Chaos control using local polynomial approximation

Rory Yorke
August 2001
Department of Electrical Engineering
University of Cape Town

Prepared in partial fulfilment of the degree of M.Sc.(Eng.)

The author was supported by a scholarship from the National Research Foundation.

Thanks to friends and family for moral and material support.

Thanks to Professor Martin Braae for general advice, and to Mr Malcolm Attfield for technical assistance.

Synopsis

Introduction

Chaotic systems may be defined as those whose behaviour is sensitively dependent on initial conditions. Such systems may be made periodic using small input perturbations, as proposed in [OGY90]; this is called Ott-Grebogi-Yorke (OGY) chaos control. The original method used a linear model for controller design; a later development of chaos control was [CCdF99], in which a polynomial model is used. This dissertation proposes using local Taylor polynomial models as a basis for chaos control.

Chaos control

Chaotic systems are deterministic, but are very sensitive to initial conditions and often appear to be random. State trajectories of chaotic systems tend asymptotically to a chaotic attractor, a complex region in state space. The attractor is ergodic, which means that a single trajectory will eventually cover the entire attractor; attractors also have embedded unstable periodic orbits (UPO's), which are stabilised by chaos control.

OGY control relies on these two properties. The chaotic system is linearly modelled in the region about a UPO, and a linear feedback controller is designed from this model. The control algorithm is to do nothing until the state trajectory enters a small neighbourhood of the UPO; at this point the linear controller is activated, the UPO is stabilised and the system becomes periodic. Control may be achieved with very small perturbations, but the more constrained the input the smaller the stabilisable region and the longer the pre-stabilisation wait. Once the system is stabilised, noise may make the state vector appear to be outside the stabilisable region causing bursts of chaotic behaviour.

The location of the UPO and the associated linear model may be found by analysis of experimental data; thus OGY control is viable even if no other model of the system is available. A fairly large amount of low-noise data is required for model fitting.

Only a scalar output from the system is necessary for modelling and control if delay space reconstruction is used. In this case, the model must take into account past input values, as the mapping between delay space and state space is dependent on the system input.

Instead of a linear model, a polynomial model may be fitted. A polynomial model of form $x(i+1) = \sum_{j=0}^{k}(a_j - b_j x(i-j))x(i-j)$ is used in [CCdF99] as a global model for chaos control.

Chaotic trajectories may be targeted at the stabilisable region to shorten the pre-stabilisation wait; for a given range of input values a region of reachable states can be predicted by forward iterating a model of the chaotic map from the current state; similarly, for a given target region a range of states which will reach the target can be found by reverse-iterating the model from the target region. The input value corresponding to the intersection of these two regions is used for targeting.

The stabilisable region about the UPO may be estimated using a Lyapunov function $V(\boldsymbol{x})$ based on the linear model of the map; however, the Lyapunov difference is computed using a non-linear model of the map. The stabilisable region is estimated by finding the minimum $V_{\max} = V(\boldsymbol{x})$ such that $\Delta V(\boldsymbol{x}) \geq 0$; the corresponding level curve is the boundary of the stabilisable region.

Chaos control has been applied to a number of experimental systems by other researchers; examples are

1. stabilising 1- and 2-periodic orbits in a chaotically vibrating magneto-elastic ribbon
2. Laminarising thermally induced turbulent fluid flow
3. stabilising many different (up to 23-periodic) orbits of a chaotic diode resonator circuit
4. controlling cardiac arrhythmia in rabbit heart tissue
5. stabilising 1- and 2-periodic orbits in an chaotic electrochemical cell
6. stabilising a fixed point of a double-linked pendulum

Local polynomial models and controllers

A chaotic map may be approximated about a point $\boldsymbol{y}_*$ by $\boldsymbol{g}(\boldsymbol{y}(i)) = \boldsymbol{Z}\boldsymbol{\phi}_p(\boldsymbol{y}(i) - \boldsymbol{y}_*)$, where vector $\boldsymbol{y}$ combines state and input, matrix $\boldsymbol{Z}$ is a matrix of Taylor coefficients and ϕ_p is a polynomial evaluation function. Given a UPO $\boldsymbol{z}(j)$, a Taylor approximation can be found for each point of the UPO giving a sequence of coefficient matrices $\boldsymbol{Z}(j)$.

The UPO $\boldsymbol{z}(j)$ and associated coefficients $\boldsymbol{Z}(j)$ may either be solved for directly if $\boldsymbol{g}$ is explicitly known, or may be discovered by analysing chaotic data produced by the map.

Chaotic data yield UPO's through the method of recurrent points, in which almost periodic points ((m, δ) points) are detected in the chaotic data; each cluster of (m, δ) points corresponds to a periodic point. The coefficients associated with each periodic point are found by fitting the model to its associated cluster of (m, δ) points. The fitting occurs in two stages; in the first, the coefficients of those terms involving state components only are found from chaotic data produced when the map is unperturbed; in the second stage the remaining coefficients are found from chaotic data produced when the map is perturbed by a small random input.

A linear feedback controller can be designed from the fitted polynomial model; the controller is based only on the linear terms of the model. When designing the controller the state vector and past input values together form an augmented state space; additionally, the periodic nature of the system must be taken into account when designing the controller.

The full polynomial model can be used in an optimising controller, which chooses an input value which minimises the predicted future distance of the state vector from the UPO. Control is only attempted when the state vector is relatively close to the UPO.

Chaos control of the driven pendulum

The dimensionless driven pendulum is defined by $\ddot{w}_1 = \dot{w}_2 = \sin(w_1) - bw_2 + a\sin(\omega t)$, which has a chaotic regime. The system was controlled in simulation, and control of an experimental pendulum system was attempted.

The input was deviation to driving amplitude a and output was rotational velocity w_2; both are sampled with period $T = 2\pi/\omega$, in phase with the driving signal, to give an implicit map $\boldsymbol{g}$. In both simulation and experiment a 2-dimensional delay space was used.

In simulation linear feedback and optimising chaos controllers successfully stabilised the chosen 1-periodic UPO.

Simulation performance comparisons revealed that constraining the input value degraded both performance (in terms of mean time to stabilisation) and noise robustness. For constrained input values, a second degree polynomial model optimising controller performed better than a linear feedback controller and a linear model optimising controller.

The experimental system could not stabilised; further observation of the system revealed low noise levels and, relative to the noise, large parameter variations. When similar noise and parameter variations were applied to the simulated system, the modelling process failed, indicating that the experimental system was uncontrollable by the method described in this dissertation.

Conclusions

The following conclusions were reached:

1. Chaos control can stabilise unstable periodic orbits (UPO's) using small input perturbations.

2. Local polynomial models can be used in simulation for chaos control.

3. Chaos control performance in simulation is dependent on the constraint on system input.

4. Modelling of the driven pendulum for chaos control is very sensitive to measurement noise and parameter variation

Contents

Notation

The notation used generally follows control engineering convention:

a	scalars and scalar-valued functions
$\boldsymbol{x}$	vectors and vector-valued functions
P	sets of vectors and scalars
$\boldsymbol{A}$	Matrices

m iterations of a function g will be denoted

$$g^{\langle m\rangle}(x) = \underbrace{g \circ g \circ \cdots \circ g}_{m \text{ times}}(x) = \underbrace{g(g(\cdots g(x) \cdots))}_{m \text{ times}}$$

Although it is conventional to denote functions of discrete time as $\boldsymbol{x}_i$, the notation $\boldsymbol{x}(i)$ has been used instead; this is to prevent confusion with vector component subscripts and exponent superscripts.

Chapter 1

Introduction to chaos control

The control of chaotic systems using constrained system inputs was first proposed in [OGY90]; since then Ott-Grebogi-Yorke (OGY) and other types of chaos control have been applied to many experimental systems.

The simplest type of chaos control is to ensure that the operating point of the system is not within a chaotic regime; however, this may not always be possible. OGY control can be used when a system to be controlled is nominally chaotic, and the input is constrained so that moving out of the chaotic regime is not possible.

The result of OGY control is stabilisation of an unstable periodic orbit embedded within the chaotic attractor; this stabilisation may be done with small input perturbations. The original OGY method used a local linear model for controller design; in [CCdF99] a global polynomial model is proposed. This dissertation presents results of using a different form of polynomial model as a local approximation of the chaotic system.

This chapter touches briefly on properties of chaotic systems, and goes on to discuss OGY control and some of its developments; the chapter ends with a number of examples of published experimental applications of chaos control.

Chapter 2 explains how chaotic systems may be locally modelled using multivariate polynomials, and gives a method of fitting polynomial coefficients to experimental data from a chaotic system.

Chapter 3 presents controllers based on the models derived in chapter 2. Two types of controller, linear feedback controllers and optimising controllers, are considered.

Chapter 4 discusses application of chaos control to the driven pendulum, both in simulation and experiment; the experimental application was unsuccessful, and possible reasons for this are given.

Conclusions reached in the dissertation are in chapter 5.

Appendix A lists the main symbols used in chapters 2 to 4; this chapter uses different symbol meanings when discussing theories from literature.

Appendix B contains background information for readers unfamiliar with chaos theory. The other appendices contain details of various points of discussion in the dissertation.

1.1 Chaotic systems

A chaotic system may be defined as one whose behaviour is very sensitive to changes in initial condition [Lor93]. More precisely, if a chaotic system is started from a particular state, the behaviour observed would be quite different if the system were started from an infinitesimally different state.

Chaotic behaviour is deterministic; it does not occur because of random effects such as noise or parametric uncertainty. The combination of chaos and randomness is long term unpredictability; since system behaviour is sensitive to small changes in initial condition, if this initial condition is imprecisely known (e.g. because of measurement noise) a wide range of behaviours is possible.

The relevance of chaos to engineering and science is still a matter of debate [Wil97]. However, there are a number of examples of chaos occurring in physical systems (as opposed to in simulations); section 1.3 has some examples.

1.2 Chaos control

The term "chaos control" is usually reserved for the Ott-Grebogi-Yorke (OGY) control method, first proposed in [OGY90], and its derivatives. The main features of OGY are that chaotic systems become periodic, and that only small control action is necessary to achieve control.

1.2.1 Ott-Grebogi-Yorke chaos control

OGY operates on dynamical maps; it can be applied to continuous time systems through Poincaré section (see section B.2.3) or other sampling methods.

OGY stabilises unstable periodic orbits (UPO's) embedded in the chaotic attractor; according to [OGY90] chaotic attractors typically have an infinite number of embedded UPO's.

UPO's and a linear model of the map about each UPO may be found by analysis of experimental data; OGY uses a linear model for controller design. More detail on the modelling procedure is given in chapter 2.

Once a suitable UPO (in terms of desired system behaviour) has been found, a linear controller can be designed to stabilise it. The stabilisation is only local because the map is non-linear (since it is chaotic).

In [OGY90], the controllers are designed by a method equivalent to pole-placement; the unstable poles of the linear model are moved to the origin and the stable poles left where they are [GL97].

The linear control law only gives local stability in some neighbourhood (in state space) of the UPO. However, chaotic attractors are ergodic, and if the system is run without any control action for long enough the state vector will eventually enter the stabilisable neighbourhood.

The OGY control algorithm is thus to run the system chaotically (i.e. without control action) until the state vector is in the stabilisable region, and then to use the linear controller to stabilise the UPO.

Some features of OGY are (from [OGY90] and [Kap98]):

1. very small control action will stabilise the system
2. different UPO's can be stabilised for the same range of input values
3. it is not necessary to fully characterise the system (i.e. obtain a good but possibly complex model) to design the control law

Two limitations of OGY are:

1. there may be a long waiting transient before the system stabilises
2. noise can destabilise the system causing bursts of chaotic behaviour

1.2.2 Chaos control in delay space

In [DN92], a modification to OGY is given which is necessary for using chaos control in delay space. (Delay space is discussed in sections B.1.6 and B.2.5.) The modification involves explicitly modelling the effect of past input values, as explained below.

There exists a mapping $\mathbf{\Phi}$ from normal state space to delay space. Elements of normal space will be denoted in this section by $\boldsymbol{x}(i)$, and those of delay space by $\boldsymbol{y}(i)$. In [DN92] it is claimed that for appropriately chosen delay dimension d and lag time τ, $\mathbf{\Phi}$ is bijective. This is a stronger claim than made elsewhere (e.g. [Wil97]); however, we will see that $\mathbf{\Phi}$ must be bijective in at least the neighbourhood of the target UPO for chaos control in delay space to be viable, and it will therefore be assumed that $\mathbf{\Phi}$ is bijective.

The mapping $\mathbf{\Phi}$ is dependent on the dynamical system, and in particular on input values u for the time interval in which the delay vector is obtained

[SO95]. For a discrete time system in which $\boldsymbol{y}(i) = (\boldsymbol{x}_k(i), \boldsymbol{x}_k(i-1), \ldots \boldsymbol{x}_k(i-d)$ the mapping is dependent on input values $u(i)$ back to an input history $d_h = d$.

When working in continuous time, the input is only changed at sampling instants, thus a similar input history concept may be employed. In this case the mapping is dependent on all values of input from current time t to $t - d\tau$, and d_h will be the smallest natural number such that $t - d_h\tau \leq t - d\tau$.

Let $\boldsymbol{v}(i) = (u(i), u(i-1), \ldots, u(i-d_h))$; $\boldsymbol{\Phi}$ and $\boldsymbol{\Phi}^{-1}$ are then parametrised by $\boldsymbol{v}(i)$, denoted by $\boldsymbol{\Phi}_{\boldsymbol{v}(i)}$ and $\boldsymbol{\Phi}^{-1}_{\boldsymbol{v}(i)}$.

We can now express the delay space map as

$$\boldsymbol{y}(i+1) = \boldsymbol{\Phi}_{\boldsymbol{v}(i+1)} \circ \boldsymbol{g}(\boldsymbol{\Phi}^{-1}_{\boldsymbol{v}(i)}(\boldsymbol{y}(i)), u(i)) \tag{1.1}$$

This map has explicit dependence on the past input values. Notice also that if $\boldsymbol{\Phi}$ is not injective (and thus not bijective), i.e. if there is more than one $\boldsymbol{x}$ such that $\boldsymbol{y} = \boldsymbol{\Phi}_{\boldsymbol{v}}(\boldsymbol{x})$, then the map is uncontrollable in delay space since the next value cannot be predicted. Bijectivity is only required in the neighbourhood of the UPO, since it is only then that control will be applied.

It follows from Eq. (1.1) that the system model must include the effect of $\boldsymbol{v}(i)$ for chaos control in delay space. See sections 2.3.3 and 3.1 for details.

1.2.3 Chaos control with polynomials

In [CCdF99] a polynomial model was used to control a chaotic system. The system input was not considered to be constrained, and the goal was to find a global model for controller design.

The model uses a delay vector $\boldsymbol{x}(i)$ of dimension n where $x_k(i) = x_{k-1}(i)$ for $k > 1$ and $x_1(i)$ is the current scalar output of the system. The model is of form

$$\boldsymbol{x}(i+1) = \boldsymbol{A}\boldsymbol{x}(i) + \boldsymbol{f}(\boldsymbol{x}(i)) + \boldsymbol{c}u(i) \tag{1.2}$$

where

$$\boldsymbol{A} = \begin{pmatrix} a_1 & a_2 & & a_{n-1} & a_n \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & & 0 & 0 \\ & \vdots & \ddots & \vdots & \\ 0 & 0 & \cdots & 1 & 0 \end{pmatrix}, \quad \boldsymbol{c} = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

and

$$\begin{aligned} f_1(\boldsymbol{x}) &= b_1 x_1^2 + b_2 x_2^2 + \ldots + b_n x_n^2 \\ f_k(\boldsymbol{x}) &= 0 \text{ for } k > 1 \end{aligned}$$

Given a setpoint $\boldsymbol{y}(i)$, it is simpler to work in an error space $\boldsymbol{e}(i) = \boldsymbol{x}(i) - \boldsymbol{y}(i)$. Substituting this into Eq. (1.2) and rearranging gives

$$\boldsymbol{e}(i+1) = (\boldsymbol{A}+\boldsymbol{B}(i))\boldsymbol{e}(i)+\boldsymbol{A}\boldsymbol{y}(i)-\boldsymbol{y}(i+1)+\boldsymbol{f}(\boldsymbol{e}(i))+\boldsymbol{f}(\boldsymbol{y}(i))+\boldsymbol{c}u(i) \quad (1.3)$$

where

$$\boldsymbol{B}(i) = \begin{pmatrix} 2b_1y_1(i) & \dots & 2b_ny_n(i) \\ 0 & & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{pmatrix}$$

The controller is $u(i) = -f_1(\boldsymbol{e}(i))-f_1(\boldsymbol{y}(i))-\boldsymbol{k}\boldsymbol{e}(i)+y_1(i+1)-\sum_{j=1}^{n} a_jy_j(i)$, giving $\boldsymbol{e}(i+1) = (\boldsymbol{A}+\boldsymbol{B}(i)-\boldsymbol{c}\boldsymbol{k})\boldsymbol{e}(i)$, which is a linear time-varying system.

The setpoint $\boldsymbol{y}(i)$ is assumed to be constant or periodic, and the elements of $\boldsymbol{k}$ are found using a Lyapunov function approach or other methods.

Some unusual features of [CCdF99] are that the delay dimension was particularly high ($n = 25$) despite the relatively low dimension of the two chaotic systems considered (3 in both cases); the delay dimension was chosen from how well the model fitted data for different n. The model does not have any cross-terms (e.g. x_1x_2), nor does it take into account the effect of past input values as discussed in section 1.2.2.

The method was tested in simulation on Duffing's oscillator and successfully stabilised both a fixed point and a 1-period cycle.

1.2.4 Other developments of chaos control

This section discusses two chaos control extensions which are not of direct relevance to the dissertation but considered by the author to be of interest.

Targeting chaotic trajectories

Chaotic trajectories can be directed to targets using constrained inputs [SOGY90, KGOY93]. In the case of chaos control such a target would be the stabilisable region about a UPO.

A trajectory starting from a given initial condition will eventually enter any chosen target region on the chaotic attractor. However, the number of iterations required for this to occur may be long; targeting can reduce the number of iterations needed. In [SOGY90] a trajectory which would otherwise reach its goal in about 6000 iterations does so in 12 with targeting, and in [KGOY93] a target is reached in 35 iterations instead of an estimated 10^{11}.

The papers cited employ slightly different methods, but both rely on the controller being able to predict system behaviour by iterating the chaotic map forwards and backwards in time. The method described below is from [SOGY90].

Given a two-dimensional chaotic map $\boldsymbol{g}(\boldsymbol{x}(i), u(i))$, $|u(i)| < u_{\max}$ for small $u_{\max}$ the possible values of $\boldsymbol{x}(i+1)$ will lie on a small almost straight line segment containing and roughly centred on $\boldsymbol{g}(\boldsymbol{x}(i), 0)$. This segment is iterated forward in time through $\boldsymbol{g}$ for p iterations, at which time its length is of the same order as the attractor size. The number of forward iterations p is estimated from the positive Lyapunov exponent of the map.

The target region is similarly iterated backwards, also growing in size until one of the reverse images intersects with the forward iterated image; the $u(i)$ value corresponding to the intersection is the value which will direct the trajectory to the target.

Targeting was found to be robust to small noise and modelling errors if the control value was recomputed at every iteration of the process [SOGY90].

Although [SOGY90] claims the model used for forward and backward iteration need not be exact, and could in principle be derived from data, it is clear that a simple linear model would be insufficient; given the nature of chaotic systems (nearby trajectories diverging exponentially) it is likely that quite good models would be needed for targeting.

Estimating the stabilisable region

OGY control guarantees stability for a small neighbourhood about the UPO and for small input values; [Vin97] gives a method to estimate the stabilisable region. In [Vin97] both continuous and discrete time cases with multiple inputs are presented, but here only the discrete time single input case will be discussed.

As in OGY it is assumed that the input is constrained and a linear model $\boldsymbol{x}(i+1) = \boldsymbol{A}\boldsymbol{x}(i) + \boldsymbol{B}u(i)$ is used to design a controller $u(i) = -\boldsymbol{K}\boldsymbol{x}(i)$ (in [Vin97] DLQR is used to design $\boldsymbol{K}$).

A Lyapunov function $V(\boldsymbol{x}) = \boldsymbol{x}\boldsymbol{P}\boldsymbol{x}'$ is constructed from the controlled linear system. $\boldsymbol{P}$ is the solution to the discrete Lyapunov equation $\boldsymbol{P} = \boldsymbol{M}'\boldsymbol{P}\boldsymbol{M} + \boldsymbol{L}$, $\boldsymbol{M} = \boldsymbol{A} - \boldsymbol{B}\boldsymbol{K}$ and $\boldsymbol{L}$ a positive definite matrix.

This Lyapunov function may be used with the original non-linear map $\boldsymbol{f}(\boldsymbol{x}(i), \boldsymbol{u}(i))$; if we let $\boldsymbol{h}(\boldsymbol{x}(i)) = \boldsymbol{f}(\boldsymbol{x}(i), -\boldsymbol{K}\boldsymbol{x}(i))$, the system is stabilisable for a region defined by $V(\boldsymbol{x}) \leq V_{\max}$ if, for the entire region,

$$\Delta V(\boldsymbol{x}, m) = V(\boldsymbol{h}^{(m)}(\boldsymbol{x}(i))) - V(\boldsymbol{x}) < 0$$

The number of iterations m is used to predict further into the future and find larger stabilisable regions than for single iterations; in this sense it is similar to the targeting method previously discussed.

The parameter defining the region, $V_{\max}$, may be maximised by finding the minimum value $V(\boldsymbol{x})$ such that $\Delta V(\boldsymbol{x}, m) \geq 0$; [Vin97] notes that this optimisation is in general non-trivial.

The final control law is of the form

$$u(i) = \begin{cases} -\boldsymbol{K}\boldsymbol{x}(i) & V(\boldsymbol{x}) \leq V_{\max} \\ 0 & \text{otherwise} \end{cases}$$

This approach has been used with the Hénon map in simulation and with a bouncing ball and a double pendulum system in experiment; the double pendulum system is further discussed in section 1.3.6.

The estimation of $V_{\max}$ requires a non-linear model (the linear model would be stable up to input constraints); it is not clear how sensitive $V_{\max}$ is to model inaccuracies.

1.3 Applications of chaos control

Chaos control has been successfully applied in a number of fields; this section includes examples from mechanics, electronics, chemistry and biology. All the examples are experimental, none having direct practical application.

1.3.1 Chaos control of a magneto-elastic ribbon

The first published application of chaos control used OGY and was to a magneto-elastic ribbon [DRS90]. The elasticity (Young's modulus) of magneto-elastic material is sensitive to small changes in magnetic field H.

In the experiment, the ribbon was mounted vertically and clamped at its base, and a vertical magnetic field $H = a + b\cos(2\pi f t)$ was applied to it; the ribbon buckled because of gravity, and moved because of changing elasticity.

The system input was deviation Δa in the mean strength a of the magnetic field. The system output was the ribbon's curvature (an indication of position), which was measured by a sensor at the ribbon's base; the output was sampled at frequency f.

For appropriate a, b and f, the curvature changed chaotically. Within such a chaotic regime, Ditto et al. were able to stabilise 1- and 2-periodic UPO's, found by data analysis. Control was achieved with $|\Delta a| < 0.01$ Oe, $a = 0.112$ Oe, $b = 2.050$ Oe and $f = 0.85$ Hz. (Oersteds (Oe) are a measurement of magnetic field strength [The73].)

The experiment had relatively low noise; the standard deviation was about 0.005 V over an output range of 2 V.

1.3.2 Laminarising of fluid flow

The second published application of chaos control was to a thermal convection loop and did not use OGY [SWB91].

In this experiment a thin pipe filled with water was joined end-to-end to form a torus which was stood in the vertical plane. The lower half of the torus was uniformly heated while the upper half was kept at a constant, uniform cool temperature; the temperature gradient induced fluid motion. The system output was the temperature difference between positions at 6 and 12 o'clock on the torus (to use the terminology of [SWB91]), and the system input was the heating rate.

For low heating rates, below 190 W, the fluid flow was steady and unidirectional. Above 190 W the system entered a chaotic regime, and the fluid flow was chaotic, in particular chaotically reversing direction. The flow became once more relatively smooth and undirectional when under control.

The control algorithm used was on-off: if the output exceeded a prechosen value, the heating rate (nominally at 600 W) was changed to 625 W, and if it went below this value, the heating rate was set to 575 W; these values are well within the chaotic regime.

The controller successfully rejected test disturbances; details of the disturbances and of noise and the method of choosing controller parameters were not given in [SWB91].

1.3.3 Stabilising a diode resonator

Hunt used an OGY based technique to control a diode resonator [Hun91]. The diode resonator circuit comprised a diode, inductor and voltage source connected in series; the voltage source produced a sinusoidal signal. The diode used was a 1N2858; the inductor was 100 mH with 25 Ω direct current (DC) resistance and the driving signal had a frequency of 53 kHz (the signal amplitude was not given in the paper).

The system output was the current, sampled at local maxima (presumably using peak detection, although this is not indicated in [Hun91]) and the input was deviation to the driving amplitude.

The control method was effectively OGY, but UPO's were not found by data analysis. The controller had three adjustable parameters, namely offset (corresponding to UPO position), a window about the offset in which control was activated, and feedback gain. The UPO's were found by experimentally scanning through the settings; in this way UPO's for most periods between 1 and 23 were found.

The scanning method was practical because the system was fast (about

0.2 ms between current peaks); [Hun91] notes that there was no systematic method for finding UPO's of a particular period, although each UPO, once found, could be reproduced by reproducing the settings recorded when the UPO was discovered.

To stabilise low period UPO's, only small changes in the amplitude on the order of 0.5 % were needed; for the higher periods up to a 10 % change was required. The controller was also robust to input disturbances; changes of 10 % to the driving amplitude did not destabilise the system in the low periodic case.

1.3.4 Controlling cardiac arrhythmia

Control of chaotic arrhythmia in rabbit heart tissue is described in [GSDW92].

The experimental apparatus was a section of rabbit heart tissue, periodically stimulated through electrodes. This stimulus caused spontaneous electrical activity in the tissue, recordable as beats. The system input was a shortening in the pulse interval, and the system output was the time between beats (the inter-beat interval). The pulse interval could only be shortened because the effect of pulse lengthening was very sensitive to external factors.

Chaos was chemically induced in the system, which would cease beating several minutes after the onset of chaos.

The control method used a 2-dimensional delay space comprising the current and last inter-beat intervals.

The controller operated in two phases. First was a learning phase, of between 5 s and 60 s, during which the controller detected and characterised a 1-periodic UPO. The characterisation was simplified by the assumption that a shortening of the pulse interval would result in an identical shortening of the inter-beat interval.

After this learning phase was a control phase, during which chaos control was applied. The chaos control algorithm used was based on OGY.

Despite finding 1-periodic UPO's in the learning phase, the controller typically stabilised 3-periodic orbits. If $\boldsymbol{x}(i)$ was inside the controllable region, the controller would set $u(i)$ to place $\boldsymbol{x}(i+1)$ on the stable manifold of the UPO. However, because of model inaccuracy, $\boldsymbol{x}(i+1)$ would typically be near but not on the stable manifold. The next point $\boldsymbol{x}(i+2)$ would typically land outside the controllable region, but still be near the stable manifold, and $\boldsymbol{x}(i+3)$ would typically be near $\boldsymbol{x}(i+1)$, restarting the 3-cycle.

It is unclear why the point $\boldsymbol{x}(i+2)$ ended up outside the controllable region; it may be that the delay space adaptation of [DN92], not used, would have solved this problem.

Chaos control was successfully applied in 8 out of 11 experimental runs.

The arrhythmia studied does not have direct medical relevance; however, the authors speculated that if other types of cardiac arrhythmia are also chaotic, advanced pacemakers might be able to employ chaos control to stabilise the heart when such arrhythmias occur.

1.3.5 Electrochemical chaos control

Chaos control of an electrochemical system is described in [KGNP97], following a similar experiment in [PSRD93].

The apparatus was an electrochemical cell containing ortho-phosphoric acid with a copper rotating disk anode, a platinum cathode and a reference electrode. The system input was change in the nominal voltage between the anode and cathode, and the system output was the current through the cell; the system was sampled at 200 Hz. For a disk speed of 1800 rpm and a nominal voltage of 532 mV the system was chaotic.

A straight-line Poincaré section was used in a two-dimensional delay space to give a one-dimensional chaotic map (for the delay space, $\tau = 0.5$ s).

1- and 2-periodic UPO's (found by data analysis) were successfully stabilised using a variant of OGY for one-dimensional maps, with absolute maximum input set to 0.5 mV or about 0.1% of the nominal voltage. In the 1-periodic case, the absolute input voltage settled down to less than 0.1 mV after the UPO was stabilised, but in the 2-periodic the input voltage varied in a 0.5 mV range. This was probably because the input value was only changed every second iteration of the map (according to [KGNP97]), which this author speculates made the control less robust. The approach described in [SO95] or section 3.1, in which the input value is computed every map iteration even for multi-periodic orbits, might have helped in this situation.

Control failed after many hours of experimentation because of significant changes in system dynamics, although it is not clear how large a parameter variation was involved.

1.3.6 Control of a double linked pendulum

The control of a double linked pendulum is described in [Vin97].

A double pendulum is a normal pendulum with a second arm attached to the bottom of the first, as shown in Fig. 1.1.

In the experimental setup described in [Vin97], both joints were driven by DC motors and the position and angular velocity of both arms was directly measured. The system input was the driving voltage to the two motors, and the output was all four state variables.

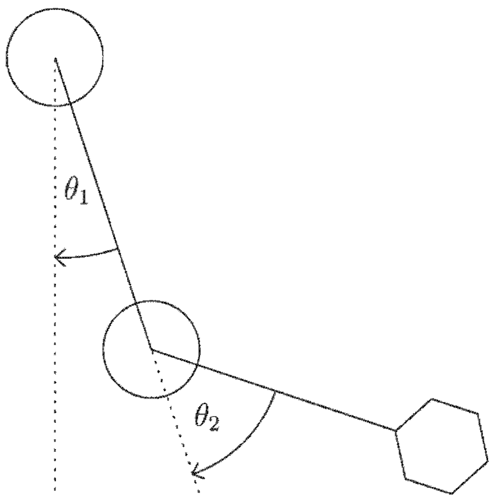


Figure 1.1: The double pendulum

The control target was to have the system at rest with the first arm pointing downwards ($\theta_1 = 0$) and the second arm pointing upwards ($\theta_2 = \pi$); this is different from the other examples in this section since this is not stabilisation of a UPO but of an unstable fixed point.

The motors used were quite weak; from a downward rest position the motors could only hold the joint angles at less than 20 ° at full torque. When the first motor provided an appropriate sinusoidally varying torque, the system behaved chaotically: in this regime, the second pendulum swung right around its axis, although the first did not.

The controller was designed by analysis of a differential equation model of the system. The system was linearised about the target point and from this a linear controller was synthesised by LQR; the stabilisable neighbourhood of the target point with this controller was conservatively estimated using a Lyapunov function. Control was only applied when the state vector was within the controllable region.

The system was deliberately operated in a chaotic regime so that the state vector came close enough to the target point. The chaotic regime chosen was safe; while the method could also target the position of both arms upright ($\theta_1 = \pi$, $\theta_2 = 0$), the (different) chaotic regime which could have reached this fixed point would have caused very high rotational velocities in the second pendulum.

Open-loop optimal control could have achieved the same goal, but [Vin97] remarks that chaos control was more robust. If, after reaching the target, the second arm were bumped, the optimal controller would have to restart from a pre-chosen initial condition (e.g. both arms at rest pointing downwards); the chaos controller would return to chaotic operation until the stabilisable region were entered again.

Chapter 2

Modelling systems for chaos control

This chapter discusses how to obtain a local polynomial model of a chaotic system, with a view to stabilizing an unstable periodic orbit (UPO). The discussion focuses on a chaotic map $\boldsymbol{g}$; it is assumed that this is a map in delay space. $\boldsymbol{g}$ is implicit, and represents either a normal (i.e. non-delay) space map, or a continuous time system through Poincaré or stroboscopic section.

Chaos control stabilises unstable periodic orbits embedded in a system's chaotic attractor; with this in mind, the models used for chaos control contain the location of a target UPO and an approximation of the system's behaviour near the UPO.

Following the use of (non-Taylor) polynomials in [CCdF99], Taylor polynomials were chosen as the approximations to be studied. The model for a particular m-periodic UPO will thus consist of a set of m points comprising the UPO, and the m corresponding Taylor approximations.

The main disadvantages of using a polynomial model are that such models are more complex, making controller implementation more expensive, and that the model fitting is more difficult, since there are more coefficients to be fitted than in the linear case.

Appendix A has a list of symbols used in chapters 2 to 4.

2.1 Taylor polynomials of vector functions

Taylor polynomials can be used to approximate scalar-valued vector functions, and a set of such polynomials contained in a vector function may be used to approximate a vector-valued function.

The pth degree Taylor polynomial approximating a scalar-valued function $f : \mathbb{R}^q \to \mathbb{R}$ about a point $\boldsymbol{y}_*$ is (from [Bra58])

$$f(\boldsymbol{y}_* + \Delta \boldsymbol{y}) \approx \sum_{k=0}^{p} \frac{1}{k!} \left(\sum_{i=1}^{q} (\Delta y_i) \frac{\partial}{\partial y_i} \right)^k f \Bigg|_{\boldsymbol{y}=\boldsymbol{y}_*} \tag{2.1}$$

which is a generalisation of Taylor polynomials in a single variable.

Higher degree polynomials usually give better approximations over a larger area of the function's domain than a lower order polynomial. Polynomials cannot always give a global fit; consider any finite-order Taylor polynomial of $\cos(x)$. For this reason the polynomial models are considered to be local.

The number of coefficients needed to represent an multivariate polynomial in q variables of degree p is $r = \binom{q+p}{p}$. It is impractical to deal with a large number of coefficients; because of the combinatorial growth of r the author has not gone beyond degree 2 for chaos control.

A particular Taylor polynomial can be expressed as the dot product of two vectors. One is a vector of coefficients and the other a "polynomial evaluation vector" of degree p, which has form

$$\boldsymbol{\phi}_p(\boldsymbol{y}) = (1, y_1, y_2, \ldots, y_q, y_1^2, y_1 y_2, \ldots, y_q^2, \ldots, y_1^p, \ldots, y_q^p) \tag{2.2}$$

Both vectors are of length r; $\boldsymbol{\phi}_p$ is taken to be a column vector.

The Taylor coefficients of an vector function $\boldsymbol{f} : \mathbb{R}^q \to \mathbb{R}^n$ about a point $\boldsymbol{y}_*$ can be contained in an $n \times r$ matrix $\boldsymbol{Z}$, giving a complete approximation

$$\boldsymbol{f}(\boldsymbol{y}) \approx \boldsymbol{Z}\boldsymbol{\phi}_p(\boldsymbol{y} - \boldsymbol{y}_*) \tag{2.3}$$

Taylor approximations of chaotic maps

To approximate a chaotic map in the general delay space case, we will identify delay space dynamical map $\boldsymbol{g}(\boldsymbol{x}(i), \boldsymbol{v}(i))$ with $\boldsymbol{f}(\boldsymbol{y}(i))$. The delay input vector $\boldsymbol{v}(i)$ is $\boldsymbol{v}(i) = (u(i), u(i-1), \ldots, u(i-d_h))$, where d_h is the input history. From [SO95], d_h is the smallest natural number such that any value in a delay state vector $\boldsymbol{x}(i)$ is measured at the same time or later than $i - d_h$. It is also possible for $\boldsymbol{g}$ to be a normal space map, in which case $d_h = 0$ and $\boldsymbol{v}(i) = u(i)$.

The identification means that $\boldsymbol{y}(i)$ is the concatenation of $\boldsymbol{x}(i)$ and $\boldsymbol{v}(i)$, i.e. $y_k(i) = x_k(i)$ for $1 \leq k \leq n$ and $y_k = v_{k-n}(i)$ for $n+1 \leq k \leq r$, where $r = n + d_h + 1$. For a given m-periodic UPO $\boldsymbol{z}(j)$ there will be a sequence of coefficient matrices $\boldsymbol{Z}(j)$.

The coefficients of the Taylor polynomial may be obtained by direct evaluation of Eq. (2.1) if the defining equations are known, or they may be fitted by analysing data produced by the chaotic system.

2.2 Symbolic modelling for chaos control

Symbolic analysis is conceptually simple, but does require that the equations defining the chaotic system be known; this knowledge may be difficult to obtain for a physical system [OGY90].

All points of an m-periodic embedded UPO will satisfy $\boldsymbol{x}_* = \boldsymbol{g}^{\langle m \rangle}(\boldsymbol{x}_*, 0)$, and solving this equation for $\boldsymbol{x}_*$ will reveal such an orbit (if one exists). Solving for a UPO may be an extremely difficult problem, possibly doable only by numerical means or not at all; since the entire symbolic analysis is somewhat idealised, this complication will be ignored.

A chosen embedded m-periodic UPO, if one exists, will be denoted by $\boldsymbol{z}(j)$ where $\boldsymbol{z}(j+1) = \boldsymbol{g}(\boldsymbol{z}(j), 0)$ and $\boldsymbol{z}(j+m) = \boldsymbol{z}(j)$ for all j. The Taylor coefficients $\boldsymbol{Z}(j)$ for each point of $\boldsymbol{z}(j)$ are found by differentiation, from Eq. (2.1).

When considering a continuous time system, all differentiations must be approximated by finite differences.

Examples of symbolic analysis of the Hénon and driven pendulum systems are given in appendix C.

When the equations of the system are not available, the models must be found by analysis of the data produced by the system.

2.3 Modelling from data for chaos control

If the equations of the system are unknown, the Taylor coefficients must be fitted to the data; the result will obviously be less accurate than the ideal symbolic case. While more practical than the ideal case, the fitting does require a large quantity of relatively low-noise data.

2.3.1 Finding embedded UPO's by data analysis

The method of recurrent points [LK89], described below, reveals UPO's embedded in the attractor. The method operates on a vector sequence $\boldsymbol{x}_u(i)$ generated by a chaotic map $\boldsymbol{g}$ such that $\boldsymbol{x}_u(i+1) = \boldsymbol{g}(\boldsymbol{x}_u(i), \boldsymbol{v}(i))$ where $u(i) = 0$ for all i. (The u subscript denotes an unperturbed sequence.)

Points in the neighbourhood of an m-periodic point $\boldsymbol{x}_*$ of a map $\boldsymbol{g}$ will be almost m-periodic; that is, if $\|\boldsymbol{x}_* - \boldsymbol{x}\|_2$ is small then $\left\|\boldsymbol{g}^{\langle m\rangle}(\boldsymbol{x}) - \boldsymbol{x}\right\|_2$ will also be small.

The method of recurrent points assumes the converse, that points which are almost m-periodic are near m-periodic points. The (m, δ) data points are all $\boldsymbol{x}_u(i)$ for which $\|\boldsymbol{x}_u(i+m) - \boldsymbol{x}_u(i)\|_2 < \delta$. For small δ it is assumed that the (m, δ) data points will be grouped around m-periodic points, including those points comprising m-periodic UPO's.

The (m, δ) data points are divided into clusters (for details of the clustering method, see appendix F) and the mean of the points in each cluster is taken to be the corresponding m-periodic point; the set of m-periodic points thus found will be denoted $\boldsymbol{a}(j)$.

2.3.2 Characterising embedded UPO's by data analysis

For a particular m-periodic point, Taylor coefficients may be found from the behaviour of its associated cluster of data points. Initially we will only find coefficients related to the state vector; in other words the function $\boldsymbol{f}(\boldsymbol{y}(i))$ is identified with $\boldsymbol{g}(\boldsymbol{x}(i), \boldsymbol{0})$ and coefficients for $\boldsymbol{v}(i)$ will not be computed. This analysis uses the data $\boldsymbol{x}_u(i)$ and clusters of (m, δ) data points of the previous section.

For a particular m-periodic point $\boldsymbol{a}(j_*)$, let $\boldsymbol{\alpha}(k)$ $(k = 1, \ldots, q)$ be the (m, δ) points from $\boldsymbol{x}_u(i)$ clustered about $\boldsymbol{a}(j_*)$ recentred about $\boldsymbol{0}$ (i.e. offset by $-\boldsymbol{a}(j_*)$). Let $\boldsymbol{\beta}(k)$ be the next-points of the $\boldsymbol{\alpha}(k)$ in $\boldsymbol{x}_u(i)$, that is if i_* corresponds to k_*, then $\boldsymbol{\alpha}(k_*) = \boldsymbol{x}_u(i_*) - \boldsymbol{a}(j_*)$ and $\boldsymbol{\beta}(k_*) = \boldsymbol{x}(i_* + 1)$; or, equivalently, $\boldsymbol{\beta}(k) = \boldsymbol{g}(\boldsymbol{\alpha}(k) + \boldsymbol{a}(j_*), \boldsymbol{0})$.

The Taylor coefficients $\boldsymbol{Y}(j_*)$ of $\boldsymbol{g}$ about $\boldsymbol{a}(j_*)$ can be fitted by least squares; from Eq. (2.3) we have

$$\begin{pmatrix} \boldsymbol{\phi}_p^T(\boldsymbol{\alpha}(1)) \\ \boldsymbol{\phi}_p^T(\boldsymbol{\alpha}(2)) \\ \vdots \\ \boldsymbol{\phi}_p^T(\boldsymbol{\alpha}(q)) \end{pmatrix} \boldsymbol{Y}^T(j_*) = \begin{pmatrix} \boldsymbol{\beta}^T(1) \\ \boldsymbol{\beta}^T(2) \\ \vdots \\ \boldsymbol{\beta}^T(q) \end{pmatrix} \tag{2.4}$$

in which $\boldsymbol{Y}(j_*)$ replaces $\boldsymbol{Z}$ used in Eq (2.3); $\boldsymbol{Y}(j_*)$ is a part of the complete matrix of coefficients $\boldsymbol{Z}(j_*)$. Eq. (2.4) can be solved for $\boldsymbol{Y}(j_*)$ by the usual least squares method [PTVF92]. This is done for each m-periodic point $\boldsymbol{a}(j)$, giving a complete set of coefficients $\boldsymbol{Y}(j)$.

The zeroth order coefficients, that is the first column of each $\boldsymbol{Y}(j)$, are the images of $\boldsymbol{a}(j)$ under $\boldsymbol{g}$; by matching these up with the points of $\boldsymbol{a}(j)$

m-periodic UPO's can be identified.

A chosen m-periodic orbit will be labelled $\boldsymbol{z}(j)$, and $\boldsymbol{Y}(j)$ is redefined (or rather, relabelled) to be the corresponding matrices of Taylor coefficients.

2.3.3 Input characterisation by data analysis

The model is completed by finding coefficients of terms involving $\boldsymbol{v}$.

Another data series $\boldsymbol{x}_p(i) = \boldsymbol{g}(\boldsymbol{x}_p(i), \boldsymbol{v}(i))$ must be generated, but this time $u(i)$ is a small random perturbation signal.

As in section 2.3.1, clusters of (m, δ) data points are found in from $\boldsymbol{x}_p(i)$; each point of the UPO must have a corresponding cluster for coefficients to be fitted.

For a point $\boldsymbol{z}(j_*)$ of the UPO, let $\boldsymbol{\alpha}(k)$ be the corresponding cluster of (m, δ) points from $\boldsymbol{x}_p(i)$, offset by $-\boldsymbol{z}(j_*)$. Let $\boldsymbol{\beta}(k)$ and $\boldsymbol{w}(k)$ be the corresponding next points from $\boldsymbol{x}_p(i)$ and next input values from $\boldsymbol{v}(i)$ respectively.

A subset of the complete matrix of Taylor coefficients, $\boldsymbol{Z}(j_*)$, has already been computed and is contained in $\boldsymbol{Y}(j_*)$. What remains to be computed, $\boldsymbol{X}(j_*)$, are coefficients of all terms involving elements of $\boldsymbol{v}$. Let $\boldsymbol{\psi}_p(\boldsymbol{v}, \boldsymbol{x})$ be a function generating a polynomial evaluation vector of all terms involving $\boldsymbol{v}$; $\boldsymbol{\psi}_p(\boldsymbol{v}, \boldsymbol{x})$ is $\boldsymbol{\phi}_p(\boldsymbol{y})$ with all the components in $\boldsymbol{\phi}_p(\boldsymbol{x})$ removed.

The problem may now be expressed as

$$\begin{pmatrix} \boldsymbol{\psi}_p^T(\boldsymbol{w}(1), \boldsymbol{\alpha}(1)) \\ \boldsymbol{\psi}_p^T(\boldsymbol{w}(2), \boldsymbol{\alpha}(2)) \\ \vdots \\ \boldsymbol{\psi}_p^T(\boldsymbol{w}(q), \boldsymbol{\alpha}(q)) \end{pmatrix} \boldsymbol{X}^T(j_*) = \begin{pmatrix} \boldsymbol{\beta}^T(1) \\ \boldsymbol{\beta}^T(2) \\ \vdots \\ \boldsymbol{\beta}^T(q) \end{pmatrix} - \begin{pmatrix} \boldsymbol{\phi}_p^T(\boldsymbol{\alpha}(1)) \\ \boldsymbol{\phi}_p^T(\boldsymbol{\alpha}(2)) \\ \vdots \\ \boldsymbol{\phi}_p^T(\boldsymbol{\alpha}(q)) \end{pmatrix} \boldsymbol{Y}^T(j_*) \tag{2.5}$$

which can again be solved using least squares. $\boldsymbol{X}(j_*)$ is computed for each point on the UPO.

The coefficients contained in $\boldsymbol{X}(j)$ and $\boldsymbol{Y}(j)$ can now be combined to form the complete set of Taylor coefficient matrices $\boldsymbol{Z}(j)$. The combination requires correctly interspersing the columns from $\boldsymbol{X}(j)$ and $\boldsymbol{Y}(j)$; for details see the source code on the attached CD-ROM.

Chapter 3

Design of chaos controllers

Two types of controller are presented: linear feedback controllers (LFC's) where the input is a simple linear combination of the state vector, and optimising controller (OC's) which minimise the predicted future error of the state vector.

3.1 Linear feedback controllers

The controller design method presented here is based on [SO95].

Linear feedback controllers are all of the form $u(i) = -\boldsymbol{k}(j)(\boldsymbol{x}_\mathrm{a}(i) - \boldsymbol{z}_\mathrm{a}(j))$. $\boldsymbol{x}_\mathrm{a}(i)$ is an augmented form of $\boldsymbol{x}(i)$:

$$\boldsymbol{x}_\mathrm{a}(i) = (x_1(i), \ldots, x_n(i), u(i-1), \ldots, u(i-s))$$

or $\boldsymbol{y}(i)$ with the $u(i)$ component removed; $\boldsymbol{z}_\mathrm{a}(j)$ is a corresponding augmented form of the UPO, with the last d_h elements of $\boldsymbol{z}_\mathrm{a}(j)$ equal to 0.

The vector sequence $\boldsymbol{k}(j)$ is found from linear augmented models; these models are

$$\boldsymbol{x}_\mathrm{a}(i+1) - \boldsymbol{z}_\mathrm{a}(j+1) = \boldsymbol{A}_\mathrm{a}(\boldsymbol{x}_\mathrm{a}(i) - \boldsymbol{z}_\mathrm{a}(j)) + \boldsymbol{b}_\mathrm{a}(j)u(i)$$

where j is such that $\boldsymbol{z}(j)$ is the closest point in the UPO to $\boldsymbol{x}(i)$. The linear coefficients $\boldsymbol{A}_\mathrm{a}(j)$ and $\boldsymbol{b}_\mathrm{a}$ may be derived from the previously computed polynomial coefficients $\boldsymbol{Z}_\mathrm{a}(j)$.

From Eq. (2.2) and (2.3), we can see that the first element of $\boldsymbol{\phi}_p(\boldsymbol{y})$ is 1 which will be multiplied by the first column of $\boldsymbol{Z}(j)$. The first column thus comprises the zeroth coefficients, and thus corresponds to $\boldsymbol{z}(j+1)$.

The next q elements of $\boldsymbol{\phi}_p(\boldsymbol{y})$ are $\boldsymbol{y}$, and the corresponding columns of $\boldsymbol{Z}(j)$ contain the linear coefficients.

The vector $\boldsymbol{y}$ is divided into a state vector $\boldsymbol{x}$ and an input vector $\boldsymbol{v}$; the input vector in turn is composed of the current input value and past input values, if any are used. Each matrix $\boldsymbol{Z}(j)$ may thus be divided as

$$\boldsymbol{Z}(j) = \left(\; \hat{\boldsymbol{z}}(j+1) \;\middle|\; \boldsymbol{A}(j) \;\middle|\; \boldsymbol{b}(j) \;\middle|\; \boldsymbol{C}(j) \;\middle|\; \boldsymbol{Z}_{\text{ho}}(j) \; \right)$$

where $\hat{\boldsymbol{z}}(j+1)$ contains the zeroth coefficients (the match between $\hat{\boldsymbol{z}}(j+1)$ and $\boldsymbol{z}(j+1)$ may not be exact), $\boldsymbol{A}(j)$, $\boldsymbol{b}(j)$ and $\boldsymbol{C}(j)$ contain the linear coefficients corresponding to the state, current input and past inputs respectively, and $\boldsymbol{Z}_{\text{ho}}(j)$ contains the higher order coefficients. All have n rows; $\boldsymbol{A}(j)$ has n columns and $\boldsymbol{C}(j)$ has d_h columns.

From the definition of $\boldsymbol{x}_a$, the augmented matrices are

$$\boldsymbol{A}_{\text{a}}(j) = \left(\begin{array}{c|ccccc} \boldsymbol{A}(j) & & & \boldsymbol{C}(j) & & \\ \hline & 0 & 0 & & 0 & 0 \\ & 1 & 0 & \cdots & 0 & 0 \\ \boldsymbol{0} & 0 & 1 & & 0 & 0 \\ & \vdots & & \ddots & \vdots & \\ & 0 & 0 & \cdots & 1 & 0 \end{array} \right), \quad \boldsymbol{b}_{\text{a}}(j) = \begin{pmatrix} \boldsymbol{b}(j) \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

These are the linearisations of chaotic map $\boldsymbol{g}$. To stabilise the UPO, we need the linearisations of $\boldsymbol{g}^{\langle m \rangle}$, which can be obtained by combining the individual linearisations above. Recall that $\boldsymbol{z}(j) = \boldsymbol{z}(j+m)$ and similarly for $\boldsymbol{Z}(j)$, $\boldsymbol{A}_{\text{a}}(j)$, etc. Let $\Theta(j,k) = \prod_{q=1}^{k} \boldsymbol{A}_{\text{a}}(j-q)$; the full-cycle linear model $(\boldsymbol{A}_{\text{f}}(j), \boldsymbol{b}_{\text{f}}(j))$ is

$$\boldsymbol{A}_{\text{f}}(j) = \Theta(j,m)$$

$$\begin{aligned} \boldsymbol{B}_{\text{f}}(j) \;\; = \;\; & (\Theta(j-1,m-1)\boldsymbol{b}_{\text{a}}(j) | \Theta(j-1,m-2)\boldsymbol{b}_{\text{a}}(j+1) | \\ & \ldots | \boldsymbol{b}_{\text{a}}(j+m-1)) \end{aligned}$$

$$\boldsymbol{x}_{\text{a}}(i+m) - \boldsymbol{z}_{\text{a}}(j) = \boldsymbol{A}_{\text{f}}(j)(\boldsymbol{x}_{\text{a}}(i) - \boldsymbol{z}_{\text{a}}(j)) + \boldsymbol{B}_{\text{f}}(j) \begin{pmatrix} u(i) \\ u(i+1) \\ \vdots \\ u(i+m-1) \end{pmatrix}$$

For each point $\boldsymbol{z}_{\text{a}}(j)$ a controller $\boldsymbol{K}(j)$ must be found so that the eigenvalues of $\boldsymbol{A}_{\text{f}}(j) - \boldsymbol{B}_{\text{f}}(j)\boldsymbol{K}(j)$ are in the unit circle. $\boldsymbol{K}(j)$ provides future input values ($u(i+1)$ etc.); since $u(i)$ will be recomputed at every i, only the first row of the $\boldsymbol{K}(j)$ is needed; this is the control law $\boldsymbol{k}(j)$.

The author has used the discrete linear quadratic regulator (DLQR) $\boldsymbol{K}(j)$ (and hence $\boldsymbol{k}(j)$); this is also used in [Vin97], though only in normal space for 1-periodic systems. In [SO95], on which the above is based, the control law is obtained from the requirement that the UPO is reached in m steps; this yields a unique solution for the $\boldsymbol{k}(j)$.

The control value $u(i)$ is constrained by $|u(i)| < u_{\max}$; additionally, control is only applied if $\|\boldsymbol{x}(i) - \boldsymbol{z}(j)\|_2 < \kappa$, where κ is an experimentally chosen constraint (usually of the same order as δ, used to select (m, δ) points in forming the model).

3.2 Optimising controllers

These controllers use a model of the system to predict the future system behaviour; the input value is found by minimising a cost function over future input values $u(i), u(i+1), \ldots, u(i+h)$, where h is the prediction horizon. Increasing h increases the dimension of the search space, and thus the difficulty of minimisation. Although the general case is presented below, for simplicity only the case of $h = 0$ was fully investigated in application.

The polynomial model derived in chapter 2 is used below, but any model of the chaotic system could be used in its place.

The cost function is based on the sum of the distances between the UPO and the current and predicted future states. To find these distances, shift the indices of UPO $\boldsymbol{z}(j)$ so that that $\boldsymbol{z}(0)$ is the point closest to current state vector $\boldsymbol{x}(i)$.

Recall that $\boldsymbol{y}(i)$ is a vector containing all the components of state vector $\boldsymbol{x}(i)$ and the input vector $\boldsymbol{v}(i)$; let $\boldsymbol{z}_c(j)$ be $\boldsymbol{z}(j)$ with $d_h + 1$ zeroes appended, so that each vector in $\boldsymbol{z}_c(j)$ is the same length as $\boldsymbol{y}(i)$. Also, define

$$\boldsymbol{Z}_c(j) = \begin{pmatrix} & & \boldsymbol{Z}(j) & & \\ 0 & 0 & & 0 & 0 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & & 0 & 0 \\ \vdots & & \ddots & \vdots & \\ 0 & 0 & \cdots & 1 & 0 \end{pmatrix}$$

Now, let $\hat{\boldsymbol{y}}(j)$ be the predicted future deviation from the UPO for inputs $u(i), \ldots, u(i+h)$; that is,

$$\hat{\boldsymbol{y}}(j) = \begin{cases} \boldsymbol{y}(i) - \boldsymbol{z}_c(0) & j = 0 \\ \boldsymbol{Z}_c(j-1)\hat{\boldsymbol{y}}(j-1) - \boldsymbol{z}_c(j) & 1 \le j \le h \end{cases}$$

and $\hat{y}_{n+1}(j) = u(i+j)$.

Now, let $\hat{\boldsymbol{x}}(j)$ be the first n components of $\hat{\boldsymbol{y}}(j)$ (i.e. the state components); cost function Q is then

$$Q(u(i),\ldots,u(i+h)) = \sum_{j=0}^{h} \left(\left\| \hat{\boldsymbol{x}}(j) \right\|_2 + c(u(i+j)) \right)$$

where

$$c(u(i)) = \begin{cases} 0 & |u(i)| < u_{\max} \\ |u(i)| & \text{otherwise} \end{cases}$$

$c(u(i))$ is a reflection of the saturation of $u(i)$ in the cost function.

Control is only applied if the minimum cost $Q < \kappa$, analogously to the linear case.

The primary problem with the OC is that it is not (to the author's knowledge) provably stable; however, simulations indicate that it works for a range of chaotic systems. The intuitive justification is that if the state-vector can be driven closer to the UPO on the next step, it can be driven even closer on the following one and so on.

Chapter 4

Controlling the driven pendulum

The driven pendulum was chosen as an experimental system to test chaos control. Simulations were used to test and develop the control algorithms previously presented; after success in simulation, control of an experimental pendulum was attempted.

In both the simulated and experimental systems the goal was to stabilise a 1-periodic UPO; a 1-periodic orbit was chosen for simplicity.

4.1 Description of the driven pendulum

The driven pendulum was chosen as an example for chaos control because: it has previously been in simulation used to study chaos control [Bak95]; it is a relatively simple system; the author has had some experience with similar systems; and because of the availability of an experimental pendulum system.

This driven pendulum is different to the one described in section 1.3.6 in a number of ways: this system is a single pendulum; the goal is to stabilize a UPO not a fixed point; the system is to be controlled with a scalar output while the double pendulum system was controlled with full state output; and the controller is to be designed without using the actual driven pendulum equations while the double pendulum controller was designed using the equations of motion of the system.

The driven pendulum is depicted in Fig. 4.1. It is a simple pendulum in the vertical plane to which an external torque may be applied.

In Fig. 4.1 θ is the angular position, with clockwise positive orientation and 0 rad at the downward position. Mass m has weight $W = mg$; L is the distance to the pendulum's centre of mass and τ is the external torque.

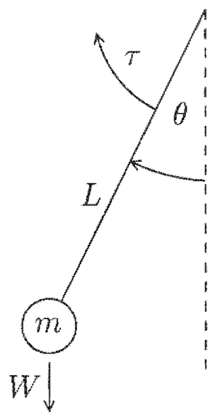


Figure 4.1: The driven pendulum

The applied torque is periodic, $\tau = \alpha \sin(\psi t)$, where α is the torque amplitude and ψ is the frequency. The torque due to gravity is $-mgL\sin\theta$ and that due to friction is $-\beta\dot{\theta}$ (where β is a coefficient of friction); the effective torque is the sum of all of these giving

$$J\ddot{\theta} = -mgL\sin\theta - \beta\dot{\theta} + \alpha\sin(\psi t) \tag{4.1}$$

where J is the rotational inertia of the system.

The equation can be non-dimensionalised with respect to time by substituting $t = s/\sqrt{mgL/J}$; from this it follows that $\dot{\theta} = d\theta/ds \cdot \sqrt{mgL/J}$ and $\ddot{\theta} = d^2\theta/ds^2 \cdot mgL/J$. If, after substitution of these equalities, s, θ and $d\theta/ds$ are relabelled t, w_1 and w_2 respectively, the dimensionless state-space equation is

$$\begin{pmatrix} \dot{w}_1 \\ \dot{w}_2 \end{pmatrix} = \begin{pmatrix} w_2 \\ -\sin(w_1) - bw_2 + a\sin(\omega t) \end{pmatrix} \tag{4.2}$$

where $b = \beta/mgL$, $a = \alpha/mgL$ and $\omega = \psi\sqrt{J/mgL}$.

The system input is torque amplitude a, and output is rotational velocity w_2; choice of output was dictated by the experimental setup.

In principle the driving frequency ω could also have been used as an input, but it was simpler to implement the simulated and experimental systems with a fixed sample frequency ω_s at some integer multiple of the driving frequency.

For appropriate values of b, a and ω the system is chaotic. Such chaotic behaviour can be tested for by looking at the system's attractor or by determining the system's Lyapunov exponents.

Although this system has only two states, it is non-autonomous; time is an implicit third state, and the system thus has three states as needed for chaos (see section B.1.5).

4.2 Control of the pendulum in simulation

Simulations were performed using a fourth-order Runge-Kutta algorithm applied to Eq. 4.2. For $b = 1/3.9$, $a = 1.5$ and $\omega = 2/3$, the system is chaotic [Bak95]; the simulation step-size was fixed at $\Delta t = T/100$, where $T = 2\pi/\omega$. An example time series plot of $w_2(t)$ is shown in Fig. 4.2.

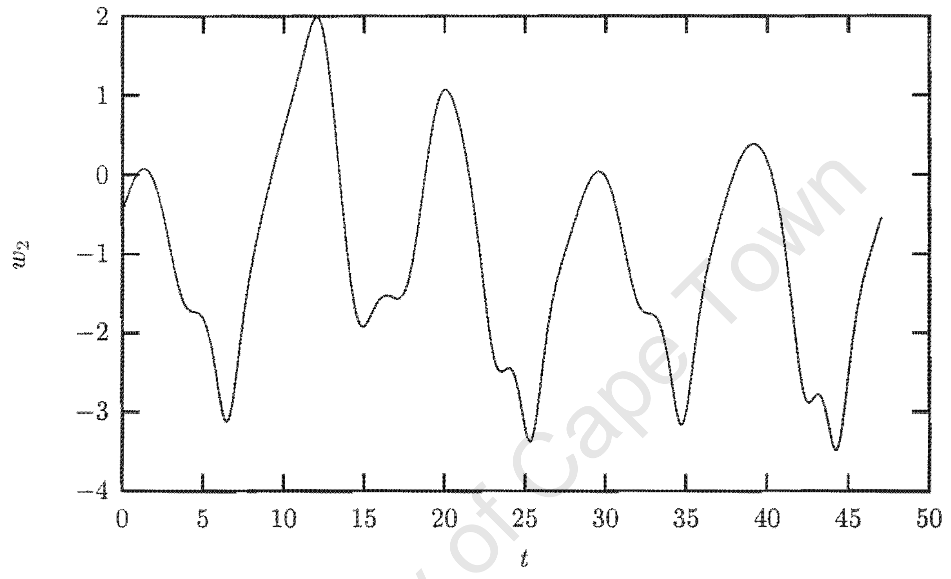


Figure 4.2: A sample time series of the simulated driven pendulum

4.2.1 Establishing experimental parameters

The experimental parameters are the sampling time and method, and the delay space lag-time and dimension.

A chaotic map, necessary for chaos control, is obtained by sampling the system in phase with the driving torque (i.e. with period $T = 2\pi/\omega$). Figure 4.3 shows the attractor for this chaotic map.

The autocorrelation function of w_2 was used to choose a lag-time; Fig. 4.4 shows this function computed from a simulation run of 100 000 time steps. The first zero crossing is at $\tau = 32\Delta t$, which was chosen as the lag-time.

For this lag-time the correlation dimension for delay dimensions 1 to 4 is shown in Fig. 4.5. The delay dimension is taken from the knee of the graph, at $d = 2$. From this and $\tau = 32\Delta t$ it follows that the input history is $d_h = 1$.

The continuous time system with state $\boldsymbol{w}(t)$ is thus converted to a discrete system with state $\boldsymbol{x}(i)$ where $x_1(i) = w_2(100i\Delta t)$ and $x_2(i) = w_2((100i -$

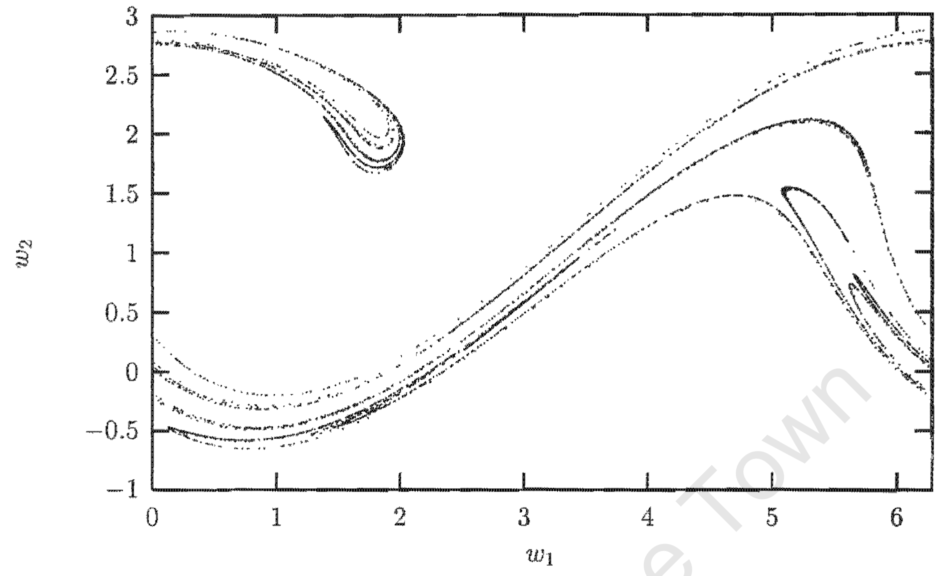


Figure 4.3: Chaotic attractor of the simulated driven pendulum

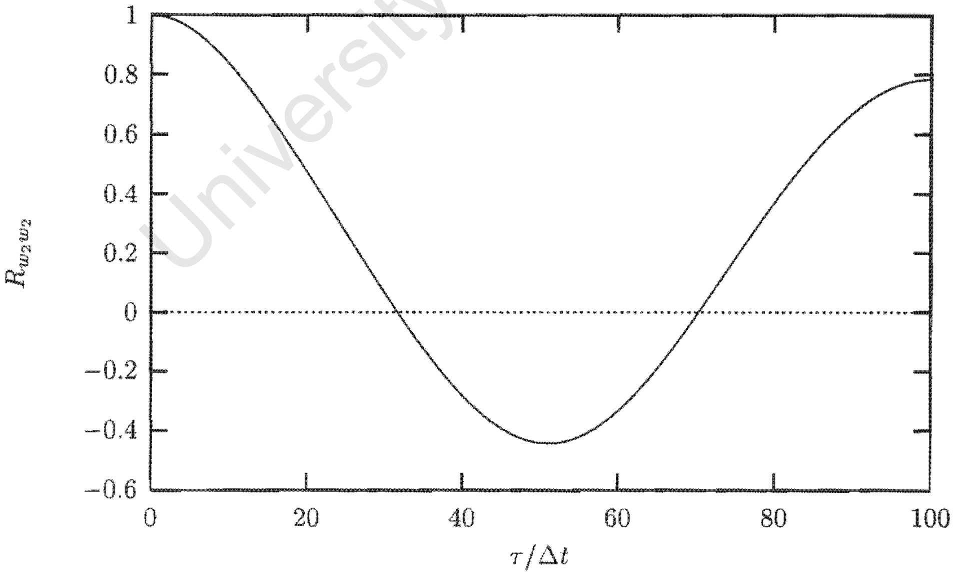


Figure 4.4: Autocorrelation function of w_2

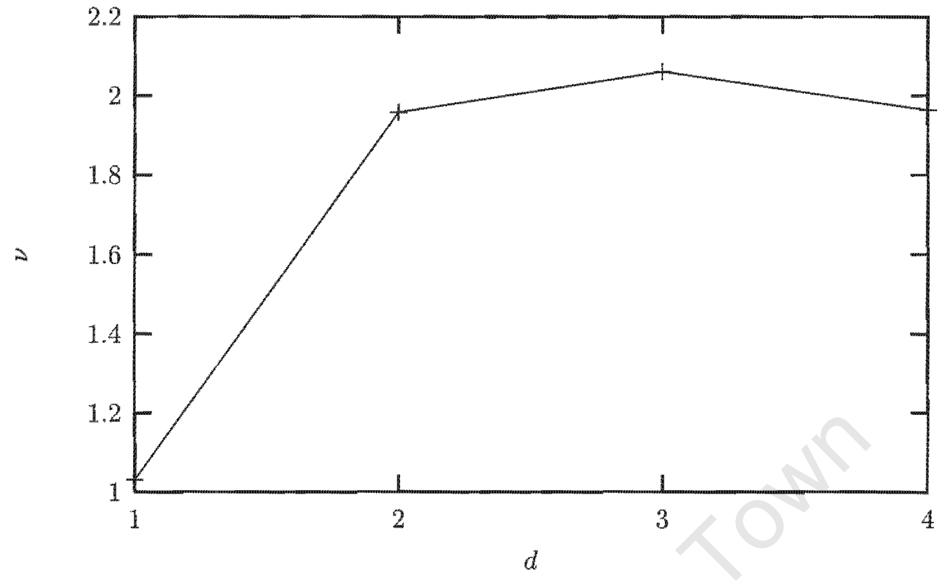


Figure 4.5: Correlation dimensions of w_2

$32)\Delta t$). The system input is only changed in phase with the driving signal; in other words $\Delta a(t) = u(i)$ for $i \leq t/T < i + 1$.

4.2.2 Data analysis and controller design

Analysis was performed on an unperturbed data sequence $\boldsymbol{x}_u(i)$ with $u_u(i) = 0$ for all i and a perturbed data sequence $\boldsymbol{x}_p(i)$ with input signal $u_p(i)$ a normally distributed random variable with mean 0 and standard deviation 0.01; both sequences were of length 50 000, corresponding to 5 000 000 simulation time steps.

The first 5 000 points are shown in Fig 4.6 as a representation of the chaotic attractor in delay space.

Also shown in Fig. 4.6 are the 100 points with the smallest next distance; there are a few isolated points, but most of them are clustered near $(1.84, 8.93\text{E}{-3})$ (62 points) and $(-0.409, -3.22)$ (30 points).

The first point was stabilisable using an optimizing controller with $h = 1$ (using first or second degree models), but could not be stabilised using a simple linear feedback controller, or an optimizing controller with $h = 0$. It is not clear why this is so; the possibility that the mapping between normal and delay space is not bijective (see section 1.2.2) at and around this UPO is ruled out by the fact that the UPO may be stabilised at all.

To allow performance comparison between linear feedback controllers and

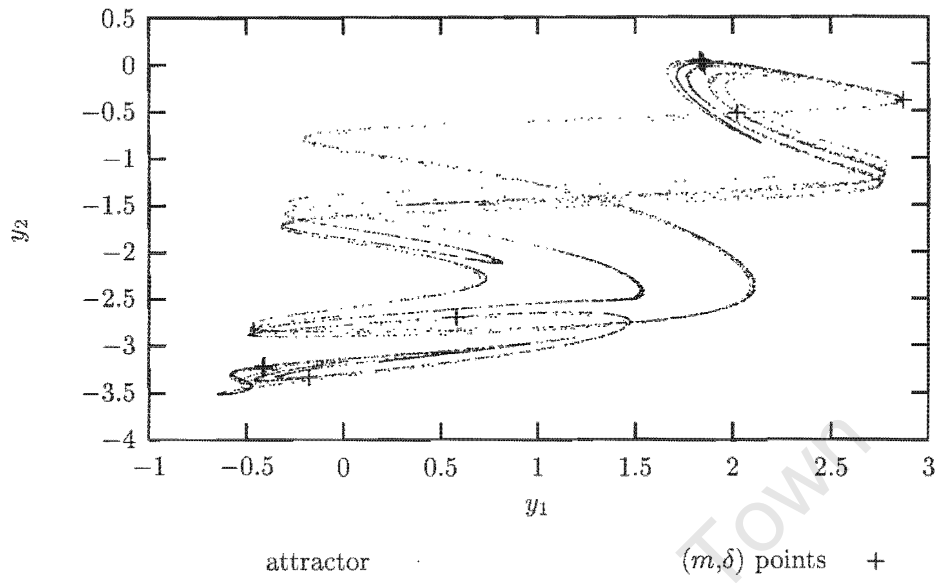


Figure 4.6: Delay space attractor of the simulated driven pendulum

optimizing controllers, the second point was chosen for further investigation, thus the UPO is $\boldsymbol{z} = (-0.409, -3.22)$. (Since $m = 1$, there is no need to write this as $\boldsymbol{z}(j)$.)

The coefficients were fitted twice, once for a linear model only, i.e. with polynomial degree $p = 1$, and once for $p = 2$.

The fitted linearisation was

$$\boldsymbol{A}_{\mathrm{a}} = \begin{pmatrix} -6.02 & 1.00 & 2.97 \\ -1.43 & 0.222 & 0.657 \\ 0 & 0 & 0 \end{pmatrix}$$

$$\boldsymbol{B}_{\mathrm{a}} = \begin{pmatrix} -7.71 \\ -2.51 \\ 1.00 \end{pmatrix}$$

Matrix $\boldsymbol{A}$ (which is the first two rows and columns of $\boldsymbol{A}_{\mathrm{a}}$) has eigenvalues -0.0158 and -5.78, indicating a saddle point as expected. The controller $\boldsymbol{k}$ is synthesised as described in chapter 3 using DLQR (although for $m = 1$, as in this case, the procedure is quite simple), with $\boldsymbol{Q} = \boldsymbol{I}$ and $R = 1$; this gives

$$\boldsymbol{k} = \begin{pmatrix} 0.740 & -0.123 & -0.365 \end{pmatrix} \tag{4.3}$$

Other values of $\boldsymbol{Q}$ and R were not tried; generally the DLQR perfomance was similar to the linear optimising controller (see below), and it was not thought necessary to optimally tune this controller.

	1	y_1	y_2	y_3	y_4
1	-0.416076	-5.90395	0.95584	-7.65546	2.88989
y_1		56.1658	-17.2869	166.472	-54.3135
y_2			0.229587	-27.5758	7.733
y_3				116.667	-82.8505
y_4					14.1244

Table 4.1: Coefficients from first row of $\boldsymbol{Z}$

	1	y_1	y_2	y_3	y_4
1	-3.21901	-1.42048	0.21523	-2.48563	0.650798
y_1		2.50284	-0.528976	10.1979	-2.29959
y_2			-0.228832	-1.7456	0.111297
y_3				6.2394	-5.23734
y_4					0.705751

Table 4.2: Coefficients from second row of $\boldsymbol{Z}$

The polynomial coefficient matrix $\boldsymbol{Z}$ is shown in Tables 4.1 and 4.2; in total $2\binom{4+2}{2} = 30$ coefficients were required.

Chaos control was successful with both the linear feedback controller $\boldsymbol{k}$ and the optimizing controller based on $\boldsymbol{Z}$. The next section discusses the relative performance of these controllers.

4.2.3 Measuring controller performance

Two types of performance were measured: ideal performance and noise robustness.

The measurements were taken with three controllers, namely a DLQR, a linear optimising controller (LOC) and a polynomial optimising controller (POC). The DLQR was the controller described in section 3.1 with $\boldsymbol{k}$ given in Eq. (4.3). The LOC and POC were the controllers described in section 3.2, the LOC using only linear and the POC using all terms of Table 4.1 and 4.2 (in other words, the LOC uses only the first row of both tables).

The LOC is included to test whether performance differences between the DLQR and POC should be attributed to differences in the model or to the optimisation used in the LOC and POC.

4.2.4 Controller ideal performance

Ideal performance of chaos controllers is measured in terms of mean time to stabilisation, following [OGY90]. The system is simulated from j different initial conditions on the chaotic attractor, and for each the time to stabilisation is measured; the system is assumed to be stable when it is within λ_1 (an experimentally chosen value) of the target UPO for k iterations of the map. The ideal performance is the mean of these times. This measures ideal performance since no noise or disturbances are introduced into the system.

The test was run for each of 10 $u_{\max}$ and 10 κ values, giving 100 points in total; the $u_{\max}$ values were logarithmically spaced between 0.01 and 0.1, and the κ values were logarithmically spaced between 0.05 and 0.5. The tests were done with $j = 40$, $k = 10$ and $\lambda_1 = 0.1$.

The mean value of time to stabilisation for the DLQR is shown in Fig. 4.7, for the LOC in Fig. 4.8 and for the POC in Fig. 4.9. Both the $u_{\max}$ and κ axes are logarithmically scaled in all figures in this section. The results are also tabulated in appendix E.

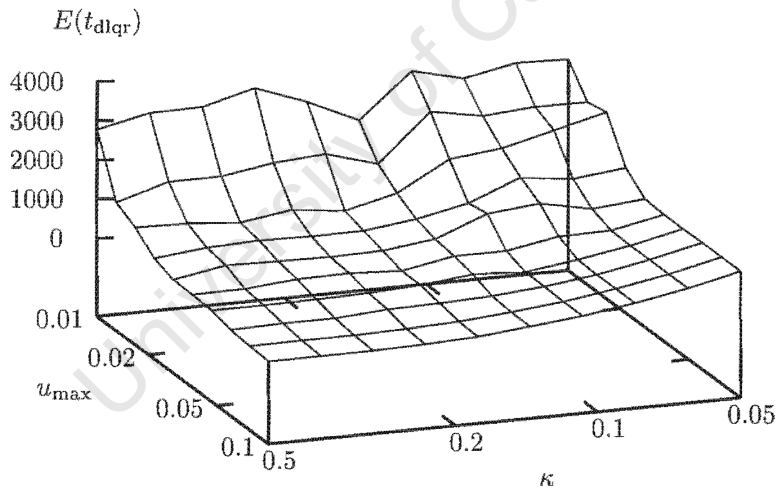


Figure 4.7: Mean time to stabilisation for the DLQR

From these figures, general trends are visible: in all three cases, performance obviously improves with increasing $u_{\max}$. For higher $u_{\max}$ values performance also improves with increasing κ, and for the POC performance noticeably improves for low $u_{\max}$ and high κ values.

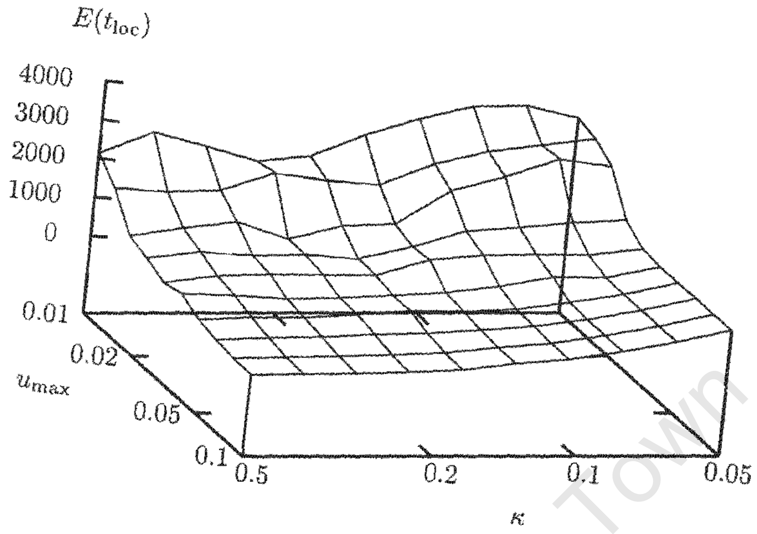


Figure 4.8: Mean time to stabilisation for the LOC

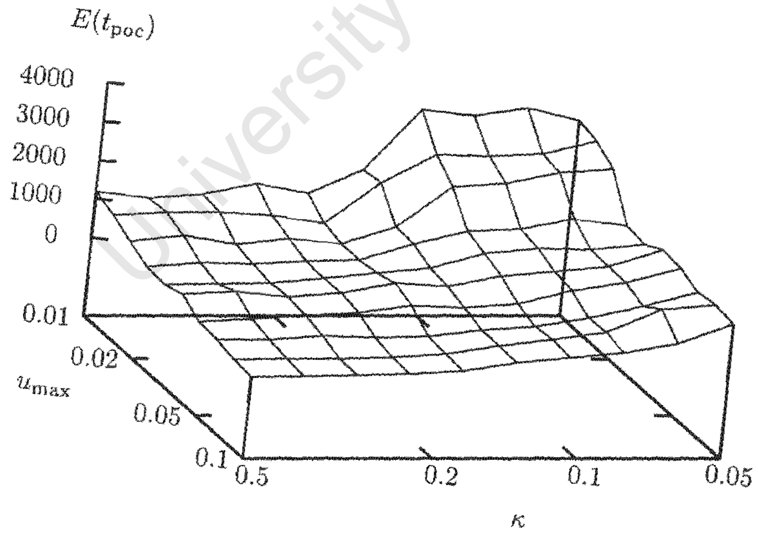


Figure 4.9: Mean time to stabilisation for the POC

To examine the difference between POC and DLQR and POC and LOC more closely, relative performance values are shown in Figures 4.10 and 4.11. These figures show the difference of the DLQR and LOC to the POC performance respectively; high positive values indicate better POC performance.

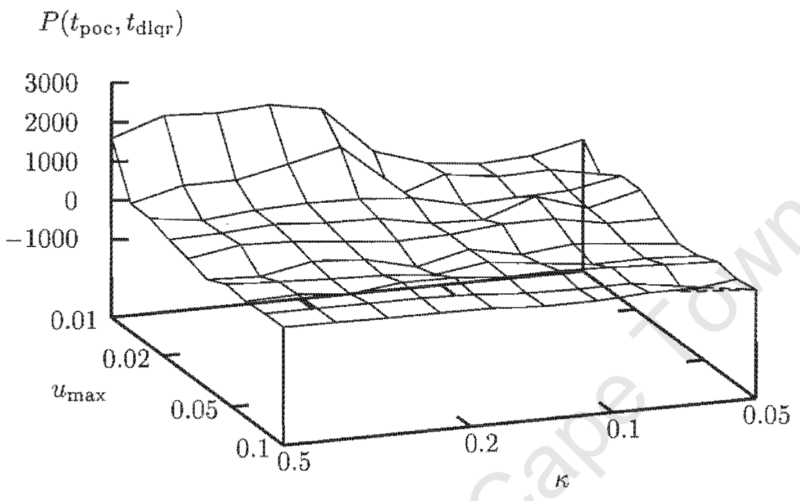


Figure 4.10: POC performance relative to DLQR

The comparative figures highlight the better performance of the POC for small $u_{\max}$ and large κ values. Figures 4.10 and 4.11 are similar, indicating that performance differences may be attributed to the polynomial model rather than other differences in the control algorithm.

The improved performance of POC for large κ is expected since the POC should be able to predict the behaviour of the pendulum over a greater region of state space about the UPO; however, the advantage of the polynomial model is offset by the improvement caused by higher $u_{\max}$ values evident in all three controllers.

From this it follows that using a POC will be advantageous when $u_{\max}$ is strongly constrained by external factors; however, if the input is relatively unconstrained the simpler DLQR controller offers similar performance.

4.2.5 Controller noise performance

Noise robustness was measured by the ratio of time the system was within λ_2 (again experimentally chosen) of the UPO when additive Gaussian noise with

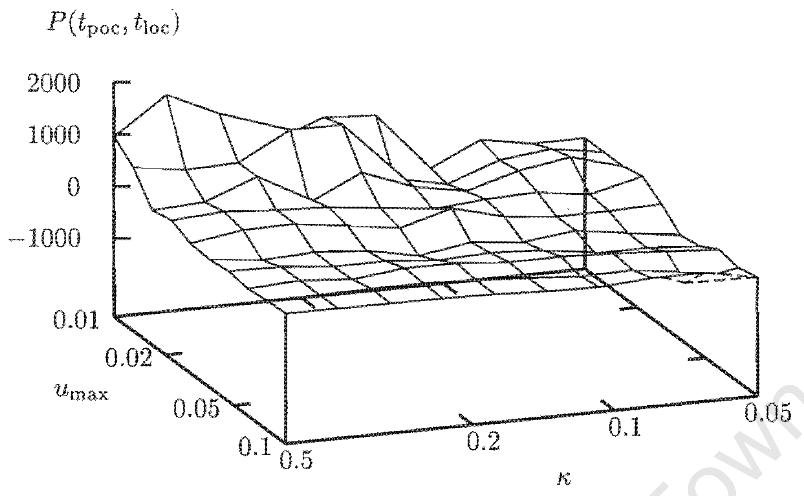


Figure 4.11: POC performance relative to LOC

standard deviation σ was introduced. For this test system was started very near the UPO to remove variations already measured by ideal performance.

Interpreted by itself this measure is misleading, since even without control the state vector will at times come arbitrarily close to the UPO; the importance is in the difference in the measure for different noises levels and between different controllers.

Noise performance was measured at three different points in $(u_{\max}, \kappa)$ space, namely $(0.01, 0.05)$, $(0.01, 0.5)$ and $(0.1, 0.5)$. These values were chosen from results of the ideal performance test; the two $u_{\max} = 0.01$ points were chosen to investigate noise effects in the region where the POC has a performance advantage, and the $u_{\max} = 0.1$ case was chosen to compare noise robustness at a higher $u_{\max}$ value.

For each point, the performance was measured for 30 σ values logarithmically spaced between 3E−4 and 3E−3 in the first two cases and 1E−3 and 0.1 in the third case. In all cases λ_2 was set at 0.1. The test simulations were run for 10 000 map iterations each.

Noise performance is shown in Fig. 4.12, 4.13 and 4.14. In these figures a higher value indicates better noise performance.

For $(u_{\max}, \kappa)$ at $(0.01, 0.05)$ and $(0.01, 0.5)$ (Fig. 4.12 and 4.13), the POC and LOC have a slight advantage over the DLQR; the difference seems to be because of the optimisation in the POC and LOC. The control is very

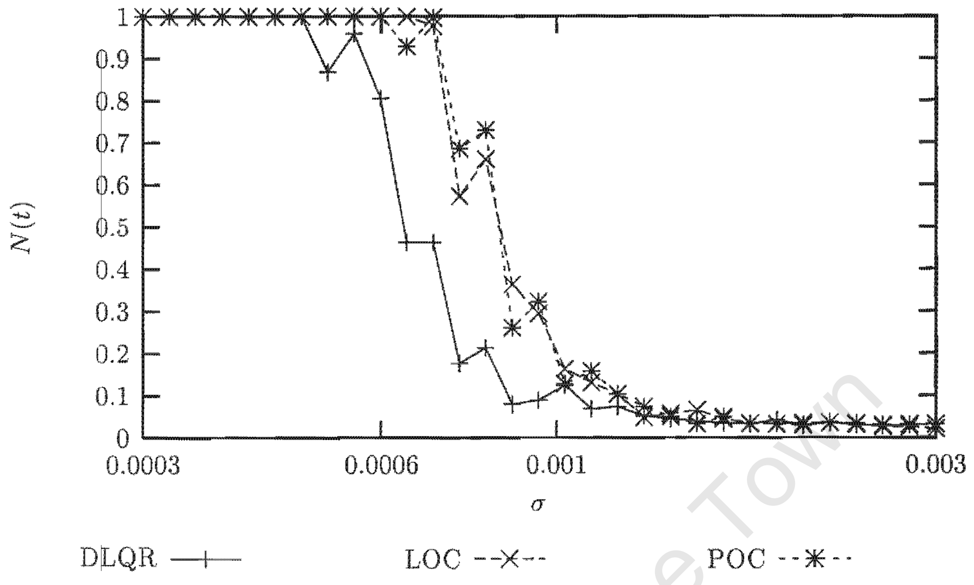


Figure 4.12: Noise performance for $(u_{\max}, \kappa) = (0.01, 0.05)$

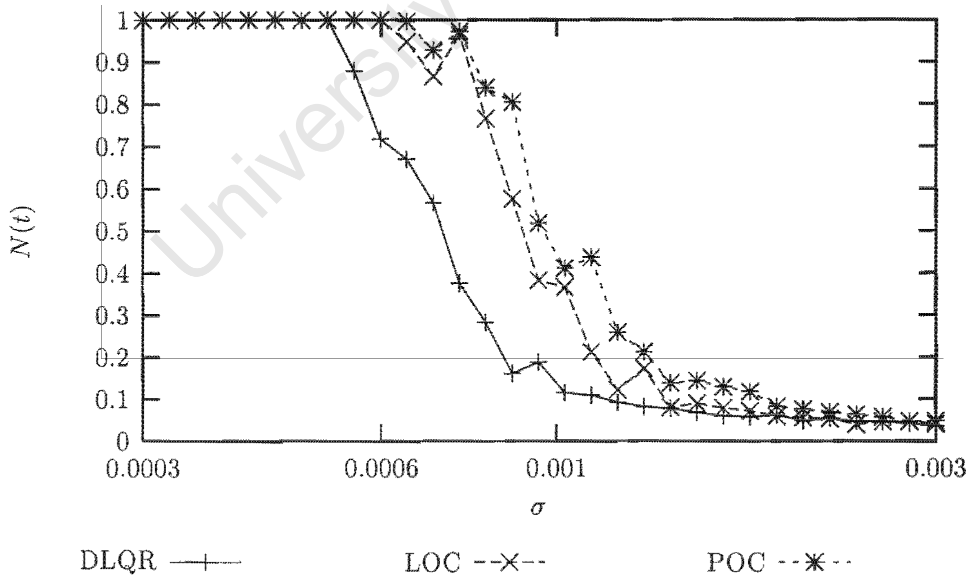


Figure 4.13: Noise performance for $(u_{\max}, \kappa) = (0.01, 0.5)$

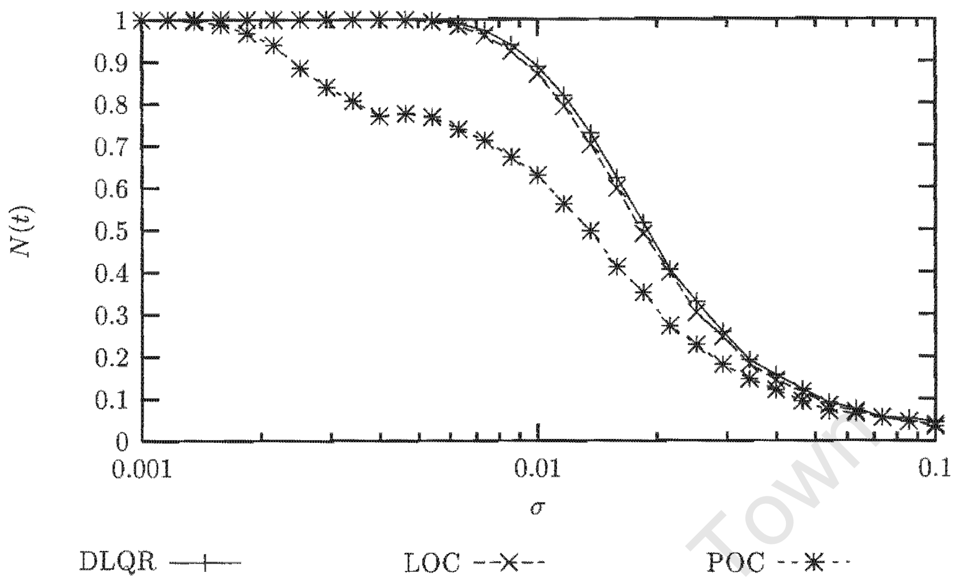


Figure 4.14: Noise performance for $(u_{\max}, \kappa) = (0.1, 0.5)$

sensitive to noise, which starts having an effect for σ between 5E−4 and 8E−4.

When $(u_{\max}, \kappa) = (0.1, 0.5)$ the POC performs worse than the LOC and DLQR, which have similar results; in this case differences can be attributed to the polynomial model. Performance is far better than for either $u_{\max} = 0.01$ case; noise only effects the POC for $\sigma >$ 1E−3 and the LOC and DLQR for $\sigma >$ 5E−3.

Prediction of system behaviour is obviously worse in the presence of noise; additionally, noise may cause the state erroneously to appear to be outside the controllable region about the target UPO, in which case control is not attempted (or abandoned). If the noise is great enough that this occurs often, chaos control becomes useless.

It is expected that noise would have a worse effect when using a polynomial model since the higher order terms will tend to amplify the error; this is visible in Fig. 4.14. In Fig. 4.12 and 4.13 it appears that the small value of $u_{\max}$ is a more significant factor, though the reason for this is unclear.

From the ideal performance and noise results above, it can be seen that constrained $u_{\max}$ values lead to degraded ideal performance and noise robustness; if $u_{\max}$ is constrained by external factors, a polynomial model may offer improved performance over a linear model.

4.3 Attempted control of the pendulum in experiment

The attempt to control an experimental pendulum unfortunately failed at an early stage in the experimental procedure, during modelling. This section presents results of modelling data from the experimental system and speculation on the cause of the failure.

The experimental system comprised an armature controlled DC motor, the pendulum arm and mass, a tachometer, a computer for measurement and control and electronics connecting the computer to the motor and tachometer. Details are given in appendix D.

Comparing this setup to Eq. 4.1, it is obvious that the DC motor will be providing the driving torque. It is assumed that the electrical time constant of the motor (from coil inductance and resistance) is negligible compared to the mechanical time constant (from inertia). The motor was voltage controlled; the effect of back EMF is lumped with the friction factor β.

4.3.1 Establishing experimental parameters

In terms of Eq. 4.1, the accessible parameters are mass m, distance to centre of mass L, driving amplitude α and driving frequency ψ. To find a chaotic regime it was simplest to fix m and L and try different values of driving amplitude and frequency; from this search driving frequency 1.1 Hz and amplitude 6.0 V were chosen as an operating point.

For analysis and control, the system was sampled at 110 Hz. The signal from the tachometer was anti-aliased by a low-pass filter with a cut-off frequency of about 48 Hz before being digitised.

The autocorrelation of 100 000 samples is shown in Fig. 4.15, and a power spectrum (from a discrete Fourier transform of 2^{16} samples) in Fig. 4.16.

In Fig. 4.15, the integer value closest to the first zero crossing is $j = 30$; this will be the lag-time used for delay space reconstruction.

A digital filter was used to compensate for the quantisation introduced by the 12-bit analog to digital converter. If the sampled signal is $w_{\text{in}}(i)$ and the filter output $w(i)$, the filter used is $w(j) = 0.3w_{\text{in}}(j) + 0.7w(j-1)$. This is a low-pass filter with cut-off at about 6 Hz; from Fig. 4.16 the spectrum has already dropped off to less than 1E$-$4 of the peak power at this frequency.

Correlation dimension for embedding dimensions 1 to 5 are shown in Fig 4.17; this was computed using 10 000 points of the filtered signal $w(j)$ with a lag-time of 30 sample steps.

There is no clear knee in Fig. 4.17 (compare with Fig. 4.5). Delay di-

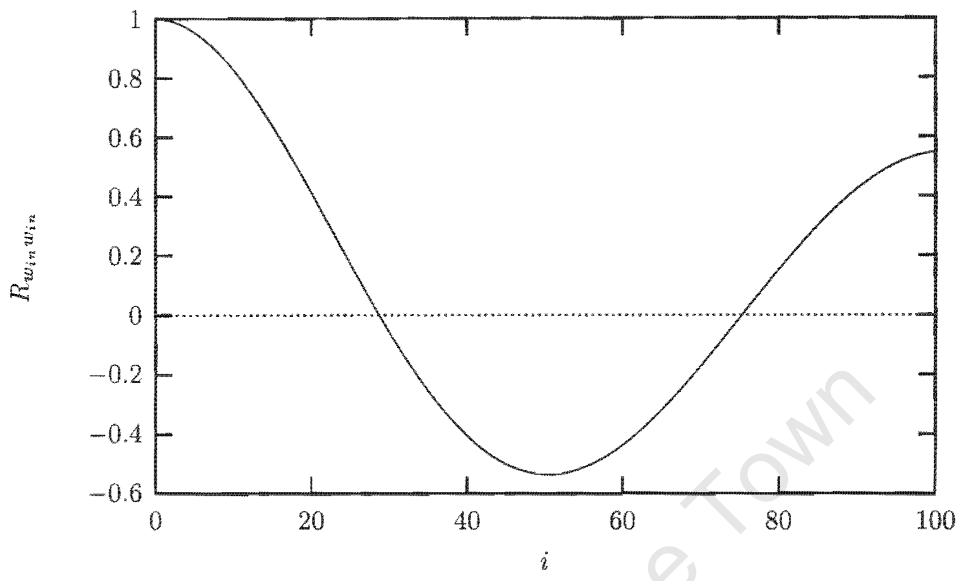


Figure 4.15: Autocorrelation of experimental data

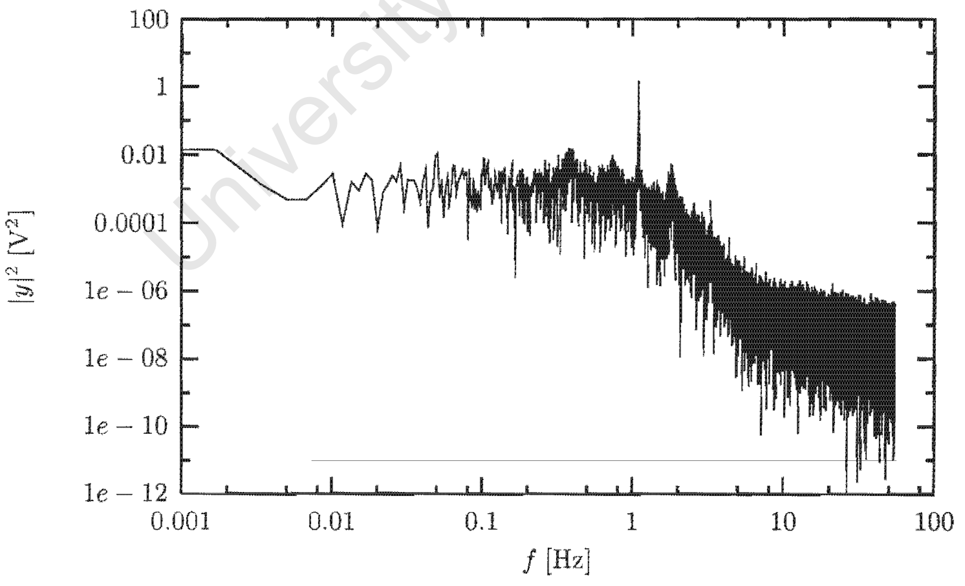


Figure 4.16: Power spectrum of experimental data

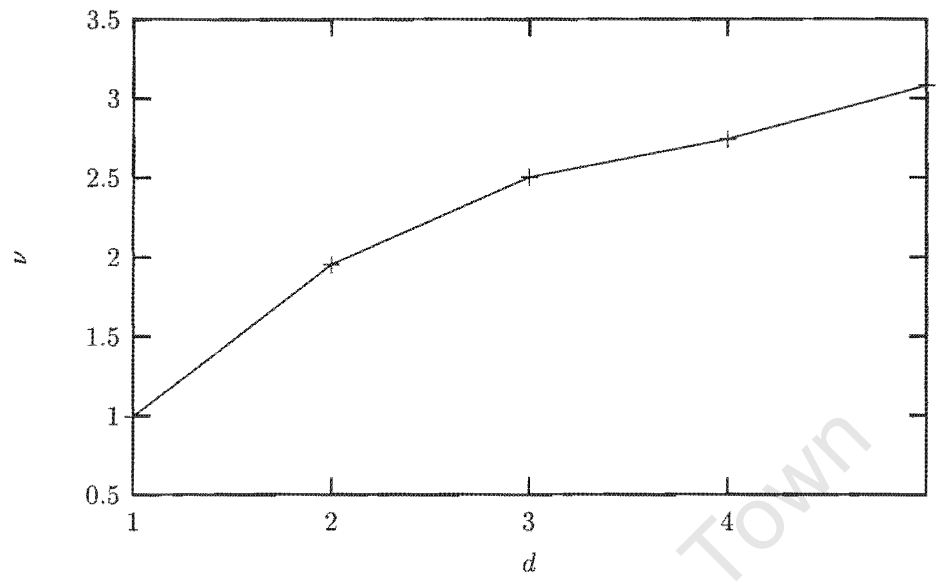


Figure 4.17: Correlation dimensions of experimental data

mension $d = 2$ was chosen because control was successful in simulation for $d = 2$, and because higher delay dimensions would require a large number of coefficients (e.g. with $p = 2$, $d = n = 3$, and thus $d_h = 1$, 63 polynomial coefficients would be required). From $d = 2$ and lag-time 30, $d_h = 1$.

The process input is only changed at the start of each driving cycle; the sampled process input is $a(j) = (6.0 + \Delta a(j)) \sin(2\pi j/110)$ and $\Delta a(j) = u(i)$ for $i \leq j/110 < i + 1$.

Analysis was performed in a delay space with lag-time 30 and dimension 2; thus state vector $\boldsymbol{x}(i) = (w(110i), w(110i - 30))$ where $w(j)$ is the digitally filtered velocity measurement.

4.3.2 Data analysis and controller design

Analysis was performed on a unperturbed data sequence $\boldsymbol{x}_u(i)$ of size 63 200 with $u(i) = 0$ for all i and on a perturbed data set $\boldsymbol{x}_p(i)$ of the same size for input $u_p(i)$ a normally distributed random variable with zero mean and standard deviation 0.05.

The first 5 000 points of $\boldsymbol{x}_u(i)$ are shown in Fig. 4.18 as a representation of the chaotic attractor. Also shown are the $(m, \delta) = (1, 0.137)$ points; δ was chosen to select 35 points.

The attractor depicted in Fig. 4.18 is noticeably fuzzier than that of Fig. 4.3; this is probably because of noise or parameter variation.

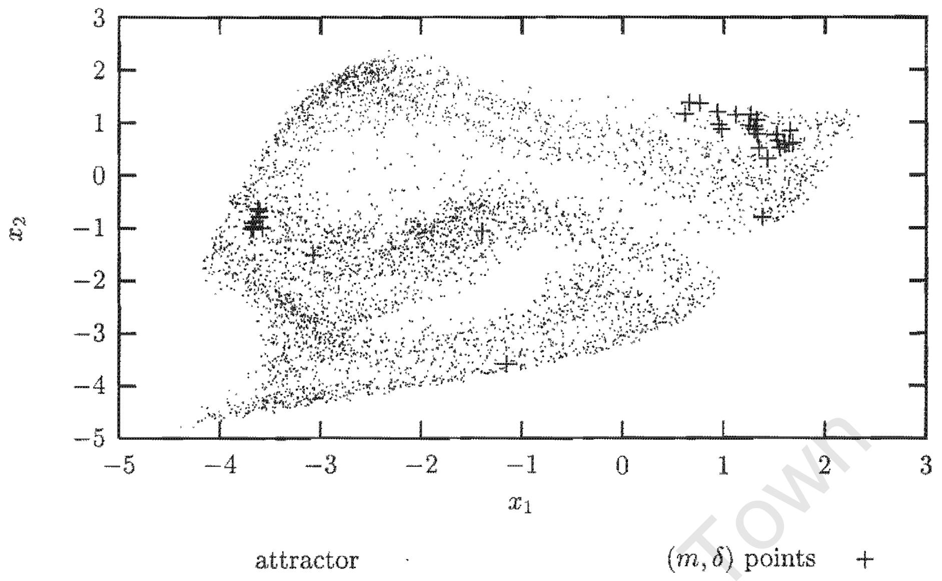


Figure 4.18: Delay space attractor of the experimental driven pendulum

In Fig. 4.18 there are two large clusters of points, one of 9 points near $(-3.5, -1)$, and one of 2 points near $(1, 1)$. The first cluster was chosen as the control target because it is less spread out; the UPO for this cluster is $\boldsymbol{z} = (-3.65, -0.870)$.

From $\boldsymbol{x}_p(i)$ 100 (m, δ) points with $\delta = 0.198$ were found, giving a cluster of 11 points near the chosen cluster from $\boldsymbol{x}(i)$; from these points the linear and polynomial models were computed. They were computed separately to compare the linear terms in the models.

The linear model is

$$\boldsymbol{A}_{\mathrm{a}} = \begin{pmatrix} 0.590 & 0.147 & 0.359 \\ 1.18 & 0.860 & 0.633 \\ 0 & 0 & 0 \end{pmatrix}, \quad \boldsymbol{B}_{\mathrm{a}} = \begin{pmatrix} 0.384 \\ 0.650 \\ 1.00 \end{pmatrix} \tag{4.4}$$

and the polynomial coefficients are given in Tables 4.3 and 4.4.

Comparing the two models it is already evident that there are some problems; the linear coefficients do not match at all.

4.3.3 Attempted control of the experimental system

Applying DLQR to Eq. 4.4 with $\boldsymbol{Q} = \boldsymbol{I}$ and $R = 1$ gives $\boldsymbol{k} = (0.723, 0.388, 0.408)$.

The system was not stabilised by either the linear or polynomial controller; control was attempted for a number of different $u_{\max}$ and κ values

	1	y_1	y_2	y_3	y_4
1	-3.68	2.93	-0.493	18.7	-4.34
y_1		-38.4	8.94	-174	-13.7
y_2			1.84	182	19.9
y_3				284	111
y_4					41.5

Table 4.3: Polynomial coefficients from the first row of $\boldsymbol{Z}$ for experimental model

	1	y_1	y_2	y_3	y_4
1	-0.927	2.17	0.478	2.98	4.21
y_1		-4.47	8.36	0.498	-10.6
y_2			1.58	5.58	19
y_3				-41.4	-7.39
y_4					-1.67

Table 4.4: Polynomial coefficients from the second row of $\boldsymbol{Z}$ for the experimental model

without success. A typical run is shown in Fig. 4.19 for $u_{\max} = 0.2$ and $\kappa = 0.2$.

4.3.4 Possible causes of controller failure

Possible causes of failure are

1. measurement noise
2. parameter variation
3. experimental or modelling error
4. control implementation error

While errors in the modelling and implementation cannot be ruled out, experimental and simulation evidence given below suggest that the measurement noise and parameter variation prevent chaos control, or the variant thereof presented in this dissertation, from being successful.

Two experiments were performed to estimate measurement noise and parameter variation, as described below.

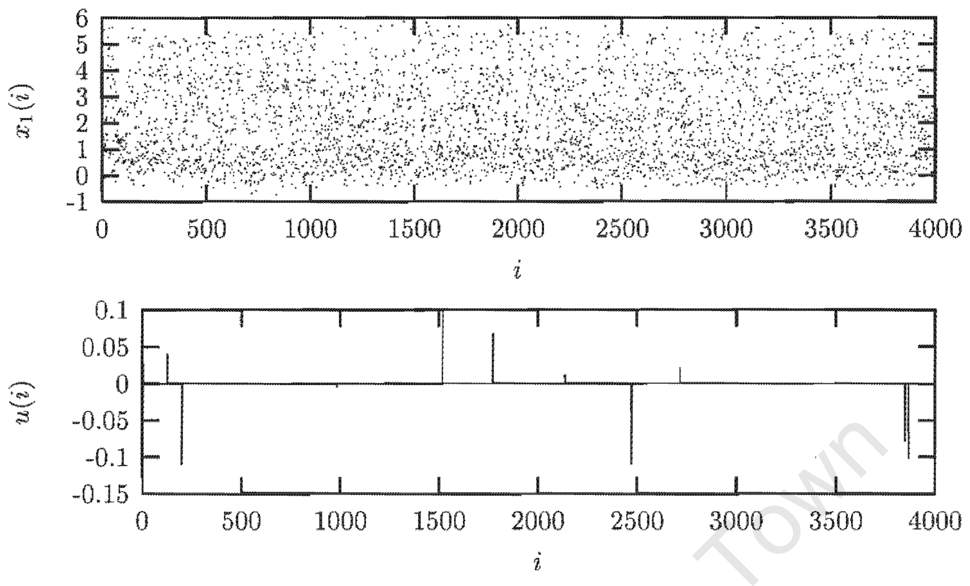


Figure 4.19: Attempted control of the experimental pendulum

Estimating measurement noise and parameter variation (1)

The first experiment was a constant speed experiment; the system was run without the pendulum attached at a constant input voltage. It was expected that the motor should run at a constant speed, and that the output voltage from the tachometer should be constant.

The system was sampled at 110 Hz, and driven at a constant input of 6.0 V. The analog gain on the tachometer signal was reduced to 1 from 10 for this experiment only, since its output was substantially higher than when the pendulum arm was attached. The experiment was run for 100 000 sample steps.

Figures 4.20 and 4.21 show a section from measurements recorded in the first experiment; Fig. 4.20 shows $w_{\text{in}}(j)$, and Fig. 4.21 is the corresponding part of the digitally filtered signal $w(j)$.

The quantisation and the effect of the digital filter are clearly visible in Fig. 4.20 and 4.21; in Fig. 4.21 the signal is no longer quantised (although the effects of quantisation are still visible).

The mean of $w_{\text{in}}(j)$ and $w(j)$ (excluding the first 20 points which are transients) is 2.00; the standard deviation of both signals is 0.013, 0.65 % of the mean.

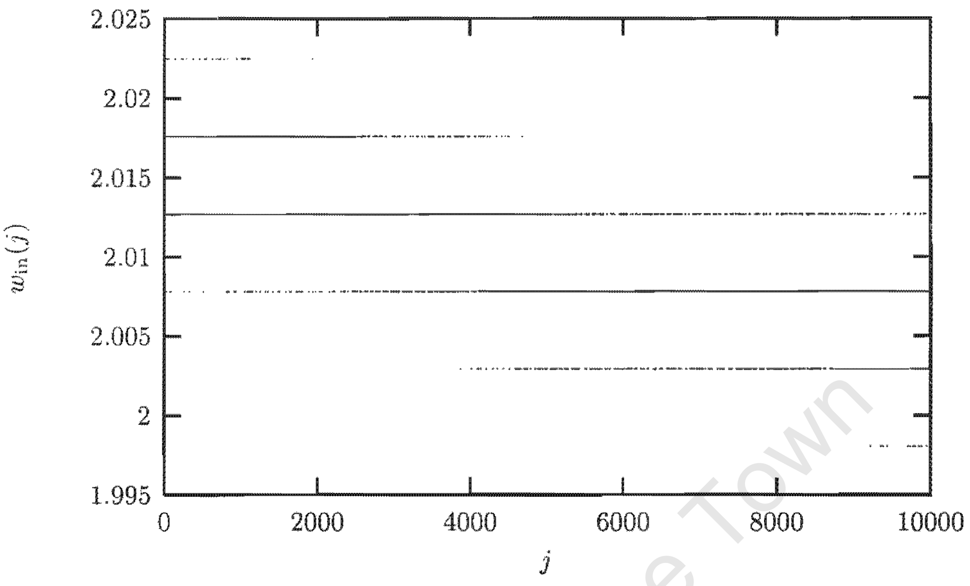


Figure 4.20: Unfiltered measurement of constant speed $w_{\text{in}}(j)$

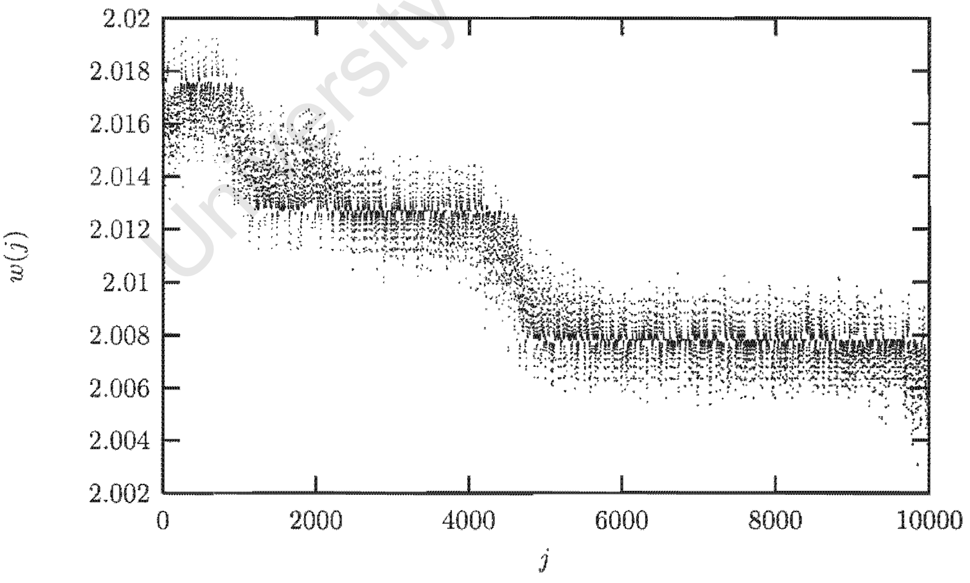


Figure 4.21: Filtered measurement of constant speed $w(j)$

Estimating measurement noise and parameter variation (2)

The second experiment was a periodic response experiment. It was run with the pendulum attached; the input was driven periodically but outside the chaotic regime. In this case periodic behaviour, in phase with the driving signal, was expected.

The experiment was performed for two different operating points: amplitude $a = 3.0$ V and driving frequency $f_d = 1.1$ Hz, and amplitude $a = 4.0$ V and driving frequency $f_d = 2.0$ Hz. In both cases the system was sampled at 100 times the driving frequency, and run for 100 000 sample steps. Digitally filtered output values in phase with the driving signal (i.e. for $i = 110j$ or $i = 200j$) are shown in Fig. 4.22 and 4.23.

The digital filter of section 4.3.1 was used; the change in sampling frequency shifts the cut-off of the digital filter to about 12 Hz in the $f_d = 2.0$ Hz case, but this does not significantly affect the results.

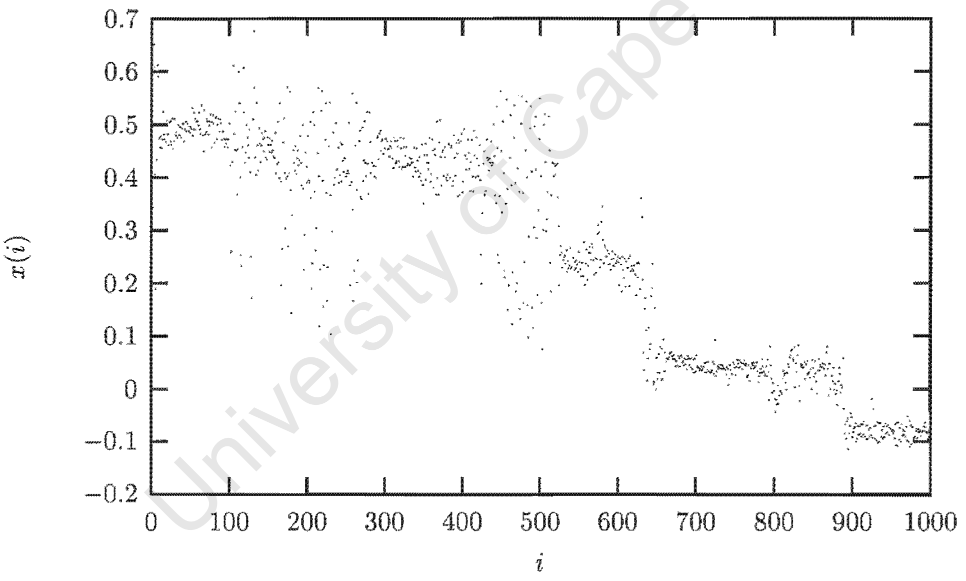


Figure 4.22: Response to periodic input for $a = 3.0$ V, $f_d = 1.1$ Hz

Figure 4.22 shows that the system behaviour is not simply periodic as expected; between time $i = 0$ and $i = 500$, the process seems to be occasionally periodic and occasionally either random or chaotic; after that there are three different phases of periodic behaviour. The final low amplitude was confirmed was visual observation: at the end of the run the pendulum was sweeping an arc of less than 20 °. This behaviour was attributed to the low voltage being applied to the motor, as the signal amplitude at the motor

Estimating measurement noise and parameter variation (2)

The second experiment was a periodic response experiment. It was run with the pendulum attached; the input was driven periodically but outside the chaotic regime. In this case periodic behaviour, in phase with the driving signal, was expected.

The experiment was performed for two different operating points: amplitude $a = 3.0$ V and driving frequency $f_d = 1.1$ Hz, and amplitude $a = 4.0$ V and driving frequency $f_d = 2.0$ Hz. In both cases the system was sampled at 100 times the driving frequency, and run for 100 000 sample steps. Digitally filtered output values in phase with the driving signal (i.e. for $i = 110j$ or $i = 200j$) are shown in Fig. 4.22 and 4.23.

The digital filter of section 4.3.1 was used; the change in sampling frequency shifts the cut-off of the digital filter to about 12 Hz in the $f_d = 2.0$ Hz case, but this does not significantly affect the results.

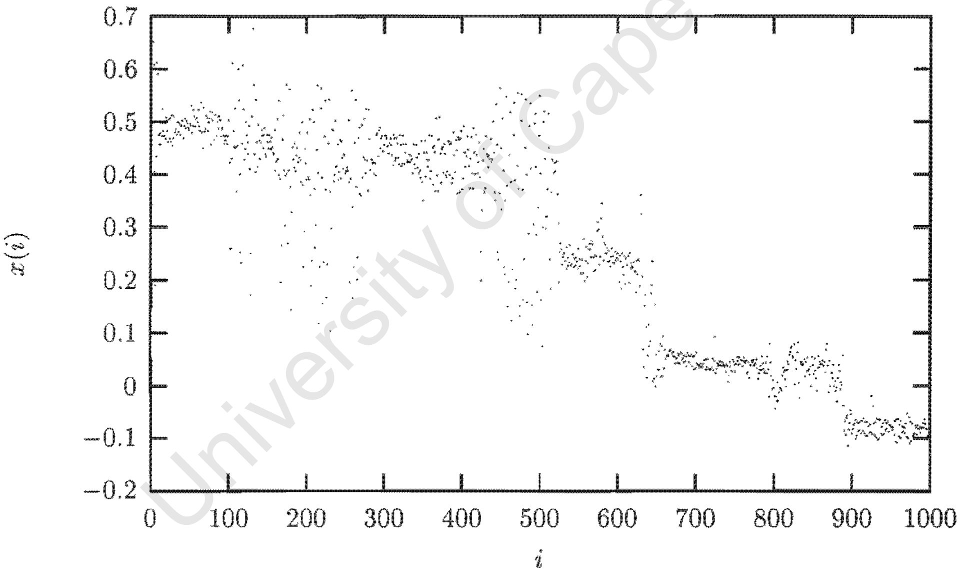


Figure 4.22: Response to periodic input for $a = 3.0$ V, $f_d = 1.1$ Hz

Figure 4.22 shows that the system behaviour is not simply periodic as expected; between time $i = 0$ and $i = 500$, the process seems to be occasionally periodic and occasionally either random or chaotic; after that there are three different phases of periodic behaviour. The final low amplitude was confirmed was visual observation: at the end of the run the pendulum was sweeping an arc of less than 20 °. This behaviour was attributed to the

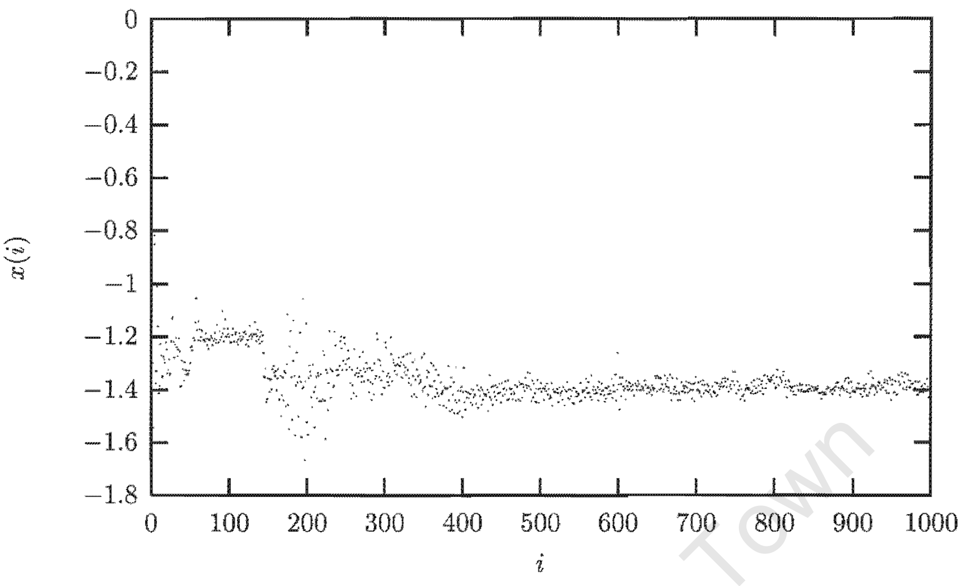


Figure 4.23: Response to periodic input for $a = 4.0$ V,$f_d = 2.0$ Hz

terminals was about 1.5 V (the signal was attenuated in buffer electronics; see Fig D.3) . Higher amplitudes for $f_d = 1.1$ Hz were in the chaotic regime, which is why a different driving frequency was used for further tests.

Figure 4.23 shows results which better match expectations; the system is almost periodic for the entire run, although the signal appears to have more deviation than shown in the constant speed experiment. The standard deviation of this signal is 0.10, which is 7.9 % of the absolute value of the mean.

Under the assumption that the measurement noise remains the same as in the constant speed experiment, the results indicate relatively large parameter variation.

The difference between the results of the constant speed and periodic response experiments are thought to be because the system is moving relatively slowly in the second experiment, well below the motor's usual operating speeds; however, the chaotic system also operates at these speeds. This suggests that parameter variation might be reduced if a gearing system were used, but this was not investigated.

The effect of measurement noise and parameter variation was tested in simulation. The models are very simple, and only indirectly reflect the experimental observations above, but they do give some insight as to how sensitive the modelling process is.

Modelling the effect of measurement noise

To test the effect of measurement noise on modelling, part of the fitting process for the simulated driven pendulum was repeated with Gaussian noise added to the simulated data set.

The modelling was done with a dataset $\boldsymbol{x}_n(i) = \boldsymbol{x}_u(i) + \sigma \boldsymbol{n}(i)$ where $n_1(i)$ and $n_2(i)$ are independent, normally distributed random variables with mean of 0 and standard deviation of 1; $\boldsymbol{x}_u(i)$ is the true simulated value.

The noise not only affects the fitting of coefficients, but also the selection and clustering of (m, δ) points.

The estimation of linear coefficients is more sensitive to noise than the estimation of the fixed point, but should be less sensitive than fitting second (or higher) degree polynomial coefficients. The accuracy of the modelling was determined by how well the fitted linear coefficients $\boldsymbol{A}^*$ matched the previously computed $\boldsymbol{A}$; $\boldsymbol{A}$ is presumed to be close to the true linearisation. The modelling accuracy is measured as the sum $\zeta = \sum_i \sum_j |A_{ij} - A^*_{ij}|$; smaller values of ζ indicate a better match.

The error estimate ζ was computed for two different noise levels and two data set sizes.

The first noise standard deviation, $\sigma_1 = 0.046$, is 0.65 % of the full range of the velocity signal of 7.0. 0.65 % is the standard deviation from the constant speed experiment. The second noise level was $\sigma_2 = 7.0\text{E}-4$, used for comparison.

The data set lengths were 50 000 and 500 000; additionally, ζ was computed for different number of (m, δ) points in the cluster about the UPO.

Results are shown in Tables 4.5 and 4.6, as well as Fig. 4.24

	Cluster size					
σ	25	100	200	500	1000	2000
0.046	15.8	1.76	2.03	4.65	4.6	4.57
7.0E−4	1.83	0.281	0.206	0.743	2.42	3.76

Table 4.5: ζ for measurement noise with 50 000 points

In all cases σ_1 produces large errors. For the smaller noise level, the estimate initially improves with cluster size but finally becomes worse; this is because, as cluster size increases, points which are further away from the UPO which fit the linear model poorly are used in the computation of $\boldsymbol{A}^*$.

It appears that the modelling method cannot tolerate the noise level of σ_1; increasing the amount of data used does not seem to help, but merely shifts

	Cluster size					
σ	25	100	200	500	1000	2000
0.046	5.81	9.32	9.94	8.87	8.67	8.45
7.0E−4	1.21	0.98	0.531	0.392	0.116	0.164

Table 4.6: ζ for measurement noise with 500 000 points

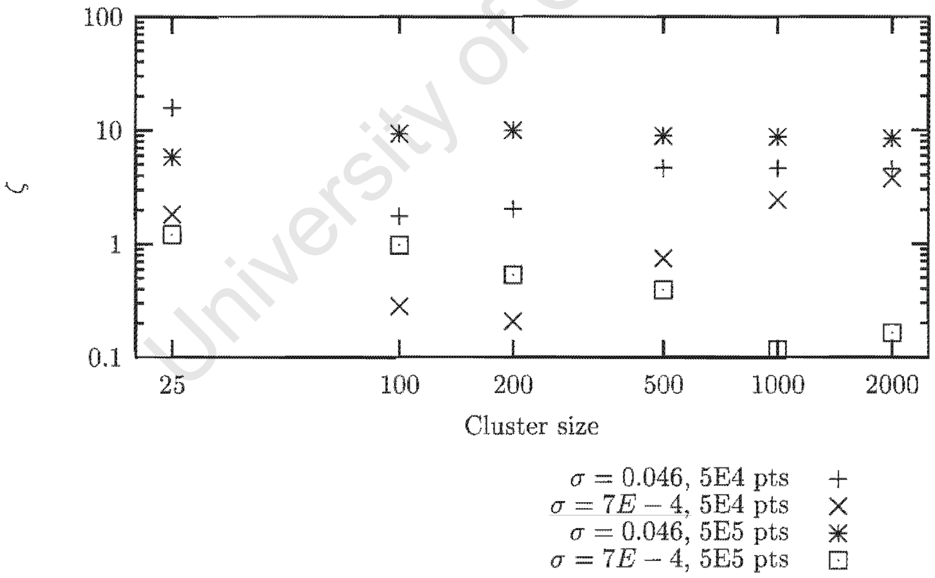


Figure 4.24: ζ for different noise levels and numbers of points

the minimum ζ value. It is speculated that a more sophisticated method of choosing points for the cluster would yield better results.

Modelling the effect of parameter variation

For the purposes of testing the effect of parameter variation, Eq. 4.2 was modified to be

$$\begin{pmatrix} \dot{w}_1 \\ \dot{w}_2 \end{pmatrix} = \begin{pmatrix} w_2 \\ -\sin(w_1) - (b + \sigma b_{\text{var}}(t))w_2 + a\sin(\omega t) \end{pmatrix} \tag{4.5}$$

where $b_{\text{var}}(t)$ is a Gaussian random variable with mean 0 and standard deviation 1; it is band-limited by the simulation step size. Other parameters are unchanged from before.

Modelling accuracy was measured as in the previous section, by comparing the summed absolute differences between the linear coefficients previously computed and linear coefficients fitted to data obtained by simulating Eq. (4.5). ζ was determined for $\sigma_1 = 0.01$ and $\sigma_2 = 0.1$, and data set lengths of 50 000 and 500 000 for various cluster sizes.

When the driving amplitude is lowered to $a = 0.5$, the system becomes periodic (other parameters are the same as in section 4.2); for measurements taken in phase with the driving signal, the velocity signal has a standard deviation of about 0.702 % (relative to the mean) for σ_1 and 6.77 % for σ_2. Results are given in Tables 4.7 and 4.8.

	Cluster size				
σ	25	100	200	500	1000
0.01	4.70	1.03	0.28	2.47	3.98
0.1	11.95	8.31	6.45	5.92	6.20

Table 4.7: ζ for parameter variation with 50 000 data points

	Cluster size				
σ	100	200	500	1000	2000
0.01	5.17	2.14	2.36	1.88	0.921
0.1	8.47	8.53	9.2	9.02	8.89

Table 4.8: ζ for parameter variation with 500 000 data points

The trend for parameter variation is similar to that observed previously with measurement noise; for small parameter variation, it is possible to obtain

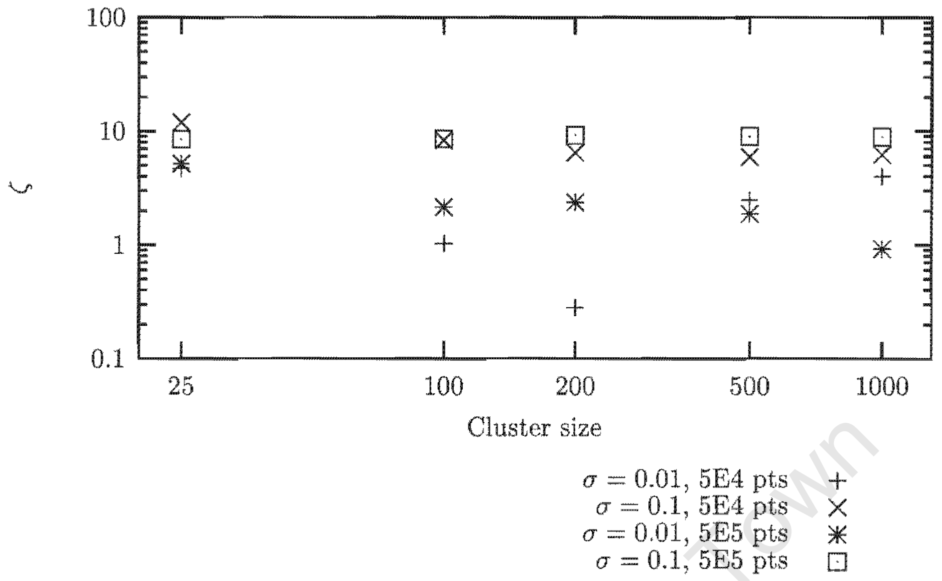


Figure 4.25: ζ for different parameter variation levels and numbers of points

a reasonably good model if the cluster size is correct, but for larger parameter variation the modelling error is very large. Also similarly to measurement noise, increasing the size of the data set does not seem to help.

From the experimental data above, it is clear that there is noise or parameter variation in the process, likely both. Although the simulations performed used approximate models of these two factors, it is clear that the coefficient fitting procedure is very sensitive to both noise and parameter variation.

Given the levels of noise and parameter variation present, no further attempts were made to control the experimental pendulum.

Chapter 5

Conclusions

From the arguments presented in this dissertation, the following conclusions have been reached:

1. Chaos control can stabilize unstable periodic orbits (UPO's) using small input perturbations.

   This was established in [OGY90]; the basic method developed there has been extended to be used in delay space and for higher periodic orbits.

   Chaos control is not just a theoretical or simulated curiosity, it has been successfully applied to experimental systems by other researchers. It has not yet, to this author's knowledge, been applied to an industrial system.

2. Local polynomial models can be used in simulation for chaos control.

   It has been demonstrated that local polynomial models can be used to stabilize UPO's in chaotic systems. These models can be fitted by analysing data from the system, and work in delay space.

   Additionally, when tested with the simulated driven pendulum, polynomial based control had a performance advantage over linear based control when the input was strongly constrained.

3. Chaos control performance in simulation is dependent on the constraint on system input.

   Performance of linear and polynomial controllers was strongly dependent on the input constraint; both performance in terms of mean time to control and noise robustness were far better when the input was less constrained.

4. Modelling of the driven pendulum for chaos control is very sensitive to measurement noise and parameter variation.

   Chaos control of an experimental driven pendulum was unsuccessful; simulating the effects of noise and parameter variation revealed that the modelling process is very sensitive to measurement noise and parameter variation.

   It is possible that parameter variation could have been reduced by operating the motor at a higher speed using a gearing system. Robust model-fitting for chaos control should be further investigated.

Bibliography

[AS92] Ralph Abraham and Christopher D. Shaw. *Dynamics: the geometry of behaviour*. Addison-Wesley, second edition, 1992.

[Bak95] Gregory L. Baker. Control of the chaotic driven pendulum. *Am. J. of Phys.*, 63(9):832–838, 1995.

[Bra58] Louis Brand. *Advanced Calculus*, chapter 4, pages 185–186. John Wiley and Sons, 1958.

[CCdF99] Gang Chen, Guanrong Chen, and Rui J. P. de Figueiredo. Feedback control of unknown chaotic dynamical systems based on time-series data. *IEEE Trans. on Circuits Syst. I*, 46(5):640–644, 1999.

[DN92] Ute Dressler and Gregor Nitsche. Controlling chaos using time delay coordinates. *Phy. Rev. Lett.*, 68(1):1–4, 1992.

[DRS90] W. L. Ditto, S. N. Rauseo, and M. L. Spano. Experimental control of chaos. *Phy. Rev. Lett.*, 65(26):3211–3214, 1990.

[ER85] J.-P. Eckmann and D. Ruelle. Ergodic theory of chaos and strange attractors. *Rev. of Mod. Phys.*, 57(3):617–656, 1985.

[GL97] Celso Grebogi and Ying-Cheng Lai. Controlling chaos in high dimensions. *IEEE Trans. on Circuits Syst. I*, 44(10):971–975, 1997.

[Gle94] Paul Glendinning. *Stability, instability and chaos: an introduction to the theory of nonlinear differential equations*. Cambridge University Press, 1994.

[GP83] Peter Grassberger and Itamar Procaccia. Characterization of strange attractors. *Phy. Rev. Lett.*, 50(5):346–349, 1983.

[GSDW92] Alan Garfinkel, Mark L. Spano, Willim L. Ditto, and James N. Weiss. Controlling cardiac chaos. *Science*, 257:1230–1234, 1992.

[Hun91] E. R. Hunt. Stabilizing high-period orbits in a chaotic system. *Phy. Rev. Lett.*, 67(15):1953–1955, 1991.

[Hut97] Michael Hutt. C implentation of Nelder-Mead simplex optimisation. `http://home.earthlink.net/ mfhutt/crosen.c`, November 1997.

[HW91] John H. Hubbard and Beverly H. West. *Differential equations: A dynamical systems approach, Part 1: Ordinary differential equations.* Springer-Verlag, 1991.

[IN98] Naohiko Inaba and Takashi Nitanai. OPF chaos control in a circuit containing a feedback voltage pulse generator. *IEEE Trans. on Circuits Syst. I*, 45(4):473–480, 1998.

[Kap98] Tomasz Kapitaniak. *Chaos for engineers: theory, applications and control.* Springer-Verlag, 1998.

[KGNP97] I. Z. Kiss, V. G'asp'ar, L. Nyikos, and P. Parmananda. Controlling electrochemical chaos in the copper-phosphoric acid system. *J. Phys. Chem.*, 101(46):8668–8674, 1997.

[KGOY93] Eric J. Kostelich, Celso Grebogi, Edward Ott, and James A. Yorke. Higher-dimensional targeting. *Phys. Rev. E*, 47(1):305–310, 1993.

[LK89] Daniel P. Lathrop and Eric J. Kostelich. Characterization of an experimental strange attractor by periodic orbits. *Phys. Rev. A*, 40(7):4028–4031, 1989.

[Lor93] Edward N. Lorenz. *The essence of chaos.* University of Washinton Press, 1993.

[Moo92] Francis C. Moon. *Chaotic and fractal dynamics*, chapter 1, page 41. John Wiley and Sons, 1992.

[OGY90] Edward Ott, Celso Grebogi, and James A. Yorke. Controlling chaos. *Phy. Rev. Lett.*, 64(11):1196–1199, 1990.

[PSRD93] P. Parmananda, P. Sherard, R. W. Rollins, and H. D. Dewald. Control of chaos in an electrochemical cell. *Phys. Rev. E*, 47:R3003–R3006, 1993.

[PTVF92] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C*. Cambridge University Press, second edition, 1992.

[SO95] Paul So and Edward Ott. Controlling chaos using time delay coordinates via stabilization of periodic orbits. *Phys. Rev. E*, 51:2955–2962, 1995.

[SOGY90] Troy Shinbrot, Edward Ott, Celso Grebogi, and James A. Yorke. Using chaos to direct trajectories to targets. *Phy. Rev. Lett.*, 65(26):3215–3218, 1990.

[Spr00] J. C. Sprott. Correlation dimension refinements. `http://sprott.physics.wisc.edu/chaos/corrdim.htm`, July 2000.

[SWB91] J. Singer, Y.-Z. Wanf, and Haim H. Bau. Controlling a chaotic system. *Phy. Rev. Lett.*, 66(9):1123–1125, 1991.

[The73] J. Thewlis. *Concise dictionary of physics*. Pergamon Press, 1973.

[TS98] Tadashi Tsubone and Toshimichi Saito. Stabilizing and destabilizing control for a piecewise-linear circuit. *IEEE Trans. on Circuits Syst. I*, 45(2):172–177, 1998.

[Vin97] Thomas L. Vincent. Control using chaos. *IEEE Contr. Sys.*, pages 65–76, December 1997.

[Wil97] Garnett P. Williams. *Chaos theory tamed*. Taylor and Francis Limited, 1997.

[WSSV85] Alan Wolf, Jack B. Swift, Harry L. Swinney, and John A. Vastano. Determining lyapunov exponents from a time series. *Physica D*, 16:285–317, 1985.

[YBOY98] Guohui Yuan, Soumitro Banerjee, Edward Ott, and James A. Yorke. Border-collision bifurcations in the buck converter. *IEEE Trans. on Circuits Syst. I*, 45(7):707–716, 1998.

Appendix A

Glossary of symbols

Symbol	Space	Definition
n	$\mathbb{N}$	state dimension
d_h	$\mathbb{N}$	input history
q	$\mathbb{N}$	$n + d_h + 1$
m	$\mathbb{N}$	period of UPO
$\boldsymbol{x}$	$\mathbb{R}^n$	state vector
u	$\mathbb{R}$	input
$\boldsymbol{v}$	$\mathbb{R}^{(1+d_h)}$	current and past input
$\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{v})$	$f : \mathbb{R}^n \times \mathbb{R}^{1+d_h} \to \mathbb{R}^n$	delay space dynamical map
$\boldsymbol{z}$	$\mathbb{R}^n$	point in UPO
$\boldsymbol{A}$	$\mathbb{R}^{n\times n}$	state linear coefficients
$\boldsymbol{B}$	$\mathbb{R}^{n\times(1+d_h)}$	input linear coefficients
$\boldsymbol{x}_a$	$\mathbb{R}^{n+d_h}$	augmented statevector
$\boldsymbol{A}_a$	$\mathbb{R}^{(n+d_h)\times(n+d_h)}$	augmented state linear coefficients
$\boldsymbol{b}_a$	$\mathbb{R}^{n+d_h}$	augmented state linear coefficients
p	$\mathbb{N}$	polynomial degree
r	$\mathbb{N}$	number of coefficients; $r = \binom{p+q}{q}$
$\boldsymbol{y}$	$\mathbb{R}^q$	state and input vector
$\boldsymbol{Z}$	$\mathbb{R}^{n\times r}$	polynomial coefficients
$\boldsymbol{Y}$	$\mathbb{R}^{n\times r_1}$	coefficients related to state only
$\boldsymbol{X}$	$\mathbb{R}^{n\times r_2}$	all coefficients related to input
$\phi_p(\boldsymbol{y})$	$f : \mathbb{R}^q \to \mathbb{R}^r$	polynomial evaluation vector function
$\psi_p(\boldsymbol{v}, \boldsymbol{x})$	$f : \mathbb{R}^{d_h} \times \mathbb{R}^n \to \mathbb{R}^{r_2}$	partial polynomial evaluation

Appendix B

Basic dynamical systems concepts

This appendix serves as a reference to basic concepts in the study of non-linear dynamics which are relevant to this dissertation. More formal treatments can be found in [Gle94] and [HW91] among many others. A pictorial introduction to dynamical systems is [AS92].

This appendix is based on [Gle94], except where otherwise noted.

B.1 Continuous time dynamics

If time is considered as a continuous value in a system, i.e. time $t \in \mathbb{R}$, differential equations are often used to model the system. A system in which dynamics occur are with respect to time only may be modelled with ordinary differential equations.

B.1.1 Ordinary differential equations

Any nth-order differential equation can be written as n first-order equations; such a system may be represented in vector form as

$$\begin{pmatrix} \dot{x_1} \\ \dot{x_2} \\ \vdots \\ \dot{x_n} \end{pmatrix} = \begin{pmatrix} f_1(x_1, x_2, \ldots, x_n, t) \\ f_2(x_1, x_2, \ldots, x_n, t) \\ \vdots \\ f_n(x_1, x_2, \ldots, x_n, t) \end{pmatrix}$$

which may be written as $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, t)$, with $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ and $\boldsymbol{f} = (f_1, f_2, \ldots, f_n)$.

The vector $\boldsymbol{x}$ is the *state* of the system; it is an element of the system's *state space*, also called the phase space.

It is possible for each element of the state vector to have some meaning, such as position, velocity or acceleration; transformation of the equations often prevents such simple (and useful) interpretations.

The state space is normally $\mathbb{R}^n$ or a subset thereof, but in general it can be more complicated. Each element of the state-vector may occupy a particular one dimensional space (e.g. the real line or the unit circle) and the state space is then the Cartesian product of the space of each of the elements of $\boldsymbol{x}$.

B.1.2 Solutions to ordinary differential equations

A curve $\boldsymbol{x}(t)$ satisfying the differential equations is a *solution* to the equations. The curve this solution traces out in state space is called a *solution trajectory*.

If $\boldsymbol{f}$ is continuously differentiable in $\boldsymbol{x}$ and t, then a solution $\boldsymbol{x}(t)$ corresponding to an initial condition $\boldsymbol{x}(t_0) = \boldsymbol{x}_0$ exists and is unique: this important result means that solution trajectories do not cross each other or meet.

The requirement that $\boldsymbol{f}$ be continuously differentiable leads to the useful uniqueness result but it does exclude some important systems, e.g. power electronics and other switched systems in which chaos control has been applied [TS98, YBOY98, IN98]. In this dissertation it is assumed that $\boldsymbol{f}$ is continuously differentiable.

For low order systems (one, two or three dimensions) it is possible and often illuminating to plot solution trajectories for a number of initial conditions; this is usually termed a *phase portrait*. Time is shown on such diagrams by an arrow on each trajectory showing which direction it moves in. For examples see Table B.1.

A system of equations $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, t)$ is said to be *autonomous* if $\boldsymbol{f}$ has no explicit dependence on time, otherwise it is *non-autonomous*. An autonomous system may be written as $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})$.

An n-dimensional non-autonomous system can be converted to an equivalent $(n + 1)$-dimensional autonomous one by including time in the state vector; this is done by adding an equation $\dot{x}_{n+1} = 1$ and substituting all occurrences of t in $\boldsymbol{f}$ by x_{n+1}.

The solutions $\boldsymbol{x}(t)$ for different initial conditions form a *flow* $\boldsymbol{\Phi}(\boldsymbol{x}, t)$ in the state space. $\boldsymbol{\Phi}(\boldsymbol{x}_0, t_0)$ is the value of the solution with initial condition $\boldsymbol{x}(0) = \boldsymbol{x}_0$ at time t_0.

A point $\boldsymbol{x}_*$ is said to be a *stationary point* of a system if $\boldsymbol{\Phi}(\boldsymbol{x}_*, t) = \boldsymbol{x}_*$ for all t, or equivalently if $\boldsymbol{f}(\boldsymbol{x}_*, t) = \boldsymbol{0}$ for all t.

An operating point in conventional control design is usually a stationary point.

A point $\boldsymbol{x}_*$ is *periodic* with period T if $\boldsymbol{\Phi}(\boldsymbol{x}_*, t) = \boldsymbol{\Phi}(\boldsymbol{x}_*, t+T)$ for all t and there is no $s, 0 \leq s < T$ such that $\boldsymbol{\Phi}(\boldsymbol{x}_*, t) = \boldsymbol{\Phi}(\boldsymbol{x}_*, t+s)$. A solution with initial condition $\boldsymbol{x}(0) = \boldsymbol{x}_*$ is a *periodic* solution—the trajectories corresponding to these solutions are closed curves in state space.

Limit cycles are a type of periodic solution. The goal of chaos control is often the stabilisation of such periodic behaviour.

A solution is *quasi-periodic* if it has frequency components which are irrational multiples of each other, e.g. $\boldsymbol{x}(t) = (\sin(t), \sin(\pi t))$—these solutions form n-toruses in state space. Quasi-periodic solutions are mentioned for completeness only.

Finally, any bounded solution (defined below) which is not stationary, periodic or quasiperiodic but is locally divergent (i.e. solutions starting close together diverge) is chaotic.

B.1.3 Stability of solutions

There are many different varieties of dynamical stability (at least 57, according to [Gle94]); three of the more commonly used are Lyapunov, quasi-asymptotic and asymptotic stability:

A point $\boldsymbol{x}$ is *Lyapunov stable* iff for all $\epsilon > 0$ there exists $\delta > 0$ such that if $\|\boldsymbol{x} - \boldsymbol{y}\|_2 < \delta$ then $\|\boldsymbol{\Phi}(\boldsymbol{x}, t) - \boldsymbol{\Phi}(\boldsymbol{y}, t)\|_2 < \epsilon$.

A point $\boldsymbol{x}$ is *quasi-asymptotically stable* iff there exists $\delta > 0$ such that if $\|\boldsymbol{x} - \boldsymbol{y}\|_2 < \delta$ then $\|\boldsymbol{\Phi}(\boldsymbol{x}, t) - \boldsymbol{\Phi}(\boldsymbol{y}, t)\|_2 \to 0$ as $t \to \infty$.

A point is *asymptotically stable* iff it is both Lyapunov and quasi-asymptotically stable. This is what is meant by "stability" in this dissertation.

Stability of a stationary point $\boldsymbol{x}_*$ of $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})$ may often (but not always) be determined by the linearisation of $\boldsymbol{f}$ about $\boldsymbol{x}_*$. $\boldsymbol{x}_*$ is stable if all the eigenvalues of $\partial \boldsymbol{f}/\partial \boldsymbol{x}$ evaluated at $\boldsymbol{x}_*$ have strictly negative real parts.

The eigenvalues of the linearisation may be used to classify the stationary point; some stationary points of interest are shown in Table B.1.

The table shows a node and vertex which are stable; phase portraits of unstable nodes and vertices may be obtained by reversing time. Time reversal changes the sign of the eigenvalues—notice that this does not change the nature of a saddle.

Stability of limit cycles can be analysed through Poincaré section; this is discussed in section B.2.3.

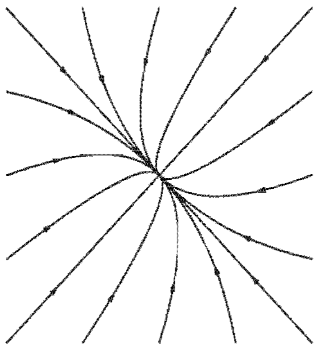
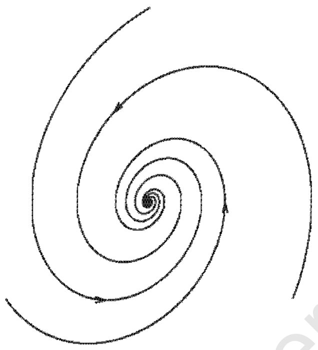
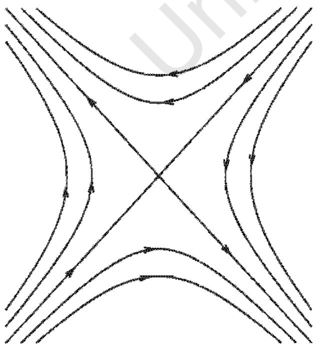
phase plot	continuous time	discrete time
	stable node	
	$\lambda_1, \lambda_2 \in \mathbb{R}$ $\lambda_1 < \lambda_2 < 0$	$\lambda_1, \lambda_2 \in \mathbb{R}$ $0 < \lambda_1 < \lambda_2 < 1$
	stable vertex	
	$\lambda_1, \lambda_2 \in \mathbb{C}$ $\lambda_1, \lambda_2 \notin \mathbb{R}$ $\Re(\lambda_i) < 0$	$\lambda_1, \lambda_2 \in \mathbb{C}$ $\lambda_1, \lambda_2 \notin \mathbb{R}$ $\|\lambda_i\| < 1$
	saddle node	
	$\lambda_1, \lambda_2 \in \mathbb{R}$ $\lambda_1 < 0 < \lambda_2$	$\lambda_1, \lambda_2 \in \mathbb{R}$ $0 < \lambda_1 < 1 < \lambda_2$

Table B.1: Important types of stationary and fixed points

Boundedness is a concept related to stability; a solution is *bounded* iff there exists α such that $\left\|\Phi(\boldsymbol{x},t)\right\|_2 < \alpha$ for all $t > 0$.

B.1.4 Attractors, repellors and saddles

An attractor is a region of state space which attracts nearby trajectories. Asymptotically stable bounded solutions are attractors; one may thus have point attractors, periodic attractors (limit cycles), quasi-periodic attractors and chaotic attractors.

An attractor has an associated *basin of attraction*: this is the set of all initial conditions giving rise to trajectories which tend to the attractor. An asymptotically stable linear system would have a point attractor with the entire state space as its basin of attraction; in non-linear systems it is possible to have multiple attractors each having its own basin of attraction.

A repellor is an attractor in negative time; repellors repel nearby trajectories. Once again one may have point, periodic, quasi-periodic and chaotic repellors.

There are also regions of state space which both attract and repel; an obvious example is the saddle node of Table B.1 but one may also have periodic, quasi-periodic and chaotic solutions with this attracting and repelling behaviour. Saddle cycles (i.e. saddle periodic trajectories) are of particular interest in chaos control because chaos control is often used to stabilise such cycles.

If post-transient experimental (real-world) data is analysed, only attractors and saddle structures can usually be detected.

B.1.5 Chaos in continuous time dynamical systems

If $\boldsymbol{f}(\boldsymbol{x})$ is continuously differentiable, then necessary conditions for chaos in the trajectories of $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})$ are a state space of dimension 3 or greater and non-linearity of $\boldsymbol{f}$ with respect to $\boldsymbol{x}$ [Moo92].

In the vicinity of chaotic attractors, trajectories are locally divergent although they remain globally bounded.

The chaotic attractor is ergodic: given a point $\boldsymbol{x}_0$ on the attractor, a trajectory near the attractor will eventually get arbitrarily close to $\boldsymbol{x}_0$. This means that a single trajectory will visits the neighbourhood of every point in the entire attractor. The way a trajectory "fills" a region of state space is similar to the behaviour of a random process and allows for statistical analysis of chaotic systems.

Two statistically based global properties of the attractor are the correlation dimension and the Lyapunov exponents.

Correlation dimension

Chaotic attractors are often fractals—geometrically complex objects which are assigned non-integer dimension. A discussion of this generalisation of dimension can be found in [Wil97] and references therein. Correlation dimension, originally defined in [GP83], is often used in the study of chaos because it is easy to compute and is not purely geometric—it contains dynamical information as well.

Given a trajectory $\boldsymbol{x}(t)$, let $\boldsymbol{y}(i)$ be the sampled time series $\boldsymbol{y}(i) = \boldsymbol{x}(i\tau)$ of length N. The correlation integral is then defined as

$$C(r) = \lim_{N\to\infty} \frac{1}{N^2} \sum_{\substack{i,j=0 \\ i\neq j}}^{N-1} \theta(r - \|\boldsymbol{y}(i) - \boldsymbol{y}(j)\|_2) \tag{B.1}$$

where θ is the unit step (Heaviside) function.

The correlation integral is the probability of two points in state space being within distance r of each other; this is a good example of how statistical methods are used to analyse chaotic data.

For small r, [GP83] claims that $C(r)$ behaves as a power law, i.e. $C(r) \propto r^\nu$. The correlation dimension ν is then

$$\nu = \lim_{r\to 0^+} \frac{\mathrm{d}\log(C)}{\mathrm{d}\log(r)} \tag{B.2}$$

(adapted from [Spr00]).

Correlation dimension is useful for estimating the correct embedding dimension for a delay space—see section B.1.6 for more on this.

Lyapunov exponents

Lyapunov exponents are a generalised stability measure and are in some sense a generalisation of the eigenvalues of a stationary point [Kap98]. Most definitions of Lyapunov exponents are a little involved—see for instance [WSSV85], [ER85] and [Kap98]. The definition below is adapted from [ER85].

Given a trajectory $\boldsymbol{x}(t)$ let $\boldsymbol{y}(i)$ be the sampled discrete time series $\boldsymbol{y}(i) = \boldsymbol{x}(i\tau)$. Defining a matrix $\boldsymbol{L}$ as

$$\boldsymbol{L} = \lim_{N\to\infty} \frac{1}{N} \sum_{i=0}^{N} \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\boldsymbol{y}(i)} \tag{B.3}$$

and defining κ_j as the eigenvalues of $\exp(\boldsymbol{L})$, the Lyapunov spectrum λ_j is

$$\lambda_j = \frac{\log_2(|\kappa_j|)}{\tau} \tag{B.4}$$

ordered such that $\lambda_j \geq \lambda_{j+1}$. Using a base 2 logarithm the Lyapunov exponents are measured in bits per unit time—one could use base e to measure in nats per unit time.

B.1.6 Reconstruction of the state space

Most of this section is based on [Kap98] and [Wil97].

It is common to have access only to a scalar measurement of a system rather than the full state. From a time series of scalar values it is possible to form a time series of vectors; these vectors are in a *delay space* which has some properties in common with the state space. Most importantly for this dissertation, the delay space vector often provides sufficient information for chaos control.

Given a scalar function of time $z(t)$ which is the output of a system with n states, the delay vector at time t is $(z(t), z(t-\tau), \ldots, z(t-(m-1)\tau))$ where m is the dimension of the delay space (also called the embedding dimension) and τ is the lag time. The invariance of certain characteristic properties of the system, e.g. Lyapunov exponents and correlation dimension, is guaranteed for $m > 2n + 1$, although it is common for smaller values of m to give acceptable results.

Choosing τ and m

τ may be set to the smallest value of time for which the time series being investigated has zero autocorrelation; other techniques of choosing τ are given in [Wil97].

The embedding dimension m can be estimated by considering the correlation dimension ν for increasing m; ν will increase with m until the embedding dimension is sufficiently large to contain the entire attractor.

B.2 Discrete time dynamics

Any system which is sampled at intervals is a discrete time system—in such a system, time may be regarded as an integer denoted i.

B.2.1 Dynamical maps

A dynamical map is a system of the form

$$\boldsymbol{y}(i+1) = \boldsymbol{g}(\boldsymbol{y}(i), u(i))$$

where $\boldsymbol{y}(i)$ is the state vector and $u(i)$ the input value at time i. The state vector is an element of the system's n-dimensional state space D. Although scalar inputs are used in this dissertation, in general one can work with multi-input maps.

A trajectory of a discrete time system is a sequence $\boldsymbol{y}(i)$ generated by the map $\boldsymbol{g}$; for low-dimensional systems these points may be plotted to give the discrete time analogy to a phase portrait. It is possible to join consecutive points in the sequence, but for chaotic systems this tends to obscure the diagram.

B.2.2 Features of dynamical maps

A *fixed point* of a map $\boldsymbol{g}$ is a point $\boldsymbol{y}_*$ such that $\boldsymbol{y}_* = \boldsymbol{g}(\boldsymbol{y}_*, 0)$. The stability of fixed points may often be determined by linearisation of the map about the fixed point. The eigenvalues of this linearisation correspond to z-plane poles of control theory.

Some types of fixed point and corresponding eigenvalues of linearisation for a two-dimensional state space are shown in Table B.1. The phase portraits in the table apply to continuous time systems only, but one can imagine a discrete system obtained by sampling a continuous system at a very high rate; this would give the corresponding appearance of continuity in a phase portrait.

More generally, an *m-periodic point* $\boldsymbol{y}_*$ is one such that $\boldsymbol{y}_* = \boldsymbol{g}^{\langle m \rangle}(\boldsymbol{y}_*, 0)$. An *m-periodic orbit* is a sequence of points $\boldsymbol{y}_*(0), \boldsymbol{y}_*(1), \ldots, \boldsymbol{y}_*(m-1)$ such that

$$\begin{aligned} \boldsymbol{y}_*(0) &= \boldsymbol{g}(\boldsymbol{y}_*(m), 0) \\ \boldsymbol{y}_*(i+1) &= \boldsymbol{g}(\boldsymbol{y}_*(i), 0) \text{ for } 0 \leq i < m \\ \boldsymbol{y}_*(i) &\neq \boldsymbol{g}^{\langle k \rangle}(\boldsymbol{y}_*(i), 0) \text{ forall } k < m \end{aligned}$$

Notice that while every point in an m-periodic orbit is an m-periodic point, not every m-periodic point is a member of an m-periodic orbit (at least, not as the terms have been defined above) since any point which is m-periodic will also be (pm)-periodic (where $p = 1, 2, 3, \ldots$).

Stability of an m-periodic orbit may (usually) be determined from the linearisation of the map $\boldsymbol{g}^{\langle m \rangle}$ about a point in the orbit; it is possible to compute this Jacobian using the chain rule:

$$\left. \frac{\partial}{\partial \boldsymbol{y}} \boldsymbol{g}^{\langle m \rangle} \right|_{\boldsymbol{y}=\boldsymbol{y}_*(0), u=0} = \prod_{k=1}^{m} \left. \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{y}} \right|_{\boldsymbol{y}=\boldsymbol{y}_*(m-k), u=0} \tag{B.5}$$

The eigenvalues of this linearisation determine stability as for the fixed point (or 1-periodic orbit) case.

B.2.3 Poincaré sections and maps

By generalising the concept of sampling a signal at regular time intervals, a discrete map may be extracted from a continuous time system—this type of map is called a Poincaré map.

In a n-dimensional state space D, the map is created by considering how trajectories pierce an $(n-1)$-dimensional surface. The Poincaré *section* is given by the piercings, while the Poincaré *map* is the map between successive piercings.

Let $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})$ be a first order system of differential equations with n-dimensional state space D and corresponding flow $\boldsymbol{\Phi}(\boldsymbol{x}, t)$. Let $p(\boldsymbol{x}) = 0$ be an $(n-1)$-dimensional surface in D; the Poincaré section P is defined by

$$P = \{\boldsymbol{x} \in D : p(\boldsymbol{x}) = 0 \text{ and } \boldsymbol{f}(\boldsymbol{x}) \cdot \nabla p(\boldsymbol{x}) > 0\}$$

$p(\boldsymbol{x})$ must satisfy $\boldsymbol{f}(\boldsymbol{x}) \cdot \nabla p(\boldsymbol{x}) \neq 0$; this means that the trajectory cannot have a turning or inflection point in the surface of section p.

Since P is $(n-1)$-dimensional, we can express its elements in a new (n-1)-dimensional space E; to convert between the spaces define a one-to-one mapping $\boldsymbol{\Psi} : P \to E$. The Poincaré map $\boldsymbol{\Gamma}$ can then be defined as

$$\begin{aligned} \boldsymbol{\Gamma} &= \{(\boldsymbol{y}(i), \boldsymbol{y}(i+1)) \in E \times E : \boldsymbol{y}(i+1) = \boldsymbol{\Psi}(\boldsymbol{\Phi}(\boldsymbol{\Psi}^{-1}(\boldsymbol{y}(i)), t_1)) \\ &\quad \text{such that } t_1 \text{ is the smallest } t > 0 \text{ for which } \boldsymbol{\Phi}(\boldsymbol{\Psi}^{-1}(\boldsymbol{y}(i)), t) \in P\} \end{aligned}$$

The importance of Poincaré maps is that they transform periodic solutions in continuous time to periodic orbits in discrete time; a periodic solution is stable if its corresponding periodic orbit is stable.

Example B.1: Poincaré section in $\mathbb{R}^2$

Figure B.1 shows trajectory $\boldsymbol{x}(t) = (\exp(-t/10)\sin(t), \exp(-t/10)\cos(t))$ for $0 \leq t \leq 50$; this trajectory satisfies

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) = \begin{pmatrix} -0.1 & 1 \\ -1 & -0.1 \end{pmatrix} \boldsymbol{x}$$

Also shown in Fig. B.1 is the Poincaré surface of section $p(\boldsymbol{x}) = x_2$, and the elements of the section itself, marked by $\times$. Notice that only the upward going intersections are taken, corresponding to the condition that $\boldsymbol{f}(\boldsymbol{x}) \cdot \nabla p(\boldsymbol{x}) = -x_1 - x_2/10 > 0$; evaluated at $p(\boldsymbol{x}) = x_2 = 0$ this gives the condition $x_1 < 0$.

Figure B.2 shows the corresponding Poincaré map, obtained by setting $y(i) = \boldsymbol{\Psi}(\boldsymbol{x}) = x_1$. □

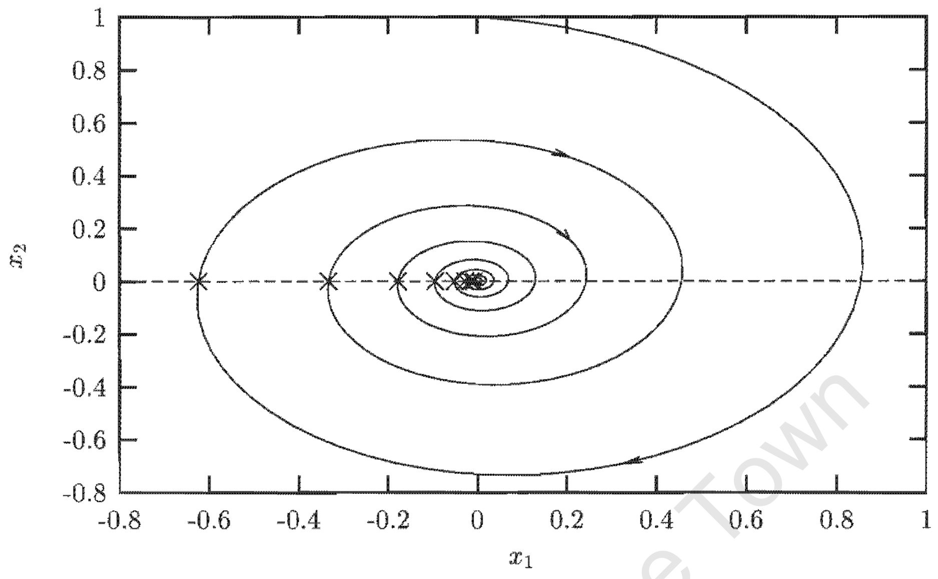


Figure B.1: A simple Poincaré section

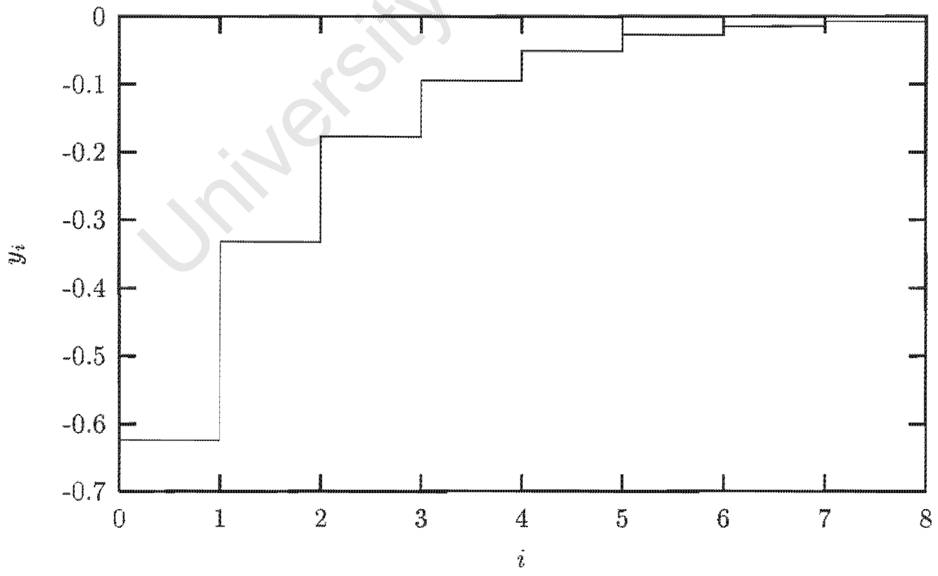


Figure B.2: The Poincaré map corresponding to Fig. B.1

Example B.2: Periodic time sampling as a special case of the Poincaré section

Given a trajectory $\boldsymbol{x}(t) \in \mathbb{R}^n$, define a new state vector

$$\boldsymbol{z}(t) = (\boldsymbol{x}(t), \sin(\omega t)) \in \mathbb{R}^{n+1}$$

The Poincaré section $p(\boldsymbol{z}) = z(n+1)$ and transformation $\boldsymbol{y}(i) = \boldsymbol{\Psi}(\boldsymbol{z}(t)) = (z_1, z_2, \ldots, z_n)$ with $i = 2\pi t/\omega$ give a sequence $\boldsymbol{y}(i)$ which is $\boldsymbol{x}(t)$ sampled with period $T = 2\pi/\omega$. □

B.2.4 Chaos in discrete time dynamics

Discrete time chaos has properties similar to continuous time chaos. Chaos can occur in non-linear discrete maps with state space of any dimension, unlike the continuous time case; however, in both cases one finds chaotic attractors (see e.g. Fig. C.1).

The correlation dimension of the attractor (Eq. (B.1)) may be computed in the obvious way for the discrete time case.

Lyapunov exponents are computed by a simple alteration of the method used for continuous time systems: if κ_j are the eigenvalues of $\boldsymbol{L}$ of Eq. (B.3)—not of $\exp(\boldsymbol{L})$, as before—the Lyapunov exponents are $\lambda_j = log_2(|\kappa_j|)$, measured in bits per iteration (or bits per cycle).

B.2.5 Reconstruction of the state space

Delay space reconstruction may be employed on discrete time data. If the scalar time series is $z(i)$, the delay space vector will simply be $(z(i), z(i-p), \ldots, z(i-mp))$ where m is the delay space dimension and p is the delay lag; p is usually 1.

B.3 Bifurcations in dynamical systems

A bifurcation is a qualitative change in a solution to a differential (or difference) equation which occurs because of a small parametric change to the system's defining equation.

Consider for instance the logistic map ([Lor93])

$$y(i+1) = ay(i)(1-y(i)) \tag{B.6}$$

for an initial condition $0 < y(0) < 1$. The steady-state behaviour for different values of a is shown in Fig. B.3.

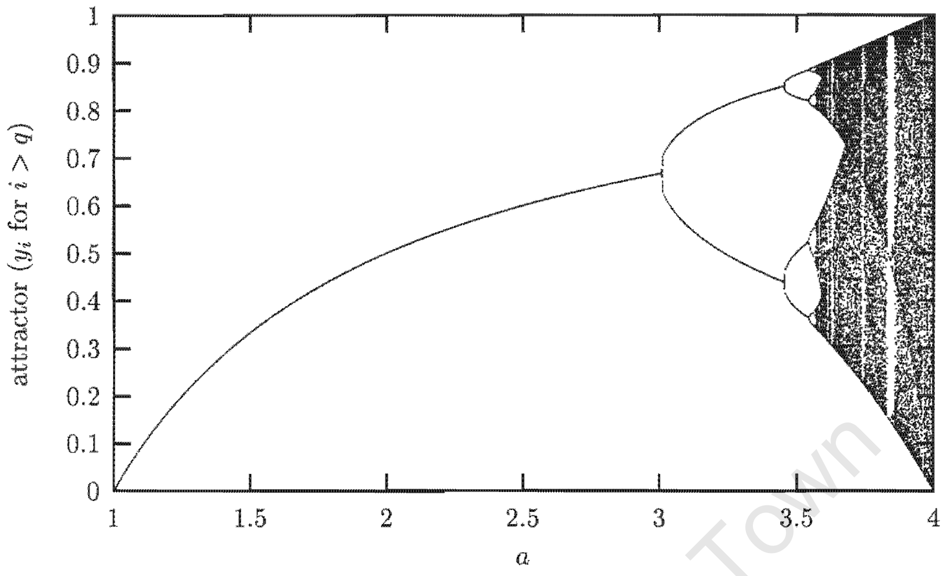


Figure B.3: Bifurcation diagram for the logistic map (Eq. B.6

Figure B.3 shows the chaotic attractor for different values of a. The diagram is obtained by computing a time series $y(i)$ for a particular value of a and plotting the points $(a, y(i))$ for $i > q$, where q is large enough that no transients are included, and repeating the procedure for different values of a.

For $1 < a < 3$ the logistic map has a single stable fixed point at $(a-1)/a$; however, at $a = 3$ a bifurcation occurs and a stable 2-periodic orbit is "born". As a increases, the periodicity of the stable attractor keeps doubling and the distance between successive doubling points becomes smaller and smaller until the system becomes chaotic, as seen in Fig. B.3.

Bifurcation patterns of this sort are very common in non-linear systems. A range of parameters for which the system is chaotic is called a chaotic regime; for chaos control to be applicable, the nominal control value (as well as all other system parameters) must be within such a chaotic regime.

Bifurcation diagrams may be used when designing systems to either purposefully avoid or cause chaos; in a sense the simplest form of "chaos control" is to ensure that system parameters are outside the chaotic regimes.

Appendix C

Chaos control worked examples

This chapter presents a worked example of chaos control of the Hénon map, with modelling by symbolic and data analysis, and an example of how to approximate symbolic analysis on the driven pendulum system.

C.1 Analysis and control of the Hénon map

The Hénon map was used in the seminal chaos control paper [OGY90], and as an example for chaos control in [Vin97].

The Hénon map may be defined as

$$\begin{pmatrix} x_1(i+1) \\ x_2(i+1) \end{pmatrix} = \begin{pmatrix} -1.4x_1^2(i) + 0.3x_2(i) + 1 + u(i) \\ x_1(i) \end{pmatrix} = \boldsymbol{g}(\boldsymbol{x}(i), u(i)) \tag{C.1}$$

This form is slightly different to that used in [OGY90] and [Vin97].

The Hénon map is particularly simple to work with; the second order Taylor approximation is the map itself, and the delay space $(x_1(i), x_1(i-1))$ is the same as the normal space (both facts are evident from Eq. C.1).

C.1.1 Symbolic analysis of the Hénon map

Solving for 1-periodic orbits is straightforward. From $\boldsymbol{x} = \boldsymbol{g}(\boldsymbol{x}, 0)$ we have $x_1 = x_2$ in the second row; substituting this into the first row gives the second order polynomial $-1.4x_1^2 - 0.7x_1 + 1 = 0$. This has two solutions, $x_1 = x_2 = (\sqrt{609} \pm 7)/28$, or approximately 0.631 and −1.131.

Both 1-periodic points are shown on Fig. C.1; only one is embedded in the attractor, namely $\boldsymbol{z}_* = (0.631, 0.631)$.

The 2nd degree approximation about this point can be found by direct

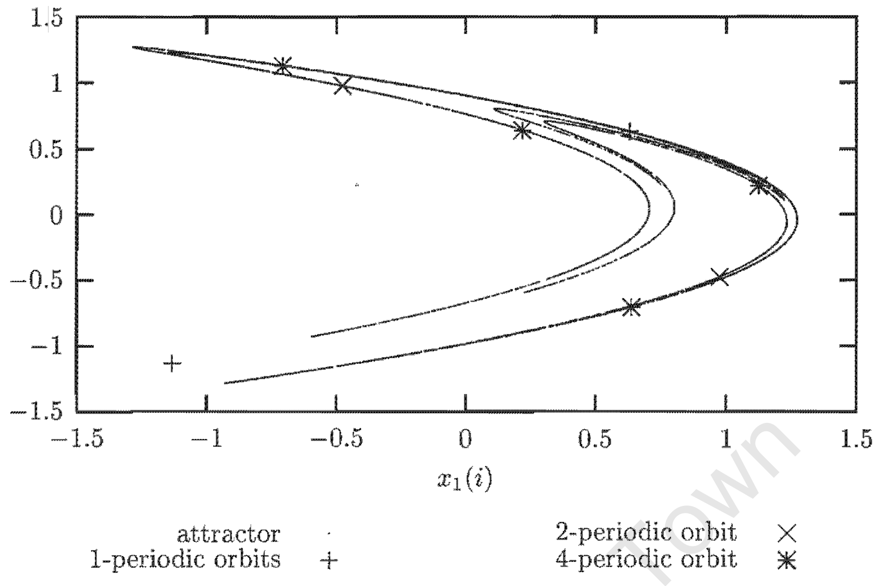


Figure C.1: The Hénon attractor and 1-, 2- and 4-periodic points of the Hénon map

differentiation, yielding

$$Z_* = \begin{pmatrix} 0.631 & -1.767 & 0.300 & 1 & -1.4 & 0 & 0 & 0 & 0 & 0 \\ 0.631 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

The approximation about any point is similar; only the zeroth and the $\partial g/\partial x_1 = -2.8x_1$ terms differ.

Finding 2-periodic points requires solving $\boldsymbol{x} = \boldsymbol{g}^{\langle 2\rangle}(\boldsymbol{x}, 0)$; four solutions were found using the MAXIMA computer algebra system (CAS). Two of these solutions are the 1-periodic points already found; the other two form a 2-periodic orbit $\boldsymbol{z}_\dagger(j)$, with $\boldsymbol{z}_\dagger(1) = (-0.476, 0.976)$ and $\boldsymbol{z}_\dagger(2) = (0.976, -0.476)$.

No 3-periodic orbits could be found by any means. The author could not find 4-periodic orbits using MAXIMA; however, one was discovered by the method of recurrent points (as discussed below). The 4-periodic orbit found is

$$\begin{aligned} \boldsymbol{z}_\ddagger(1) &= (-0.707, 1.125) \\ \boldsymbol{z}_\ddagger(2) &= (0.638, -0.707) \\ \boldsymbol{z}_\ddagger(3) &= (0.218, 0.638) \\ \boldsymbol{z}_\ddagger(4) &= (1.125, 0.218) \end{aligned}$$

C.1.2 Data analysis of the Hénon map

Data analysis was performed on two data sequences, $\boldsymbol{x}(i)$ and $\boldsymbol{x}_p(i)$ with $\boldsymbol{x}(i+1) = \boldsymbol{g}(\boldsymbol{x}(i), 0)$ and $\boldsymbol{x}_p(i+1) = \boldsymbol{g}(\boldsymbol{x}_p(i), u_p(i))$ for $1 \leq i < 50\text{E}3$. $u_p(i)$ is a sequence of independent normally distributed random values with mean 0 and variance 1E−3.

The (m, δ) clusters of $\boldsymbol{x}(i)$ for $m = 1, 2, 3, 4$ are shown in Fig. C.2; in each case δ was chosen to select 100 points.

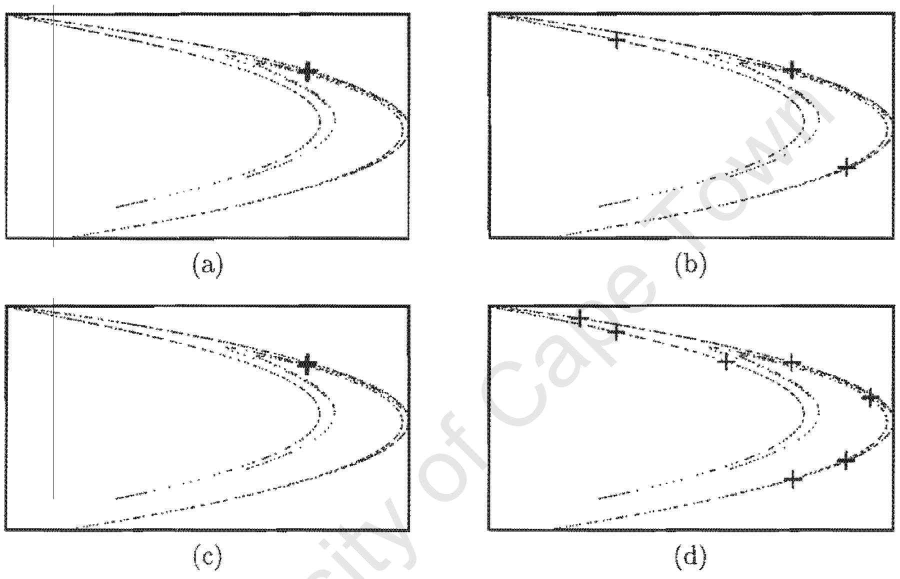


Figure C.2: Hénon attractor showing (a) $(1, \delta)$, (b) $(2, \delta)$, (c) $(3, \delta)$ and (d) $(4, \delta)$ points

In Figure C.2 we can see that among the $(2, \delta)$ points are some $(1, \delta)$ points, and among the $(4, \delta)$ points are both $(2, \delta)$ and $(1, \delta)$ points. The $(3, \delta)$ points appear to all be $(1, \delta)$ points, hence the assumption that there are no 3-periodic orbits embedded in the attractor.

Deciding on numbers of clusters is quite straightforward; 1, 3 and 7 for the 1-, 2 and 4-periodic cases respectively. The coefficient fitting is performed exactly as described in section 2.3.2; since the model is a perfect fit, the parameters are extremely close to their actual values.

C.1.3 Results of control of the Hénon map

Figure C.3 shows the results of chaos control for the three different UPO's found; the controller was a POC with $\kappa = u_{\max} = 0.1$.

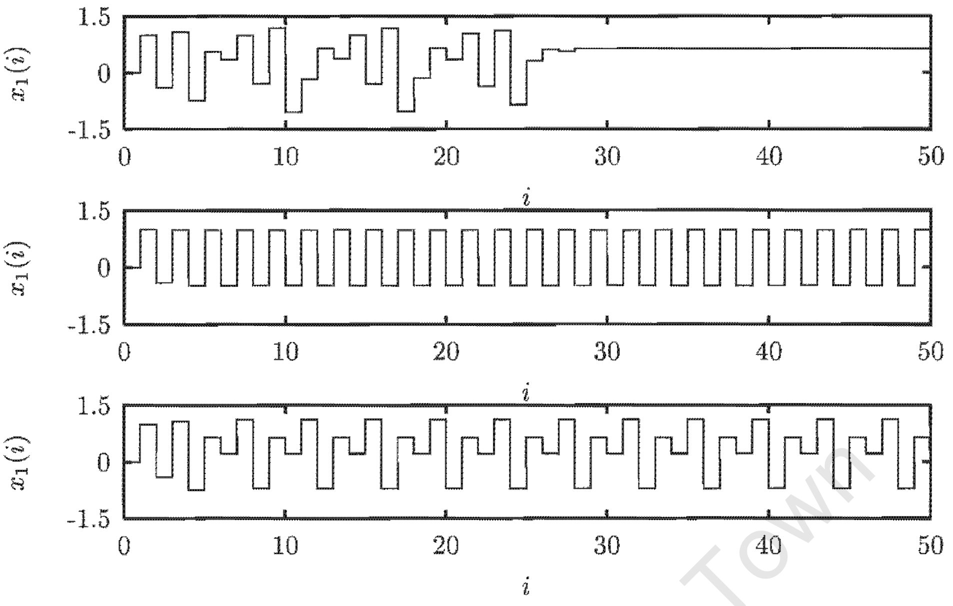


Figure C.3: Chaos control of the Hénon map

Chaos control is successful in all cases. Figures C.4, C.5 and C.6 show mean time to stabilisation from 20 different initial conditions on the chaotic attractor for the DLQR, LOC and POC.

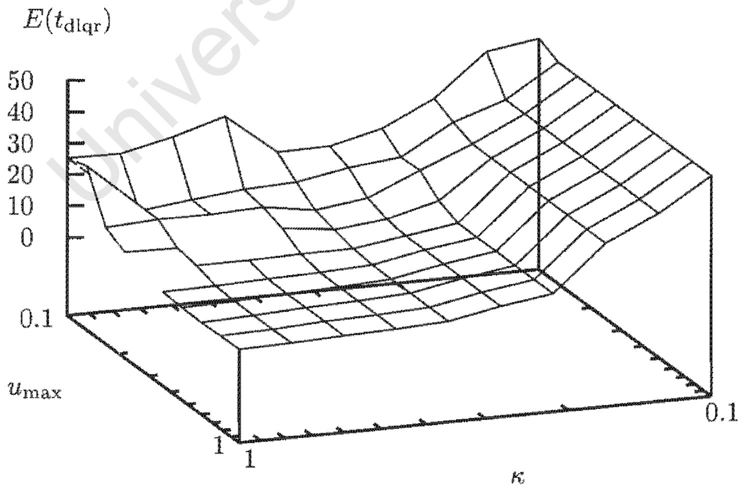


Figure C.4: Mean time to stabilisation for DLQR

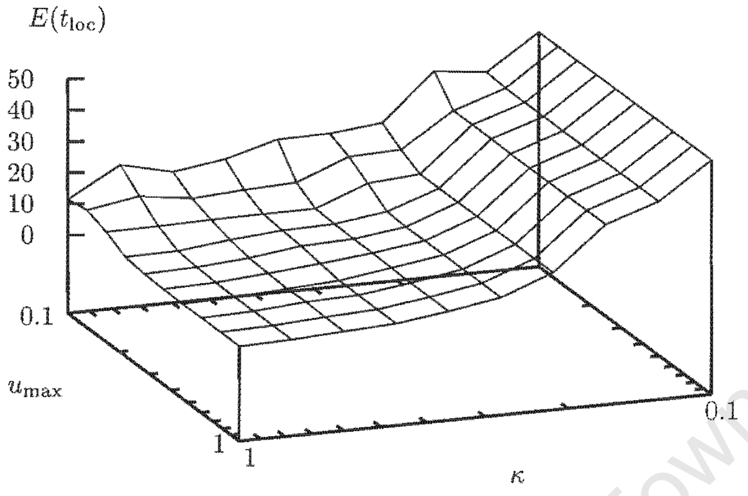


Figure C.5: Mean time to stabilisation for LOC

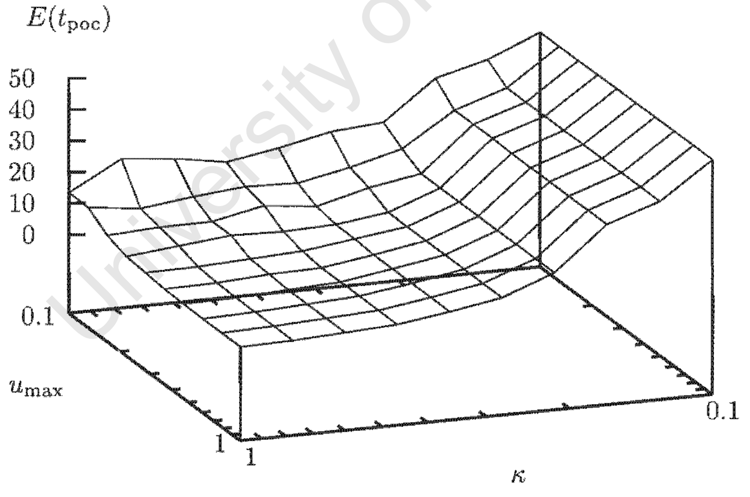


Figure C.6: Mean time to stabilisation for POC

The values of the two unplotted points in Fig. C.4 are very large; this is because the chaotic attractor has a relatively small basin of attraction, outside which the Hénon map is unstable; it is thus possible for the controller to make the system uncontrollable, and the corresponding time to stabilisation is set to the maximum simulation time (20 000 in this case).

Except for these isolated cases, the results are fairly similar; the neither the DLQR, LOC or POC have any clear advantage.

C.1.4 Effect of measurement noise on modelling

For comparison with the results of measurement noise on modelling the driven pendulum, below are similar results for the Hénon system.

In this test measurement additive noise of the form $\boldsymbol{x}_n(i) = \boldsymbol{x}(i) + \sigma\boldsymbol{n}(i)$ was used, where $n_1(i)$ and $n_2(i)$ were normally distributed random variables. This ignores the fact that $x_2(i) = x_1(i-1)$, which could be used in modelling to improve the results below.

The error measurement ζ is, as before, the sum of the absolute value of the elements of the difference between the true linearisation, $\boldsymbol{A}$ and the linearisation obtained from the noisy data; data sets of length 50 000 and 500 000 were used. The results are shown in Tables C.1 and C.2.

	Cluster size					
σ	200	500	1000	2000	5000	10 000
0.01	1.59	0.787	0.199	0.0857	0.0532	1.8
0.1	4.18	3.68	3.34	2.82	2.35	2.03

Table C.1: ζ for 50 000 data points

	Cluster size					
σ	200	500	1000	2000	5000	10 000
0.01	4.67	4.14	3.42	2.46	0.827	0.343
0.1	4.95	4.95	4.91	4.84	4.6	4.22

Table C.2: ζ for 500 000 data points

The results are similar to those of the driven pendulum; too much noise makes modelling impossible, and increasing the size of the dataset does not improve results.

C.2 Symbolic analysis of the driven pendulum

This section briefly demonstrates how to model a continuous time system in normal space by "pseudo-symbolic" analysis. It is assumed that the equations describing the system are known; the approach is to use numerical methods to approximate a chaotic map, and to find fixed points and derivatives from this map. The author has only investigated this method for normal space (i.e. non-delay space), but it should be extendable to delay space.

The driven pendulum system used is as defined in Eq. (4.2), with $b = 1/3.9$, $a = 1.5$ and $\omega = 2/3$ as in section 4.2.

The state vector is $\boldsymbol{x}(i) = \boldsymbol{w}(iT)$; the system input is as before deviation in driving amplitude, changed only at times iT and denoted $u(i)$. This implicitly defines map $\boldsymbol{x}(i+1) = \boldsymbol{g}(\boldsymbol{x}(i), u(i)$, which will be approximated by numerically solving the underlying differential equations.

To find a 1-periodic UPO we must solve $\boldsymbol{x}_* = \boldsymbol{g}(\boldsymbol{x}_*, 0)$; one obvious approach is to refine a UPO found by recurrent points (section 2.3.1) by using it as an initial estimate in minimizing $\|\boldsymbol{x}_* - \boldsymbol{g}(\boldsymbol{x}_*, 0)\|_2$.

In normal space, the method of recurrent points yields two UPO's, one of which is $\boldsymbol{z}_a = (1.977038, 1.846086)$; the refinement describes above yields $\boldsymbol{z}_f = (1.997501, 1.846086)$.

The linear coefficients may be found by approximating the derivative. In this case this yields

$$\boldsymbol{A}_f = \begin{pmatrix} -1.49175 & -3.41586 \\ -1.86090 & -4.32097 \end{pmatrix}, \boldsymbol{B}_f = \begin{pmatrix} -1.15438 \\ -1.08132 \end{pmatrix}$$

The linearization obtained by fitting the model to chaotic data is

$$\boldsymbol{A}_a = \begin{pmatrix} -1.49943 & -3.39697 \\ -1.88312 & -4.27279 \end{pmatrix}, \boldsymbol{B}_a = \begin{pmatrix} -1.14265 \\ -1.11378 \end{pmatrix}$$

The recurrent points solution is not far from the refined solution (presumed to be more accurate).

Appendix D

Technical details of the driven pendulum system

This appendix presents details of the experimental pendulum system.

The pendulum system is shown in Fig. D.1 and D.2.

The motor used was a Maxon Motor model S 2332-966-51.276-200, which is a DC motor with an integrated tachometer. Some relevant specifications are:

Specification	Unit	Value
Motor speed constant	rpm/V	415
Power	W	12
Nominal voltage	V	12
Maximum speed	rpm	9200
Tachometer output	V/rpm	5.2E−4
Tachometer peak-to-peak ripple		6%

The rod was 130 mm long; the mass, which was a hexagonal nut, was 20 mm in diameter and its centre was 116 mm from the motor shaft. The rod's mass was 18.9 g and the nut's 19.8 g.

The motor was connected to a computer via some buffer electronics, depicted in Fig. D.3.

In the Fig. D.3, V_{DAC} and V_{ADC} are the connections to the digital to analog (DAC) and analog to digital (ADC) connections on the computer, respectively. V_{MOTOR} and V_{TACHO} are the connection to the motor and tachometer. The circuitry to the right of the dashed line was built by another student for a separate project. The simple electronics to the left of the dashed line are to scale the DAC output down, and to scale and filter (for anti-aliasing) the tachometer output. It is this scaling which is changed in the constant speed experiment (by replacing the 33 kΩ resistor with one of

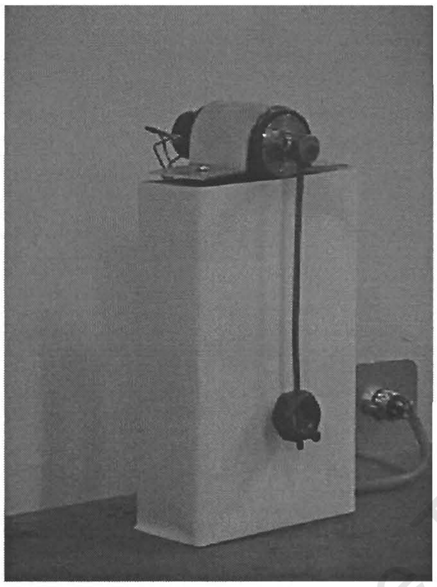


Figure D.1: Front of pendulum system, showing rod and mass

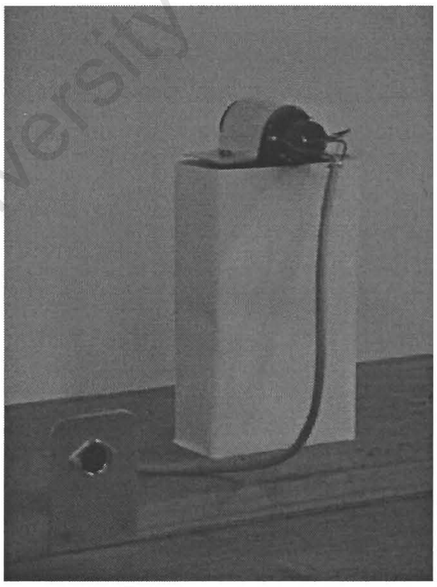


Figure D.2: Back of pendulum system, showing tachometer

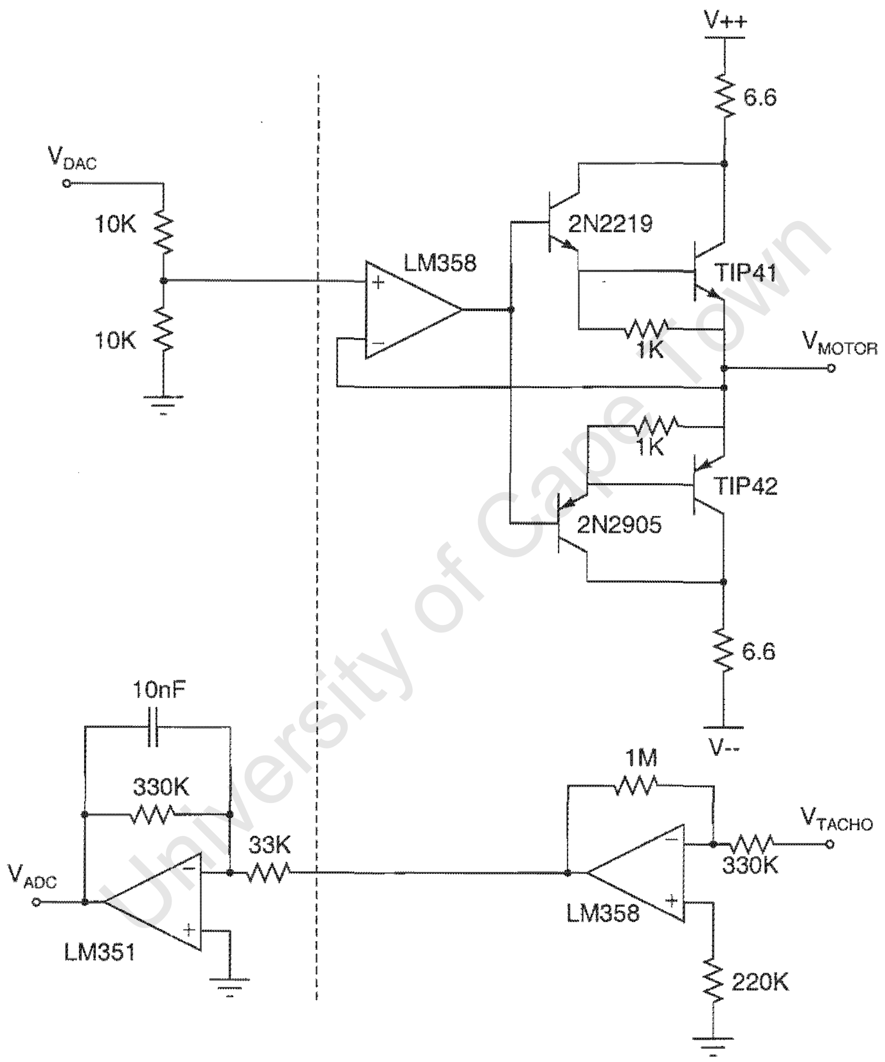


Figure D.3: Schematic of electronics linking computer to experimental system

330 kΩ). Notice that the tachometer output is already amplified by about 3 when it reaches this filter.

The computer used had a Intel Celeron processor running at 333 MHz; the DAC and ADC connections were provided by a Data Translation DT302 card, which has 12 bit resolution. The control software was run on Microsoft Windows NT 4.0.

Appendix E

Simulation performance results

Below are the tabulated results of the ideal performance tests, as described in section 4.2.4.

	$u_{\max}$				
	1.00E−2	1.29E−2	1.67E−2	2.15E−2	2.78E−2
5.00E−2	3.59E4	3.63E4	3.35E4	3.68E4	2.49E4
6.46E−2	2.80E4	2.82E4	2.71E4	2.97E4	1.70E4
8.34E−2	2.92E4	2.56E4	2.22E4	2.07E4	1.29E4
0.11	1.80E4	1.64E4	1.82E4	1.26E4	9.65E3
0.14	1.34E4	1.21E4	1.50E4	1.36E4	8.13E3
0.18	1.30E4	1.01E4	9.91E3	7.25E3	7.24E3
0.23	1.30E4	9.76E3	7.31E3	5.48E3	6.19E3
0.30	1.30E4	9.76E3	7.17E3	5.36E3	4.16E3
0.39	1.30E4	9.76E3	7.17E3	5.06E3	3.60E3
0.50	1.30E4	9.76E3	7.17E3	5.06E3	3.59E3

Table E.1: Mean time to stabilisation for DLQR part 1

	$u_{\max}$				
	3.59E−2	4.64E−2	5.99E−2	7.74E−2	0.10
5.00E−2	3.12E4	3.64E4	3.26E4	3.24E4	2.94E4
6.46E−2	2.11E4	2.00E4	1.78E4	1.77E4	1.35E4
8.34E−2	9.49E3	1.12E4	9.09E3	1.02E4	1.05E4
0.11	7.93E3	7.80E3	8.04E3	7.43E3	5.98E3
0.14	6.31E3	6.02E3	6.20E3	6.49E3	4.03E3
0.18	4.72E3	5.41E3	5.21E3	3.98E3	3.85E3
0.23	4.64E3	3.95E3	3.58E3	3.36E3	3.35E3
0.30	3.56E3	3.32E3	2.25E3	2.28E3	1.93E3
0.39	2.47E3	1.89E3	1.54E3	1.54E3	1.36E3
0.50	2.18E3	1.55E3	1.18E3	1.09E3	1.01E3

Table E.2: Mean time to stabilisation for DLQR part 2

	$u_{\max}$				
	1.00E−2	1.29E−2	1.67E−2	2.15E−2	2.78E−2
5.00E−2	3.20E4	3.55E4	3.52E4	3.20E4	2.76E4
6.46E−2	3.08E4	2.91E4	2.73E4	2.40E4	1.84E4
8.34E−2	2.75E4	2.95E4	2.48E4	2.09E4	1.19E4
0.11	1.72E4	1.65E4	1.53E4	1.42E4	9.91E3
0.14	1.37E4	1.19E4	1.21E4	1.19E4	1.07E4
0.18	1.30E4	1.01E4	1.02E4	7.72E3	5.92E3
0.23	1.30E4	9.76E3	7.57E3	5.71E3	5.14E3
0.30	1.30E4	9.76E3	7.24E3	5.41E3	4.38E3
0.39	1.30E4	9.76E3	7.17E3	5.06E3	3.100E3
0.50	1.30E4	9.76E3	7.17E3	5.06E3	3.92E3

Table E.3: Mean time to stabilisation for LOC part 1

	$u_{\max}$				
	3.59E−2	4.64E−2	5.99E−2	7.74E−2	0.10
5.00E−2	2.19E4	2.04E4	2.45E4	2.84E4	2.27E4
6.46E−2	1.94E4	2.16E4	1.68E4	1.65E4	1.73E4
8.34E−2	1.13E4	7.72E3	1.25E4	1.08E4	8.64E3
0.11	6.72E3	7.68E3	7.48E3	8.55E3	6.86E3
0.14	6.41E3	6.50E3	6.66E3	6.30E3	4.46E3
0.18	5.19E3	4.53E3	4.29E3	5.22E3	5.76E3
0.23	5.28E3	4.29E3	3.48E3	3.60E3	3.02E3
0.30	3.55E3	3.23E3	2.59E3	2.36E3	1.86E3
0.39	2.53E3	1.100E3	1.64E3	1.39E3	1.18E3
0.50	2.19E3	1.62E3	1.26E3	1.07E3	976.00E2

Table E.4: Mean time to stabilisation for LOC part 1

	$u_{\max}$				
	1.00E−2	1.29E−2	1.67E−2	2.15E−2	2.78E−2
5.00E−2	3.20E4	3.55E4	3.34E4	3.51E4	1.86E4
6.46E−2	3.08E4	2.73E4	2.67E4	2.71E4	1.94E4
8.34E−2	2.75E4	2.65E4	2.35E4	2.26E4	1.16E4
0.11	1.63E4	1.82E4	1.55E4	1.48E4	1.14E4
0.14	1.56E4	1.42E4	1.45E4	1.31E4	9.10E3
0.18	1.84E4	1.39E4	1.25E4	1.12E4	8.44E3
0.23	1.78E4	1.38E4	1.16E4	8.05E3	6.94E3
0.30	1.55E4	1.17E4	8.46E3	6.17E3	5.21E3
0.39	1.64E4	1.51E4	8.95E3	6.98E3	5.22E3
0.50	1.54E4	1.02E4	7.64E3	6.32E3	4.95E3

Table E.5: Mean time to stabilisation for POC part 1

	$u_{\max}$				
	3.59E−2	4.64E−2	5.99E−2	7.74E−2	0.10
5.00E−2	1.21E4	1.48E4	1.18E4	1.08E4	1.26E4
6.46E−2	8.32E3	1.05E4	1.13E4	1.13E4	1.05E4
8.34E−2	7.09E3	8.50E3	6.86E3	9.30E3	7.75E3
0.11	6.96E3	6.58E3	5.89E3	6.18E3	5.13E3
0.14	7.74E3	6.29E3	4.98E3	4.76E3	4.62E3
0.18	9.34E3	6.59E3	5.58E3	7.05E3	6.33E3
0.23	7.49E3	5.63E3	4.83E3	3.97E3	3.17E3
0.30	3.50E3	3.12E3	3.68E3	2.57E3	2.66E3
0.39	3.06E3	2.08E3	1.75E3	1.71E3	1.54E3
0.50	2.69E3	1.93E3	1.51E3	1.23E3	1.09E3

Table E.6: Mean time to stabilisation for POC part 1

Appendix F

Clustering points

This discussion describes the two methods used to separate (m, δ) points into clusters, each presumed to correspond to an m-periodic point.

The first method is, given a list of (m, δ) points, simply to find the all (m, δ) points within distance η of the first available point, using the Euclidean norm. All points thus selected are removed from the list of available points and the procedure repeated. This method is simple, but it is not obvious how to choose η; for 2-dimensional systems plotting the (m, δ) points can help.

A method which gives guidance in choosing a clustering threshold is described below. This method could not be used for large numbers of (m, δ) points because of computational complexity.

Given a vector sequence $\boldsymbol{x}(i)$, the n (m, δ) points may be indexed by an integer sequence $\alpha(p)$, $p = 1, \ldots, n$.

Let $\beta(p)$ be a rearrangement of $\alpha(p)$ such that

1. $\beta(1) = \alpha(1)$
2. if $p < q$, then for all $r < p$

$$\left\|\boldsymbol{x}(\beta(r)) - \boldsymbol{x}(\beta(p))\right\|_2 \leq \left\|\boldsymbol{x}(\beta(r)) - \boldsymbol{x}(\beta(q))\right\|_2$$

The sequence $\beta(p)$ is constructed sequentially; the next index to be added is always the index of the (m, δ) point not yet in the sequence closest to points already in the sequence. In this way points close to $\boldsymbol{x}(\beta(1))$ are found early in the sequence, and in general points which are close together will be clustered in the sequence $\beta(p)$.

To separate the clusters, define $\gamma(q) = \left\|\boldsymbol{x}(\beta(q+1)) - \boldsymbol{x}(\beta(q))\right\|_2$, $q = 1, \ldots, n-1$. To divide $\beta(p)$ into k clusters, find the $k - 1$ largest values in $\gamma(q)$; let the corresponding indices be $q_1, \ldots, q_{k-1}$ ordered by $q_i < q_j$ for $i < j$

and let $q_0 = 0$ and $q_k = n$. The ith cluster is then indexed by $\beta(q_{i-1} + 1)$ through $\beta(q_i)$.

The value k may be chosen by examining the $\gamma(q)$ values sorted by decreasing magnitude; if the clusters are far apart relative to distances between points in the clusters, there should be a clear threshold in this sequence which indicates the number of clusters.

Appendix G

Source code

Source code for simulations, controller code, and the LaTeX source for this document are on the included CD-ROM. The simulations were run on a GNU/Linux system using Octave; they may work on other systems. The control code requires a Microsoft Windows NT system with a Data Translation DT302 card. Instructions for building and running the code are on the CD-ROM.

C code for the Nelder-Mead simplex optimisation was taken from [Hut97].