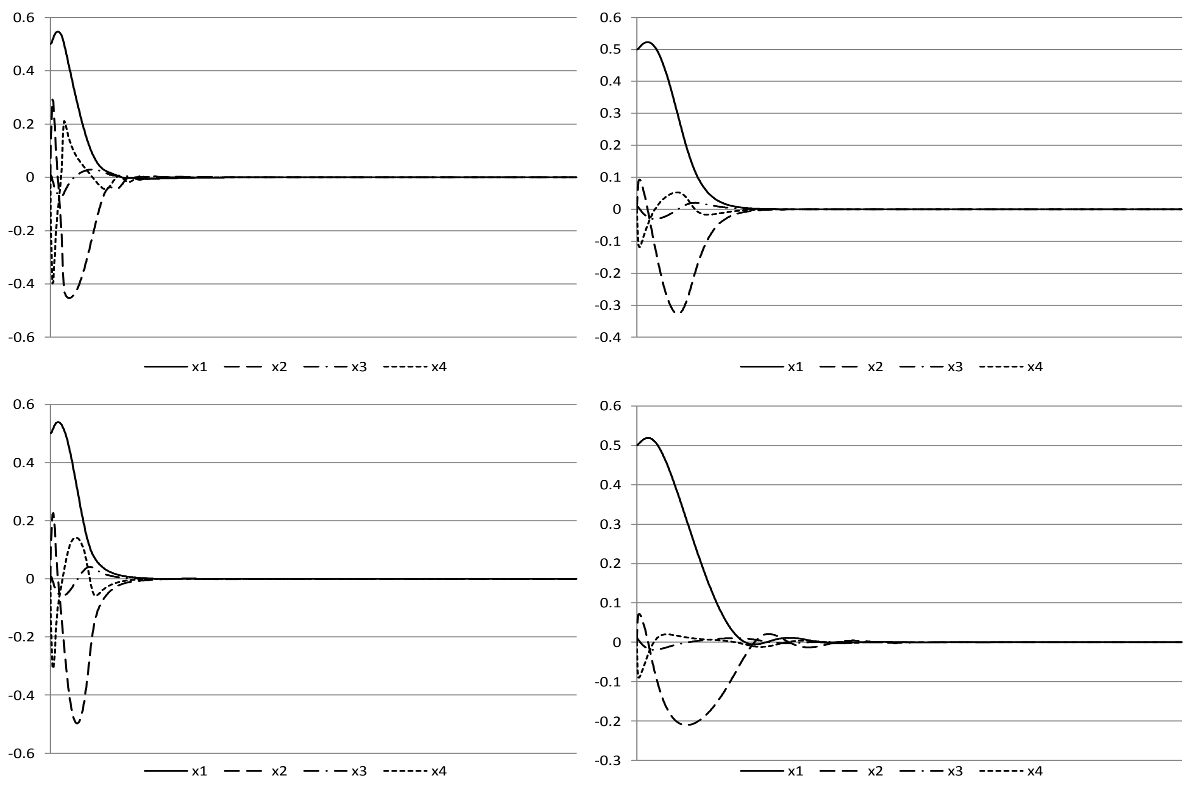

Figure 9.5 State variables convergence greedy L3-34-1-2 uniform, L3-34-1-2 random (top), L3-34-2-1 uniform, L3-34-2-1 random (bottom).

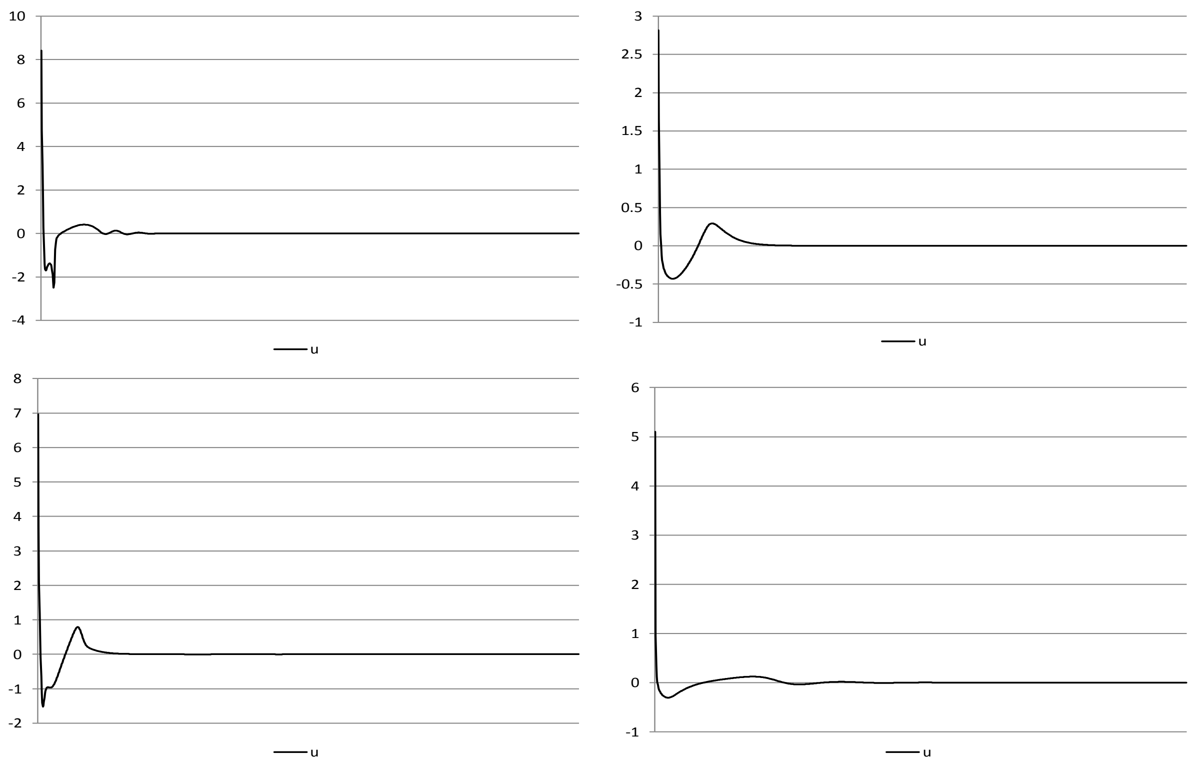

Figure 9.6 Control magnitude and smoothness of the control action for greedy L3-34-1-2 uniform, L3-34-1-2 random (top), L3-34-2-1 uniform, L3-34-2-1 random (bottom).

Control magnitude and smoothness of the control action over $[0,T]$ interval is shown in Figure 9.4 for explorative co-evolutionary examples: *L3-34-1-2* uniform, *L3-34-1-2* random (top), *L3-34-2-1* uniform, *L3-34-2-1* random (bottom). State variables convergence for greedy co-evolutionary examples is shown in Figure 9.5: *L3-34-1-2* uniform, *L3-34-1-2* random (top), *L3-34-2-1* uniform, *L3-34-2-1* random (bottom). Comparison of control magnitude and smoothness of the control action over $[0,T]$ interval is shown in Figure 9.6 for greedy co-evolutionary examples: *L3-34-1-2* uniform, *L3-34-1-2* random (top), *L3-34-2-1* uniform, *L3-34-2-1* random (bottom).

As can be seen in figures showing state variables convergence and control action for both explorative and greedy co-evolutionary algorithm examples, the smoothest and relatively low magnitude of control is exhibited by the controller with topology *L3-34-2-1* (EA with uniform initial population of membership functions), see left bottom part of Figure 9.4, and by the controller with topology *L3-34-2-1* (EA with random initial population of membership functions), see left bottom part of Figure 9.6. Compared to control action shown in Figure 9.2, it shows smooth time history without oscillations. Only initial large magnitude control action is required at the very beginning of system stabilisation and then the controller quickly stabilises the system. This is true for all co-evolutionary controllers which exhibit smoother control action than controllers designed without fine-tuning of membership functions. Based on this comparison it can be concluded that co-evolutionary algorithm delivers better controller performance than any controller investigated in Chapter 5.

Comparison of single objective EA (init. cond. no 88) and MOEA state variables convergence for test-1 (init. cond. 78), controller no 8 and 1, and test-2 (init. cond. 122) are shown in Figure 9.7. Controller action for the same examples is shown in Figure 9.8. Both single objective and MOEA solutions exhibit low magnitude control action. MOEA controllers have relatively fast stabilisation time. In case of the compositional method controllers the crucial performance index was the number of initial conditions from which the controller stabilises the system. In this respect, the controller no 8, see top right of Figure 9.7, is beyond doubt the best solution for the control problem with 94.5% success rate (241 convergences to the TR out of total 255 initial conditions) even though the state variable convergence is not as regular as shown in other examples. As can be seen in all examples for the compositional method, the position of the cart x_1 was the most difficult to control. None of the MOEA solutions achieves smoothness of state variables convergence and control action of co-evolutionary algorithm solutions. This needs to be seen as a trade-off between quality of control and meeting its major objective (the largest controllability region). In the end, the MOEA based compositional method strikes a good compromise, delivering controllers with better performance than initially investigated in Chapter 5 and slightly worse performing controllers than delivered by the co-evolutionary algorithm.

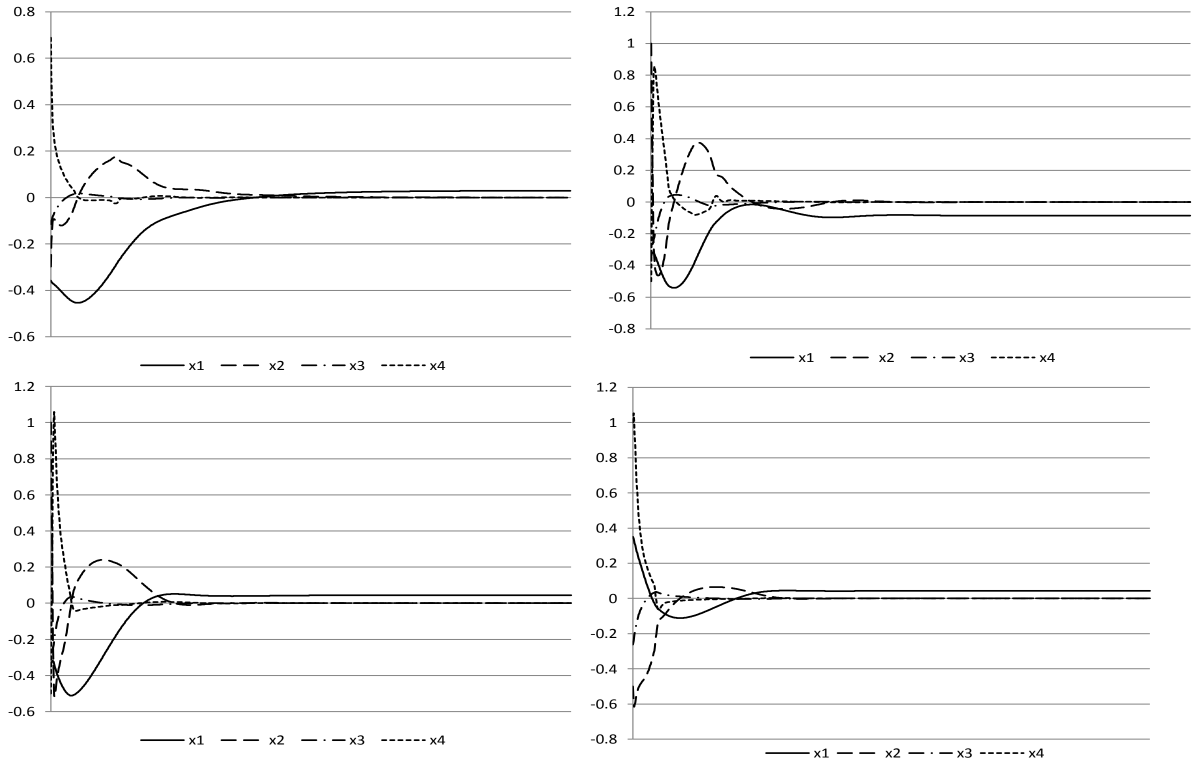

Figure 9.7 State variables convergence for single objective EA init. cond. no 88 (top left) and MOEA test-1 (init. cond. 78), controller no 8 (top right) and 1 (bottom left), and test-2 init. cond. 122 (bottom right).

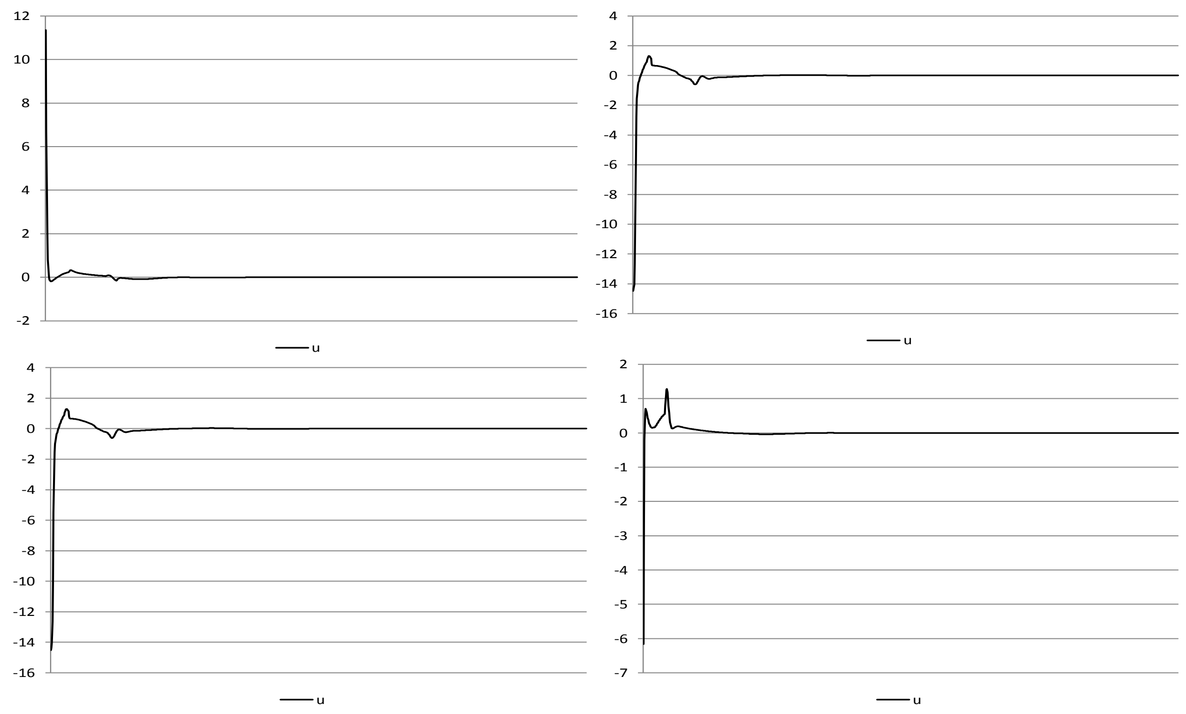

Figure 9.8 Control action for single objective EA init. cond. no 88 (top left) and MOEA test-1 (init. cond. 78), controller no 8 (top right) and 1 (bottom left), and test-2 init. cond. 122 (bottom right).

9.6.3 Success rate for single EA and MOEA based compositional method

The difference between single EA and MOEA based compositional method is illustrated in the table with the number of initial conditions from which the selected controllers developed by both methods stabilised the system to the target region.

Table 9.4 The success rate for single EA and MOEA based composition method.

	EA best controller	EA typical	MOEA test-1 controller no 8	MOEA test-1 typical
No of init. cond. converged to TR	205	≈ 100	241	≈ 207

The average for test-1 was 207.6 (including failed controller no 10) and 212.4 without taking into account controller no 10 (see Section 8.4 for more details). Therefore, controller no 1 represents typical controller performance for test-1. Full results for MOEA success rate test-1 can be found Table 8.1.

9.7 Summary of conclusions

The major conclusions of the thesis can be summarized as:

- Selection of the right topology, both input configuration and hierarchical fuzzy system structure, plays crucial role in the control system performance. It is also vital for the fast evolutionary algorithm convergence to a desired solution. The number of layers is not an important factor in performance of the controller (in terms of control magnitude and stabilisation rate of the state variables). This fact allows large and simple structure knowledge base (for example: single layer) to be replaced with the HFS without loss of controller performance. In fact, the HFS produced controllers that performed more efficiently in terms of system stabilisation than single layer controller except for magnitude of control which for single layer control system is the lowest. A particular configuration of input in the HFS layers is more important than the number of layers. This indicates that interdependence of variables plays a crucial role in finding the ‘optimal’ HFS for a particular problem. Examining the nature of variables interdependence is a key to an automated determination of the decomposition of the fuzzy model of control.
- It was observed that for every simulation of co-evolutionary algorithm a different knowledge base is evolved. The controller is different partly because the EA starts from different initial population. With the increased number of generations (5000—10000) co-evolutionary algorithm still converges to different solutions but very similar in performance. The knowledge bases are thus different but control systems performance very similar. One possible explanation is that there are several or more suboptimal solutions in search space and co-evolutionary algorithm converges to one of them in every simulation.
- The co-evolutionary algorithm starting from a uniform membership functions population generally produces better performing controllers. However, in most cases, the controllers perform in a very similar fashion, i.e., having similar control time history and similar character of state variables convergence. The membership functions centres resulting from the EA with uniform membership function population change very little from their original definitions and better results from such EAs suggest that control systems perform better with evenly spaced membership functions.
- The proposed MOEA based compositional method shows significant improvement over single objective EA especially in terms of consistency of results from different

controllers. Further improvements are expected with refinement of objective functions definitions and use of more efficient MOEA. The MOEA based compositional method can be applied to a wide range of dynamical systems with a re-definition of system equations, encoding of evolutionary population, objective functions, and appropriate set of initial conditions. The proposed MOEA based compositional method is specifically tailored to control the system from a user defined set of initial conditions; this being its greatest advantage.

9.8 Future directions

A common problem encountered with designing the evolutionary algorithm is selection of the algorithm parameters. Development of a general approach to fine-tuning of the EA parameters is required. This can be achieved by using pre-defined set of EA parameters, such as a range of crossover operators, mutation rates, selection methods, population size, encoding methods, etc. Definition of the objective function (or functions) is case dependant and therefore it is not possible to define objective function via any automated process unless the problem is simple enough to use one of the standard objective functions, such as Euclidean distance to the target region, sum of squared differences between state vector and the origin, etc.

The obvious extension of this research would be investigation into the method of an automated determination of the optimal or sub-optimal topology of the hierarchical fuzzy control system. There are difficulties to overcome in such an investigation, namely handling variable length of strings in the population and their different encoding methods. One possible approach is to use multiple populations. Another approach would be using extended strings with another vector containing information about encoded structure. With a number of additional assumptions such an automated method is feasible.

The methods presented in this thesis used the inverted pendulum as a test system. To fully test the applicability of the MOEA based compositional method a more complex test system needs to be used. Multi-link robotic manipulator would be a good example. Other applications include financial systems, production control, traffic control, etc.

Presented methods show a potential for future development and refinement.

APPENDIX

The table below contains a set of regularly interspaced 255 initial conditions within the region defined by: $|x_1| \leq 0.75$, $|x_2| \leq 1.0$, $|x_3| \leq \pi/12$, $|x_4| \leq 1.0$. This set of initial conditions was used to test evolutionary algorithm based compositional method.

Table Appendix. Initial conditions set.

x_1	x_2	x_3	x_4	x_1	x_2	x_3	x_4
0.75	1.0	0.261799	1.0	0.35	0.5	0.261799	1.0
0.75	1.0	0.261799	0.5	0.35	1.0	0.261799	1.0
0.75	1.0	0.261799	0.5	0.35	1.0	0.261799	0.5
0.75	1.0	0.261799	1.0	0.35	1.0	0.261799	0.5
0.75	0.5	0.261799	1.0	0.35	1.0	0.261799	1.0
0.75	0.5	0.261799	0.5	0.35	1.0	0.261799	1.0
0.75	0.5	0.261799	0.5	0.35	1.0	0.261799	0.5
0.75	0.5	0.261799	1.0	0.35	1.0	0.261799	0.5
0.75	0.5	0.261799	1.0	0.35	1.0	0.261799	1.0
0.75	0.5	0.261799	0.5	0.35	0.5	0.261799	1.0
0.75	0.5	0.261799	0.5	0.35	0.5	0.261799	0.5
0.75	0.5	0.261799	1.0	0.35	0.5	0.261799	0.5
0.75	1.0	0.261799	1.0	0.35	0.5	0.261799	1.0
0.75	1.0	0.261799	0.5	0.35	0.5	0.261799	1.0
0.75	1.0	0.261799	0.5	0.35	0.5	0.261799	0.5
0.75	1.0	0.261799	1.0	0.35	0.5	0.261799	0.5
0.75	1.0	0.130899	1.0	0.35	0.5	0.261799	1.0
0.75	1.0	0.130899	0.5	0.35	1.0	0.261799	1.0
0.75	1.0	0.130899	0.5	0.35	1.0	0.261799	0.5
0.75	1.0	0.130899	1.0	0.35	1.0	0.261799	0.5
0.75	0.5	0.130899	1.0	0.35	1.0	0.261799	1.0
0.75	0.5	0.130899	0.5	0.35	1.0	0.130899	1.0
0.75	0.5	0.130899	0.5	0.35	1.0	0.130899	0.5
0.75	0.5	0.130899	1.0	0.35	1.0	0.130899	0.5
0.75	0.5	0.130899	1.0	0.35	1.0	0.130899	1.0
0.75	0.5	0.130899	0.5	0.35	0.5	0.130899	1.0
0.75	0.5	0.130899	0.5	0.35	0.5	0.130899	0.5
0.75	0.5	0.130899	1.0	0.35	0.5	0.130899	0.5
0.75	1.0	0.130899	1.0	0.35	0.5	0.130899	1.0
0.75	1.0	0.130899	0.5	0.35	0.5	0.130899	1.0
0.75	1.0	0.130899	0.5	0.35	0.5	0.130899	0.5
0.75	1.0	0.130899	1.0	0.35	0.5	0.130899	0.5
0.75	1.0	0.130899	1.0	0.35	0.5	0.130899	1.0
0.75	1.0	0.130899	0.5	0.35	1.0	0.130899	1.0
0.75	1.0	0.130899	0.5	0.35	1.0	0.130899	0.5
0.75	1.0	0.130899	1.0	0.35	1.0	0.130899	0.5
0.75	0.5	0.130899	1.0	0.35	1.0	0.130899	1.0
0.75	0.5	0.130899	0.5	0.35	1.0	0.130899	1.0
0.75	0.5	0.130899	0.5	0.35	1.0	0.130899	0.5

0.75	0.5	0.130899	1.0	0.35	1.0	0.130899	1.0
0.75	0.5	0.130899	1.0	0.35	0.5	0.130899	1.0
0.75	0.5	0.130899	0.5	0.35	0.5	0.130899	0.5
0.75	0.5	0.130899	0.5	0.35	0.5	0.130899	0.5
0.75	0.5	0.130899	1.0	0.35	0.5	0.130899	1.0
0.75	1.0	0.130899	1.0	0.35	0.5	0.130899	1.0
0.75	1.0	0.130899	0.5	0.35	0.5	0.130899	0.5
0.75	1.0	0.130899	0.5	0.35	0.5	0.130899	0.5
0.75	1.0	0.130899	1.0	0.35	0.5	0.130899	1.0
0.75	1.0	0.261799	1.0	0.35	1.0	0.130899	1.0
0.75	1.0	0.261799	0.5	0.35	1.0	0.130899	0.5
0.75	1.0	0.261799	0.5	0.35	1.0	0.130899	0.5
0.75	1.0	0.261799	1.0	0.35	1.0	0.130899	1.0
0.75	0.5	0.261799	1.0	0.35	1.0	0.261799	1.0
0.75	0.5	0.261799	0.5	0.35	1.0	0.261799	0.5
0.75	0.5	0.261799	0.5	0.35	1.0	0.261799	0.5
0.75	0.5	0.261799	1.0	0.35	1.0	0.261799	1.0
0.75	0.5	0.261799	1.0	0.35	0.5	0.261799	1.0
0.75	0.5	0.261799	0.5	0.35	0.5	0.261799	0.5
0.75	0.5	0.261799	0.5	0.35	0.5	0.261799	0.5
0.75	0.5	0.261799	1.0	0.35	0.5	0.261799	1.0
0.75	1.0	0.261799	1.0	0.35	0.5	0.261799	1.0
0.75	1.0	0.261799	0.5	0.35	0.5	0.261799	0.5
0.75	1.0	0.261799	0.5	0.35	0.5	0.261799	0.5
0.75	1.0	0.261799	1.0	0.35	0.5	0.261799	1.0
0.35	1.0	0.261799	1.0	0.35	1.0	0.261799	1.0
0.35	1.0	0.261799	0.5	0.35	1.0	0.261799	0.5
0.35	1.0	0.261799	0.5	0.35	1.0	0.261799	0.5
0.35	1.0	0.261799	1.0	0.35	1.0	0.261799	1.0
0.35	0.5	0.261799	1.0	0.75	1.0	0.261799	1.0
0.35	0.5	0.261799	0.5	0.75	1.0	0.261799	0.5
0.35	0.5	0.261799	0.5	0.75	1.0	0.261799	0.5
0.35	0.5	0.261799	1.0	0.75	1.0	0.261799	1.0
0.35	0.5	0.261799	1.0	0.75	0.5	0.261799	1.0
0.35	0.5	0.261799	0.5	0.75	0.5	0.261799	0.5
0.35	0.5	0.261799	0.5	0.75	0.5	0.261799	0.5
0.35	0.5	0.261799	1.0	0.75	0.5	0.261799	1.0
0.35	1.0	0.261799	1.0	0.75	0.5	0.261799	1.0
0.35	1.0	0.261799	0.5	0.75	0.5	0.261799	0.5
0.35	1.0	0.261799	0.5	0.75	0.5	0.261799	0.5
0.35	1.0	0.261799	1.0	0.75	0.5	0.261799	1.0
0.35	1.0	0.130899	1.0	0.75	1.0	0.261799	1.0
0.35	1.0	0.130899	0.5	0.75	1.0	0.261799	0.5
0.35	1.0	0.130899	0.5	0.75	1.0	0.261799	0.5
0.35	1.0	0.130899	1.0	0.75	1.0	0.261799	1.0
0.35	0.5	0.130899	1.0	0.75	1.0	0.130899	1.0
0.35	0.5	0.130899	0.5	0.75	1.0	0.130899	0.5
0.35	0.5	0.130899	0.5	0.75	1.0	0.130899	0.5

0.35	0.5	0.130899	1.0	0.75	1.0	0.130899	1.0
0.35	0.5	0.130899	1.0	0.75	0.5	0.130899	1.0
0.35	0.5	0.130899	0.5	0.75	0.5	0.130899	0.5
0.35	0.5	0.130899	0.5	0.75	0.5	0.130899	0.5
0.35	0.5	0.130899	1.0	0.75	0.5	0.130899	1.0
0.35	1.0	0.130899	1.0	0.75	0.5	0.130899	1.0
0.35	1.0	0.130899	0.5	0.75	0.5	0.130899	0.5
0.35	1.0	0.130899	0.5	0.75	0.5	0.130899	0.5
0.35	1.0	0.130899	1.0	0.75	0.5	0.130899	1.0
0.35	1.0	0.130899	1.0	0.75	1.0	0.130899	1.0
0.35	1.0	0.130899	0.5	0.75	1.0	0.130899	0.5
0.35	1.0	0.130899	0.5	0.75	1.0	0.130899	0.5
0.35	1.0	0.130899	1.0	0.75	1.0	0.130899	1.0
0.35	0.5	0.130899	1.0	0.75	1.0	0.130899	1.0
0.35	0.5	0.130899	0.5	0.75	1.0	0.130899	0.5
0.35	0.5	0.130899	0.5	0.75	1.0	0.130899	0.5
0.35	0.5	0.130899	1.0	0.75	1.0	0.130899	1.0
0.35	0.5	0.130899	1.0	0.75	0.5	0.130899	1.0
0.35	0.5	0.130899	0.5	0.75	0.5	0.130899	0.5
0.35	0.5	0.130899	0.5	0.75	0.5	0.130899	0.5
0.35	0.5	0.130899	1.0	0.75	0.5	0.130899	1.0
0.35	1.0	0.130899	1.0	0.75	0.5	0.130899	1.0
0.35	1.0	0.130899	0.5	0.75	0.5	0.130899	0.5
0.35	1.0	0.130899	0.5	0.75	0.5	0.130899	0.5
0.35	1.0	0.130899	1.0	0.75	0.5	0.130899	1.0
0.35	1.0	0.261799	1.0	0.75	1.0	0.130899	1.0
0.35	1.0	0.261799	0.5	0.75	1.0	0.130899	0.5
0.35	1.0	0.261799	0.5	0.75	1.0	0.130899	0.5
0.35	1.0	0.261799	1.0	0.75	1.0	0.130899	1.0
0.35	0.5	0.261799	1.0	0.75	1.0	0.261799	1.0
0.35	0.5	0.261799	0.5	0.75	1.0	0.261799	0.5
0.35	0.5	0.261799	0.5	0.75	1.0	0.261799	0.5
0.35	0.5	0.261799	1.0	0.75	1.0	0.261799	1.0
0.35	0.5	0.261799	1.0	0.75	0.5	0.261799	1.0
0.35	0.5	0.261799	0.5	0.75	0.5	0.261799	0.5
0.35	0.5	0.261799	0.5	0.75	0.5	0.261799	0.5
0.75	0.5	0.261799	1.0	0.75	1.0	0.261799	1.0
0.75	0.5	0.261799	1.0	0.75	1.0	0.261799	0.5
0.75	0.5	0.261799	0.5	0.75	1.0	0.261799	0.5
0.75	0.5	0.261799	0.5	0.75	1.0	0.261799	1.0
0.75	0.5	0.261799	1.0				

REFERENCES

Abbass, H.A., Sarker, R., Newton, C. (2001). PDE: a Pareto-frontier differential evolution approach formulti-objective optimization problems. *Proceedings of the 2001 Congress on Evolutionary Computation,* Vol. 2, pp. 971–978.

Abbass, H.A. (2002). The self-adaptive Pareto differential evolution algorithm. *CEC '02, Proceedings of the 2002 Congress on Evolutionary Computation*, Vol. 1, pp. 831–836.

Abraham, A. (2005). *Adaptation of fuzzy inference system using neural learning, fuzzy system engineering: Theory and practice*. Eds. Edjah, N. *et al.* Springer-Verlag, Chapter 3, pp. 53–83.

Abraham, A., Jain, L.C, Goldberg, R. (2005). *Evolutionary multiobjective optimizations: theoretical advances and applications*. Springer Verlag, London.

Ahn, C.W., Ramakrishna, R.S. (2003). Elitism-based compact genetic algorithms. *IEEE Transactions on Evolutionary Computation*, Vol. 7, No. 4, pp. 367–385.

Allamehzadeh, H., Cheung, J.Y. (2002). Chattering-free sliding mode fuzzy control with continuous inherent boundary layer. *FUZZ-IEEE'02, Proceedings of the 2002 IEEE International Conference on Fuzzy Systems,* Vol. 2, pp. 1393–1398.

Anderson, C.W. (1987). Strategy learning with multilayer connectionist representations. *Methods of A.M. Liapunov and their applications, Proceedings 4th International Workshop Machine Learning*, Pub. Morgan Kaufmann, pp. 103–114.

Andersen, H.C. Lotfi, A., Tsoi, A.C. (1997). A new approach to adaptive fuzzy control: The controller output error method. *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 27, No. 4, pp. 686–691.

Anderson, C.W. (1989). Learning to control an inverted pendulum using neural networks. *IEEE Control Systems Magazine*, pp. 31–36.

Astrom, K.J., Furuta, K. (2000). Swinging up a pendulum by energy control. *Automatica,* Vol. 36, pp. 287–295.

Babuska, R. (1998). *Fuzzy modelling for control*. Kluwer Academic Press.

Babuska, R. (2009). *Computational intelligence in modelling and control*. Delft University of Technology, http://www.dcsc.tudelft.nl/~rbabuska/CTU/transp/lecture_notes_ctu.pdf.

Baturone, I., Moreno-Velo, F.J., Sanchez-Solano, S., Ollero, A. (2004). Automatic design of fuzzy controllers for car-like autonomous robots. *IEEE Transactions on Fuzzy Systems*, Vol. 12, No. 4, pp. 447–465.

Beceriklia, Y., Celik, B.K. (2007). Mathematical and computer modeling. *Proceedings of the International Conference on Computational Methods in Sciences and Engineering 2004,* Vol. 46, Issues 1-2, pp. 24–37.

Bellman, R.E. (1961). *Adaptive Control Processes*. Princeton University Press, Princeton, NJ.

Bellman, R.E., Zadeh, L.A. (1970). Decision making in a fuzzy environment. *Management Science,* Vol.17, No. 4, pp. 141–164.

Belarbi, K., Titel, F. (2000). Genetic algorithm for the design of a class of fuzzy controllers: an alternative approach. *IEEE Transactions on Fuzzy Systems,* Vol. 8, No. 4, August 2000, pp. 398–405.

Brown, M., Bossley, K.M., Mills, D.J., Harris, C.J. (1995). High dimensional neurofuzzy systems: Overcoming the curse of dimensionality. *Proceedings of the 4th International Conference on Fuzzy Systems*, pp. 2139–2146.

Brunetti, C., Dotoli, M. (2004). Rule-based decoupled fuzzy sliding mode control for inverted pendulum swing-up. *2004 IEEE International Symposium on Industrial Electronics,* Vol.1, pp. 495–500.

Campos, J., Lewis, F.L. (1999). Deadzone compensation in discrete time using adaptive fuzzy logic. *IEEE Transactions on Fuzzy Systems*, Vol. 7, No. 6, pp. 697–707.

Cao, S.G., Rees, N.W., Feng, G. (1995). Analysis and design of fuzzy control systems using dynamic fuzzy global models. *Fuzzy Sets and Systems*, Vol. 75, pp. 47–62.

Cao, S.G., Rees, N.W., Feng, G. (1997). Analysis and design for a class of complex control systems—Part II: Fuzzy controller design. *Automatica*, Vol. 33, pp. 1029–1039.

Caroll, D.L. (1996). Chemical laser modelling with genetic algorithms. *AIAA Journal*, Vol. 34, No. 2, pp. 338–346.

Castillo, O., Cazarez, N., Rico, D. (2006). Intelligent control of dynamic systems using type-2 fuzzy logic and stability issues. *International Mathematical Forum*, Vol. 1, No 28, pp. 1371–1382.

Chang, W., Park, J.B., Joo, Y.H., Chen, G. (2002). Design of robust fuzzy-model based controller with sliding mode control for SISO nonlinear systems. *Fuzzy Sets and Systems,* Vol. 125, pp. 1–22.

Chen, B.S., Uang, H.J., Tseng, C.S. (1999). Robustness design of nonlinear dynamic systems via fuzzy linear control. *IEEE Transactions on Fuzzy Systems*, Vol. 7, No. 5, pp. 571–585.

Chen, C.L., Chang, M.H. (1998). Optimal design of fuzzy sliding mode control: A comparative study. *Fuzzy Sets and Systems*, Vol. 93, pp. 37–48.

Chen, C.S., Chen, W.L. (1998). Robust adaptive sliding-mode control using fuzzy modelling for an inverted-pendulum system. *IEEE Transactions on Industrial Electronics*, Vol. 45, No. 2, pp. 297–306.

Chen, G. (1996). Conventional and fuzzy PID controllers: An overview. *International Journal on Intelligent Control Systems*, Vol. 1, pp. 235–246.

Chen, Y.Y., Tsao, T.C. (1989). A description of the dynamical behavior of fuzzy systems. *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 19, No. 4, pp. 745–755.

Chen, Y., Yang, B., Abraham, A., Peng, L. (2007). Automatic design of hierarchical Takagi-Sugeno type fuzzy systems using evolutionary algorithms. *IEEE Transactions on Fuzzy Systems*, Vol. 15, No. 3, pp. 385–397.

Cheng, F.Y., Zhong, G.M., Li Y.S., Xu, Z.M. (1996). Fuzzy control of a double inverted pendulum. *Fuzzy Sets and Systems*, Vol. 79, pp. 315–321.

Cheong, F., Lai, R. (2007). Designing a hierarchical fuzzy controller using the differential evolution approach. *Applied Soft Computing***,** Vol. 7, No. 2, pp. 481–491.

Chiaberge, M., Di Bene,G., Di Pascoli, S., Lazzerini, B., Maggiore, A., Reyneri, L.M. (1995). An integrated hybrid approach to the design of high-performance intelligent controllers. *Proceedings of the 1995 International IEEE/IAS Conference on Industrial Automation and Control: Emerging Technologies*, pp. 436–443.

Chiu, S. (1998). Using fuzzy logic in control applications: Beyond fuzzy PID control. *IEEE Control Systems Magazine*, Vol. 18, No. 5, pp. 100–104.

Chopra, S., Mitra, R., Kumar, V. (2005). Fuzzy controller: Choosing an appropriate and smallest rule set. *International Journal of Computational Cognition*, Vol. 3, No. 4, pp. 73–79.

Coello Coello, C.A., Van Veldhuizen, D.A., Lamont, G.B. (2002). *Evolutionary algorithms for solving multiobjective problems*. Kluwer Academic Publishers.

Cooper, M.G., Vidal, J.J. (1994). Genetic design of fuzzy controllers: the cart and jointed-pole problem. *The Third IEEE International Conference on Fuzzy Systems*, Orlando, Florida, Vol. 2, pp. 1332–1337.

Cordon, O., Herrera, F., Zwir, I. (2002). Linguistic modeling of hierarchical systems of linguistic rules. *IEEE Transactions on Fuzzy Systems*, Vol. 10, No. 1, pp. 2–20.

Cordon, O., Herrera, F., Hoffmann, F., Magdalena, L. (2001a). *Genetic fuzzy systems: evolutionary tuning and learning of fuzzy knowledge bases. Advances in Fuzzy Systems Applications and Theory*, *Vol. 19*, World Scientific Publishing.

Cordon, O., Herrera, F., Villar, P. (2001b). Generating the knowledge base of a fuzzy rule-based system by the genetic learning of the data base. *IEEE Transactions on Fuzzy Systems*, Vol. 9, No. 4, pp. 667–674.

Cordon, O., Herrera, F., Hoffman, F., Magdalena, L. (2001c). *Genetic fuzzy systems*. World Scientific, Singapore.

Cordon, O., Herrera, F., Zwir, I. (2002). Linguistic modelling by hierarchical systems of linguistic rules. *IEEE Transactions on Fuzzy Systems*, Vol. 10, No. 1, pp. 2–20.

Cordon, O., Gomide, F., Herrera, F., Hoffman, F., Magdalena, L. (2004). Ten years of genetic fuzzy systems: current framework and new trends. *Fuzzy Sets and Systems*, Vol. 141, pp. 5–31.

Czogala, E., Pedrycz, W. (1981). On identification in fuzzy systems and its applications in control problems. *Fuzzy Sets and Systems*, Vol. 6, pp. 73—83.

Czogala, E., Pedrycz, W. (1982). Control problems in fuzzy systems. *Fuzzy Sets and Systems*, Vol. 7, pp. 257–273.

Daley, S., Gill, K.F. (1986). A design study of a self-organsing fuzzy logic controller. *Proc. Institute of Mechanical Engineers*, Vol. 200, C1, pp. 59–69.

Dadios, E.P., Williams, D.J. (1996). A fuzz-genetic controller for the flexible pole-cart balancing problem. *IEEE International Conference on Evolutionary Computing*, Nagoya, Japan, pp. 223–228.

Damousis, I.G., Satsios, K.J., Labridis, D.P., Dokopoulos, P.S. (2002). Combined fuzzy logic and genetic algorithm techniques-application to an electromagnetic field problem. *Fuzzy Sets and System*, Vol. 129, No. 3, pp. 371–386.

Deb, K., Agraval, S., Pratap, A., Meyarivan, T. (2000). A fast elitist non-dominated sorting genetic algorithm for multiobjective optimization NSGA II. *Parallel Problem Solving from Nature* – PPSN VI, Springer, Berlin, pp. 849–858.

Deb, K. (2001). *Multiobjective optimization using evolutionary algorithms*. John Wiley & Sons.

Deb, K., Goel, T. (2001). Controlled elitist non-dominated sorting genetic algorithims for better convergence. *Proceedings of the First International Confrence on Evolutionary Multi-Criterion Optimization* (EMO-2001), pp. 67–81.

De Jong, K.A., Potter, M.A. (1995). Evolving complex structures via cooperative coevolution. *Fourth Annual Conference on Evolutionary Programming*, San Diego, CA, pp. 307–317.

Demirci, M. (2004). Design of feedback controllers for linear system with applications to control of a double-inverted pendulum. *International Journal of Computational Cognition*, Vol. 2, No. 4, pp. 65–84.

Desylva, M.J. (1994). Hybrid fuzzy logic control to stabilize an inverted pendulum from arbitrary initial conditions. *M.S. Thesis Air Force Inst. of Tech*., Wright-Patterson AFB, OH.

Di Nola, A., Lettieri, A., Perfilieva, I., Novak, V. (2007). Algebraic analysis of fuzzy sets. *Fuzzy Sets and Systems*, Vol. 158, No 1, pp. 1–22.

Doostfatemeh, M., Kremer, S.C. (2005). Developing a new fuzzy controller. *Fuzzy Information Processing Society*, NAFIPS 2005, pp. 187–192.

Duan, J.C., Chung, F.L. (2002). Multilevel fuzzy relational systems: Structure and identification. *Soft Comp*. Vol. 6, pp. 71–86.

Feng, M., Harris, C.J. (2001). Piecewise Lyapunov stability conditions of fuzzy systems. *IEEE Transactions on Systems, Man, and Cybernetics* – Part B: Cybernetics, Vol. 31, No. 2, pp. 259–262.

Feng, G. (2002). An approach to adaptive control of fuzzy dynamic systems. *IEEE Transactions on Fuzzy Systems,* Vol. 10, No. 5, pp. 676–697.

Feng, G. (2006). A survey on analysis and design of model-based fuzzy control systems. *IEEE Transactions on Fuzzy Systems,* Vol. 14, No. 2, pp. 268–275.

Fischle, K., Schroder, D. (1999). An improved stable adaptive fuzzy control method. *IEEE Transactions on Fuzzy Systems*, Vol. 7, No. 1, pp. 27–40.

Fonseca, C. M., Fleming, P. J. (1993). Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. *Genetic Algorithms: Proceedings of the Fifth International Conference*, Urbana-Champaign, USA, pp. 416–423.

Fonseca, C.M., J. Fleming, P.J. (1998). Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms—Part I: A Unified Formulation**.** *IEEE Transactions on Systems, Man, and Cybernetics*, Part A: Systems and Humans**,** Vol. 28, No. 1, pp. 26–37.

Fonseca, C.M., J. Fleming, P.J. (1998). Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms—Part II: A Application Example. *IEEE Transactions on Systems, Man, and Cybernetics,* Part A: Systems and Humans**,** Vol. 28, No. 1, pp. 38–47.

Gao, Y., Err, M.J. (2003). Online adaptive fuzzy neural identification and control of a class of MIMO nonlinear systems. *IEEE Transactions on Fuzzy Systems*, Vol.11, Issue 4, pp. 462–477.

Goldberg, D.E. (1989). *Genetic algorithms for search, optimisation, and machine learning*. Reading, MA, Addison-Wesley.

Goldberg, D.E. (2002). *Genetic algorithms*. Addison Wesley, USA.

Golea, N., Golea, A., Benmahammed, K. (2002). Fuzzy model reference adaptive control. *IEEE Transactions on Fuzzy Systems*, Vol. 10, No. 6, pp. 803–805.

Graham, P, Newell, R. (1988). Fuzzy identification and control of a liquid level rig. *Fuzzy Sets and Systems*, Vol. 26. pp. 255–273.

Gupta, M.M., (1992). Fuzzy logic and neural networks. *Proceedings of the Tenth International Conference on Multiple Criteria Decision Making*, Taipei, pp. 281–294.

Han, H., Su, C.Y., Stepanenko, Y. (2001). Adaptive control of a class of nonlinear systems with nonlinearly parameterized fuzzy approximators. *IEEE Transactions on Fuzzy Systems,* Vol. 9, No. 2, pp. 315–323.

Harris, C.J., Moore, C.G., Brown, M., (1993). *Intelligent control: Aspects of fuzzy logic and neural nets*. World Scientific, Singapore.

Hills, D.W. (1990). *Co-evolving parasites improve simulated evolution as an optimization procedure*. Artificial Life II, Addison Wesley, pp. 313–324.

Holland, J.H. (1975). *Adaptation in neural and artificial systems*. University of Michigan Press.

Holmblad, L.P., Ostergaard, J.J. (1982). Control of a cement kiln by fuzzy logic. Eds. Gupta, M.M., Sanchez, E., *Fuzzy Information and Decision Processes*, Amsterdam, pp. 398–409.

Homaifar, A., McCormick, E. (1995). Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms. *IEEE Transactions on Fuzzy Systems*, Vol.3, No. 2, pp. 129–139.

Horn, J., Nafploitis, N., Goldberg, D. (1994). A niched Pareto genetic algorithm for multi-objective optimisation. *Proceeding of the First IEEE Conference on Evolutionary Computation*, pp. 82–87.

Hsu, Y.C., Chen,G., Li, H.X. (2001). A fuzzy adaptive variable structure controller with applications to robot manipulators. *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, Vol. 31, No. 3, pp. 331–340.

Huang, Y.P., Wang, S.F. (2000). Designing a fuzzy model by adaptive macroevolution genetic algorithms. *Fuzzy Sets Syst*., Vol. 113, pp. 367–379.

Hughes, E.J. (2005). Evolutionary many-objective optimisation: many once or one many? *Evolutionary Computation*, Vol. 1, pp. 222–227.

Ichihashi, H., Tokunaga, M. (993). Neuro-fuzzy optimal control of backing up a trailer truck. *Proceedings of IEEE International Conference on Neural Networks*, ICNN'93, San Francisco, pp. 306–311.

Ioannou, P.A., Sun, J. (1995). *Stable and robust adaptive control*. Englewood Cliffs, Prentice-Hall.

Jamshidi, M., Coelho, L.S., Krohling, R.A., Fleming, P.J. (2003). *Robust control systems with genetic algorithms*. CRC Press.

Johansen, T.A., Shorten, R., Murray-Smith, R. (2000). On the interpretation and identification of dynamic Takagi–Sugeno models. *IEEE Transactions on Fuzzy Systems*, Vol. 8, No. 3, pp. 297–313.

Joo, M.G., Lee, J.S. (2002). Universal approximation by hierarchical fuzzy system with constrains on the fuzzy rul. *Fuzzy Sets and Systems*, Vol. 130, pp. 175–188.

Kandel, A., Manor, O., Klein, Y., Fluss, S. (1999). ATM traffic management and congestion control using fuzzy logic. *IEEE Transactions on Systems, Man, and Cybernetics, Part C,* Appl. Rev., Vol. 29, No. 3, pp. 474–480.

Kaynak, O., Erbatur, K., Ertugnrl, M. (2001). The fusion of computationally intelligent methodologies and sliding-mode control – A survey. *IEEE Transactions on Industrial Electronics*, Vol. 48, No. 1, pp. 4–17.

Khwan-on, S., Kulworawanichpong, T., Srikaew, A., Sujitjorn, S. (2004). Neuro-tabu-fuzzy controller to stabilize an inverted pendulum system**.** *TENCON 2004, 2004 IEEE Region 10 Conference*, Vol. D, pp. 562–565.

Kickert, W.J.M., Lemke, H.V.N. (1976). Application of a fuzzy controller in a warm water plant. *Automatica*, Vol. 12, No. 4, pp. 301–308.

Kickert, W.J.M., Mamdani, E. (1978). Analysis of a fuzzy logic controller. *Fuzzy Sets and Systems*, Vol. 1, pp. 29–44

Kingham, M., Mohammadian, M., Stonier, R.J. (1998). Prediction of interest rate using neural networks and fuzzy logic. *Proceedings of ISCA 7th International Conference on Intelligent Systems,* Melun, Paris, pp. 72–75.

Kiszka, J.B., Gupta, M.M., Nikiforuk, P.N. (1985). Energestistic stability of fuzzy dynamic systems. *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 15, No. 6, pp. 783–792.

Konar, A. (2005). *Computational intelligence*. Springer Verlag, Berlin.

Koo, T.J. (2001). Stable Model Reference Adaptive Fuzzy Control of a Class of Nonlinear Systems. *IEEE Transactions on Fuzzy Systems*, Vol. 9, No. 4, pp. 624–636.

Kosko, B. (1992). *Neural networks and fuzzy system: a dynamical approach to machine intelligence*. Englewood Cliffs, Prentice Hall, USA.

Kosko, B. (1993). *Fuzzy thinking: the new science of fuzzy logic*. Hyperion, NY, USA.

Krstic, M., Kanellakopoulos, I., Kokotovic, P. (1995). *Nonlinear and adaptive control design*. New York, Wiley.

Kumar, M., Garg, D.P. (2004). Intelligent learning of fuzzy logic controllers via neural network and genetic algorithm. *Proceedings of 2004 JUSFA, 2004 Japan – USA Symposium on Flexible Automation*, Denver, Colorado, pp. 1–8.

Kwon, Y.H., Kim, B.S., Lee,S.Y., Lim, M.T. (2001). Swing up controller for inverted pendulum system. *Proceedings of the 32nd ISR (International Symposium on Robotics)*, Mexico City, pp. 896–901.

Larkin, L.I. (1985). A fuzzy logic controller for aircraft flight control. Ed. M. Sugeno, *Industrial Applications of Fuzzy Control*, Amsterdam, pp. 87–104.

Laumans, M., Thiele, L., Deb, K., Zitzler, E. (2002). Combining convergence and diversity in evolutionary multiobjective optimization. *Evolutionary Computation*, Vol. 10 No. 3, Massachusetts Institute of Technology, pp. 263–282.

Lee, C.C. (1990). Fuzzy logic in control systems: fuzzy logic controller, part I and II. *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 20, No. 2, pp. 404–436.

Lee, M.A. Takagi, H. (1993). Integrating design stages of fuzzy systems using genetic algorithms. *Proceedings IEEE International Conference Fuzzy Systems*, Vol. I, pp. 612–617.

Lee, M.A. Takagi, H. (1993a). Dynamic control of genetic algorithms using fuzzy logic techniques. *Proceedings of the Fifth International Conference on Genetic Algorithms*, pp. 76–83.

Lee, M.L., Chung, H.Y., Yu, F.M. (2003). Modeling of hierarchical fuzzy systems. *Fuzzy Sets and Systems*, 138, pp. 343–361.

Lee, Y.M., Jang, S.I., Chung, K.W., Lee, D.Y., Kim, W.C., Lee, C.W. (1994). A fuzzy-control processor for automatic focusing. *IEEE Transactions on Consumer Electronics*, Vol. 40, No. 2, pp. 138–144.

Lee, Y.G., Zak, S.H. (2004). Uniformly ultimately bounded fuzzy adaptive tracking controllers for uncertain systems. *IEEE Transactions on Fuzzy Systems*, Vol. 12, No. 6, pp. 797–811.

Lei, S., Langari, R. (2000). Hierarchical fuzzy logic control of a double inverted pendulum. *Fuzzy System 2000, FUZZ IEEE 2000, The Ninth IEEE International Conference,* Vol. 2, pp. 1074–1077.

Lei, S., Langari, R. (2003). Synthesis and approximation of fuzzy logic controllers for nonlinear systems. *International Journal of Fuzzy Systems*, Vol. 5, No 2, pp. 98–105.

Leung, F.H.F., Lam, H.K., Tam, P.K.S., Lee, Y.S. (2003). Stable fuzzy controller design for uncertain nonlinear systems: genetic algorithm approach. *FUZZ '03, The 12th IEEE International Conference on Fuzzy Systems*, Vol. 1, pp. 500–505.

Lin, C.T., Lee, C.S.G. (1996). *Neural fuzzy systems*. Prentice Hall.

Lin, S.C., Chen, Y.Y. (1997). Design of self learning fuzzy sliding mode controllers based on genetic algorithms. *Fuzzy Sets and Systems*, Vol. 86, pp. 139–153.

Lin, L.C., Lee, G.Y. (1999). Hierarchical fuzzy control for C-axis of CNC tuning centres using genetic algorithms. *Journal of Intelligent Robotic Systems*, Vol. 25, No. 3, pp. 255–175.

Lin, C.M., Mon, Y.J. (2005). Decoupling control by hierarchical fuzzy sliding-mode controller. *IEEE Transactions on Control Systems Technology*, Vol. 13, Issue 4, pp. 593–589.

Lo, J.C., Kuo, Y.H. (1998). Decoupled fuzzy sliding mode control. *IEEE Transactions on Fuzzy Systems*, Vol. 6, No. 3, pp. 426–435.

Magdalena, L. (1998). Hierarchical fuzzy control of a complex system using meta-knowledge. *Proceedings of the 7th International conference on Information Processing and Management of Uncertainty in Knowledge-based Systems*, Paris, pp. 630–637.

Margaliot, M., Langholz, G. (1999). Fuzzy Lyapunov-based approach to the design of fuzzy controllers. *Fuzzy Sets and Systems*, Vol. 106, pp. 49—59.Mamdani, E., Assilian, S. (1975). An experiment in linguistic synthesis with fuzzy logic controller. *International Journal of Man-Machine Studies*, Vol. 7, No. 1, pp. 1–13.

Mamdani, E. (1976). Advances in the linguistic synthesis of fuzzy controllers. *International Journal of Man-Machine Studies*, Vol. 8, No. 6, pp. 669–678.

Mao, X., Stonier, R.J., Thomas, P. (2001). An effective fuzzy image filter. *Proceedings of the International Conference on Computational Intelligence for Modelling Control and Automation* (CIMCA'01), Las Vegas, pp. 280–285.

Matellan, V., Fernandez, C., Molina, J.M. (1998). Genetic learning of fuzzy reactive controllers. *Robotics and Autonomous Systems*, Vol. 25, pp. 33–41.

Melba, M.P., Marimuthu, N.S. (2008). Design of intelligent hybrid controller for swing-up and stabilisation of rotary inverted pendulum. *ARPN Journal of Engineering and Applied Science*, Vol. 3, No. 4, pp. 60–70.

Michalewicz, Z. (1994). *Genetic Algorithms + Data Structures = Evolution Programs*. 2nd Ed., Springer Verlag.

Mitra, S., Hayashi, Y.(2000). Neuro-fuzzy rule generation: Survey in soft computing framework. *IEEE Transactions on Neural Networks*, Vol. 11, No. 3, pp. 748–768.

Mlynski, M.F., Zimmermann H.J. (2008). An efficient method to represent and process imprecise knowledge. *Applied Soft Computing*, 8, pp. 1050–1067.

Mohammadian, M., Stonier, R.J. (1995a). Adaptive two layer fuzzy logic control of a mobile robot system. *Proceedings of IEEE International Conference on Evolutionary Computing*, Perth, WA, Australia, Vol.1, pp. 204–209.

Mohammadian, M., Stonier, R.J. (1995b). Self learning hierarchical fuzzy logic controller in multi-robot systems. *Proceedings of the IEA Conf. Control95*, Melbourne, Australia, pp. 381–386.

Mohammadian, M., Stonier, R.J. (1996). Fuzzy rule generation by genetic learning for target tracking. *Proc. of the 5th Int. Intelligent Systems Conference*, Reno, Nevada, pp. 10–14.

Mohammadian, M.,Stonier, R.J. (1996a). Evolutionary learning in fuzzy logic control systems. Eds. R.Stocker *et al.*, *Complex Systems 96; From Local Interactions to Global Phenoma*, IOS Press, pp. 193–212.

Mohammadian, M., Stonier, R.J. (1996b). Intelligent hierarchical control for obstacle-avoidance. *Computation Techniques and Applications: CTAC95*, World Scientific, pp. 733–740.

Mohammadian, M., Stonier, R.J. (2000). Hierarchical fuzzy control. Eds. Bouchon-Meunier, B., Yager, R.R., Zadeh, L.A., *Uncertainty in Intelligent and Information Systems, Advances in Fuzzy Systems - Application and Theory* Vol. 20, World Scientific, pp. 119–130.

Mohammadian, M. (2002). Designing customised hierarchical fuzzy systems for modelling and prediction. *Proceedings of the International Conference on simulated Evolution and Learning (SEAL'02)*, Singapore, CD ISBN: 981047523.

Mohammadian, M., Kingham, M. (2004). An adaptive hierarchical fuzzy logic system for modelling of financial systems. *Journal of Intelligent Systems in Accounting, Finance, and Management,* Wiley Interscience, Vol. 12, pp. 61—82.

Mon, Y.J., Lin, C.M. (2002). Hierarchical fuzzy sliding-mode control. *The 2002 IEEE World Congress on Computational Intelligence*, pp. 656–661.

Muskinja, N., Tovornik, B. (2006). Swinging up and stabilization of a real inverted pendulum. *IEEE Transactions on Industrial Electronics,* Vol. 53, Issue 2, pp. 631–639.

Nainar, I., Mohammadian, M., Stonier, R.J., Millar, J. (1996). An adaptive fuzzy logic controller for control of traffic signals. *Proceedings of the 4th International Conference on Control, Automation, Robotics and Computer Vision* (ICARCV'96), Singapore, pp. 578–582.

Odetayo. M.O., McGregor, D.R. (1989). Genetic algorithm for inducing control rules for a dynamic system. *Proc. of the 3rd International Conference on Genetic Algorithms,* George Mason University, pp. 177–182.

Paulo, S. (2005). Clustering and hierarchization of fuzzy systems. *Soft Comp. Journal,* Vol. 9, No. 10, pp. 715–731.

Pal, T., Pal, N.R. (2003). SOGARG: A self-organized genetic algorithm-based rule generation scheme for fuzzy controllers. *IEEE Transactions on Evolutionary Computation,* Vol. 7, No. 4, pp. 397–415.

Passino, K.M., Yurkovich, S. (1998). *Fuzzy control*. Addison-Wesley.

Pedrycz, W. (1984). An identification algorithm in fuzzy relational systems. *Fuzzy Sets and Systems,* Vol. 13, pp. 153–167.

Pedrycz, W. (1993). *Fuzzy control and fuzzy systems*. New York, Wiley.

Pena-Reyes, C.A., Sipper, M. (2001). Fuzzy CoCo: A cooperative-coevolutionary approach to fuzzy modeling. *IEEE Transactions on Fuzzy Systems*, Vol. 9, No. 5, pp. 727–737.

Phan, P.A., Gale, T. (2007). Two-mode adaptive fuzzy control with approximation error estimator. *IEEE Transactions on Fuzzy Systems,* Vol. 15, Issue 5, pp. 493–955.

Potter, M.A., De Jong, K. (1994). A cooperative coevolutionary approach to function optimization. *Proceedings of Parallel Problem Solving from Nature III* (PPSN III), Berlin, Springer Verlag, pp. 249–257.

Potter, M.A., De Jong, K. (1995). Evolving neural networks with collaborative species. *Summer Computer Simulation Conference*. The Society of Computer Simulation, pp. 340–345.

Procyk, T., Mamdani, E. (1979). A linguistic self-organising process controller. *Automatica,* Vol. 15, No. 1, pp. 15–30.

Qiao, F., Zhu, Q.M., Winfield, A., Melhuish, C. (2003). Fuzzy sliding mode control for discrete nonlinear systems. *Transactions of China Automation Society,* Vol. 22, No 2, pp. 313–315.

Rajapakse, A., Furuta, K., Kondo, S. (2002). Evolutionary learning of fuzzy logic controllers and their adaptation through perpetual evolution. *IEEE Transactions on Fuzzy Systems*, Vol. 10, No. 3, pp. 309–321.

Raju, G.V.S., Zhou, J., Kisner, R.A., (1990). Fuzzy logic control for steam generator feedwater control. *Proceedings of American Control Conference,* San Diego, CA, pp. 1491–1493.

Raju, G.V.S., Zhou, J., Kisner, R.A., (1991). Hierarchical fuzzy control. *International Journal on Control,* Vol. 54, No. 5, pp. 1201–1216.

Raju, G.V.S, Zhou, J. (1993). Adaptive hierarchical fuzzy controller. *IEEE Transactions on Systems, Man, Cybernetics,* Vol. 23, No. 4, pp. 973–980.

Ray, K.S., Majumder, D.D. (1984). Application of circle criteria for stabilisaty analysis of linear SISO and MIMO systems associated with fuzzy logic controllers. *IEEE Transactions on Systems, Man, Cybernetics,* Vol. 14, No. 2, pp. 345–349.

Ramos, M.C., Koivo, A.J. (2002). Fuzzy logic-based optimization for redundant manipulators. *IEEE Transactions On Fuzzy Systems*, Vol. 10, No. 4, pp. 498–509.

Rojas, I., Pomares, H., Ortega, J., Prieto, A. (2000). Self-organized fuzzy system generation from training examples. *IEEE Transactions On Fuzzy Systems*, Vol. 8, No. 1, pp. 23–36.

Russo, M. (2000). Genetic fuzzy learning. *IEEE Transactions on Evolutionary Computing*, Vol. 4, No. 3, pp. 259–273.

Saifuzul, A.A., Abu Osman, N.A., Azlan, C.A., Ungku Ibrahim, U.F.S. (2006). Intelligent control for self-erecting inverted pendulum via adaptive neuro-fuzzy inference system. *American Journal of Applied Sciences*, 3 (4), pp. 1795–1802.

Sazonov, E.S., Klinkhachorn, P., Klein, R.L. (2003). Hybrid LQG-Neural controller for inverted pendulum system. *Proceedings of the 35th Southeastern Symposium on System Theory*, pp. 206–210.

Sbalzarini, I.F., Muller, S., Koumoutsakos, P. (2001). Multiobjective optimization using evolutionary algorithms. *Proceedings of the Summer Program 2000*, Center for Turbulance Research, pp. 63–74.

Shaffer, J.D. (1985). Multiple objective optimisation with vector evaluated genetic algorithms. *Proceedings of the 1st International Conference on Genetic Algorithms*, pp. 93–100.

Sharma, S.K., Irwin, G.W. (2003). Fuzzy coding of genetic algorithms. *IEEE Transactions on Evolutionary Computation*, Vol. 7, No. 4, pp. 344–355.

Shimojima, K., Fukuda, T., Hasegawa, Y. (1995). Self-tuning fuzzy modelling with adaptive membership function, rules, and hierarchical structure based on genetic algorithm. *Fuzzy Sets Syst.*, Vol. 71, pp. 295–309.

Slotine J.J.E., Li, W. (1991). *Applied nonlinear control*. Prentice Hall.

Srinivas, N, Deb, K. (1994). Multio-bjective function optimisation using non-dominated sorting genetic algorithms. *Evolutionary Computation Journal*, Vol. 2, No. 3, pp. 221–248.

Stonier, R.J., Mohammadian, M. (1995). Self learning hierarchical fuzzy logic controller in multi-robot systems. *Proceedings of the IEA Conference Control95*, Melbourne, Australia, pp. 381–386.

Stonier, R.J., Mohammadian, M. (1996a). Evolutionary learning in fuzzy logic control systems. *Invited Address, Proceedings of the third National Conference on Complex Systems 96*; From Local Interactions to Global Phenomena, IOS Press, pp. 193–212.

Stonier, R.J., Mohammadian, M. (1996b). Intelligent hierarchical control for obstacle-avoidance. *Computational Techniques and Applications: CTAC95*, World Scientific, pp. 733–740.

Stonier, R.J. (1999). Evolutionary learning of fuzzy logic controllers over a region of initial states. *Proceedings of the 1999 Congress on Evolutionary Computation*, Vol. 2, Washington, pp. 2131–2138.

Stonier, R.J., Stacey, A.J., Messom, C. (1998). Learning fuzzy controls for the inverted pendulum. *Proceedings of ISCA 7th International Conference on Intelligent Systems*, Melun, pp. 64–67.

Stonier, R.J., Mohammadian, M. (1998). Knowledge acquisition for target capture. *Proceedings of the IEEE International Conference on Evolutionary Computing*, ICEC'98, Anchorage, Alaska, pp. 721–726.

Stonier, R.J., Stacey, A., Mohammadian M., Smith, S.F. (1999). Application of evolutionary learning in fuzzy logic and optimal control. Ed. M. Mohammadian, *Computational Intelligence for Modelling, Control and Automation*, IOS Press, pp. 76–85.

Stonier, R.J., Drumm, M.J., Bell, J. (2003). Optimal boundary control of a tracking problem for a parabolic distributed system using hierarchical fuzzy control and evolutionary algorithms. *Proceedings of the 18th International Conference on Computers and their Applications*, Honolulu, Hawaii, on CD ISBN 1-880843-46-3, pp. 214–218.

Stonier, R.J. Mohammadian, M. (2004). Multi-layered and hierarchical fuzzy modelling using evolutionary algorithms. *Proceedings of the International Conference on Computational Intelligence for Modelling Control and Automation (CIMCA'2004),* Gold Coast, Australia, pp. 321–344.

Stonier, R.J., Zajaczkowski, J. (2003). Model reference control using sliding mode with Hamiltonian dynamics. *The ANZIAM Journal (The Australian & New Zealand Industrial and Applied Mathematics Journal)* Vol. 45 Part E, 2003, pp. E1–E40.

Stonier, R.J., Zajaczkowski, J. (2003). Hierarchical fuzzy controllers for the inverted pendulum. *Proceedings of CIRAS 2003*, on CD ISSN: 0219-613, PS01-4-03, Singapore.

M. Sugeno, M. (1999). On stability of fuzzy systems expressed by fuzzy rules with singleton consequents. *IEEE Transactions on Fuzzy Systems*, Vol. 7, No. 2, pp. 201–224.

Sun, Y.L., Er, M.J. (2004). Hybrid fuzzy control of robotics systems. *IEEE Transactions on Fuzzy Systems*, Vol. 12, Issue 6, pp. 755—765.

Suykens, J.A.K., Vandewalle, J., de Moor, B. (2001). Optimal control by least squares support vector machines. *Neural Networks,* Vol. 14, pp. 23–35.

Takagi, T., Sugeno, M. (1985). Fuzzy identification of systems and its applications to modelling and control. *IEEE Transactions on Systems, Man, and Cybernetics,* Vol. 15, No. 1, pp. 116–132.

Tanaka, K., Wang, H.O. (2001). *Fuzzy Control Systems Design and Analysis: A LMI Approach*. New York, Wiley.

Tang, K.S, Man, K.F, Liu, Z.F., Kwong, S. (1998). Minimal fuzzy memberships and rules using hierarchical genetic algorithms. *IEEE Transactions on Industrial Electronics*, Vol. 45, No. 1, pp. 162–169.

Tang,Y., Velez-Diaz, D. (2003). Robust control of mechanical systems. *IEEE Transactions On Fuzzy Systems,* Vol. 11, No. 3, pp. 411–410.

Thomas, P.J., Stonier, R.J. (2003). Fuzzy control in robot-soccer, evolutionary learning in the first layer of control. *Journal of Systems, Cybernetics and Informatics*, Vol. 1, No. 1, pp. 75–80.

Thomas, P.J., Stonier, R.J. (2003). Evolutionary learning of a 5I2O fuzzy controller including wheel lift constraint. *Proceedings of the 2nd International Conference on Computational Intelligence, Robotics and Autonomous Systems (CIRAS 2003),* Singapore, CD, ISSN: 0219-613, PS04-4-01.

Thomas, P.J., Stonier, R.J. (2003). Hierarchical fuzzy control in robot soccer using evolving algorithms. *Proceedings of the International Congress on Evolutionary Computation (CEC2003),* Canberra, Vol. 4, pp. 2434–2440.

Thrift, P. (1991). Fuzzy logic synthesis with genetic algorithms. Eds. Belew, R.K., Booker, L.B., *Proceedings of the Fourth International Conference on Genetic Algorithms*, Morgan Kaufmann Pub., San Mateo CA, pp. 509–513.

Tong, R.M., Beck, M. B., Latten, A. (1980). Fuzzy control of the activated sludge wastewater treatment process. *Automatica*, Vol. 6, pp. 695–701.

Tong, S.C., Li, H.X. (2003). Fuzzy adaptive sliding-mode control for MIMO nonlinear systems. *IEEE Transactions on Fuzzy Systems*, Vol. 11, No. 3, pp. 354–360.

Torra, V. (2002). A review of the construction of hierarchical fuzzy systems. *IJIS*, Vol. 17, pp. 531–543.

Tu, K.Y., Lee, T.T, Wang, W.J. (2000). Design of a multilayer fuzzy logic controller for multi-input multi-output systems. *Fuzzy Sets and Systems*, Vol. 111, pp. 199–214.

Utkin, V. I. (1992). *Sliding Modes in Control Optimization*. Berlin, Germany, Springer Verlag.

Van Veldhuizen, D.A., Lamont, G.B. (2000). On measuring multiobjective evolutionary algorithm performance. *Evolutionary Computation 2000*, Vol.1, pp. 204–211.

Varsek, A., Urbancic, T., Filipic, B. (1993). Genetic algorithms in controller design and tuning. *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 23, No. 5, pp. 1330–1339.

Velez-Diaz, D., Tang, Y. (2004). Adaptive robust fuzzy control of nonlinear systems. *IEEE Transactions on Systems, Man,* and Cybernetics, Part B, Vol. 34, No. 3, pp. 1596–1601.

Wai, R.J., Lee, J.D., Chang, L.J. (2003). Development of adaptive sliding-mode control for nonlinear dual-axis inverted-pendulum system**.** AIM 2003, Proceedings of 2003 IEEE/ASME International Conference on *Advanced Intelligent Mechatronics*, Vol. 2, pp. 815–820.

Wang, L.X. (1993). Stable adaptive fuzzy control of nonlinear systems. *IEEE Transactions on Fuzzy Systems,* Vol. 1, No. 2, pp. 146–155.

Wang, L.X. (1994). *Adaptive fuzzy systems and control*. Prentice Hall, Englewood Cliffs, New Jersey, USA.

Wang, L.X. (1997). *A Course in fuzzy systems and control*. Prentice Hall, NJ, USA.

Wang, L.X. (1999). Analysis and design of hierarchical fuzzy Systems. *IEEE Transactions on Fuzzy Systems,* Vol. 7, No. 5, pp. 617–624.

Wang,W.J., Yen, T.G., Sun, C.H. (2004). GA-based fuzzy rules generation. *International Symposium on Computer and Communication Engineering*, Korea, pp. 123–129.

Wang, W., Yi, J., Zhao, D., Liu, X. (2005). Design of cascade fuzzy sliding-mode controller. *2005 American Control Confrence,* Portland, USA, pp. 4649–4654.

Weicker, K., Weicker, N. (1999). On the improvement of coevolutionary optimizers by learning variable interdependencies. *Congress on Evolutionary Computation*, Washington, IEEE Press, pp. 1627–1632.

Wu, T.P., Chen, S.M. (1999). A new method for constructing membership functions and fuzzy rules from training examples. *IEEE Transactions on Systems, Man, and Cybernatics*, Vol. 29, No. 1, pp. 25–40.

Wu, A., Tam, P.K.S. (2000). A fuzzy neural network based on fuzzy hierarchy error approach. *IEEE Transactions on Fuzzy Systems,* Vol. 8, Issue 6, pp. 808–816.

Xiao, J., Xiao, J.Z., Xi, N., Tummala, R. L., Mukherjee, R. (2004). Fuzzy controller for wall-climbing microrobots. *IEEE Transactions on Fuzzy Systems,* Vol. 12, No. 4, pp. 466–480.

Yager, R.R. (1998). On the construction of hierarchical fuzzy systems models. *IEEE Transactions on Systems, Man, and Cybernetics* – Part C: Applications and Reviews, Vol. 28, No. 1, pp. 55–66.

Yeh, Z.M., Li, K.H. (2004). A systematic approach for designing multistage fuzzy control systems. *Fuzzy Sets and Systems,* Vol. 143, No. 2, pp. 251–273.

Yen, G.G., Lu, H. (2003). Dynamic multiobjective evolutionary algorithm: Adaptive cell-based rank and density estimation. *IEEE Transactions on Evolutionary Computation*, Vol. 7, No. 3, pp.253–274.

Yi, J., Yubazaki, N. (2000). Stabilization fuzzy control of inverted pendulum systems. *Artificial Intelligence in Engineering*, Vol. 14, pp. 153–163.

Yi, J., Yubazaki, N., Hirota, K. (2002). A new fuzzy controller for stabilization of parallel-type double inverted pendulum system. *Fuzzy Sets and Systems*, Vol. 126, pp. 105–119.

Young, N., Stonier, R.J. (2003). Co-evolutionary learning and hierarchical fuzzy control for the inverted pendulum. *Proceedings of the International Congress on Evolutionary Computation (CEC2003),* Canberra, Vol 1, pp. 467–473.

Yu, W.S., Sun, C.J. (2001). Fuzzy model based adaptive control for a class of nonlinear systems. IEEE Transactions on *Fuzzy Systems,* Vol. 9, pp. 413–425.

Zadeh, L.A. (1965). Fuzzy sets. *Information and Control*, Vol. 8, pp. 338–353.

Zadeh, L.A. (1968). Fuzzy algorithms. *Information and Control*, Vol. 12, No. 2, pp. 94–102.

Zadeh, L.A. (1971). Towards a theory of fuzzy systems. Eds. Kalman, R.E, DeClaris, N., *Aspects of Network and System Theory,* pp. 209–245.

Zadeh, L.A. (1973). Outline of a new approach to the analysis of complex systems and decision processes . *IEEE Transactions on Systems, Man, and Cybernetics,* Vol. 3, No. 1, pp. 177–200.

Zadeh, L.A. (1975). The concept of a linguistic variable and its application to approximate reasoning I, II, III. *Information Sciences*, Vol. 8, pp. 199–251, pp. 301–357, pp. 43–80.

Zadeh, L.A. (1978). Fuzzy sets as a basis for theory of possibility. *Fuzzy Sets and Systems*, Vol. 1, pp. 3–28.

Zadeh, L.A. (1996). Fuzzy logic = computing with words. *IEEE Transactions on Fuzzy Systems*, Vol. 4, pp. 103–111.

Zajaczkowski, J. (2000). *Model reference adaptive control in hamiltonian systems*. Central Queensland University, Rockhampton, MSc Thesis.

Zajaczkowski, J., Stonier, R.J. (2004). Analysis of hierarchical control for the inverted pendulum. *Proceedings of Complex2004*, Cairns, (CD-ROM).

Zajaczkowski, J., Stonier, R.J. (2006). Co-evolutionary algorithm for hierarchical fuzzy control of the inverted pendulum. *Proceedings of WCCI2006, IEEE International Conference on Fuzzy Systems*, Vancouver, Canada, pp. 737–744.

Zajaczkowski, J., Stonier, R.J. (2008). Analysis of hierarchical control for the inverted pendulum. *Complexity International*, Vol. 12, Paper ID: msid49, URL: http://www.complexity.org.au/vol12/msid49/

Zajaczkowski, J, Verma, B. (2008). Hierarchical fuzzy control for the inverted pendulum over the set of initial conditions. Eds. Xiaodong Li *et al*. *Simulated Evolution and Learning. LNCS 5361, SEAL2008*. Springer, Heidelberg, pp. 534–543.

Zajaczkowski, J, Verma, B. (2009a). A compositional method using an evolutionary algorithm for finding fuzzy rules in 3-layered hierarchical fuzzy structure. *International Journal of Computational Intelligence and Applications*, Vol. 8, No 4, pp. 467–485.

Zajaczkowski, J, Verma, B. (2009b). A multiobjective evolutionary algorithm based compositional method for hierarchical fuzzy control. *ICMI 2009: International Conference on Machine Intelligence*, Bangkok, Thailand, pp. 1009–1016.

Zajaczkowski, J, Verma, B. (2010a). MOEA based hierarchical fuzzy control over the set of user-defined initial conditions. *2010 IEEE Conference on Fuzzy Systems, WCCI2010* Barcelona, Spain (accepted).

Zajaczkowski, J, Verma, B. (2010b). An evolutionary algorithm based approach for selection of topologies in hierarchical fuzzy systems. *2010 IEEE Congress on Evolutionary Computation, WCCI2010,* Barcelona, Spain (accepted).

Zeng, X.J., Singh, M.G. (1994). Approximation theory of fuzzy systems-SISO case. *IEEE Transactions on Fuzzy Systems*, Vol. 2, No. 2, pp. 162–176.

Zeng, X.J., Cai, L.L. (2002). Nonlinear adaptive control using the Fourier integral and its application to CSTR systems. *IEEE Transactions on Systems, Man, Cybernetics*, Part B, vol. 32, no. 3, pp. 367–372.

Zhong, W., Rock, H. (2001). Energy and passivity based control of the double inverted pendulum on a cart. *2001 IEEE Conference on Control Applications*, pp. 896–901.

Zhou,Y.S., Lai, L.Y. (2000). Optimal design for fuzzy controllers by genetic algorithms. *IEEE Transactions on Industry Applications*, Vol. 36, No. 1, pp. 93–97.

Zitzler, E. (1999). Evolutionary algorithms for multiobjective optimization: methods and applications. *PhD dissertation*, Computer Engineering and Networks Laboratory, Swiss Federal Institute of Technology, Zurich.

Zitzler, E., Thiele, L. (1999). Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, Vol. 3, No 4, pp. 257–271.

Zitzler, E., Deb, K., Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: empirical results. *Evolutionary Computation*, Vol. 2, No. 2, pp. 173–195.

Zitzler, E., Laumanns, M., Thiele, L. (2001). SPEA2 improving the strength Pareto evolutionary algorithm. *Technical Report 103*. Computer Engineering and Networks Laboratory TIK, Swiss Federal Institute of Technology, Zurich.

Zurada, J., Marks, R., Robinson, C. (Editors) (1994). *Computational intelligence: imitating life*. IEEE Press.