Robust Nonlinear Controller based on Set Propagation

Garron A. Fish

A dissertation submitted in partial fulfilment of the requirements
for the degree of Master of Science in Electrical Engineering
in the University of Cape Town, August 2003

Declaration

I declare that this dissertation is my own, unaided work. It is being submitted for the degree of Masters of Science in Engineering in the University of Cape Town. It has not been submitted before for any degree or examination in another university.

Signature of Author.....

Cape Town

August 31, 2003

Abstract

A novel control method, based on interval analysis, that optimises the control surface (or *u-surface*) for sampled systems with output disturbances is demonstrated on a driven pendulum with actuator constraints. The fitness function to be maximized is the probability of each state of the system being controlled to the setpoint without being perturbed to regions that are more iterations away from the setpoint. The u-surface is designed by finding all the states that could go to the setpoint in an interval and optimising these states. This process is repeated (backwards in time) by optimising states that go to the previously optimised states until no more states that have not been optimised are found. The proposed control method has been applied to the problem of swinging up a driven pendulum from rest to the inverted position with constraints on the torque of the motor. This method is computationally intensive and time constraints limit its current application to systems of low order.

Acknowledgements

Thank you to Martin Braae for his supervision, Malcom Attfield for his technical assistance and to Carly for the moral support she gave me over the years.

I would also like to thank UCT for the scholarship I received from them while doing this thesis.

Contents

List of Figures

List of Tables

List of Abbreviations

ADC	Analogue to digital converter
DAC	Digital to analogue converter
PDF	Probability distribution function
SISO	Single input single output
Sys.*x*	**Refers to different types of pendulum settings defined in Appendix A**

Chapter 1

Introduction

The aim of this thesis was originally to implement a robust neural network based controller on a nonlinear system with constraints. However no satisfactory method was found in the literature. After experimenting with a number of ideas to develop a controller to satisfy the requirements, the idea of u-surface optimal control became apparent. However, during the implementation of the u-surface optimal control a state space model was used to reduce the computational intensity.

Objectives of the thesis

The objectives of this thesis were to investigate the u-surface control method and to implement the method on a nonlinear control system. There are many nonlinear processes used in industry, of which one important example is the robot arm. The nonlinear dynamics of the robot arm are similar in nature to that of the driven pendulum system that we have in the laboratory. Therefore the u-surface controller was implemented on the driven pendulum.

Scope and limitations

The software designed to generate control surfaces in this thesis synthesises a robust controller for sampled second order single input single output nonlinear systems with constraints on the control input. The control surface found is optimized to be robust to output disturbances. A method for altering the control surface to have better performance in the presence of noise has also been implemented.

Historical background

While this thesis was underway papers were discovered (Jaulin, *et al*, 2002, Jaulin and Walters, 1997) that used methods similar in nature to the one described in this thesis.

Plan of development

The literature review follows in the next chapter. In Chapter 3, the u-surface optimisation method is described and then this method is realized in software in Chapter 4. A controller, designed using this software, is simulated on computer and applied to the driven pendulum experiment in Chapter 5. Finally, conclusions are drawn from the study and future development is proposed in Chapter 6.

Chapter 2

Literature review

The robust control of nonlinear systems with constraints on control and states of a system is a problem that is currently receiving a lot of attention in control literature. A number of control methods solve this nonlinear problem. Some of these methods are: Nonlinear model predictive control (Findeisen and Allgöwer, 2002); Global optimisation methods such as αBB:A global optimisation method (Androulakis, *et al.*, 1995; Harding and Floudas, 1997); Interval analysis optimisation techniques (Malan, *et al.*, 1997; VanAntwerp, *et al.*, 1999; Jaulin and Walter, 1997; Didrit, *et al.*, 1997; Jaulin, *et al.*, 2002; Vehi, *et al.*, 2001; Malan, *et al.*, 1997); Optimal trajectory methods (El-Farra and Christofides, 2000; Milam, *et al.*, 2000); Energy control (Åström and Furuta, 1996; Chatterjee D., Patra, *et al.*, 2002).

The control method described in this paper is very similar to the paper by Jaulin and Walter, (1997) that uses interval analysis. There are only a few papers about Interval analysis applied to control that have been written (see references above). But "interest is growing, as illustrated by a recent special issue of Reliable Computing" (quoted from (Garloff and Walter, 2000)).

The method for designing interval analysis and set computation proposed in Jaulin L., *et al.* (2002) finds a control surface that ensures that all solutions are robust and that guarantees that all the states from x_k go to x_{k+1} as in the disturbed system. The

controller described in this paper is a surface that contains control values for states of the system (the u-surface). Very few papers written about systems of this type of controller were found (Jaulin L., *et al.*, 2002; Jaulin and Walter, 1997).

The method investigated should be well suited for low order highly nonlinear systems and potentially for neural network modelled systems. Some examples of low order nonlinear systems are: Highly exothermic chemical reactions (Henson and Seborg, 1997), pH neutralizations (Henson and Seborg, 1997) and Hydraulic servomechanisms (Rong-Hong, *et al.*, 1997) and robot manipulators (Gupta K.,1997).

2.1 Interval analysis

In Jaulin and Walter (1997) they state that:

> "Interval analysis (Moore, 1993) is a fundamental numerical tool for proving properties of sets, solving sets of nonlinear equations or inequalities, and optimising nonconvex criteria in a guaranteed way."

Section 2.1.1 and Section 2.1.2 summarizes the work presented by Jaulin and Walter (1997) and by Jaulin, *et al.* (2002) respectively.

2.1.1 Introduction to intervals analysis

Nonmenclature:

x	Scalars
X	Scalar intervals
$\mathbf{x}$	Vectors
X	Vector intervals

An interval is a region defined by $X_i = [x_i^-, x_i^+]$ where X_i contains all the elements from x_i^- to x_i^+. A *vector interval* ($\mathbf{X} \in \mathbb{R}^n$) is a box that is the Cartesian product of n real intervals X_i.

$$\mathbf{X} = [x_1^-, x_1^+] \times \cdots \times [x_n^-, x_n^+] = X_1 \times \cdots \times X_n \tag{2.1}$$

The intervals are manipulated in a very similar way as their scalar counterparts. For example let **f**: $\mathbb{R}^n \rightarrow \mathbb{R}^p$ be a vector function; the "equivalent" set-valued function is **F:** $\mathbb{R}^n \rightarrow \mathbb{R}^p$ where $\mathbb{R}^r \in \mathbb{R}^n$. **F** is a (convergent) *inclusion function* of **f** if, for any vector interval **X**, it satisfies the two following conditions:

$$\mathbf{f}(X) \subset \mathbf{F}(X) \tag{2.2}$$

$$w(X) \rightarrow 0 \Rightarrow w(\mathbf{F}(X)) \rightarrow 0 \tag{2.3}$$

where $w(\mathbf{X})$ is the *width* of the vector interval of the box **X.** The *width* is the length of its largest side(s). Note that **f(X)** is not confined to being a box, compared to **F(X)** that is by definition. Example 1, taken directly from Jaulin and Walter (1997), illustrates the calculation of the inclusion function:"

"Example 1: An inclusion function for

$$\mathbf{f}(u, x_1, x_2) = \begin{pmatrix} x_1 \times \cos(x_1 \times x_2) + u \\ 3x_1^2 - \sin(u \times x_2) \end{pmatrix} \tag{2.4}$$

is

$$\mathbf{F}(\mathbf{X}, U) = \begin{pmatrix} X_1 \times \cos(X_1 \times X_2) + U \\ 3X_1^2 - \sin(U \times X_2) \end{pmatrix} \tag{2.5}$$

If, for instance, $\mathbf{X} = [0,1] \times [0, \pi/3]$ and $U = [-2,1]$, then the vector interval $\mathbf{F}(\mathbf{X}, U)$ is computed as follows:

$$\mathbf{F}(\mathbf{X}, U) = \begin{pmatrix} [0,1] \times \mathbf{cos}([0,1] \times [0, \pi/3]) + [-2,1] \\ 3 \times [0,1]^2 - \mathbf{sin}([-2,1] \times [0, \pi/3] \end{pmatrix}$$

$$= \begin{pmatrix} [0,1] \times \cos([0,\pi/3]) + [-2,1] \\ 3 \times [0,1]^2 - \sin([-2\pi/3, \pi/3]) \end{pmatrix}$$

$$= \begin{pmatrix} [0,1] \times [0.5,1] + [-2,1] \\ [0,3] - [-1, \sqrt{3}/2] \end{pmatrix}$$

$$= \begin{pmatrix} [-2,2] \\ [-\sqrt{3}/2, 4] \end{pmatrix} \tag{2.6}$$

The operators +,x,- and functions, cos, sin, $(.)^2$ and $(.)^{1/2}$ in (2.6) are interval counterparts of those in (2.5)."

Note that an inclusion function is any function that satisfies (2.2) and (2.3). In Example 1 a method to get an inclusion function was done by replacing each operator by its interval counterpart. It is possible to use other inclusion functions that have different performance characteristics as described in Tóth and Csendes (2002).

2.2 Set computation for robust nonlinear control

In Jaulin, *et al.* (2002) an efficient method based on set computation and constraint propagation is described for robust nonlinear control. This method is described in this section.

A robust controller for the discrete system with perturbations described in (2.7) is designed.

$$\boldsymbol{x}_k = \mathbf{f}(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}, \boldsymbol{w}_{k-1}) \quad k \in \{1, \cdots, \bar{k}\} \tag{2.7}$$

where $\boldsymbol{w}_k$, $\boldsymbol{u}_k$ and $\boldsymbol{x}_k$ are real vectors that belong to W_k, U_k and X_k domains respectively. $\boldsymbol{w}_k$ is the disturbance, $\boldsymbol{u}_k$ is the control value and $\boldsymbol{x}_k$ is the state at sampling time k. The robust control problem is to control the system from $\boldsymbol{x}_0$ to the final terminal set $X_{\bar{k}}$ under the presence of perturbations $\boldsymbol{w}_k$. The perturbations $\boldsymbol{w}_k$ are generally assumed to be bounded.

The robust control problem can be formulated as in (2.8).

As stated in Jaulin, *et al.* (2002):

"finding $\boldsymbol{u}_k \in U_k$ such that

$$\begin{cases} \exists \boldsymbol{u}_{k+1} \in U_{k+1},...,\exists \boldsymbol{u}_{\bar{k}-1} \in U_{\bar{k}-1} \\ \forall \boldsymbol{w}_0 \in W_0;...;\forall \boldsymbol{w}_{\bar{k}-1} \in W_{\bar{k}-1}; \\ \forall \boldsymbol{x}_0 \in X_0;\exists \boldsymbol{x}_1 \in X_1;...;\exists \boldsymbol{x}_{\bar{k}} \in X_{\bar{k}}; \\ \boldsymbol{x}_1 = \mathrm{f}(\boldsymbol{x}_0, \boldsymbol{u}_0, \boldsymbol{w}_0);...;\boldsymbol{x}_{\bar{k}} = \mathrm{f}(\boldsymbol{x}_{\bar{k}-1}, \boldsymbol{u}_{\bar{k}-1}, \boldsymbol{w}_{\bar{k}-1}) \\ \boldsymbol{x}_{k+1} = \mathrm{f}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{w}_k);...;\boldsymbol{x}_{\bar{k}} = \mathrm{f}(\boldsymbol{x}_{\bar{k}-1}, \boldsymbol{u}_{\bar{k}-1}, \boldsymbol{w}_{\bar{k}-1}) \end{cases} \tag{2.8}$$

"

We assume that the sets computations are possible. It is often not possible to do set computation exactly, but for systems with low dimensions it is possible to approximate the set operations by using interval solvers

.

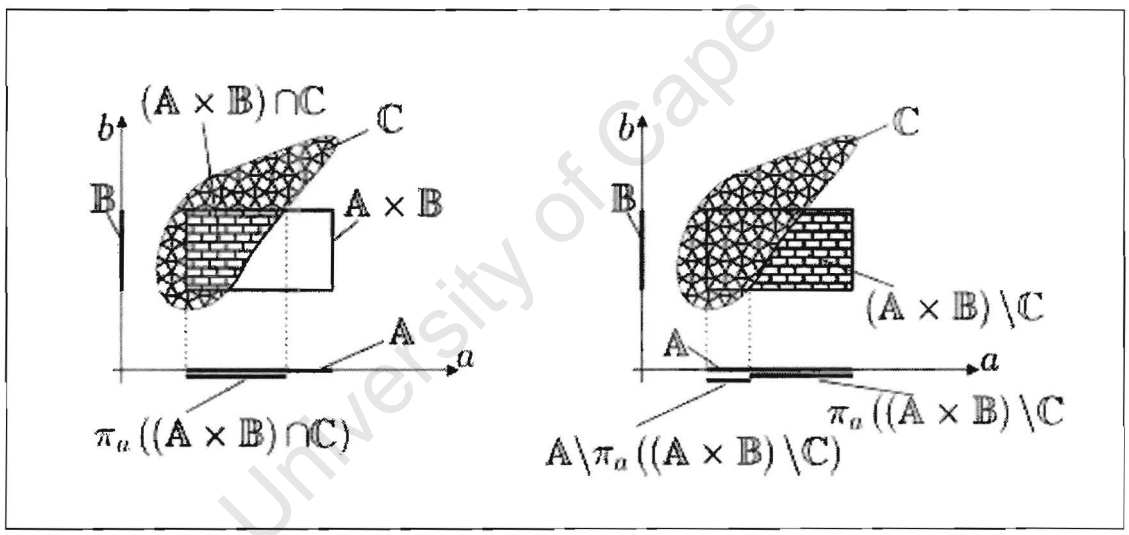


Figure 2.1 **Illustration of the set operations used in Table 2.1**

Figure 2.1. Illustration of set operations used in Table 2.1. It illustrates the operation of the following set functions: $\cap$,$\times$, \ , π_a

Table 2.1 **Algorithm of robust set computation control**

1	$\bar{X}_{\bar{k}} := X_{\bar{k}}$
2	for k:= $\bar{k}$ downto 2
3	$\bar{X}_{k-1} := X_k \cap \pi_x(X_{k-1} \times U_{k-1} \setminus \pi_{x,u}(X_{k-1} \times U_{k-1} \times W_{k-1} \setminus \mathbf{f}^{-1}(\bar{X}_k)))$;
4	endfor
5	$\hat{U}_0 := U_0 \setminus \pi_u(\bar{X}_0 \times U_0 \cap \pi_{x,u} \setminus (X_0 \times U_0 \times W_0 \setminus \mathbf{f}^{-1}(\bar{X}_1)))$;
6	if $\hat{U}_0 = \varnothing$ "The robust control problem has no solution"; end;
7	choose $\bar{\boldsymbol{u}} \in \hat{U}_0$;
8	$\bar{X}_0 := X_0$
9	for k:= 1 to $\bar{k}$ -1
10	$\bar{X}_k := X_k \cap \mathrm{f}(\bar{X}_{k-1}, \tilde{u}_{k-1}, W_{k-1})$;
11	$\hat{U}_k := U_k \setminus \pi_u(\bar{X}_k \times U_k \times W_k \setminus \mathbf{f}^{-1}(\bar{X}_{k+1}))$;
12	choose $\tilde{\boldsymbol{u}}_k \in \hat{U}_k$;
13	endfor

Fig. 2.1 and Table 2.1 are from **Jaulin, *et al.* (2002)**.

As stated from Jaulin, *et al.* (2002):

> "The set $\bar{X}_k$ represents the set of all state vectors at time k such that if $\boldsymbol{x}_k \in \bar{X}_k$ then for all feasible future perturbations, it is possible to find a control such that all future specification are satisfied.
>
> The set $\bar{X}_k$ represents all possible (i.e. for all feasible initial state and for all feasible past perturbations) state vectors that can be reached at time k if the past controls $\boldsymbol{u}_0 = \tilde{\boldsymbol{u}}_0, \ldots, \boldsymbol{u}_k = \tilde{\boldsymbol{u}}_k$. "

The algorithm described in Table 2.1 synthesises robust control sets, if all operations are executable and if there exists a robust solution. The method used finds $\bar{X}_k \subseteq X_k$ from the terminal set down to the set $\bar{X}_2$ in the presence of output disturbances W_k. Then $\bar{X}_k$ sets are found that can be controlled from X_0 to the terminal set passing in the presence of W_k. Note that a $\tilde{\boldsymbol{u}}_k$ value is selected from $\hat{U}_k$ (i.e. the set that contains all the solutions $\boldsymbol{u}_k$ for $\bar{X}_{k-1}$ to pass to $\bar{X}_k$) and this $\boldsymbol{u}_k$ value is used to generate $\bar{X}_k$. In general (but not always), the set operations can only be approximated. See Jaulin, *et al.* (2002) for proof that this algorithm is a solution to the robust control problem.

Chapter 3

u-Surface optimisation

This chapter explains u-surface method. First the model type for which control surfaces can be generated is described in Section 3.1 and then the sets and variables are stipulated which are used throughout the rest of the thesis (Section 3.2). Finally the u-surface synthesis method is described in Section 3.3.

3.1 The model

The type of control system that can potentially be controlled by this method is a sampled state space system of the form:

$$x_n = \mathrm{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{w}) \tag{3.1}$$

where $\boldsymbol{x} = [x_{n-1},\ldots,x_{n-p}]$ are the states, $\boldsymbol{u} = [u_1,\ldots,u_q]$ are the control values, $\boldsymbol{w} = [w_1,\ldots,w_p]$ are the disturbances at different sampling times. The states and control values are bounded (i.e. $x_1 \in [x_1^-, x_1^+]$, …, $x_n \in [x_n^-, x_n^+]$, and $u_1 \in [u_1^-, u_1^+]$,…, $u_n \in [u_n^-, u_n^+]$). Note that x_k can be a vector containing more then one state at a given sampling time.

u-Surface Optimisation control is limited to time invariant systems. Presently fixed setpoint has been implemented, but setpoint tracking is possible but it requires a large amount of computer resources (memory) and so might not be practically feasible on real systems without further research into technical matters.

The software that generates the control surfaces is called "The reflection generator with output disturbance optimisation" (RD) and has been written in Matlab (as described in Chapter 4). It has been written to optimise 2nd order sampled state space systems of the type:

$$\dot{\boldsymbol{x}}_n = \begin{bmatrix} \hat{x}_{2n-1} \\ \mathbf{f}(\hat{\boldsymbol{x}}) + ku_{n-1} \end{bmatrix} \tag{3.2}$$

where $\boldsymbol{x}_n = \begin{bmatrix} x_{1n} \\ x_{2n} \end{bmatrix}$ are the states and $\hat{\boldsymbol{x}}_n = \begin{bmatrix} \hat{x}_{1n} + w_1 \\ \hat{x}_{2n} + w_2 \end{bmatrix}, x_{1n} \in [x_1^-, x_1^+]$ and $x_{2n} \in [x_2^-, x_2^+]$ a forth order Runga-Kutta as described in Zill and Cullen (2000) was used to calculate $\boldsymbol{x}_n$ from $\dot{\boldsymbol{x}}_n$. The output disturbances are w_1 and w_2 and have a known probability distribution, and u_n is the control signal to the system ($u_n \in [u^-, u^+]$). This software was used to generate a u-surface controller for a driven pendulum. The model of the pendulum and results of the simulations are given in Chapter 5 (Simulations).

3.1.1 Disturbance of the system

Most real systems have disturbances affecting the system and to control a system optimally these disturbances need to be considered. Figure 3.1 contains most of the disturbances, noise and parasitics present in a system. In the control method examined, the controller is only designed to be robust with regard to output disturbances (**d**(x)). An example of an output disturbance would be knocking the pendulum of the driven pendulum.

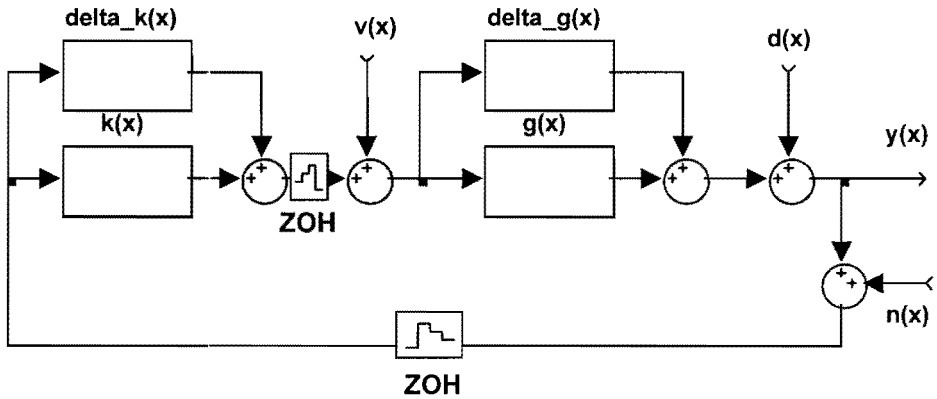


Fig. 3.1 **Model of the system with disturbances, noise and parasitics.** The signal lines from the output of **g**(x) to the input of **k**(x) contains all the states of the system. A State estimator may be required to calculate the states of the system.

If the probability distributions of v(x) and delta_g(x) are known, it is possible to calculate the equivalent output disturbance with stochastic methods (Náprstek, 2000). So a controller design method that only generates a robust controller for a system with output disturbances can be used to design a robust controller for a system with v(x) and delta_g(x) disturbances as well. It is important to note that the equivalent output disturbance distribution will be dependent on the model of the system and current state(s) of the system. The development of this equivalent output disturbance is beyond the scope of this project.

3.2 Sets and variables definitions

The following intervals sets have been adapted from Jaulin L., *et al.* (2002) and are used throughout the rest of the thesis. These sets have been written here to be referred back to while reading the thesis.

1. a set A of n subsets $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_n$ (that are domains of the x states in Equation (3.1). Note that the subscript i is not the same as the subscript of x). The $\boldsymbol{c}_i$ domains are called regions and the i subscript is the value of the region. $\boldsymbol{c}_i$ contains the $\boldsymbol{x}$ states that are an equal number of iterations away from the fixed setpoint.

2. a set B of n subsets $\boldsymbol{b}_{1j}, \ldots, \boldsymbol{b}_{nj}$, where $\boldsymbol{b}_{ij} \in \boldsymbol{c}_i$. The u-surface is approximated by a grid of squares with piecewise flat surfaces. $\boldsymbol{b}_{ij}$ represents a square domain where i indicates what $\boldsymbol{c}_i$ region the domain is in and j is an index to the particular block. The

center value of the $\boldsymbol{b}_{ij}$ block is "center($\boldsymbol{b}_{ij}$)" and the center of a $\boldsymbol{b}$ block closest to x_k is given by "round(x_k)". If $\boldsymbol{b}_{ij}$ value is at the edge the $\boldsymbol{c}_i$ region then it will have corner values (rim values) at the boundary of the region (see Figure 4.1.). These are referred to as "rim($\boldsymbol{b}_{ij}$)".

3. a set T of n subsets V_1 , ...,V_n (the V_i are $\forall V_i$: $\mathbf{f}(V_i, \boldsymbol{c}_i) \in \{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_{i-1}\}$. The V_k subset is a region that contains all the possible control values to control the system from $\boldsymbol{c}_i$ to $\{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_{i-1}\}$ without the effect of disturbances.

4. a set Y of n subsets $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n$. The $\boldsymbol{v}_i$'s are the control surfaces such that $\mathbf{f}(\boldsymbol{v}_i, \boldsymbol{c}_i) \in \{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_{i-1}\}$. Note that $\boldsymbol{v}_i$ is not unique in general and $\boldsymbol{v}_i \subseteq V_i$. The control value or domain for a state or domain z_i respectively is represented by $\boldsymbol{v}_i(z_i)$.

5. a set G of n subsets $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_n$. The $\boldsymbol{r}_i$'s are the fitness surfaces of the regions $\boldsymbol{c}_i$. Note that $\boldsymbol{r}_i$ is not unique in general and depends on the origins of the states of $\boldsymbol{c}_i$. The fitness or fitness surface for a state or domain of z_i are referred by $\boldsymbol{r}_i(z_i)$. The *value* of G or $\boldsymbol{r}_i$ refers to the integral of the surface.

6. a set D of n subsets $C_1, \ldots, C_n$ (called domains) where $C_i = \mathbf{f}^{-1}(\boldsymbol{U}, \boldsymbol{c}_{i-1})$ (C_i can be referred to as all the possible "children" of $\boldsymbol{c}_{i-1}$ see 7. below) and $\boldsymbol{U}$ is the set of all possible control values. Note that $\boldsymbol{c}_i \subseteq C_i$

7. The origin of z_{i+1} is defined as $z_i = \mathbf{f}(z_{i+1}, \boldsymbol{v}_{i+1}(z_{i+1}))$, where z_i can be a state or a domain that z_{i+1} is controlled to when no disturbances are present. Children of z_i are states or domains that have origins in z_i. A domain $\boldsymbol{b}$ is the child of z_i if the center($\boldsymbol{b}$) values of the domain are children of z_i and the origin of rim($\boldsymbol{b}$) is in $\{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_{i-1}\}$.

3.3 Synthesis of u-surface

The u-surface optimisation method is similar to the method used in Jaulin, *et al.* (2002), which is described in Section 2.1.2. The method used by u-surface optimisation is to go backward in time from the terminal state to find sets that can be controlled to the terminal state. These sets are optimised in such a way that the probability of all points inside the sets can be controlled to the terminal state with the highest probability.

A description of the method to generate a u-surface controller is illustrated for a system without (Section 3.3.1) and with disturbances (Section 3.3.1).

3.3.1 Synthesis of u-surface without optimisation

The synthesis algorithm in Table 3.1 generates the u-surface without optimisation.

Table 3.1 **Synthesis of the u-surface without optimisation**

Step	Instruction
1	let $C_1 = x_{setpoint}$ (set the first domain of $\boldsymbol{c}_i$ to the setpoint value)
2	$i = 1$
3	while $x_0 \notin D$
4	$i = i + 1$
5	find C_i
6	choose $\boldsymbol{v}_i$ arbitrarily from $\boldsymbol{V}_i\|_{Ci}$

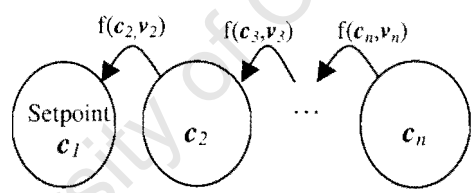


Fig. 3.2 **Methodology of u-surface control**

Figure 3.2 sets out the methodology used in u-surface control.

The control surface generated in Table 3.1. is computationally intensive and requires a large amount of computer memory to implement as the sets C_i become very large. This makes this method difficult to implement even as a computer simulation. However by noticing that there is no advantage in keeping $C_i \cap \{C_{i-1},...,C_1\}$ in C_i it can be seen that a more computationally feasible algorithm is demonstrated in Table 3.2.

Table 3.2 **Feasible computation method to generate u-surface without optimisation**

Step	Instruction
1	let $\boldsymbol{c}_1 = x_{setpoint}$ (set the first domain of $\boldsymbol{c}_i$ to the setpoint value)
2	$i = 1$
3	while $\boldsymbol{c}_i$ ~= null
4	$i = i + 1$
5	find C_i
6	let $c_i = C_i - C_i \cap \{\boldsymbol{c}_{i-1}, \boldsymbol{c}_{i-2}, ..., \boldsymbol{c}_1\}$
7	choose $\boldsymbol{v}_i$ arbitrarily from $\boldsymbol{V}_i\|_{ci}$

The algorithm in Table 3.2 has been implemented in code, and creates a controller that is near time optimal control, but is not robust with respect to output disturbances. The reason the controller is not time optimal is largely due to the surface used to save $\boldsymbol{c}_i$ and $\boldsymbol{v}_i$ being discrete (see Section 4.1 and 4.2).

3.3.2 Synthesis of u-surface with optimisation

The u-surface can be optimised by selecting the origin of every state that increases the fitness function the most. The idea that a state can have a number of origins is illustrated in the Fig. 3.3.

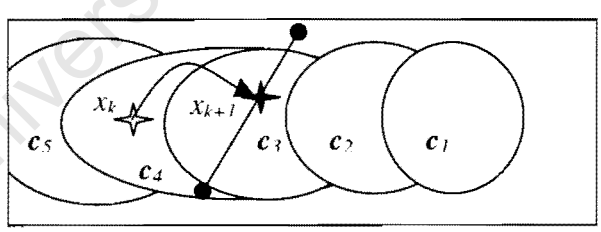


Fig. 3.3 **Selection optimisation technique**. x_k is a state in $\boldsymbol{c}_4$ that can be controlled by f($\boldsymbol{U}$,x_k) to a number of states some of which are in $\boldsymbol{c}_3$ (call the states that fall in $\boldsymbol{c}_3$, $\boldsymbol{X}_{k+1}$). The origin $x_{k+1} \in \boldsymbol{X}_{k+1}$ is selected as the state that increases the fitness function the most.

The other method used to maximize the fitness function is to select the edge of $\boldsymbol{c}_i$ bordering on $\boldsymbol{c}_{i-1}$ in a way that increases the fitness function the most. This is done by expanding the $\boldsymbol{c}_i$ region by including $\boldsymbol{b}_{4i}$ if this expansion increases the fitness function. This technique is illustrated in Figure 3.4.

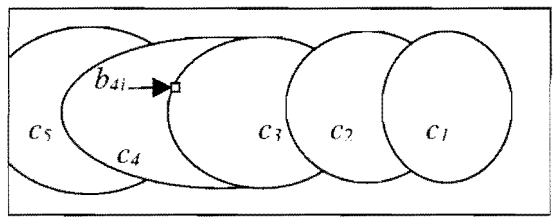


Fig. 3.4 **Optimal edge formation technique.** An optimising strategy is to "replace" $\boldsymbol{b}_{3j}$ by $\boldsymbol{b}_{4i}$ if this results in the fitness function increasing.

When $\boldsymbol{b}_{4i}$ replaces $\boldsymbol{b}_{3j}$ the point is not lost from the $\boldsymbol{c}_3$ set; that is if the system is in a state in $\boldsymbol{c}_3$ and the state is perturbed to $\boldsymbol{b}_{3j}$ then $\boldsymbol{v}_3(\boldsymbol{b}_{3j})$ will control the system to $\boldsymbol{c}_2$. The overlapping ensures that the fitness of the states in the lower region is not decreased as the boundary of the higher region expands.

Unfortunately, to reduce the computational intensity limited overlapping of sets is used (see Section 4.1 for further details). However a method that involves overlapping surfaces and trimming of the overlapped regions has been envisaged and has been described in Future Development (Chapter 7). Due to time constraints it was not possible to implement this method.

3.3.3 Robustness fitness function of the u-surface

Robust control described in Jaulin L., *et al.* (2002) controls an initial state to the setpoint by being controlled to pass though monotonically lower $\boldsymbol{c}_i$ regions with constraints that guarantee robustness. It is not possible to control the driven pendulum in such a method to the setpoint, due to the fact that the setpoint is a single state, and that once the system settles to the setpoint any output disturbance will cause the current state of the system to be moved to a higher $\boldsymbol{c}_i$ region, and so cause the system not to be robust. For this reason the control method described in Jaulin L., *et*

al. (2002) could not be applied to the driven pendulum being driven to a setpoint under the presence of a disturbance.

The measure of robustness for a state is the probability that a state under the influence of disturbances can be controlled to the setpoint. The measure of robustness of a system is the sum of all the above-mentioned probabilities for all the states of the u-surface of the system. The sum of all the probabilities is used as the fitness function of the u-surface optimisation method - hence the robustness of the system is maximised.

So the probability that a $\boldsymbol{b}_{ik}$ will be controlled to the setpoint with output disturbances of a known probability has been estimated and is used as the fitness function to calculate the optimal surface. An example of a probability surface is illustrated in Fig. 3.5.

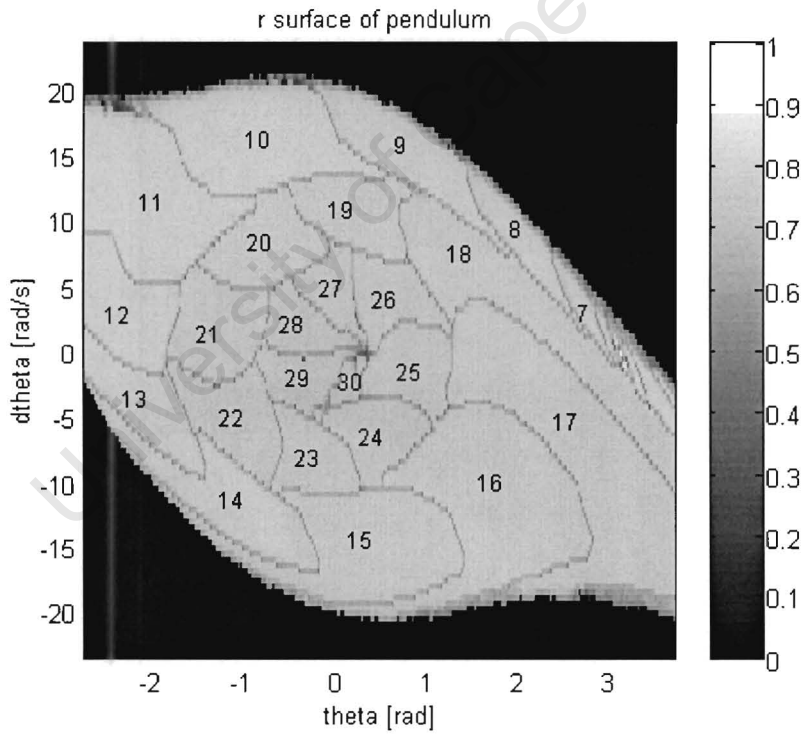


Fig. 3.5 **Probability surface of regions for a simulated pendulum** (Sys.20)

Figure 3.5 is a probability surface of the different regions for the system Sys.20 that is defined in Appendix A. The setpoint [0,0] is when the pendulum is inverted (standing

upright). The regions spiral down toward the initial value (that is when the pendulum is at rest at the bottom). The setpoint's probability is set to 1 (it is the small white dot). The regions that are black cannot be controlled to the setpoint. There are lines of lower probability that separate the regions, these lines would largely be removed if overlapping regions as described in Section 3.2 were used. But in this example the effect of not using overlapping regions appears to be limited to the boundary regions, since the probability values inside the regions do not decrease significantly as the regions become higher.

Chapter 4

The program

The Reflection generator with disturbance optimisation (RD) is the u-surface synthesis software that has been designed to control sampled second order SISO systems as described in (3.2). The aim of RD has been to generate the u-surface controller for the driven pendulum (see Section 5.1.2 for model of pendulum).

Table 4.1 gives a basic overview of the RD program. Note that G is a fitness surface created for $\{\boldsymbol{c}_1,....\boldsymbol{c}_i\}$ and that $\boldsymbol{b}_k \subseteq \{\boldsymbol{c}_{i-1},\ldots,\boldsymbol{c}_1\}$. The column to the right of Table 4.1 has been include so the reader can investigate the implementation of the algorithms in code. A CD that contains the software is at the back of the thesis book.

Certain functions in the program have been designed for working effectively for the pendulum model and may not work effectively for other systems (examples of such functions are the function that generates C_i sets (`cal_cpnts`) and the function that estimates the fitness function when the surface is modified. (`eval_cost_fn`)).

Table 4.1 **Overview of RD**

Step	Overview	Main unctions
1	Initialisation	`set_initsurfr_n`,
2	let $i = 1$	`cal_Darray`
3	evaluate rim values and $\boldsymbol{r}_1$ values for region 1 and $\boldsymbol{c}_{1=}$ setpoints	`cal_Nrim`, `cal_R`
4	**while** $\boldsymbol{c}_i$ is not empty	
5	$i = i + 1$	
6	calculate next C_i and V_i	`cal_cpnts`
7	set let $\boldsymbol{c}_i = C_i - C_i \cap \{\boldsymbol{c}_{i-1}, \boldsymbol{c}_{i-2}, ..., \boldsymbol{c}_1\}$	
8	optimise surface	`cal_ir_del`
9	initialise fitness function	
10	create array of **center**($\boldsymbol{b}_k$) to be examined	`cal_RaveOptA_init`,`cal_R`
11	**while** array to be examined not empty	
12	calculate fitness function with $\boldsymbol{b}_{kj}$ replaced and create updated G	`cal_ChgPnts`, `Mod_xRo_xR`,
13	if fitness increases without $\boldsymbol{b}_{kj}$	`eval_cost_fn`
14	use updated G	
15	**loop**	
16	calculate $\boldsymbol{b}_{i-1}$ without children	`cal_Imps`
17	**loop**	
18	include any domains that can converge to the setpoint but have not been included and optimise these domains	`add_YBflgz` `cal_YBi_Rmax`
19	determine all optimum $\boldsymbol{v}_i$ surfaces from $\boldsymbol{c}_i$	

The program generates an optimised control surface. This surface is implemented as a look up table for controlling the system.

In Section 4.1 the interpolation surface used to store the control surface is described and then some of the important functions that effect the final optimal solution are described in Sections 4.2-4.6. Chapter 4 makes extensive use of the Definitions in Section 3.2.

4.1 Description of interpolation surface

The u-surface contains control values for every controllable state of the system. A grid was created on which the control values for certain states are stored (i.e.

v_i(**center**(b_i)) and v_i(**rim**(b_i))). From this grid every state's control ($v_i(x_k)$) value is interpolated. This grid method was also used to interpolate the probability of a state being controlled to the setpoint (i.e. $r_i(x_k)$).

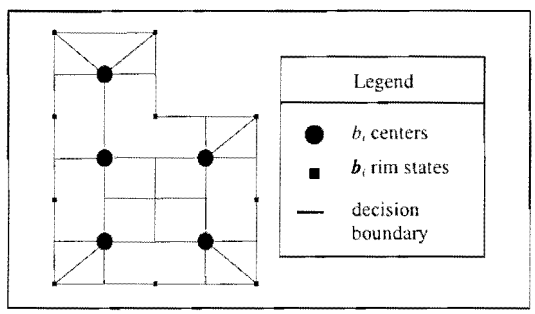


Fig. 4.1 **Grid used for interpolation for region *i***

Figure 4.1 is the grid that was used for interpolating control and probability values. Every region is made up of b_i blocks about the b_i centers and the b_i rim values. The **rim**(b_i) states lie on the boundary of the c_i region and other regions overlap with rim states from other regions.

The $v_i(x_k)$ value of a state that is in b_{ij} is interpolated from the surface that passes through the values of the closest three neighbouring v_i(**center**(b_{ij})) or if there are only one or two v_i(**center**(b_{ij})) the v_i(**rim**(b_{ij})) are used to make up the difference (call these **center**(b_{ij}) state(s) and any **rim**(b_{ij}) state(s) *construction points*).

In Figure 4.1 the layout of the **center**(b_{ij}) and **rim**(b_{ij}) states are illustrated. The areas between the decision boundaries are surfaces that are generated from the interpolation on the surface passing though the same three closest *construction points*.

A number of other interpolation methods were investigated. The u-surface was first generated using a grid were the control value v_i(**center**(b_i)) was given to any point in b_i. This technique was not adequate, as the surface could not control the pendulum to the setpoint even without disturbances, due to the unstable nature of the inverted pendulum about the setpoint.

It was also found that rim points were necessary to reduce the amount that states near the edge of $\boldsymbol{c}_i$ would not be controlled to states in $\{\boldsymbol{c}_{i-1},\ldots,\boldsymbol{c}_1\}$. The rim states improved the interpolation at the edge of $\boldsymbol{c}_i$ and this reduced the amount of states not being controlled to $\{\boldsymbol{c}_{i-1},\ldots,\boldsymbol{c}_1\}$.

Both $\boldsymbol{v}_i(\mathbf{center}(\boldsymbol{b}_i))$ and $\boldsymbol{v}_i(\mathbf{rim}(\boldsymbol{b}_i))$ control values control their respective states to the less than i regions. However the cause value $\boldsymbol{v}_i(x_k)$ generated by interpolating between construction points that are far away can control the state x_k to be controlled to a region which is not less than i. Figure 4.2 illustrates an example when this can occur.

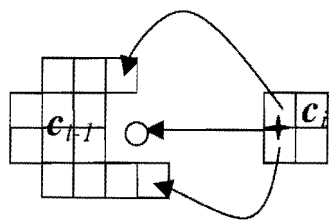


Fig. 4.2 **State between $\boldsymbol{b}_i$ centers controlled to state not in a region less than i.** The two $\boldsymbol{b}_i$ (above and below the star) are controlled to their origins in $\boldsymbol{c}_{i-1}$ but the state that falls between them is not controlled to a region in $\boldsymbol{c}_{i-1}$

However for the driven pendulum the number of times the system is not controlled to a region less than i is negligible (less than 0.14% with 5% variation of the mean and 95% confidence determined by 800000 random samples of the u-surface for a driven pendulum type 1. It was found that these errors occurred most at regions near edges of the surface (see Appendix E).

Due to the general convex shape of the regions and because neighbouring states tend to originate from points in close proximity to each other (for two states separated by 0.33 the origins of two states are separated by 0.14 on average with less then 5% variation with 95% confidence determined by 10000 random samples of the u-surface (the same system is used in the previous paragraph). Hence the number of x_k from $\boldsymbol{c}_i$ that will be controlled to a state a region less the than i is small.

4.2 Initialisation and the effect of program variables

The controller is initialised by defining the model of the driven pendulum, the range of the control values and the range of the states of the system, the magnitude and type of probability distribution of the output disturbance and the size of the grid used to construct the u-surface.

4.2.1 Grid size

The size of the grid has to be chosen correctly to ensure that the setpoint ($\boldsymbol{b}_1$) will be stable (i.e. all states in $\boldsymbol{b}_1$ should be controlled to $\boldsymbol{b}_1$), because if $\boldsymbol{b}_1$ is unstable then $\boldsymbol{c}_1$ is empty and RD stops. A function `cal_max_block` is used to calculate the maximum dimension of the grid for which the setpoint $\boldsymbol{b}_1$ is stable.

The grid size also affects the formation of the $\boldsymbol{c}_i$ regions; because large $\boldsymbol{b}_i$ do not pack in as well into the possible $\boldsymbol{c}_i$ region as smaller blocks (see Figure 4.3). Unfortunately the smaller the size of the blocks the longer the time taken to calculate each surface as seen in Fig. 4.3 (refer to Section 5.4.1).

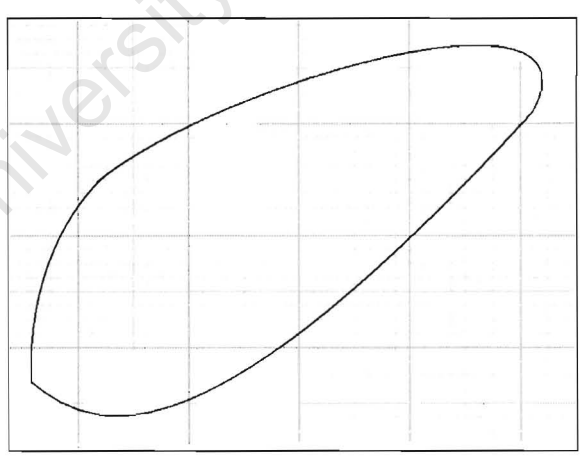


Fig. 4.3 **Packing of region with varying block sizes**

4.2.2 Calculation of disturbance probability of $\boldsymbol{b}$

All the states in a domain $\boldsymbol{b}_{ij}$ are perturbed by an output disturbance $w \in W$ with the probability distribution $\mathbf{P}_W(w)$. The PDF of a uniformly randomly chosen state starting in $\boldsymbol{b}_{ij}$ after it has been perturbed with $\mathbf{P}_W(w)$ can be calculated by

$$\mathbf{P}_Z(x) = \iiint \mathbf{P}_{W,b_{ij}}(w, x_{b_{ij}}) dx_1 dx_2 dw \tag{4.1}$$

where the limits of the integral are over x_1 and x_2 to z_1-x_1 and z_2-x_2 respectively and all w. $x_{b_{ij}}$ is a state with uniform distribution in $\boldsymbol{b}_{ij}$ and Z = W + $\boldsymbol{b}_{ij}$).

The PDF of a uniform randomly chosen state starting in $\boldsymbol{b}_{ij}$ (i.e. $x_{b_{ij}}$) being perturbed into a domain neighbouring $\boldsymbol{b}_{ij}$ (call these $\boldsymbol{b}_{ij_{neigh}}$ where a neighbour is any $\boldsymbol{b}$ domain of $\boldsymbol{b}_{ij}$ th can be perturbed to) can be calculated by:

$$\mathbf{P}_Z(\boldsymbol{b}_{ij_{neighk}}) = \iint \mathbf{P}_Z(x) dx_1 dx_2 \tag{4.2}$$

where the area over which the integration occurs is $\boldsymbol{b}_{ij_{neighk}}$

Figure 4.4 is an example of the probability distribution $\mathbf{P}_Z(\boldsymbol{b}_{ij_{neigh}})$ of a number of $\boldsymbol{b}_{ij_{neighk}}$, where $\mathbf{P}_W(w)$ is gaussian, this is the same type of $\mathbf{P}_Z(\boldsymbol{b}_{ij_{neigh}})$ that is used for calculating the fitness function.

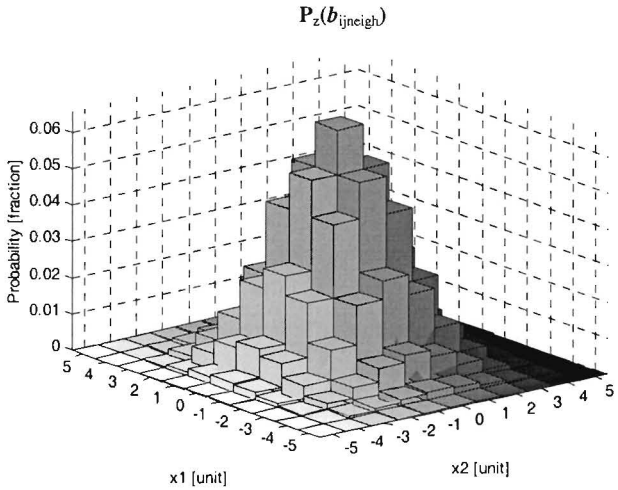


Fig. 4.4 **Probability distribution function** $\mathbf{P}_Z(\boldsymbol{b}_{ij_{neigh}})$

For the pdf $\mathbf{P}_Z(\boldsymbol{b}_{ij_{neigh}})$ illustrated in Figure 4.4 the x_1 axis is number of blocks from $\boldsymbol{b}_{ij}$ along the x_1 dimension axis and the x_2 axis is number of blocks from $\boldsymbol{b}$ along the x_2. the negative and positive signs in the x_1 and x_2 axis correspond to left and right of $\boldsymbol{b}_{ij}$ respectively. The probability axis is probability of a state originally in the centre $\boldsymbol{b}$ domain (i.e. $\boldsymbol{b}_{ij}$ which is at (0,0) in the figure) being perturbed into a $\boldsymbol{b}$ domain by a disturbance $\boldsymbol{w}$.

The calculation of all $\mathbf{P}_Z(\boldsymbol{b}_{ij_{neigh}})$ for values above a threshold is performed during the initialization phase of the program.

4.3 Calculation of possible points in the *i* region

The surface expands outward from the setpoint through the process of finding the C_i region that has its origins in $\boldsymbol{c}_{i-1}$. The function `cal_cpnts` is used to calculate C_i and the functions algorithm is outlined in the algorithm in Table 4.2.

Table 4.2 **Calculation of C_i algorithm**

Step	Instruction
1	calculate seed values (i.e. **center($\boldsymbol{b}$)** values)
2	**while** unchecked seed values
3	calculate origins of a seed value
4	**if** any origins from less then i then
5	add to C_i
6	add neighbours (refer to text for which neighbours can be added)
7	**endif**
8	**loop**
9	include **center($\boldsymbol{b}_l$)** who's rim values now have origins (refer to text for explanation)
	calculate rim values
10	**while** still unchecked rim values
11	calculate origins of rim value
12	**if** any origins are not less then i then
13	remove **center($\boldsymbol{b}$)** using removed rim value
14	add rim values to new edge $\boldsymbol{b}$ domains
15	**endif**
16	**loop**

The algorithm in Table 4.2 calculates the C_i region by finding the **center($\boldsymbol{b}_{seed}$)** that go to less than i region and then the checking that the rim values go to the less than i region.

The seed states are the **center($\boldsymbol{b}_{seed}$)** states that are expected to originate in C_i (Step 1 in Table 4.2). These states are calculated by finding the extreme $\boldsymbol{b}_{i-1}$ domains of $\boldsymbol{c}_{i-1}$ (domains with highest and lowest x_1 and x_2 states) and the $\boldsymbol{b}_{i-1j}$ that have only one or no neighbouring $\boldsymbol{b}_{i-1j}$ and finding the children of these domains by going backwards in time. All the $\boldsymbol{b}_j$ domains that do not have children ($\boldsymbol{b}_{lmps}$) are also added to the seed values. The reason why $\boldsymbol{b}_{lmps}$ are included in the seeds is because these domains can be replaced by states in lower regions while only $\boldsymbol{b}_{i-1}$ values that have children can be replaced (see Section 4.4.2).

The **center($\boldsymbol{b}_{seed}$)** origins are found by determining all origins for these states and checking that they come from a $\{\boldsymbol{c}_1,\ldots,\boldsymbol{c}_{i-1}\}$. If they are from this region their neighbours centers are added to the **center($\boldsymbol{b}_{seed}$)** states. (the eight surrounding $\boldsymbol{b}$ domains surrounding the $\boldsymbol{b}_{seed}$); except for the neighbours that are or have been a seed state and except for neighbours that are part of $\{\boldsymbol{c}_1,\ldots,\boldsymbol{c}_{i-2}\}$ (Step 6 from Table 4.2).

So the **center(b_{seed})** that originate from $\{c_1,\ldots,c_{i-1}\}$ causes the C_i region to be found by spreading out from these seed values. For the driven pendulum, C_i tends to consist of one or two continuous regions so this method is efficient for calculating C_i.

When running the algorithm in Table 4.2 for regions 1 to i-1, there could have been a number of **center(b_{seed_k})** states having rim states that did not originate from the lower regions and so were rejected from C_k. These rim states are added into an array (the rim checking is performed in Steps 10-16 of Table 4.2) The b_{seed_k} domains are examined to determine if they now have origins in $\{c_1,\ldots,c_{i-1}\}$. Those that do are added to the **center(b)** (where $b \in C_i$). This method results in b domains to be included to i regions with low values (i.e. 2-5) that would otherwise not have been found. This is beneficial as these low regions tend to be small. Consequently the adding of extra domains makes a significant increase to the fitness of these regions.

The final task of the algorithm is to check that all the **center(b)** rim states originate in $\{c_1,\ldots,c_{i-1}\}$. This is performed by calculating the rim values and checking that they have origins in $\{c_1,\ldots,c_{i-1}\}$.

4.4 Optimising the u-surface

The possible states that pass to the previous region have been calculated (i.e. C_i). From these states an optimal surface is chosen according to a robust fitness function.

4.4.1 Calculate robustness probability values for points

The output disturbance can move the states in b_{ij} to a number of different b domains ($b_{ij_{neigh}}$) (refer to Section 4.2.2). The probability of the states in b_{ij} being perturbed to $b_{ij_{neigh_k}}$ is $\mathbf{P}_Z(b_{ij_{neigh}})$ (refer to Section 4.2.2).

Estimation of r_i

If $\boldsymbol{b}_{ij_{neighk}}$ is in $\{\boldsymbol{c}_1,\ldots,\boldsymbol{c}_i\}$, then the maximum probability of the origin of $\mathbf{center}(\boldsymbol{b}_{ij_{neighk}})$ being controlled to the setpoint is $\boldsymbol{r}_j(\mathbf{f}^{-1}(\mathbf{center}(\boldsymbol{b}_{ij_{neighk}})))$ (where $j<i$). The $\boldsymbol{r}_j$ probabilities have already being calculated. If $\boldsymbol{b}_{ij_{neighk}}$ is not in $\{\boldsymbol{c}_1,\ldots,\boldsymbol{c}_i\}$ then this domain does not contribute to the fitness of $\boldsymbol{b}_{ij}$

If $\boldsymbol{b}_{ij_{neigh}}$ falls into a region not in $\{\boldsymbol{c}_1,\ldots,\boldsymbol{c}_i\}$ better results for calculating the fitness of $\boldsymbol{b}_{ij}$ could possibly be attained by using $\boldsymbol{r}_j(\mathbf{f}^{-1}(\mathbf{center}(\boldsymbol{b}_{ij_{neighk}})))$. This method was not implemented as it could result in cyclic referencing (i.e. $\boldsymbol{r}_j(\mathbf{f}^{-1}(\mathbf{center}(\boldsymbol{b}_{ij_{neighk}})))$ which would be dependent on $\boldsymbol{r}_j(\mathbf{center}(\boldsymbol{b}_{ij_{neighk}}))$ and visa versa). This would result in the calculation of G being difficult and probably far more time consuming.

Then the probability of $\mathbf{center}(\boldsymbol{b}_{ij})$ reaching the setpoint is estimated as:

$$\boldsymbol{r}_i(\mathbf{center}(\boldsymbol{b}_{ij})) \approx \sum_k \mathbf{P}_z(\boldsymbol{b}_{ij_{neighk}})\boldsymbol{r}_j(\mathbf{f}^{-1}(\mathbf{center}(\boldsymbol{b}_{ij_{neighk}}))) \tag{4.3}$$

where $j<i$

Figure 4.5 helps clarify the process above used to estimate the probability of the states in $\boldsymbol{b}_{ij}$ reaching the setpoint.

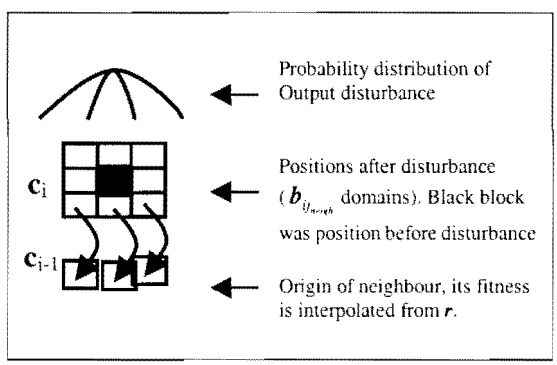


Fig. 4.5 **Methodology of robustness probability calculation for $\boldsymbol{b}_{ji}$**

Ideal r_i calculation

The method for calculating the $r_j(\mathbf{center}(b_{ij}))$ probability of a point is implemented in the script `cal_R` and is used to calculate the robustness surface G (refer to Section 3.2). This method is an approximation, the ideal method would be:

$$r_i(b_{ij}) = \iint \mathbf{P}_z(x) r_j(\mathbf{f}^{-1}(x) dx_1 dx_2) \tag{4.4}$$

where the limits of the integral are over all x_1 and x_2).

However this calculation is computationally intensive. The calculation of $\mathbf{P}_Z(b_{ij_{neigh}})$ to with an accuracy of 0.1% for $\mathbf{P}_w(x)$ having a bivariate gausian distribution with variance of 1 and mean 0 $\mathbf{P}_Z(b_{ij_{neigh}})$ takes 5.3 s. This is a less computationally intensive calculation and is only calculated once compared to the ideal method that would be calculated a number of times for every b domain. Thus the ideal $r_i(b_{ij})$ would make RD computationally unfeasible.

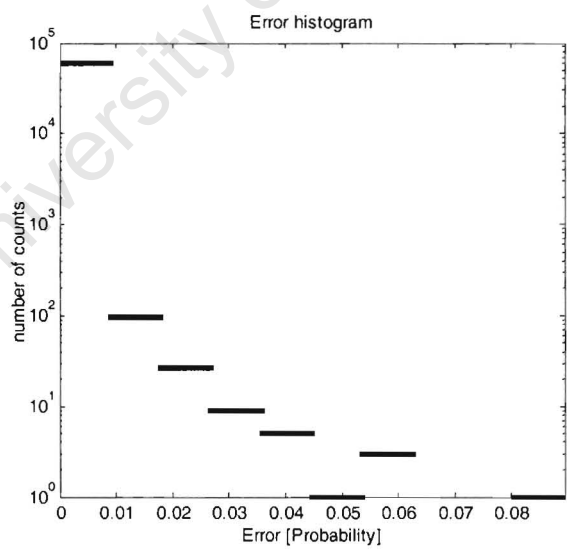


Fig. 4.6 **Error histogram for G surface** (Sys.20)

Comparison using Monte Carlo method

A Monte Carlo method was used to calculate the G surface after the u-surface has been calculated. Figure 4.6 is the histogram of the error between the G surface calculated by the RD program, and the same G surface calculated using a Monte Carlo method. The Monte Carlo method took a number of perturbed states (x_k) originally from $\boldsymbol{b}_{ij}$ (perturbed with the output disturbance) and calculates the $\boldsymbol{r}_i(\mathbf{f}^{-1}(x_k))$ fitness for these states and sums these together to get an approximation for $\boldsymbol{r}_i.(\boldsymbol{b}_{ij})$ The Inverse of the Student's T cumulative distribution function is used to determine that a large enough number of states was used to calculate the mean of $\boldsymbol{r}_i.(\boldsymbol{b}_{ij})$ with a variance of 5% on the mean and with a confidence of 95% (Hickman ,1971). The Monte Carlo method was computationally intensive and took 6.3 hrs to calculate G compared to 54s using the estimation method.

The error histogram shows that the accuracy to which RD calculates the G surface is high. The mean of the G surface is 0.4722 and less than 0.2% of the G probabilities have an error that is larger than 2% of the average. The largest errors were found mostly on the boundary regions between regions and the children of these boundary regions (see Appendix E). This difference could be explained by states near the edge of a region being more likely to not be controlled to states in higher regions without disturbances (see Section 4.1). Note that the approximation of the surface is dependent on the structure of the G surface. But in (Appendix E) there is an example where the error due to mapping is significance.

4.4.2 Choice of $\boldsymbol{b}$ domains to be replaced

The only $\boldsymbol{b}$ domains that can be replaced by $\boldsymbol{b}_i$ are those that are in the $\boldsymbol{c}_{i-1}$ region or the domains that have no children. This is because replacing domains with children

could result in removing in the children and the children's children and so on, which requires a lot of calculation.

The $\boldsymbol{b}_{i-1}$ domains can also be removed and not replaced.

The fitness of the surface is calculated with the $\boldsymbol{b}_{i-1}$ that are along the edge of the $\boldsymbol{c}_{i-1}$ being removed or replaced (see Table 4.1 Step 12). Only the edge domains are considered as the $\boldsymbol{b}_{i-1}$ away from the edge have higher $\boldsymbol{r}_i(\boldsymbol{b}_{i-1})$ fitness values, for the case of the pendulum. This is due to the general concave shape of the regions. When $\boldsymbol{b}_{i-1}$ points are removed, their children in the $\boldsymbol{c}_i$ region that no longer have origins are removed as well.

4.4.3 Estimation of *G* surface

The aim of the optimisation is to maximise the *G* surface. The ideal way to do this (without using overlapping regions, see Section 3.3.2) would be to calculate all the $\boldsymbol{c}_i$ surfaces for the entire surface and the fitness surface *G* without optimisation. Then replace $\boldsymbol{b}$ domains, starting in the lowest region and working to highest. In this method the exact amount of *G* can be calculated to see if replacing a $\boldsymbol{b}$ region increases the fitness. But this method is not computationally feasible even for low order systems. Therefore an optimisation method has been used.

Over ten methods have been used to estimate *G* to calculate if the entire region has been increased or not due to the replacement of a $\boldsymbol{b}$ domain. All the more successful methods optimise the $\boldsymbol{c}$ regions from 1 to *i* by replacing $\boldsymbol{b}_{i-1}$ and $\boldsymbol{b}_{\text{Imps}}$ domains and estimate the fitness $\boldsymbol{r}_i(\boldsymbol{c}_i)$ and uses this to estimate the change in the fitness of *G* due to a modification to $\{\boldsymbol{c}_1,\ldots,\boldsymbol{c}_{i-1}\}$. Other methods that did not use an *i* region (mentioned above) have the tendency of decreasing the size of $\boldsymbol{c}_{i-1}$ to a degree where the children of $\boldsymbol{c}_{i-1}$ have a much lower fitness than the $\boldsymbol{c}_{i-1}$ region.

Estimate of fitness of *i* region

The estimate of the i region is performed by removing the boundary values of $\boldsymbol{b}_i$ that increase the mean value of $\boldsymbol{c}_i$ until the optimum is reached or until the number of $\boldsymbol{b}_i$ equal the number of $\boldsymbol{b}_{i-1}$ domains. Limiting the number of $\boldsymbol{b}_i$ was necessary to avoid the $\boldsymbol{b}_i$ region from getting too small.

The time needed to estimate the fitness of the $\boldsymbol{c}_i$ region was reduced by estimating the optimal $\boldsymbol{c}_i$ and then only removing $\boldsymbol{b}_i$ from these optimal $\boldsymbol{c}_i$ if it would have been removed due to a $\boldsymbol{b}_{i-1}$ domain being replaced, and optimising the $\boldsymbol{c}_i$ region from this stage.

An example comparing the actual $\boldsymbol{r}_i$ value with the estimated $\boldsymbol{r}_i$ ($\boldsymbol{r}_{iest}$) value is illustrated in Figure 4.7. In the Figure the mean error of the estimated value was 0.016 (4.3% error on mean) and the maximum error was 0.070. This estimate was considered to be accurate enough for the purpose of estimating the value of $\boldsymbol{r}_i$. The method for estimating the $\boldsymbol{r}_i$ value does not take into account the effect that optimising $\boldsymbol{r}_i$ will have on $\boldsymbol{r}_{i+1}$ and so is often too greedy and so exceed the actual $\boldsymbol{r}_i$ values.

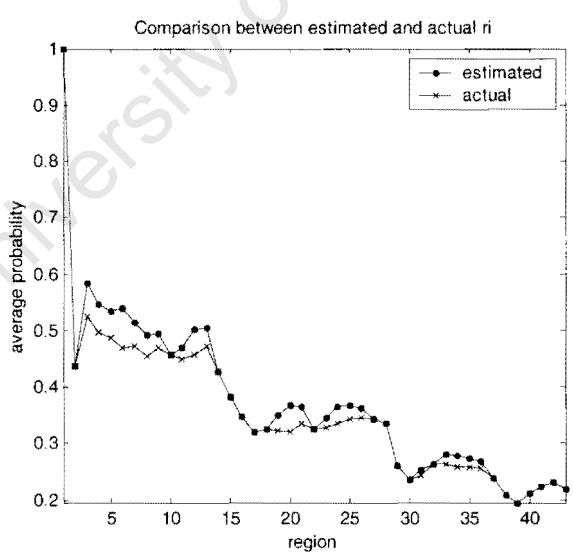


Fig. 4.7 **Comparison of estimated and actual r_i**

Methods to estimate the *G* surface

The first successful method implemented (`Rsum_est`), estimated the entire *G* surface by assuming that the regions that have not been calculated (i.e. regions larger than *i*) had a mean equal to the weighted mean of $\boldsymbol{c}_{i-1}$ and $\boldsymbol{c}_{iest}$. (see Table 4.3 for a list of methods to estimating the change in *G* surface). This method can generally generate satisfactory u-Surfaces for the pendulum with a gaussian output disturbance with a variance less than half the grid size and without state constrains. An example in which a *G* surface was optimised using `Rsum_est` is plotted in Figure 3.5. In this example the average of the fitness of the $\boldsymbol{c}$ surface is well approximated with a constant, for regions larger than 5 see Figure 4.8 (the model used was Sys.20). For many systems this method does not work because the weighted mean of c_{i-1} (see Figure 4.8 model 2) does not approximate a constant.

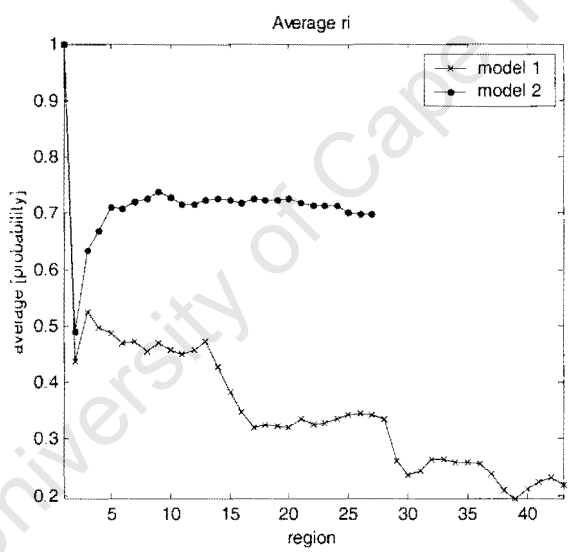


Fig. 4.8 **Comparison of mean value of c region for model 1 and model 2**

Another method (`Rsum_repl`) was devised to take into account the change in mean of the following sections. This was done by noticing that

$$\mathbf{mean}(\boldsymbol{r}_i) \approx \mathbf{mean}(\boldsymbol{r}_{i-1}) \times \frac{\mathbf{mean}(\mathbf{f}^{-1}(\boldsymbol{c}_{i-1}))}{\mathbf{mean}(\mathbf{f}^{-1}(\boldsymbol{c}_{i-2}))} \tag{4.5}$$

Based on equation 4.5 the estimate for the mean of r_{i+1} was estimated as

$$\mathbf{mean}(r_{i+1}) \approx \frac{(\mathbf{mean}(r_{iest}) + \mathbf{mean}(r_{i-1}))}{2} \times \frac{\mathbf{mean}(\mathbf{f}^{-1}(c_{i-1}))}{\mathbf{mean}(\mathbf{f}^{-1}(c_{i-2}))} \tag{4.6}$$

A number of other approximations were used to estimate this mean, most using the more logical $\mathbf{mean}(\mathbf{f}^{-1}(c_{esti}))/\mathbf{mean}(\mathbf{f}^{-1}(c_{i-1}))$ but none of these performed well compared to this approximation because the error between $\mathbf{mean}(\mathbf{f}^{-1}(c_{esti}))$ and $\mathbf{mean}(\mathbf{f}^{-1}(c_i))$ was too large. The Figure 4.8 below compares the actual and estimated $\mathbf{mean}(r_{i+1})$ calculated by using (4.6).

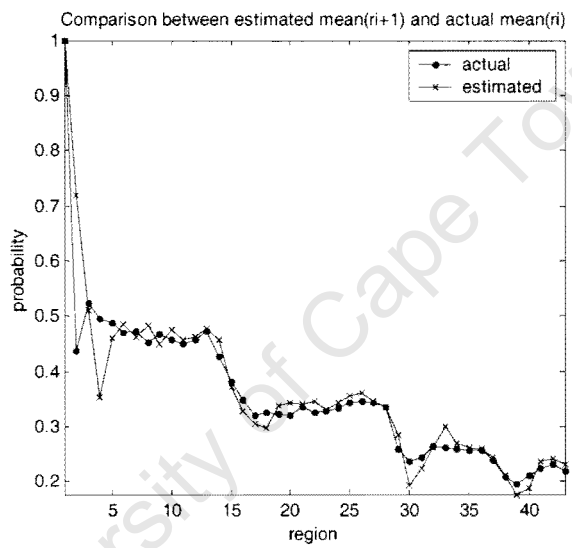


Fig. 4.9 **Comparison between estimated and actual mean(r_{i+1})**

In Figure 4.9 the actual and estimated values of r_{i+1} are compared. The mean of the absolute error was 0.023 (6 % error on the mean of actual r_{i+1}) and the maximum absolute error was 0.28 that occurred at the start. The estimation was considered sufficiently accurate.

It was no longer possible to calculate a reasonable estimate the G value (as done in `Rsum_repl`) so a new method was used. This method works by comparing the fitness of the region before replacing or removing $\boldsymbol{b}_{replace}$ and the fitness of the same area afterwards.

The method used equation (4.6) to estimate the b_{i+1} domains that would replace the removed domains An equation that estimated the value of the domains that were removed and not replaced was also used; this equation is in Table 4.3 (It is a poor estimator as it is difficult to estimate the r_i value of b in regions many regions away from the current the region).

There were two situations that still caused the u-surface not to be generated satisfactorily. One of these situations was when a constraint on the system would cause the c_i regions to become very small and result in the children c regions to have very low fitness values. The solution implemented for this was to use the `Rsum_repl_tot` estimation method but to reduce the estimated replacement values of the b domains when the size of the c_{i-1} region was smaller than the c_{i-2} region. By reducing the estimated replacement values of the removed b domains the optimisation would be less prone to making the c_i region too small. This method was called `Rsum_repl_tot` (see Appendix B for example were `Rsum_repl_tot` was required to perform optimisation).

The other situation where a surface was not well optimised occurred when the c_i region with low **mean**(r_i) was neighbouring on a much lower region with much higher **mean**(r_j). When this happens some of the b_i neighbouring the c_j would attain much higher $r_i(b_n)$ fitness than the rest of the b_i, as disturbances would cause the states in these b_i to be perturbed into c_j. However in the rest of $r_i(b_i)$ values are very low so this causes the c_i region to be optimised into a very small region around these points, and so causes the rest of the surface not to be generated satisfactorily. The method to alleviate the problem (`Rsum_repl_filt`) was to find the b_i that had $r_i(b_n)$ values that were much higher than the rest of the $r_i(b_n)$. For a system in which this method was implemented, histograms of $r_i(b_n)$ were plotted and the most extreme $r_i(b_n)$ values were excluded if they were more then 5 times the average of $r_i(b_n)$ without filtering out the extreme $r_i(b_n)$. The extreme $r_i(b_n)$ values were added back into the c_i but their fitness was set to the average of $r_i(b_n)$ (see Appendix B for example were `Rsum_repl_filt` was required to perform optimisation).

Table 4.3 **Methods for estimating the *G* surface.** Note that *dif* is the expected change in the overall fitness function.

Method	Fitness function	Comment
`Rsum_est`	$a = (\mathbf{mean}(\boldsymbol{r}_{i-1}) + \mathbf{mean}(\boldsymbol{r}_{iest}))/2$ m is the number of elements in the regions higher then $\boldsymbol{r}_{est} = m \times a + \sum \boldsymbol{r}_{<i} + \sum \boldsymbol{r}_{iest}$ $dif = \boldsymbol{r}_{est} - \boldsymbol{r}_{est\_old}$	for system that has mean($\boldsymbol{r}_i$) that is approximately constant $\boldsymbol{r}_{est\_old}$ is $\boldsymbol{r}_{est}$ calculated before the modification to the surface
`Rsum_repl`	$a = (\mathbf{mean}(\boldsymbol{r}_{i-1}) + \mathbf{mean}(\boldsymbol{r}_{iest}))/2$ m is the number of elements in the regions higher then $\boldsymbol{r}_{est} = \sum \boldsymbol{r}_{<i} + \sum \boldsymbol{r}_{iest}$ $Sd = \mathbf{mean}(Porg(\boldsymbol{c}_{i-1}))/\mathbf{mean}(Porg(\boldsymbol{c}_{i-2}))$ bd is the number of points that are removed from $\boldsymbol{c}_i$ and $\boldsymbol{c}_{i-1}$ if $\boldsymbol{b}_{i-1}$ being replaced with $\boldsymbol{b}_i$ $flg = bd \times Sd \times a$ elseif $\boldsymbol{b}_{i-1}$ *removed* $flg = (\text{bd}+1) \times Sd^n$ elseif $\boldsymbol{b}_{Imp}$ being *replaced* $flg = 0$ end $dif = \boldsymbol{r}_{est} - \boldsymbol{r}_{est\_old} + flg$	for systems in which mean($\boldsymbol{r}_i$) is not well approximated with a constant $\boldsymbol{b}_{n+i}$ domain is the estimated domain to replace $\boldsymbol{b}$ domains *removed* $\boldsymbol{r}_{est\_old}$ is $\boldsymbol{r}_{est}$ calculated before the modification to the surface
`Rsum_repl_tot`	$a = (\mathbf{mean}(\boldsymbol{r}_{i-1}) + \mathbf{mean}(\boldsymbol{r}_{iest}))/2$ m is the number of elements in the regions higher then $\boldsymbol{r}_{est} = \sum \boldsymbol{r}_{<i} + \sum \boldsymbol{r}_{iest}$ $Sd = \mathbf{mean}(Porg(\boldsymbol{c}_{i-1}))/\mathbf{mean}(Porg(\boldsymbol{c}_{i-2}))$ $wtot$ = (no. $\boldsymbol{c}_{i-2}$)/(no. $\boldsymbol{c}_{i-1}$) bd is the number of points that are removed from $\boldsymbol{c}_i$ and $\boldsymbol{c}_{i-1}$ if $\boldsymbol{b}_{i-1}$ being replaced with $\boldsymbol{b}_i$ $flg = bd \times Sd \times a \times wtot$ elseif $\boldsymbol{b}_{i-1}$ *removed* $flg = (\text{bd}+1) \times (Sd \times wtot)^n$ elseif $\boldsymbol{b}_{Imp}$ being *replaced* $flg = 0$	for systems in which constraints on the states cause the $\boldsymbol{c}_{i-1}$ to become too small $\boldsymbol{b}_{n+i}$ domain is the estimated domain to replace $\boldsymbol{b}$ domains *removed* $\boldsymbol{r}_{est\_old}$ is $\boldsymbol{r}_{est}$ calculated before the modification

	end $dif = \boldsymbol{r}_{est} - \boldsymbol{r}_{est\_old} + flg$	the modification to the surface
`Rsum_repl_filt`	Same as `Rsum_repl` but with $\boldsymbol{r}_{i-1}, \boldsymbol{r}_{iest}, Porg(\boldsymbol{c}_{i-1})$ and $Porg(\boldsymbol{c}_{i-2})$ filtered by removing all $\boldsymbol{r}\|_b$ higher then the mean of rest of the $\boldsymbol{r}$ multiplied by some factor and setting the values of the higher states equal to the mean of the rest of the state.	for systems that c_{i-1} borders on c_j where i>>j and this causes the surface to become too small

4.5 Calculating any remaining controllable $\boldsymbol{b}$

During the process of optimising the surface $\boldsymbol{b}_j$ domains are removed and are not replaced by any $\boldsymbol{b}_i$ domains (where $j<i$). These potential $\boldsymbol{b}$ domains would become uncontrollable states if they were not added back. The Matlab script `add_YBflgz` adds the $\boldsymbol{b}$ domains that were removed and not replaced and its children to the u-surface and optimises these values by only replacing $\boldsymbol{b}$ domains and not removing any (Step 18 in Table 4.1). This process ensures that all the controllable $\boldsymbol{b}$ domains for that grid were found.

4.6 Calculating the final u-surface

After all $\boldsymbol{c}_i$ have been optimised, what is obtained are the values of all $\boldsymbol{b}_i$ and the mappings between $\boldsymbol{b}_i$ and $\boldsymbol{b}_j$ where $i>j$ and $\boldsymbol{V}_i(\boldsymbol{c}_i)$ are obtained. All values $\boldsymbol{c}_i$ have been chosen to be optimal.

This gives some flexibility in selecting the final surface. The optimal solution of the u-surface with regard to the output disturbance would be to select the origins of $\boldsymbol{b}_i$ that give the maximum $\boldsymbol{r}_j(\mathbf{f}^{-1}(\mathbf{center}(\boldsymbol{b}_i), V_i(\boldsymbol{b}_i))$ and storing the respective optimal $\boldsymbol{v}_i$. The program `cal_YBi_Rmax` performs this task.

Chapter 5

Driven pendulum simulations and implementation

The aim of the simulation is to compare the performance of the u-surface control method with another technique that is a robust nonlinear controller that can control a system with constraints on the control inputs. The control method with which the u-surface is compared to is a modified bang-bang type control method (see Section 5.2).

The driven pendulum is described in the next section and the implementation of the bang-bang type controller (Section 5.2) and the u-surface control method (Section 5.3) are explained thereafter. Finally the results of the simulations done on the computer and on the driven pendulum experiment are presented in Section 5.4.

5.1 The driven pendulum

The experimental example that was implemented had to satisfy certain requirements to be computationally feasible (see Section 3.1). These requirements were that the system be of low order, preferably single input single output; that it be time invariant; and that all the states of the system could be measured or estimated with sufficient

accuracy for control. The driven pendulum satisfied these requirements and had been implemented for fixed setpoint control by Yorke (1998).

The description of the inverted pendulum system follows in the next two sections.

5.1.1 The driven pendulum system

The driven pendulum consists of a motor with a pendulum attached to the shaft in such a way that the pendulum can only rotate in a fixed plane (see Figure 5.1).

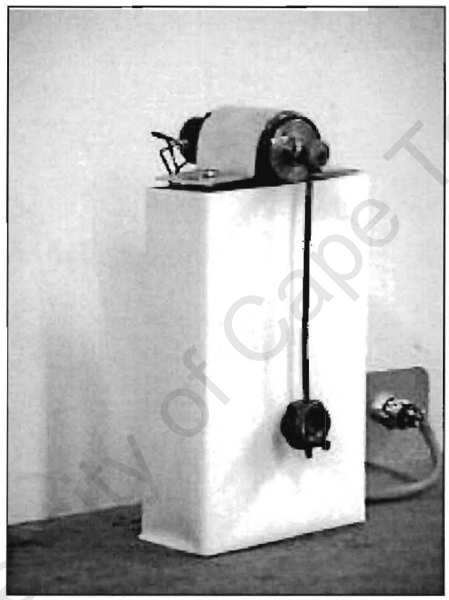


Fig. 5.1 **The driven pendulum** (photo by R. Yorke)

A potentiometer was attached to the shaft of the motor to record the position of the pendulum (the velocity is estimated).

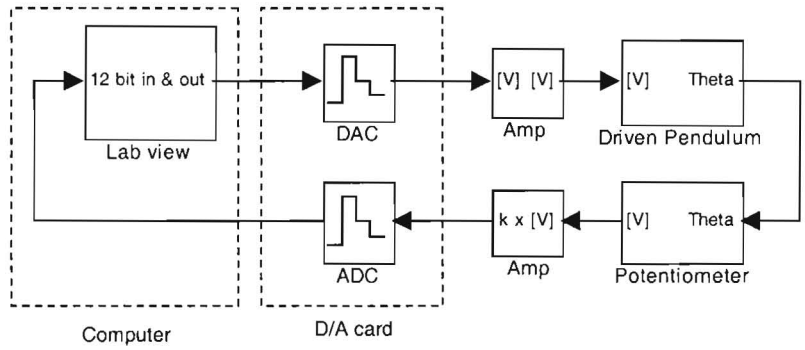


Fig. 5.2 **Layout of driven pendulum experiment**

Figure 5.2 describes the layout of the system. The computer being used is a P3 930MHz with 128Mb of RAM, Lab View was used to implement the controllers and to interface with the DT302 analogue to digital card. See Appendix C for a more detailed description of the system.

A number of tests were performed on the system and it was found to be working satisfactorily. The scaling factor and offset value to convert the ACD count values to a theta value was calculated. The input data was over-sampled by a factor of 8. The time taken to acquire the 8 samples was 0.7 ms, this was the most time consuming process in the control loop for both controllers examined.

The velocity was calculated by taking an extra sample 2ms before the sampling period and calculating the change in position over the 2ms period (a 2ms period gave more accurate results then a 1ms period, this is probably due to errors in the sampling period). This velocity was compared with the velocity determined by fitting a B-spline though the sampled Theta values and calculating the derivative at the sample points (this was done by using the Matlab Spline toolbox). Figure 5.3 is an example showing the differences between the dTheta/dt calculated and the dTheta/dt values calculated using the differentiation of the spline. As seen from the Figure the spline gives a much smoother approximation of dTheta/dt. The analysis of the data using six examples of the driven pendulum swinging up (the total of 229 points) was used to estimate the gaussian distribution of the error between the spline and the sampled value. The results were a mean of -0.55 [rad/s] and variance of 1.5 [rad/s] and a

maximum error of 4.6 [rad/s] with the mean of the spline dTheta sampled signal being 5.3 [rad/s] (calculated using Matlab's Statistics toolbox). It was assumed that the noise present in the driven pendulum system is less than or equal to these values.

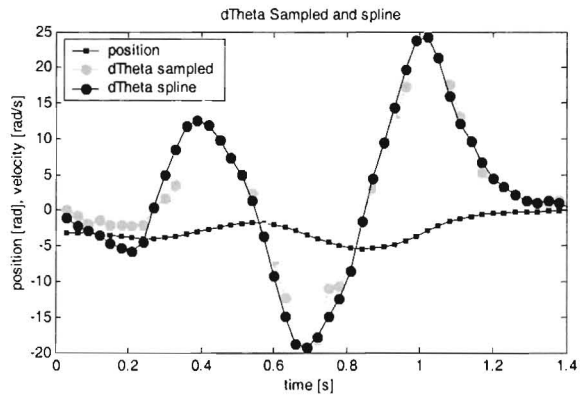


Fig. 5.3 **Comparison of dTheta sampled and dTheta spline** (Sys.5)

After the tests and set-up mentioned above the driven pendulum was modelled.

5.1.2 Modelling the driven pendulum

Modelling and controlling of the driven pendulum has previously been performed by Yorke (1998). The same choice of model was implemented, as it described the system with sufficient accuracy for fixed setpoint control. The model is:

$$\dot{\boldsymbol{x}}_n = \begin{bmatrix} \hat{x}_{2n-1} \\ -\frac{Mgl}{J}\sin(\hat{x}_{1n-1}) - \frac{\beta}{J}\hat{x}_{2n-1} + \frac{K}{J}u_{n-1} \end{bmatrix} \tag{4.1}$$

where $\boldsymbol{x}_n = \begin{bmatrix} x_{1n} \\ x_{2n} \end{bmatrix}$ are the states, and $\hat{\boldsymbol{x}}_n = \begin{bmatrix} \hat{x}_{1n} + w_1 \\ \hat{x}_{2n} + w_2 \end{bmatrix}$, $x_{1n} \in [x_1^-, x_1^+]$ and $x_{2n} \in [x_2^-, x_2^+]$. The output disturbances are w_1 and w_2, and u_n is the control to the system ($u \in [u^-, u^+]$). x_1 is the angle between the pendulum and the vertical position (Theta), J is the inertia of the pendulum and β is the coefficient of friction, M is the

mass the pendulum, g the is earth gravitational constant and l is the length of the pendulum. K is a constant and $u \in$ [Umin, Umax] is the input to the actuator.

The driven pendulum was modelled in three stages:

- drop measurements
- holding measurements
- fine tuning based on simulation results

(Model fitting of data is presented in Appendix D)

Drop tests

This test consisted of bringing the pendulum to the upright position and allowing the pendulum to fall and settle at the equilibrium state at the bottom. The Matlab script `nonlinlsq_MB` was used to calculate the least squares error fit of the data to the model in (4.1), and so used to calculate the values of Mgl/J and β/J.

Holding measurements

The relationship between the voltage and torque was calculated by supplying a voltage to the motor, vibrating the stand of the pendulum to eliminate the effects of static friction, and measuring the position at which the respective voltage held the pendulum. A deadband was present in the motor, and was calculated to be 0.97 volts symmetrically about zero volts. Deadband compensation has been implemented in the Lab View code. From equation (4.1) the following equation was derived for when the pendulum is being held stationary:

$$\frac{K}{J}u_{n-1} = -\frac{Mgl}{J}\sin(x_{1_{n-1}}) \tag{4.2}$$

Using equation (4.2) and the holding measurements, a least squares error fit for K/J was calculated.

Fine tuning based on simulation results

The bang-bang controller was now implemented (see Section 5.2) and the results of some simulations were used with `nonlinlsq_MB` to further refine the model of the pendulum.

The values attained for the pendulum model were:
Mgl/J = 144, B/J = 3.8, K/J = 0.014

5.2 Bang-bang type controller

The qualities required for the comparison method were: that it would work with a nonlinear constrained sampled system; give near optimal response with respect to rise time and output disturbance robustness and that the results be repeatable (given the same disturbances and noises). A number of methods were investigated for the purpose of being the comparison method for the driven pendulum; they are shown in Table 5.1.

Table 5.1 **Comparison of possible comparative control methods**

Method	Comment
Nonlinear Model Predictive Control	Requires training and difficult to implement in real time in Lab View
Optimal Nonlinear Control	Cannot be implemented in real time
Variable Structure Control	Difficult to implement on constrained system
Energy Control	Not flexible, it was not possible to adjust the system to be more robust
Adaptive Control	None were found that could give robust nonlinear control to a constrained system
Feedback Linearisation	Does not take into consideration constraints on control

After examining the controllers a method was implemented that used ideas from Energy Control (Åström and Furuta, 1996) and Variable Structure Control (Soltine and Li, 1991) and was designed to control the driven pendulum. This controller was referred to as the bang-bang type controller.

The bang-bang type controller uses switching surfaces and an energy pump to move the driven pendulum from the initial condition to a region near enough to the switching surfaces that will guarantee that the driven pendulum will be controlled to the setpoint.

The design of this controller is based on the following stable manifolds in the state-space plane of the driven pendulum (they are illustrated in Figure 5.4). The Umax, Uzero, Urobust and Umin stable manifold are defined for the driven pendulum with the control value equal to Umax, zero, fac×Umax (where fac is set to 0.7 for the example in Figure 5.4) and Umin respectively.

The Umax and Umin are important manifolds as all states starting between these manifolds (called the *Inside states*) can be controlled to the Uzero surface with the control set to either Umax or Umin. The states on the side of the Umax or Umin surface away from the Uzero (called the *Outside states*) cannot be controlled to the setpoint with any constant control value $u \in$ [Umin, Umax] (for the simulations examined Umin was equal to –Umax).

The optimal switching surface converging on the setpoint with regard to output disturbances would be the surface exactly between the Umin and the Umax manifolds. For this surface the largest output disturbance would be required to make the system unstable (where unstable means that the states of the system travel from *Inside states* to *Outside states*). The Uzero surface falls nearly exactly between the Umax and Umin solution (in the region near the setpoint) and so this switching surface was implemented on the driven pendulum.

Because *Outside states* cannot be controlled to the setpoint with a constant input an energy pump, defined as

$$U = \text{Umax} \times \mathbf{sign}(x_2) \tag{5.1}$$

This is derived from the energy control method described in Åström and Furuta (1996). This method increases the energy as fast as possible moving the driven pendulum from the initial value to beyond the Urobust surface, and then the Zero switching surface is used to determine the control values. The Urobust is used to give the controller more robustness as the states are driven further away from the Umax (see Figure 5.4) surface and so larger disturbances are required to make the system unstable.

Once the controller gets to the region between the Urobust surface and the Zero surface the switching controller takes over. It is defined as

$$\text{u} = \text{Umax} \times \mathbf{sat}_s(\hat{\text{w}} \times \text{d}) \tag{5.2}$$

where $\hat{\text{w}}$ is the unit vector on the line going from left to right, and d is the vector that is the minimum distance from the state x_k and the Zero surface and

$$\mathbf{sat}_s(x) = \begin{cases} -s & if \quad x <= -s \\ x & if \quad -s < x < s \\ s & if \quad x >= -s \end{cases} \tag{5.3}$$

The saturation function (5.3) is used to avoid chatter about the Zero surface; as in Åström and Furuta (1996).

When the states get close enough to the setpoint the controller switches to a LQR.

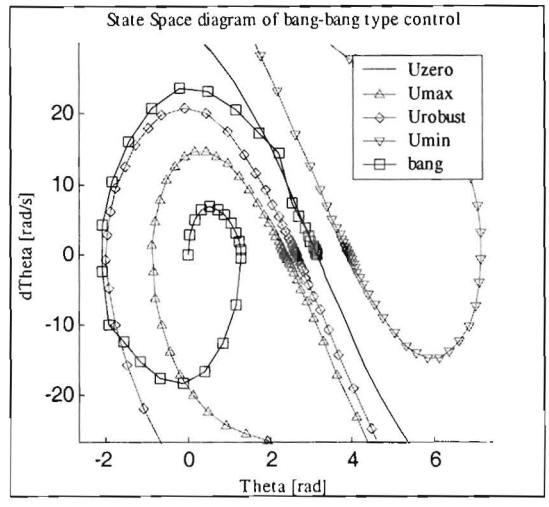


Fig. 5.4 **Bang-bang control in state space** (Sys.6)

The manifolds for the bang-bang type controller were found using a fourth order Runge-Kutta, in the Matlab script `design_slide`. The resulting bang-bang type controller was implemented in the Matlab script `ctrler_plus` and the results are presented in section 5.4.1.

The bang-bang type controller was also implemented on the driven pendulum. For the pendulum implementation the manifolds were approximated as polynomial sections to reduce the computation necessary to determine what side of the manifolds the state of the system was in. The LQR controller has been implemented on the pendulum but the saturation function has not been implemented on the pendulum.

The computational time taken by the controller from when the sampling began to when the control value was made available to the DAC was 0.8 ms. This was satisfactory as the sampling time was 30ms.

5.3 u-Surface control

The algorithm in Chapter 4 was used to determine the u-surface of the control. This surface was simulated in Matlab using `ctrler_plus` and the results are presented in Section 5.4.1.

The u-surface controller was also simulated on the driven pendulum. To implement the controller the u-surface was sampled with a fine grid to make an array that was used as a lookup table (as small as a tenth the size of the grid spacing were used (see Section 4.2.1)). This reduced the computational intensity as arrays containing the values for the rim($\boldsymbol{b}_i$) values were not required and no interpolation had to be performed.

The performance of the system was not noticeable degraded due to this the modification (as is seen in Figure 5.5)

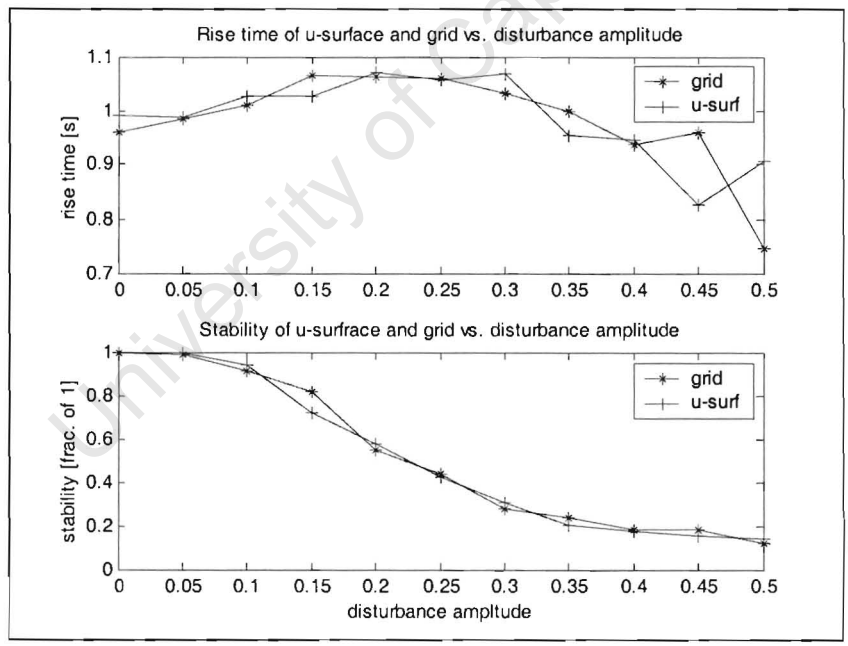


Fig. 5.5 **Rise time and stability vs. disturbance for u-surface and grid** (Sys.7)

The computational time taken from when the data is sampled to when the control value is made available to the DAC was 0.8 ms although the time taken to load up the u-surface controller was 4.9 s.

5.4 Simulations and implementation

In the simulations emphasis is placed on the swing up of the driven pendulum from the initial state of $x_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ to the setpoint $x_0 = \begin{bmatrix} \pi \\ 0 \end{bmatrix}$ and not the stability about the setpoint as this can be implemented by a LQR controller based on the linearisation of the driven pendulum model.

Throughout Section 5.4.1 the probability distribution of the output disturbance is taken as bi-variate gausian. The variance of the output disturbance in theta and dtheta was calculated as the disturbance amplitude multiplied by the respective grid spacing. The relationship between the magnitude of theta and dtheta probability distributions was chosen arbitrarily and in no way relates to the real system.

5.4.1 Simulations and performance calculations

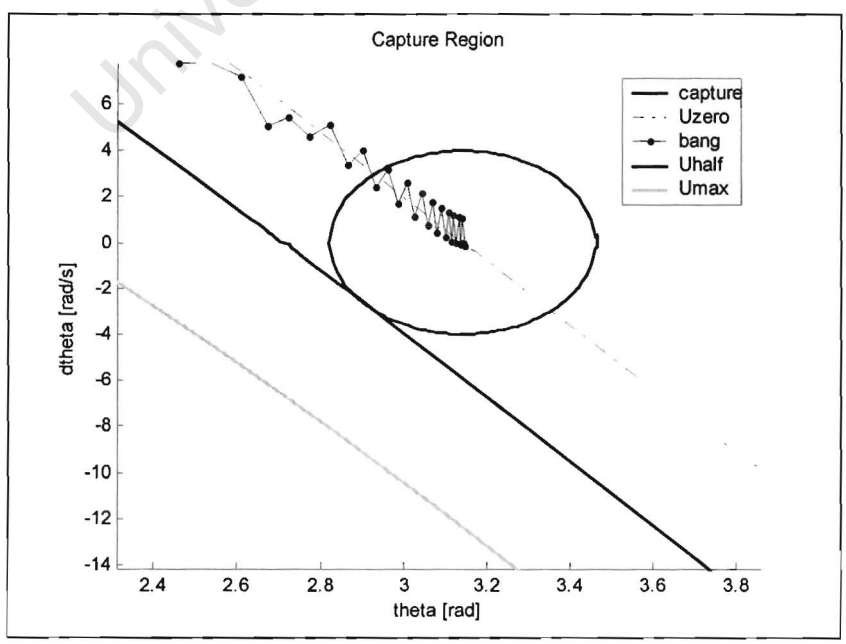


Fig. 5.6 **Capture region**

Performance measurements

Two of the performance measurements taken of the systems in this section are *rise time* and *stability*. They are defined as follows:

Capture time is the time taken for the driven pendulum to go from the initial state to a circle centred about the setpoint (see Figure 5.6). The circle is tangent to the manifolds Umax/2 and Umin/2 (only Umax/2 is shown in Figure 5.6).

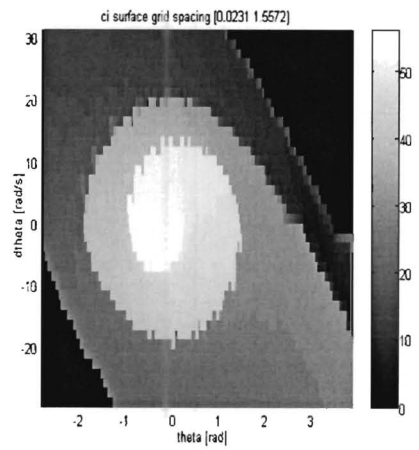
The Maximum Likelihood estimation of the *capture time* can be seen as the expected *capture time* of the system. Runs where the *capture times* are much larger than this expected *capture time* are likely to require more swings of the driven pendulum to get the pendulum near tc the setpoint. So the *stability* of the system was taken as the fraction of the *capture time* values that were less than 1.5 times more than the medium of all the *capture times* of the system (based on a histograph consisting of twenty uniform bins ranging from the smallest time to the longest time for 1500 swing up runs) divided by the total number of runs.

Rise time is the average of the *capture times* less than 1.5 times more than the medium *capture time*.

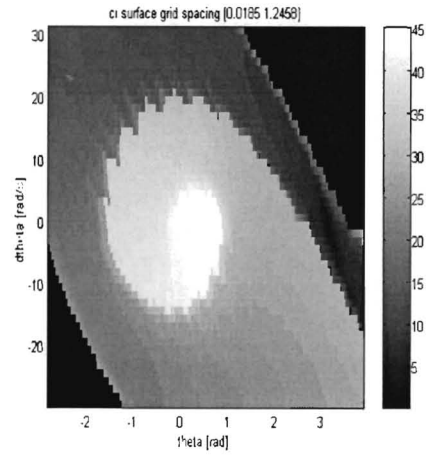
Effect of grid size on performance

The Figure 5.7 shows the performance trade-off between *rise time* and grid spacing, by using surfaces that have not been optimised for disturbance rejection. The reason for not optimising the surface is that the results are more consistent and therefore shows the trade off better.

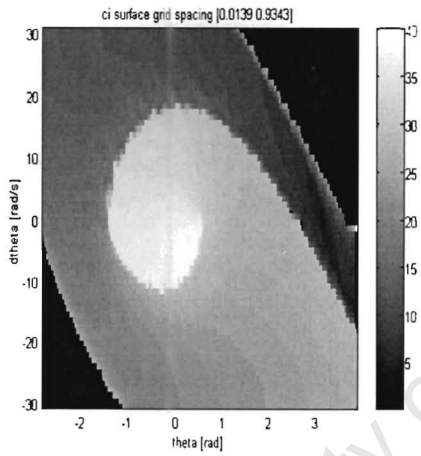
In Figure 5.7 the *rise time* is calculated from the maximum c_i surfaces. These are plotted in Figure 5.8. In Figure 5.8 the grid spacing is a percentage of the maximum grid spacing calculated by `cal_max_block`. In the example in Figure 5.8 the maximum grid spacing was [0.0231 1.5572].



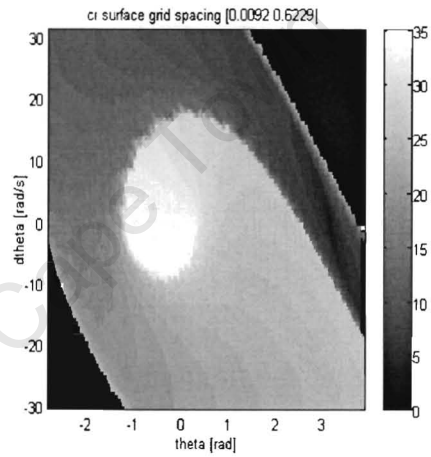
(a) grid spacing [0.023 1.557] (Sys.1)



(b) grid spacing [0.018 1.245] (Sys.2)



(c) grid spacing [0.014 0.934] (Sys.3)



(d) grid spacing [0.009 0.622] (Sys.4)

Fig. 5.7 **Non-optimised surface with varying grid spacing**

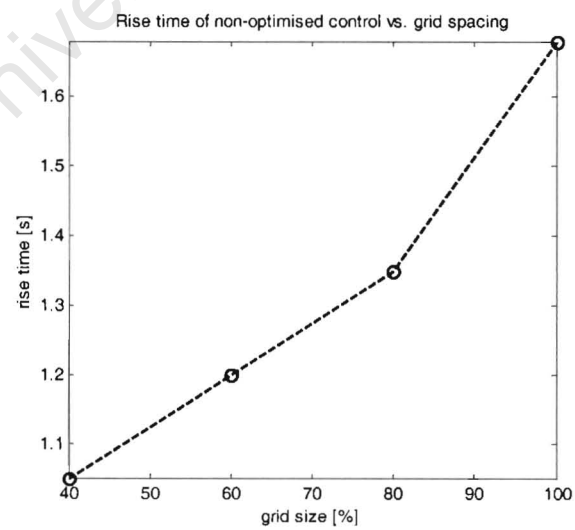


Fig. 5.8 **Rise time of non-optimised control vs. grid spacing**

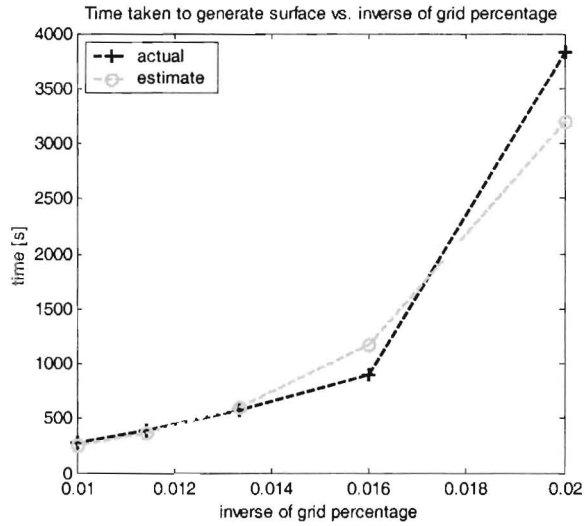


Fig. 5.9 **Time to generate u-surface vs. the inverse of the grid percentage**

The axis of Figure 5.9 is the inverse of the percentage of the maximum grid spacing (the same type of percentage of grid spacing as described for Figure 5.8).

The u-surface optimised is not represented by the surfaces above but are described in Section 5.5; they are Sys.8, Sys.9, Sys.10, Sys.11, Sys.12 corresponding to grid percentages percentage 50, 62.5, 75, 87.5 and 100 (as in Figure 5.6) respectively. But the state space was limited to a fifth of the normal size. The state space was reduced to limit the time required to calculate the surfaces. An equation estimating the computational time required versus grid size was modelled by an equation of exponential form and the parameters were determined using least squares. The resulting model was:

$$Tcal = e^{3.0 + 250/g\%} \tag{5.4}$$

where *Tcal* is the time taken to calculate the u-surface and $g_{\%}$ is the grid percentage value.

Performance of surfaces optimised for $P_w(w)$

Every controller is designed for a known variance of $P_w(w)$. In Figure 5.9 the performance of a number of controllers designed with different $P_w(w)$ variances are

plotted. The bang-bang controller is also plotted with these plots to for comparison reasons.

The systems used in Figure 5.10 are: bang-bang (Sys.6) and u-surface controllers with output disturbances with standard deviation of 0.1, 0.7, 1.1 and 1.5 correspond to systems Sys.13, Sys.15, Sys.16, Sys.17 respectively.

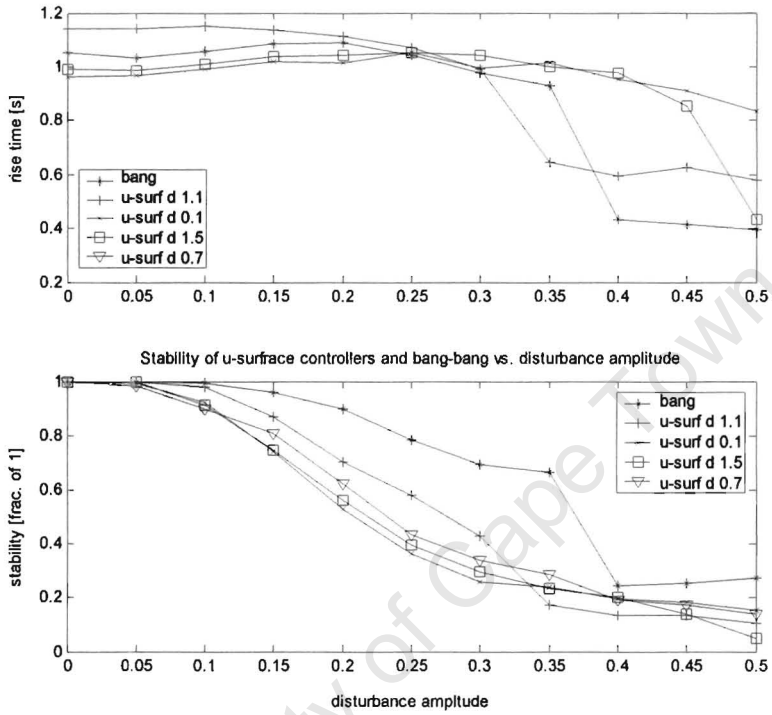


Fig. 5.10 **Computational time to calculate u-surface**

Refer to "Performance measurements" in Section 5.4.1 for definitions of *rise time* and *stability* in Figure 5.10.

In the systems used for Figure 5.10 the pendulum makes a number of swings before the pendulum attains sufficient energy to swing to the capture region. During these oscillations it is possible that a disturbance could cause the pendulum to be perturbed to a state where less or more swings are necessary to get to the capture region, generally corresponding to the *capture time* decreasing or increased respectively. It is found that the medium *capture time* calculated as in "Performance Measurements" decreases while a large fraction of *capture times* increase, this agrees with the results seen in Figure 5.10 of shorter *rise times* and lower *stability* with the increase of disturbance amplitude

Performance with regard to model uncertainty

In Figures 5.11 to 5.14 the bang-bang controller (Sys.6) and a u-surf controller's (Sys.18) performance are compared with the parameters for *Mgl*, and β varying from less than 40% to more than 40% of the design values in 20% increments. The design values for *Mgl* and β were 144 and 3.8 respectively. The plots have varying viewing angles to make it possible to examine all the data values.

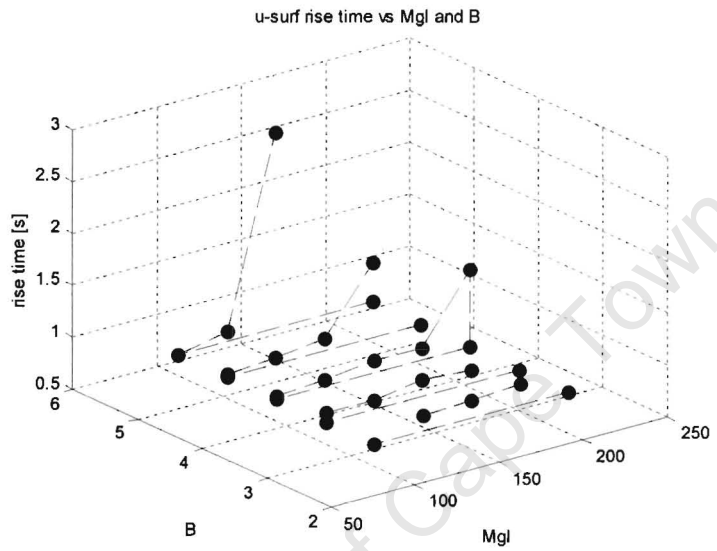
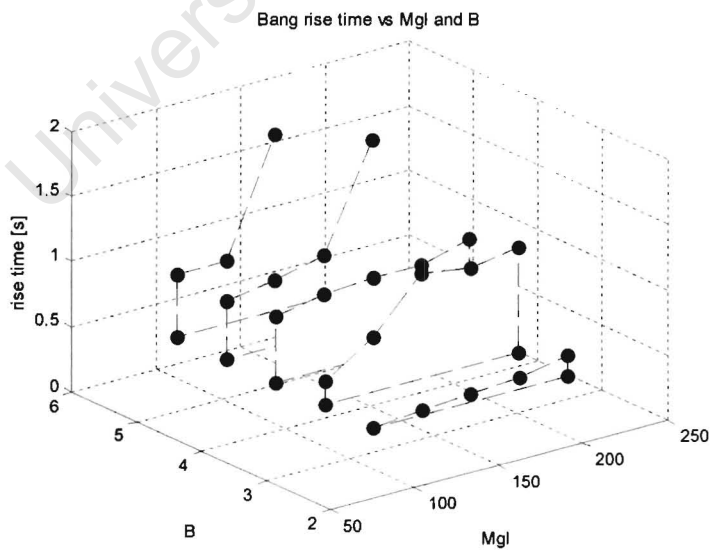


Fig. 5.11 **u-surface *rise time* vs. B and Mgl**



F'g. 5.12 **Bang-bang *rise time* vs. B and Mgl**

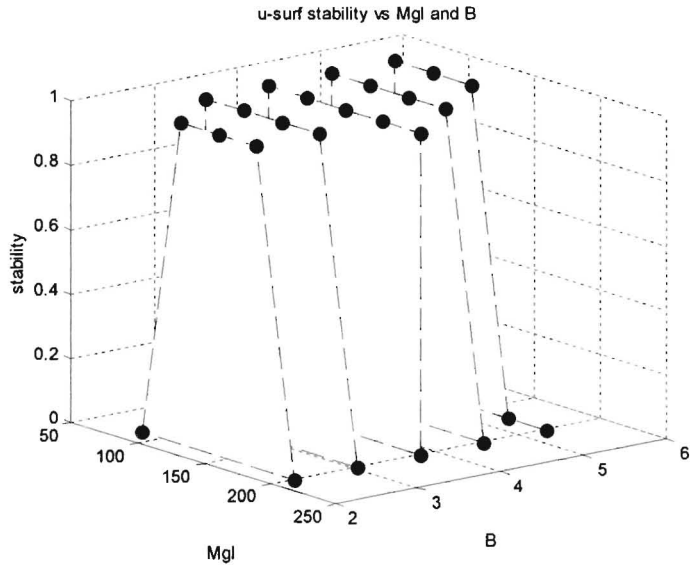


Fig. 5.13 **u-surface *stability* vs. Mgl and B**

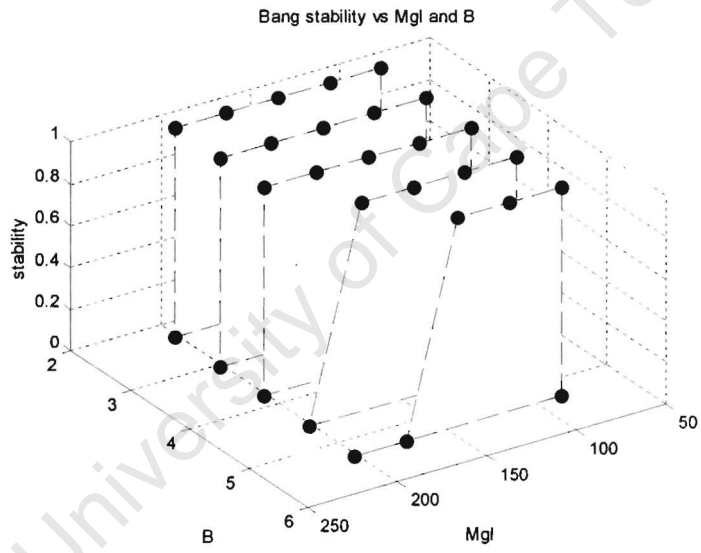


Fig. 5.14 **Bang-bang *stability* vs. Mgl and B**

It is seen in both figures that the *stability* values are generally close to 1 although it drops to about zero as the disturbance amplitude increases. The *rise time* values for the u-surface controller are comparable with that of the bang-bang controller but the *stability* values are lower.

The effect of a longer time sample on u-surface control

An example of the pendulum for a time period of Ts 0.05 is displayed in Figure 5.15. The Sys.18 and Sys.19 are the models and set-up used to generate the u-surface controllers. The bang-bang controller used Sys.6.

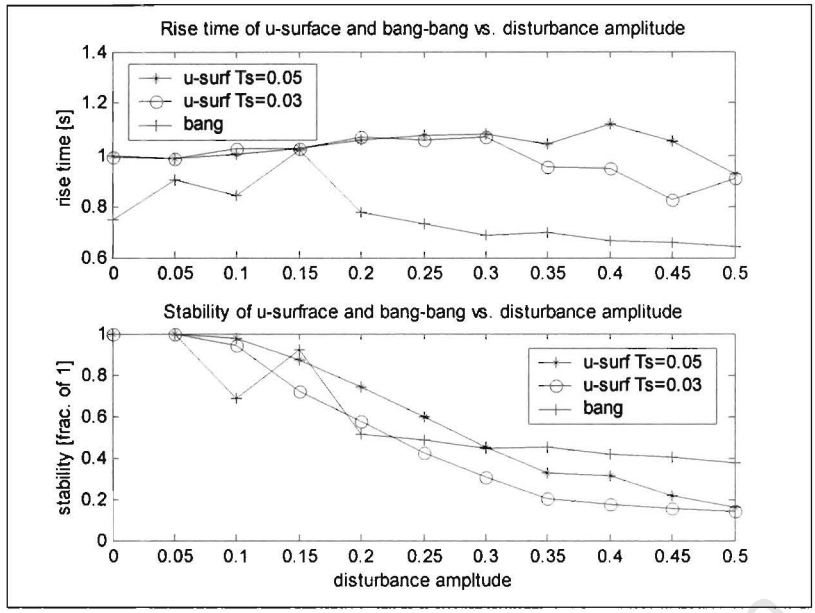


Fig. 5.15 Varying sampling time

It is seen in both figures that the *stability* values are generally close to 1 although it drops to about zero as the values make the parameters vary too much for the controller to achieve its task. The *rise time* values for the u-surface controller are comparable with that of the bang-bang controller but the *stability* values are lower.

Time plots and state space for the driven pendulum

The bang-bang type controller was compared with the u-surface controller on the pendulum. The results are in Figures 5.16-5.18.

5.4.2 Implementation of the driven pendulum

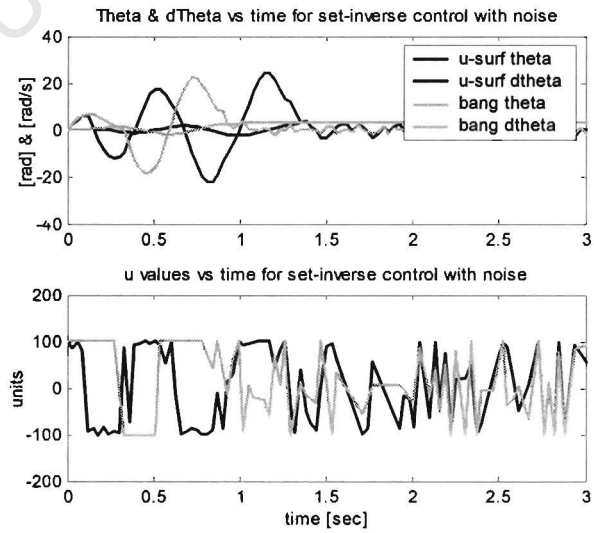


Fig. 5.16 **u-surface and bang-bang controller for the driven pendulum vs. time**

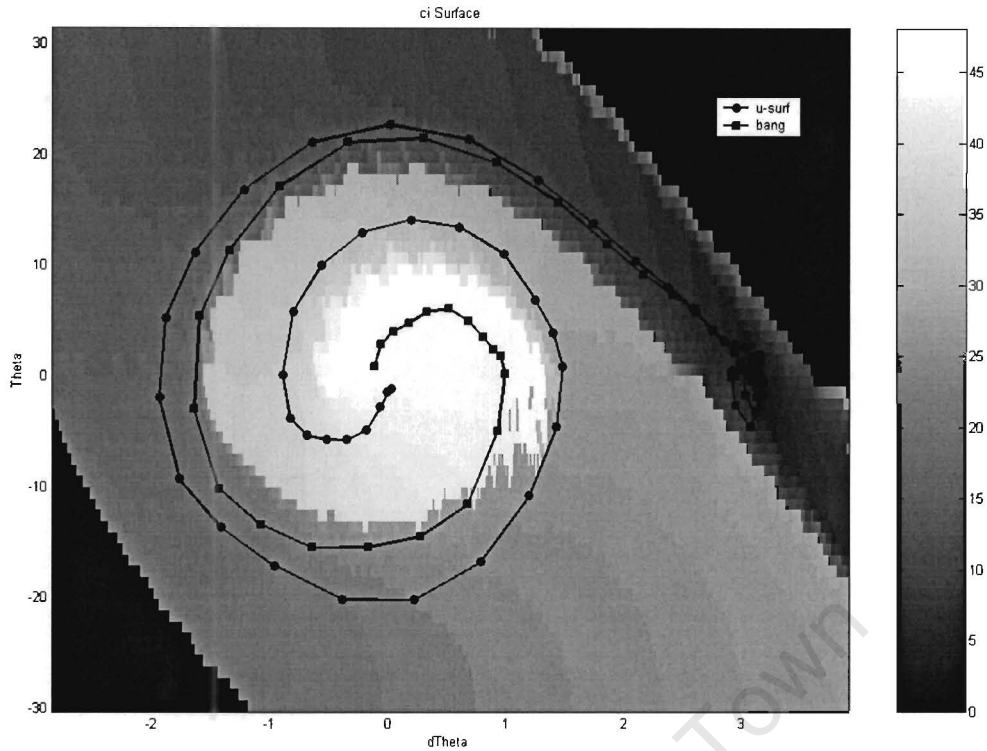


Fig. 5.17 c_i **surface with bang-bang and u-surface optimal control**

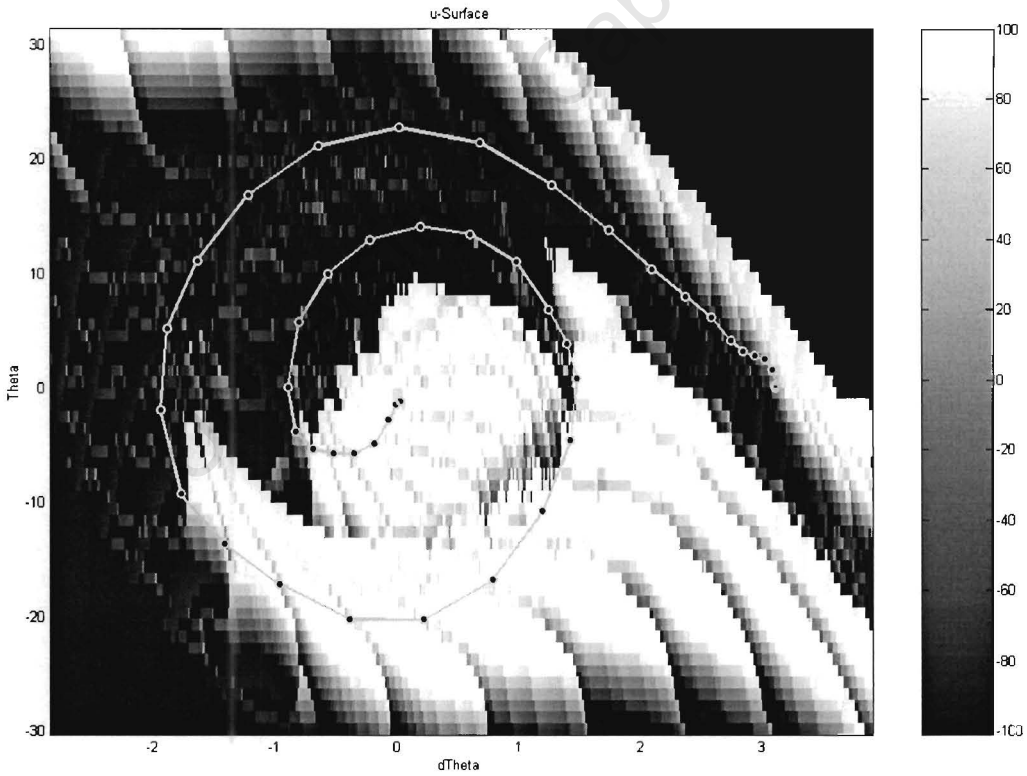


Fig. 5.18 **u-surface with u-surface control**

5.5 Discussion about results

Figures 5.7 and 5.8 shows how the shape of the surface changes as the block sizes decrease and that the *rise time* of the non-optimised u-surface decreases as the grid spacing decreases. In Figure 5.8 it is seen that the *rise time* decreases with the grid spacing decreasing.

The time taken to calculate the u-surface is plotted against the inverse of the block size can be seen in Figure 5.9. It was found that the computational time increases exponentially with the inverse of the block size.

In Figure 5.10 is a comparison between controllers designed for different $P_w(w)$ variances. From this graph we can see that the performance of the bang-bang controller is generally more robust, while there is no definite pattern between $P_w(w)$ variance and performance. This is further evidence that the surfaces generated are not optimal. However it could also have been that the number of samples (1500) taken was too small.

In figures 5.11 to 5.14 the performance with respect of model uncertainty is examined. It should be noted that both controllers could control the system over most of the region defined by a 40% variation in both Mgl and β.

Figure 5.15 supports the idea that longer sampling time could result in more robust systems. This would be logical as the C_i regions developed would be larger than for longer time samples.

It is seen in Figure 5.16 that the bang-bang controller implemented on the driven pendulum is faster than the u-surface controller. Figure 5.17 demonstrates that the u-surface controller is working as the fitness function envisaged (i.e. the states travel to the setpoint via monotonically decreasing regions). Figure 5.18 shows that the u-surface control inputs are similar to that of the bang-bang energy pump, while the states are in the region about the origin (see equation 5.1).

Chapter 6

Conclusions and future development

6.1 Conclusions

The following conclusions can be drawn from the results in this thesis.

1. It was shown that u-surface control could be used to swing up a driven pendulum.

2. The u-surface controller solution can be synthesised requiring only the model for the driven pendulum system and the probability disturbance of the output disturbance.

3. The bang-bang controller has a shorter rise time and gave a more robust response. The reason for the lower performance of the u-surface controller was because the u-surface that was generated is sub-optimal.

4. Better performance can be obtained by reducing the size of the grid used, but the computational intensity increases exponentially with the inverse of the area of a domain from the grid.

6.2 Future development

Throughout the duration of this project a number of problems and improvements have been contemplated. These are:

1. Solutions for the u-surface do not give optimal performance; a method that would increase the performance would be an overlapping set method. This method would not limit the u-surface to a surface but would allow the higher regions to overlap with the lower regions; this would result in the lower regions being more robust especially along the boundaries between regions. To avoid the overlapped regions becoming too large a method to trim the overlapping regions will have to be devised. This method could be built on the software already written.

2. A major limiting factor to the performance of the u-surfaces and their computation is the use of a grid. A method employing splines to map the children of the higher regions should greatly improve the performance of the system.

3. A method should be investigated to efficiently find the children of a region. This would decrease the computation required, to calculate the u-surface.

4. The development of a method to incorporate the effect of input disturbances, noise and parasitics in the fitness function requires further study.

5. An investigation into the nature of the fitness function and alternative fitness functions is required. In particular methods where the fitness of higher regions contributes to the fitness of lower regions require further investigation.

6. The method of cell mapping became known to the author two weeks before the hand in date. This method has a number of similarities with this control method and requires investigation before undertaking further development of this work.

Appendix A

System descriptions

In this project a number of control examples are given. These examples are given for the pendulum model described in Section 5.1.2, but with differing values for Mgl, β, Umax and the disturbances variance. The grid spacing and the optimisation technique also differ for different models. Some values were constant for all the examples, they were $J = 1$, $K = 1$ and Umin = -Umax. K/J was set to one during simulations and scaled when implemented on the actual pendulum, as $K/J = 0.014$ for the pendulum).

The different set-ups for the examples are given in Table A.1.

Table A.1 **Systems used in examples in project**

System	Mgl	β	Umax	Ts	Disturb.	Grid	Optimisation
1	144	3.8	100	0.03	0	0.023 1.557	none
2	144	3.8	1090 0	0.03	0	0.018 1.245	none
3	144	3.8	100	0.03	0	0.014 0.934	none
4	144	3.8	100	0.03	0	0.009 0.622	none

5	Pendulum	-	-	0.03	-	-	-
6	144	3.8	100	0.03	0	N/A	bang-bang
7	144	3.8	100	0.03	1.5	0.015 1.012	Rsum_repl
8	144	3.8	100	0.03	0.3	0.023 1.557	Rsum_repl
9	144	3.8	100	0.03	0.3	0.019 1.334	Rsum_repl
10	144	3.8	100	0.03	0.3	0.017 1.167	Rsum_repl
11	144	3.8	100	0.03	0.3	0.014 0.973	Rsum_repl
12	144	3.8	100	0.03	0.3	0.012 0.778	Rsum_repl
13	144	3.8	100	0.03	0.1	0.015 1.012	Rsum_repl
14	144	3.8	100	0.03	0	0.015 1.012	Rsum_repl
15	144	3.8	100	0.03	0.7	0.015 1.012	Rsum_repl
16	144	3.8	100	0.03	1.1	0.015 1.012	Rsum_repl
17	144	3.8	100	0.03	1.5	0.015 1.012	Rsum_repl
18	144	3.8	100	0.03	0.3	0.015 1.012	Rsum_repl
19	144	3.8	100	0.05	0.3	0.015 1.012	Rsum_repl
20	60	0.7	22	0.1	0.6	0.019 0.412	Rsum_est

In Table A.1, the column labelled Disturb. stands for Disturbance, and it is the disturbance in both states of the system with the standard deviation equal to the number in the disturbance column multiplied by the grid spacing.

The units used in the table for the grid spacing are radians and radians per second for the top and bottom number respectively.

Appendix B

Sub-optimal solutions

In Section 4.4.3 a number of optimisation techniques are developed because the previously implemented optimisation technique would fail to generate a satisfactory surface. In this chapter two cases are illustrated where a different optimisation technique was required to the one that was chosen.

The case where constraints reduce the surface to a very small region with low fitness values is seen in Figure B.1 and Figure B.2.

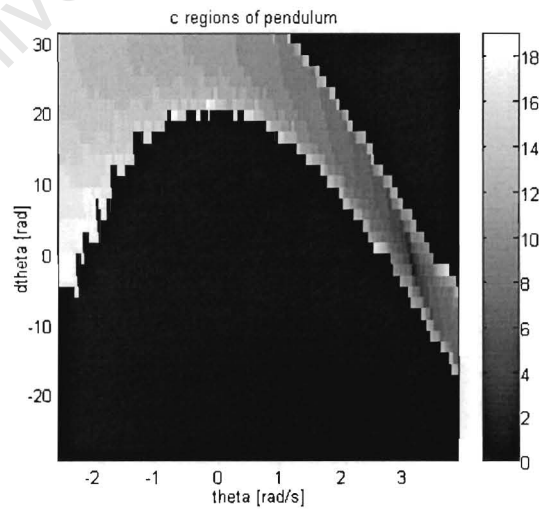


Fig. B.1 **Constraints reducing the size of c_i regions**

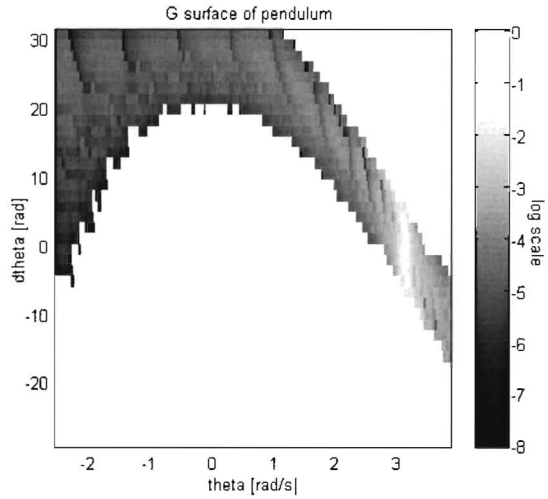


Fig.B.2 **Constraints reducing the fitness of c_i region**

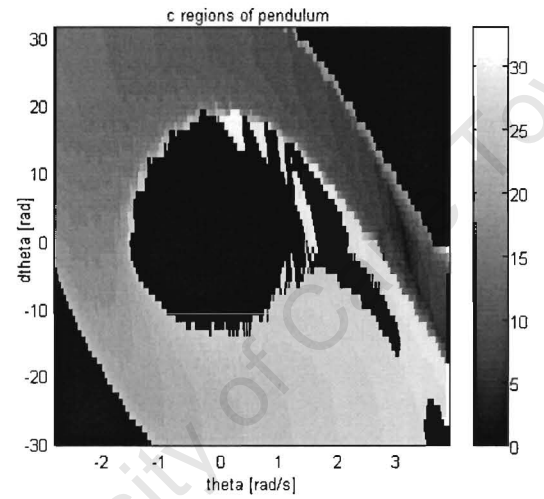


Fig.B.3 **Neighbouring regions with much higher fitness reduce size of c_i**

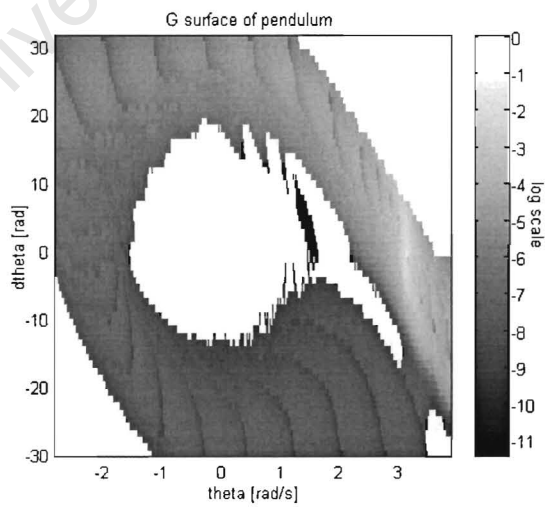


Fig B.4 **Neighbouring regions with much higher fitness reducing fitness of c_i**

Appendix C

Driven pendulum system

The driven pendulum system is described in this section. The picture below describes the layout of the system.

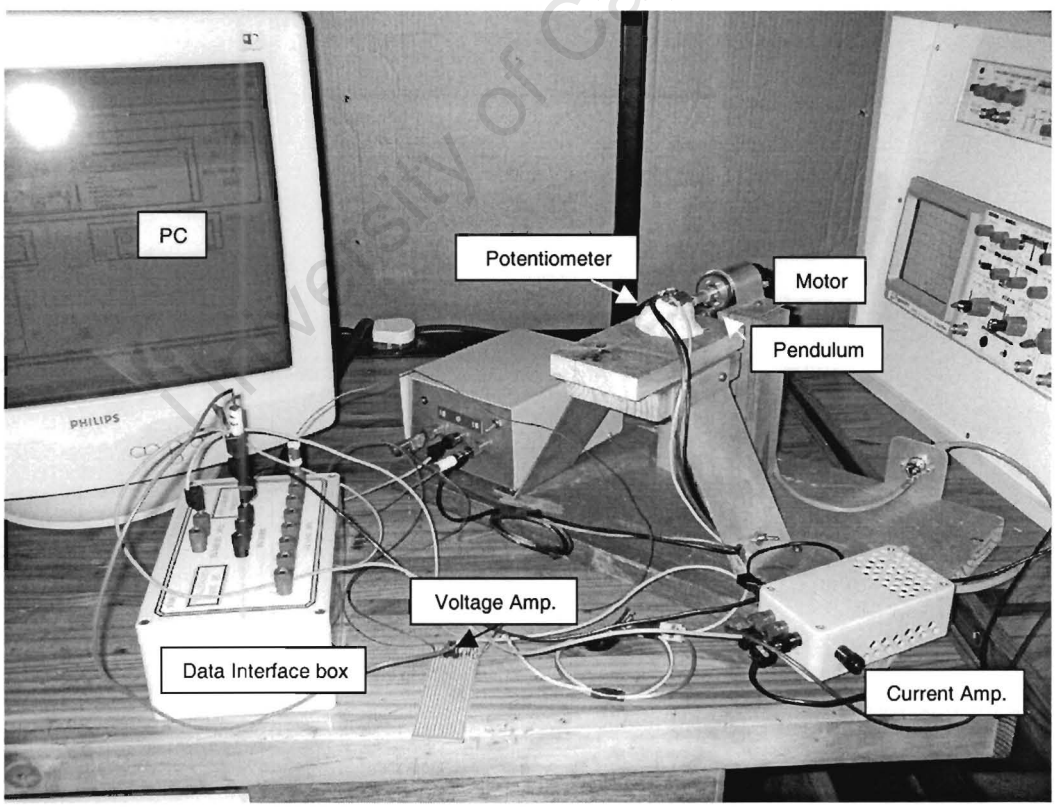


Fig C.1 **Photo of layout of the system**

The motor is a Maxon Motor model S 2332-966-51-276-200, which is a DC motor with an integrated tachometer. The following are the relevant specifications of the motor:

Table C.1 Motor specifications:

Nominal Voltage	12 V
Starting Current	3.78 A
Max. permissible speed	9200 rpm
Max. continuous current	1.2 A
Max. continuous torque	27.60 mNm

The Motor was driven using an AB class amplifier containing TIP41 and TIP42 power transistors.

The input to the ADC was filtered though a low pass filter with a time constant of 4.7ms. This limited the accuracy to which sampling of high frequency signals could be measured and filtered out noise.

D302 Data Translation cards were installed in the computers, these had 12 bit resolution and a ±10V input and output voltage range. The computer was a Pentium 3, 930 MHz with 128MB or RAM.

The system was in place before this project began. The following changes were made to the system.

A tachometer is attached to the motor but for the nature the measurement required a potentiometer was simpler to implement, and was more accurate than the tachometer. However the potentiometer did increase the damping in the system. A LF353N operational amplifier was applied to the output voltage of the potentiometer with a gain of –6.8. This resulted in a voltage change of 2.95V for a 2π change in radians of the pendulum and a sensitivity of 0.0052 [rad/ADC count].

It was found that the measurements of the potentiometer were being disturbed when switching occurred. The reason for this was that the power supply current was limited

to 1A and during switching the current would exceed this limit for a short period of time. This resulted in the voltage of the voltage supply to drop and affect the measurements output of the operational amplifier. To ensure that this was not affecting the measurement readings a separate power supply was used for the operational amplifier. Mr. Attfield found the source of this error.

The region that the driven pendulum was operated in was not limited by the hard limits of the system. Instead, soft limits were imposed on the system, where the voltage to the motor was limited to ±1.92 V.

To decrease the time required for the pendulum to travel from the rest position to the upright position, the weight was moved to the up most position on the pendulum rod. The purpose of this was to reduce the computational intensity required to calculate the u-surface controller. This resulted in the centre of pendulum being 18mm from the shaft of the motor, further increasing the effective damping of the system.

After these changes to the system, the interface to Lab View was implemented using a Dynamic link library, which had been written by a previous student in Visual Basic. Then the controllers were implemented in Lab View.

Appendix D

Model fitting of the driven pendulum

The model of the driven pendulum was obtained by implementing the methods described in Section 5.1.2. The results of the model fitting are presented below.

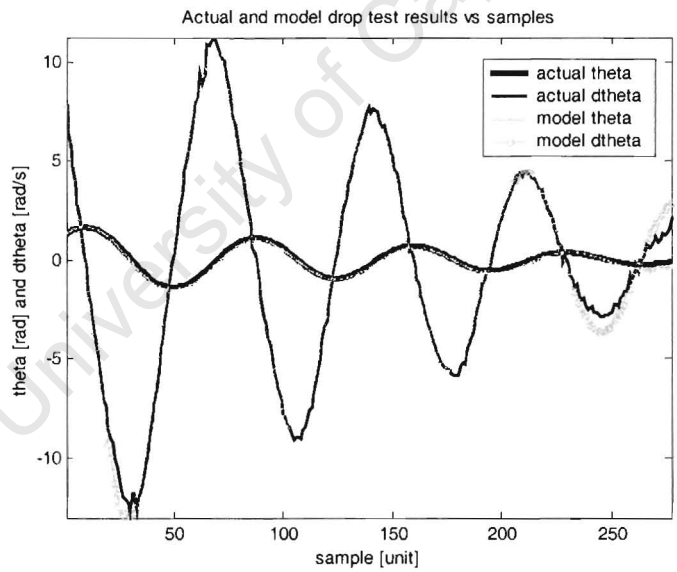


Fig. D.1 **Actual and model results for drop test**

This model fitting was performed utilising the nonlinear least squares method, `nonlinlsq` that is a function in the Matlab optimisation toolbox.

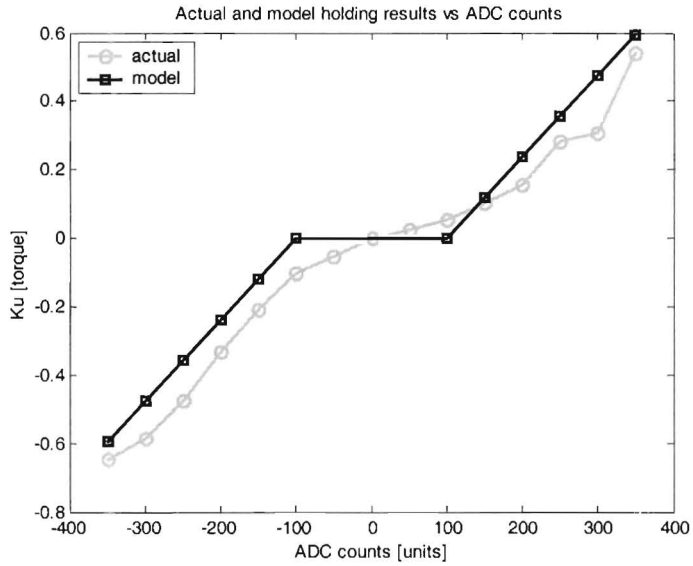


Fig. D.2 **Actual and model results for holding test**

The deadband was chosen and the model fitting in Figure D.2 was done by least squares.

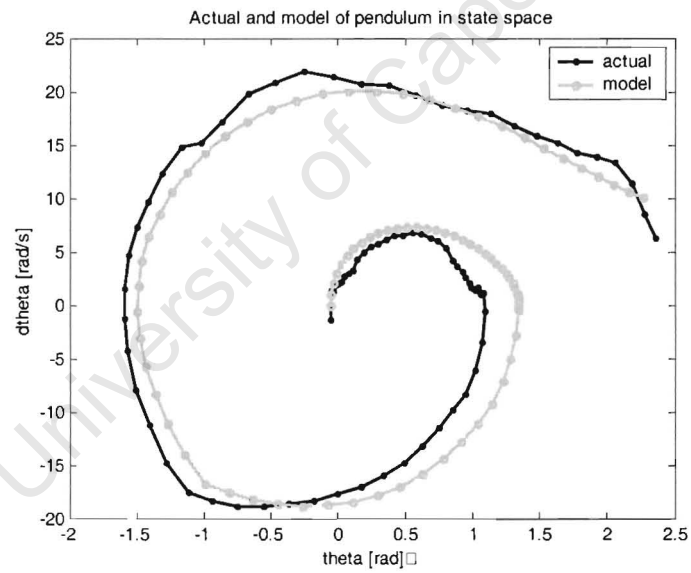


Fig. D.3 **Fine-tuning model using bang-bang control of pendulum**

The model was fine-tuned by modelling the swing up of a bang-bang controller process, by adjusting the parameters a small amount until the dynamics of the system were similar.

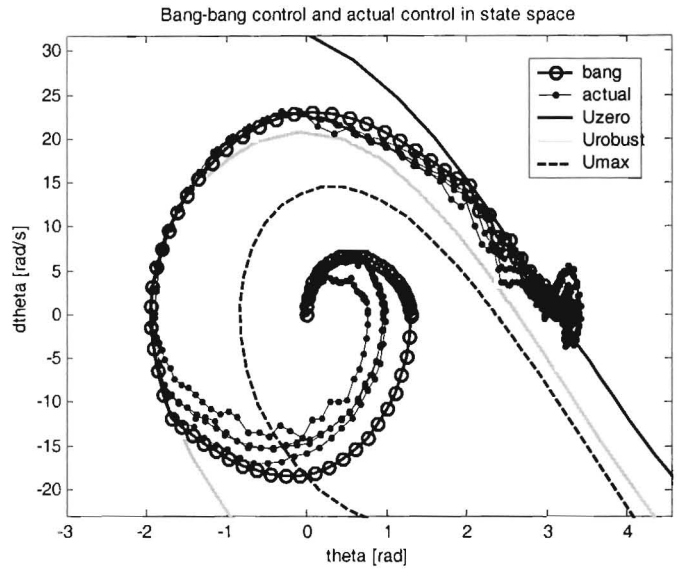


Fig. D.4 **Bang-bang control and actual control in state space**

Figure D.4 shows a number of examples of bang-bang control and the model used for the pendulum in state space. It was difficult to limit the error about the origin of the process. It was assumed that the cause of the poor modelling around the start of the pendulum was due to static friction which is ignored in the model.

Appendix E

Errors between G estimated with RD and Monte Carlo

The G surface is calculated using the Monte Carlo method described in Section 4.4.1, (see Figure E.2) and is compared with the estimated G surface in Figure E.1 calculated using RD. The results of the subtraction of the two surfaces are in Figure E.3. The histograms for the surfaces and the difference are in Figure E.5 and Figure E.6 respectively.

The number of domains with more than 10% error on the mean of the Monte Carlo estimated surface is over 50%. The reason for the very high error between the fitness surfaces is believed to be because of poor mapping between the $\boldsymbol{c}_i$ and $\boldsymbol{c}_{i-1}$ regions due to the irregular boundary of the $\boldsymbol{c}_i$ regions. Further work is required to reduce these errors.

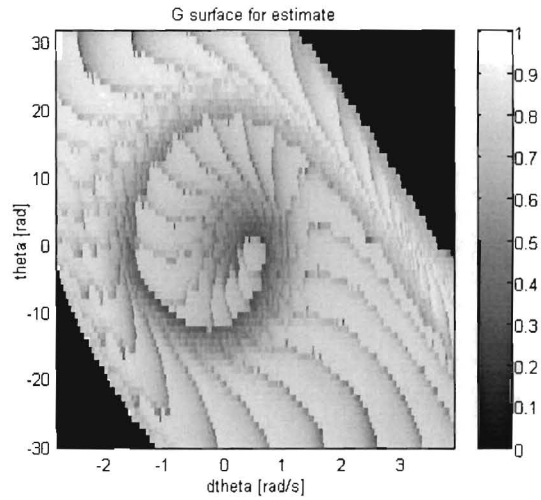


Fig. E.1 ***G* surface estimate calculated by RD** (Sys.13)

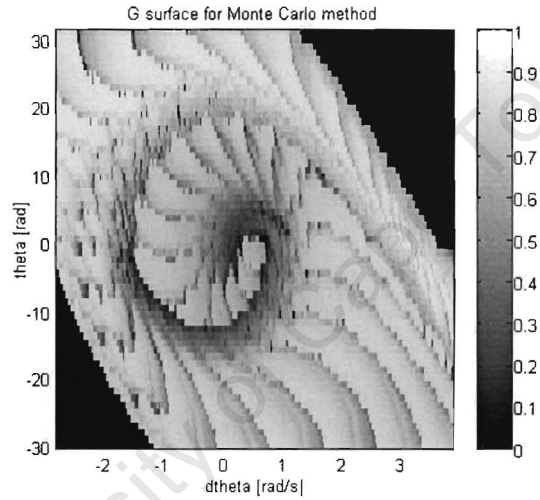


Fig. E.2 ***G* calculated by Monte Carlo method** (Sys.13)

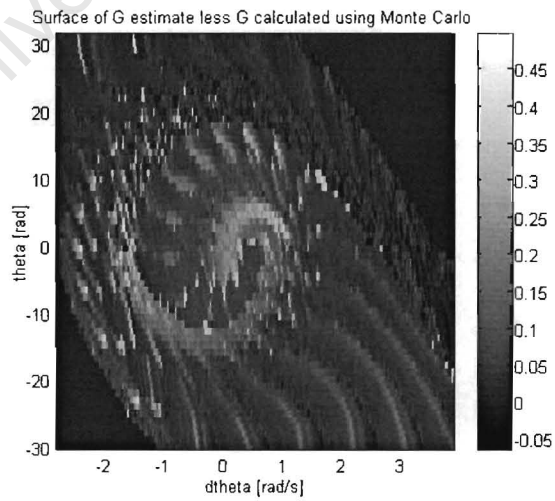


Fig. E.3 ***G* estimate less *G* calculated with Monte Carlo** (Sys.13)

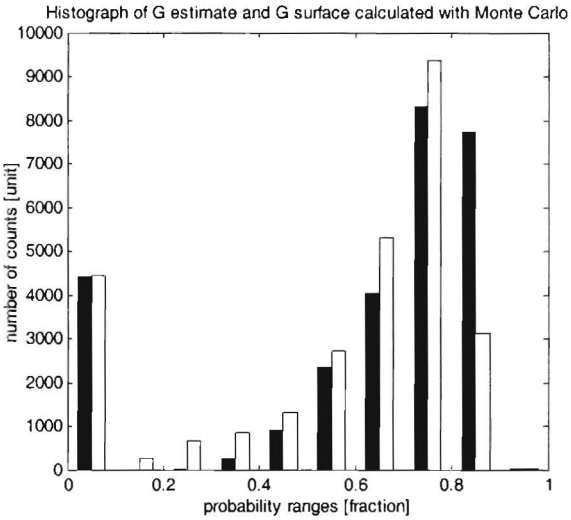


Fig. E.4 ***G* estimate and *G* calculated using Monte Carlo**

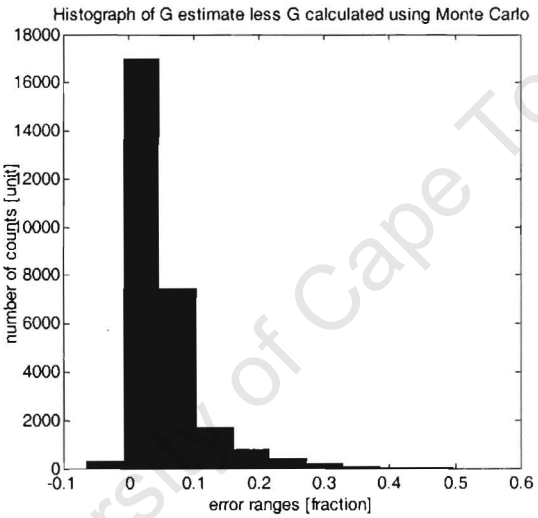


Fig. E.5 ***G* surface estimated less *G* calculated with Monte Carlo**

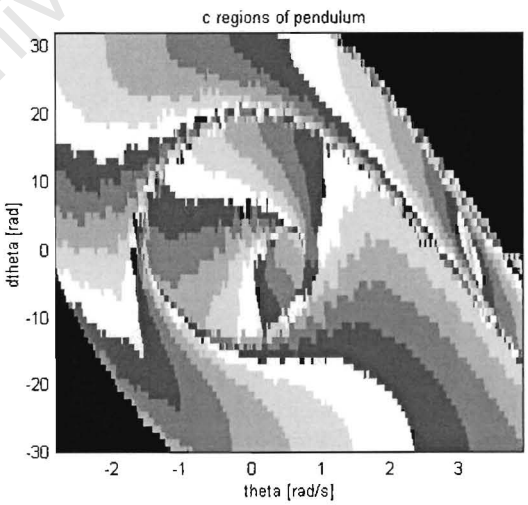


Fig. E.6 **c_i regions of model**

References

Åström K. J. and K. Furuta (1996). Swinging Up a Pendulum by Energy Control. *13th IFAC World Congress*, San Francisco, USA, http://www.control.lth.se/~kja/furutapaper.pdf

Androulakis L.P, C.D Maranas, and C.A. Floudas (1995). αBB: A Global Optimization Method for General Constrained Nonconvex Problems LP. Androulakis. *Journal of Global Optimization*, **7**(4), 337-363

Chatterjee D., A. Patraand H. K. Joglekar (2002). Swing-up and stabilization of a cart–pendulum system under restricted cart track length. *Systems & Control Letters*, **47**, 355 – 364

Didrit O., L. Jaulin and E. Walter (1997). Guaranteed Analysis and Optimization of Parametric Systems with Application to their Stability Degree. *European Journal of Control*, **3**(1), 68-80

El-Farra N. H. and P. D. Christofides (2000). Robust optimal control and estimation of constrained nonlinear processes. *Computers and Chemical Engineering* **24**, 801-807

Fung R., Y. Wang, R. Yang and H. Huang (1997). A variable structure control with proportional and integral compensations for electrohydraulic position servo control system. *Mechatronics* **7**(1), 67-81

Findeisen R. and F. Allgöwer (2002). An Introduction to Nonlinear Model Predictive Control. *21st Benelux Meeting on Systems and Control*, Veldhoven, Netherlands www.cts.sysbio.de /reports/pdf/2001-8.pdf

Gupta K., (1997). *Mechanics and control of Robots.* Springer, New York, USA

Garloff .J and E. Walter. Editors (2000). Special Issue on Applications to Control. Signals and Systems. *Reliable Computing* **6**(3), 229-362

Hickman E. and Hilton J., (1971). *Probability and Statistical Analysis.* Intext Educational Publishers, Scranton, USA

Harding S.T. and C.A. Floudas (1997). Global Optimization in Multiproduct and Multipurpose Batch Design Under Uncertainty. Industrial and Engineering Chemistry Research, **36**, 1644-1664

Hauser J. and H. Osinga (2001). On the Geometry of Optimal Control: The inverted pendulum example. *Proceedings of the American Control Conference*, **2**, 1721-1726

Henson M.A. and D.E. Seborg (1997). *Nonlinear Process Control*, Prentice Hall PTR New Jersey

Jaulin L. and E. Walter (1997). Global numerical approach to nonlinear Discrete-Time Control. *IEEE Transactions on Automatic Control* **42**(6), 872-875

Jaulin L., I. Braems, M. Kieffer and E. Walter (2002). Interval methods for nonlinear identification and robust control. *IEEE Conference on Decision and Control,* Las Vegas, USA www.istia.univangers.fr/~jaulin/publications.html

Jaulin L., S. Ratschan and L. Hardouin (2002) Set Computation for Nonlinear Control *Submitted to Reliable Computing.* http://www.istia.univangers.fr/~jaulin/ publications.html

Kwon .Y, Beom-Soo K. , L. Sang-Yup and L. Myo-Taeg (2001). Swing Up Controller for Inverted Pendulum System. *Proceedings of the 32nd International Symposium on Robotics*, Seoul, Japan, http://cml.korea.ac.kr/pdf/FB5-1.pdf

Malan S., M. Milanese and M. Taragna (1997). Robust Analysis and Design of Control Systems Using Interval Arithmetic. *Automatica* **33**(2), 1363-1372

Milam M. B., K. Mushambi and R. M. Murray (2000). A New Computational Approach to Real-Time Trajectory Generation for Constrained Mechanical Systems. *2000 Conference on Decision and Control*, Sydney, Australia 845-851

Moore R.E. (1979), *Methods and Applications of Interval Analysis.* Philadelphia, PA: SIAM

Náprstek J., (2000). A posteriori estimate of the random response of a dynamic system with autocorrelated additive noises. *Probabilistic Engineering Mechanics*, **15**(1), 73-80

Slotine J. E. and W. Li (1991) *Applied nonlinear control*, Chapter 7. Prentice-Hall, Englewood Cliffs, N.J

Tóth B. and T. Csendes (2002). Empirical investigation of the convergence speed of inclusion functions. Submitted for publication

VanAntwerp J. G., R. D. Braatz and N. V. Sahinidis (1999). Globally optimal robust process control. *Journal of Process Control* **9**(5) 375-383

Vehí .J., N. Luo, J. Rodellar and J. Armengol (2001). Digital Control via Interval Analysis. *Nonlinear Analysis* 47(1), 203-212

Yorke R., (1998), *An investigation into variable structure control*, B.Sc(Eng) Elec thesis.

Zill D.G. and Cullen (2000), *Advanced Engineering Mathematics*, Jones & Bartlett, United Kingdom, London