

# Chapter 7

## Controlling of Chaos

### 7.1 Introduction

In this chapter we discuss different strategies for controlling classical chaos (Ott et al [156], Kapitaniak [112], Schöll and Schuster [183], Zhang et al [227]). Controlling chaos plays a central role in engineering in particular in electronics and mechanical systems. Roughly speaking, there are two kinds of ways to control chaos: feedback control and non-feedback control.

The frame of a chaotic attractor is given by infinitely many unstable periodic orbits. The task is to use the unstable periodic orbits to control chaos. We describe the Ott-Grebogi-Yorke method which uses this fact. By applying small, judiciously chosen temporal perturbations to an accessible control parameter of the dissipative system, they demonstrated that an original chaotic trajectory can be converted to a desired fixed point or periodic orbit. The Ott-Grebogi-Yorke method [157] belongs to feedback control. We apply this method to one and two-dimensional maps. The maps under consideration are the logistic map and the Hénon map. Controlling can also be achieved by coupling periodic modulations to the control parameter. We study the Lorenz model with this technique in section 7.3. This is a non-feedback control. Resonant perturbation and control is considered in section 7.4 and applied to a driven anharmonic oscillator.

### 7.2 Ott-Yorke-Grebogi Method

#### 7.2.1 One-Dimensional Maps

The basic idea of the Ott-Yorke-Grebogi method [157] for stabilizing unstable periodic orbits embedded in a chaotic attractor can be understood by considering a simple model system. We consider the *logistic map*

$$x_{t+1} = f(x_t, r) = rx_t(1 - x_t) \quad r \in [3, 4]$$

where  $x_0 \in [0, 1]$  and thus  $x_t \in [0, 1]$ . The logistic map develops chaos via the period-doubling bifurcation route. The period-doubling cascade accumulates at  $r = r_\infty \approx 3.57\dots$ , after which chaos can arise. Consider the case  $r = 3.8$ . The system is apparently chaotic for this value of  $r$  and the chaotic attractor is contained in the interval  $[0, 1]$ . The chaotic attractor contains an infinite number of unstable periodic orbits embedded within it and they are dense in it. For example, a fixed point ( $r = 3.8$ )

$$x^* = 1 - \frac{1}{r} = 0.7368\dots$$

and a period-2 orbit,

$$x(1) \approx 0.3737\dots, \quad x(2) \approx 0.8894\dots$$

where

$$x(1) = f[x(2)], \quad x(2) = f[x(1)].$$

We find  $x(1)$  and  $x(2)$  by solving the quartic equation (second iterate)

$$x^* = r^2 x^* (1 - x^*) (1 - r x^* (1 - x^*)).$$

Besides the solutions  $x^* = 0$ ,  $x^* = 1 - 1/r$  we obtain  $x(1)$  and  $x(2)$  as fixed points of the second iterate of the map.

Suppose we want to avoid chaos at  $r = 3.8$ . In particular, we want trajectories resulting from a randomly chosen initial condition  $x_0$  to be as close as possible to the period-2 orbit assuming that this period-2 orbit gives the best system performance. We can choose the desired asymptotic state of the map to be any of the infinite number of unstable periodic orbits, if that periodic orbits gives the best system performance. To achieve this goal, we suppose that the parameter  $r$  can be finely tuned in a very small range around the value  $r_0 = 3.8$ , namely, we allow  $r$  to vary in the range  $[r_0 - \delta, r_0 + \delta]$ , where  $\delta \ll 1$ . Due to the ergodicity of the chaotic attractor, the trajectory that begins from an arbitrary value of  $x_0$  will fall, with probability one, into the neighbourhood of the desired period-2 orbit at some later time. The trajectory would diverge quickly from the period-2 orbit if we do not intervene. We program the parameter perturbations in such a way that the trajectory stays in the neighbourhood of the period-2 orbit for as long as the control is present. The small parameter perturbations will be time-dependent in general. The logistic map in the neighbourhood of a periodic orbit can be approximated by a linear equation expanded around the periodic orbit. Let the target period- $m$  orbit to be controlled be  $x(i)$ ,  $i = 1, \dots, m$ , where  $x(i+1) = f[x(i)]$  and  $x(m+1) \equiv x(1)$ . Assume that at time  $t$ , the trajectory falls into the neighbourhood of the  $i$ th component of the period- $m$  orbit. The linearized dynamics in the neighbourhood of the  $(i+1)$ th component is then

$$x_{t+1} - x(i+1) = \frac{\partial f(x, r)}{\partial x} \Big|_{x=x(i), r=r_0} (x_t - x(i)) + \frac{\partial f(x, r)}{\partial r} \Big|_{x=x(i), r=r_0} \Delta r_t$$

where the partial derivatives are evaluated at  $x = x(i)$  and  $r = r_0$ . Thus we obtain

$$x_{t+1} - x(i+1) = r_0(1 - 2x(i))(x_t - x(i)) + x(i)(1 - x(i))\Delta r_t.$$

We require  $x_{t+1}$  to stay in the neighbourhood of  $x(i+1)$ . Hence, we set

$$|x_{t+1} - x(i+1)| = 0.$$

From this condition it follows that

$$\Delta r_t = r_0 \frac{(2x(i) - 1)(x_t - x(i))}{x(i)(1 - x(i))}.$$

This equation holds only when the trajectory  $x_t$  enters a small neighbourhood of the period- $m$  orbit, i.e., when  $|x_t - x(i)| \rightarrow 0$ . Hence, the required parameter perturbation  $\Delta r_t$  is small. Let the length of a small interval defining the neighbourhood around each component of the period- $m$  orbit be  $2\epsilon$ . In general, the required maximum parameter perturbation  $\delta$  is proportional to  $\epsilon$ . Since  $\epsilon$  can be chosen to be arbitrarily small,  $\delta$  also can be made arbitrarily small. However, the average transient time before a trajectory enters the neighbourhood of the target periodic orbit depends on  $\epsilon$  (or  $\delta$ ). A larger  $\delta$  will mean a shorter average time for the trajectory to be controlled. Of course, if  $\delta$  is too large, nonlinear effects become important and the linear control strategy might not work. When the trajectory is outside the neighbourhood of the target periodic orbit, we do not apply any parameter perturbation and the system evolves at its nominal parameter value  $r_0$ . We usually set  $\Delta r_t = 0$  when  $\Delta r_t > \delta$ . The parameter perturbations  $\Delta r_t$  depend on  $x_t$  and are therefore time-dependent.

In the following C++ program `control1.cpp` we control the fixed point  $x^* = 1 - 1/r$ , where  $r = 3.8$  for the logistic equation.

```
// control1.cpp

#include <iostream>
#include <cmath>      // for fabs
using namespace std;

int main(void)
{
    double x = 0.28;      // initial value
    double r0 = 3.8;     // control parameter
    double T = 650;      // number of iterations
    double eps = 0.001;  // neighbourhood
    double xf = 1.0 - 1.0/r0; // fixed point
    double x1;
    int count = 0;
    double r = r0;
    for(int t=0;t<T;t++)
```

```

{
x1 = x; x = r*x1*(1.0-x1);
count++;
if(fabs(x-xf) < eps)
{
cout << "in eps neighbourhood" << endl;
cout << "count = " << count << endl;
double delta = r0*(2.0*xf-1.0)*(x-xf)/(xf*(1.0-xf));
cout << "delta = " << delta << endl;
r = r0+delta; // change to control parameter for controlling
cout << "x = " << x << endl;
} // end if
} // end for loop
return 0;
}

```

In the following C++ program `control2.cpp` we control a period two-orbit for the logistic map with  $r = 3.8$ .

```

// control2.cpp

#include <iostream>
#include <cmath> // for sqrt, fabs
using namespace std;

int main(void)
{
double x = 0.28; // initial value
double r0 = 3.8; // control parameter
double T = 353; // number of iterations
double eps = 0.001; // neighbourhood
int count = 0;
double x1;
// periodic points
double ab = (1.0+1.0/r0);
double xp1 = ab/2.0+sqrt(-ab/r0+ab*ab/4.0);
double xp2 = ab/2.0-sqrt(-ab/r0+ab*ab/4.0);
cout << "xp1 = " << xp1 << endl; // periodic point 0.88942
cout << "xp2 = " << xp2 << endl; // periodic point 0.373738

double r = r0;
for(int t=0;t<T;t++)
{
x1 = x; x = r*x1*(1.0-x1);
count++;
if((fabs(x-xp1) < eps) || (fabs(x-xp2) < eps))
{
cout << "in eps neighbourhood of xp1 or xp2" << endl;
cout << "count = " << count << endl;
}
}
}

```

```

if(fabs(x-xp1) < eps)
{
double delta1;
delta1 = r0*(2.0*xp1-1.0)*(x-xp1)/(xp1*(1.0-xp1));
cout << "delta1 = " << delta1 << endl;
r = r0+delta1;
cout << "x = " << x << endl;
}
else
{
double delta2;
delta2 = r0*(2.0*xp2-1.0)*(x-xp2)/(xp2*(1.0-xp2));
cout << "delta2 = " << delta2 << endl;
r = r0+delta2;
cout << "x = " << x << endl;
}
} // end for loop
return 0;
}

```

### 7.2.2 Systems of Difference Equations

Here we consider a system of difference equations (Ott et al [157])

$$\mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t, r)$$

where  $\mathbf{x}_t \in \mathbb{R}^n$ ,  $r \in \mathbb{R}$  and  $\mathbf{f}$  is sufficiently smooth in both variables. Here,  $r$  is considered a real parameter which is available for external adjustment but is restricted to lie in some small interval  $|r - r_0| < \delta$  around a nominal value  $r_0$ . We assume that the nominal system (i.e., for  $r = r_0$ ) contains a chaotic attractor. We vary the control parameter  $r$  with time  $t$  in such a way that for almost all initial conditions in the basin of the chaotic attractor, the dynamics of the system converge onto a desired time periodic orbit contained in the attractor. The control strategy is the following. We find a stabilizing local feedback control law which is defined on a neighbourhood of the desired periodic orbit. This is done by considering the first order approximation of the system at the chosen unstable periodic orbit. Here we assume that this approximation is stabilizable. Since stabilizability is a generic property of linear systems, this assumption is quite reasonable. The ergodic nature of the chaotic dynamics ensures that the state trajectory eventually enters into the neighbourhood. Once inside, we apply the stabilizing feedback control law in order to steer the trajectory towards the desired orbit.

Thus the control of unstable periodic orbits  $\mathbf{x}_{P,i} \in \mathbb{R}^n$  with  $i = 1, \dots, P$ , where  $P$  denotes the period length, can be achieved by small, exactly determined variations of the system parameter during each iteration  $t$  given by the control  $r_t \in \mathbb{R}$ . Hence

we have to consider the parametrised system

$$\mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t, r_t).$$

By changing  $r_t$  slightly the periodic points are also shifted slightly, i.e.,  $\mathbf{x}_{P,i}(r_t)$  for  $i = 1, \dots, P$ .

We describe the method as applied to the stabilization of fixed points (i.e., period one orbits) of the map  $\mathbf{f}$ . The consideration of periodic orbits of period larger than one is straightforward. Let  $\mathbf{x}^*(r)$  denote an unstable fixed point on the attractor. For values of  $r$  close to  $r_0$  and in the neighbourhood of the fixed point  $\mathbf{x}^*(r_0)$  the map can be approximated by the linear map

$$\mathbf{x}_{t+1} - \mathbf{x}^*(r_0) = A[\mathbf{x}_t - \mathbf{x}^*(r_0)] + B(r - r_0)$$

where  $A$  is an  $n \times n$  Jacobian matrix and  $B$  is an  $n$ -dimensional column vector

$$A := D_{\mathbf{x}}\mathbf{f}(\mathbf{x}, r), \quad B := D_r\mathbf{f}(\mathbf{x}, r).$$

The partial derivatives are evaluated at  $\mathbf{x} = \mathbf{x}^*(r_0)$  and  $r = r_0$ . We now introduce the time-dependence of the parameter  $r$  by assuming that it is a linear function of the variable  $\mathbf{x}_t$  of the form

$$r - r_0 = -K^T(\mathbf{x}_t - \mathbf{x}^*(r_0)).$$

The  $1 \times n$  matrix  $K^T$  is to be determined so that the fixed point  $\mathbf{x}^*(r_0)$  becomes stable. We obtain

$$\mathbf{x}_{t+1} - \mathbf{x}^*(r_0) = (A - BK^T)(\mathbf{x}_t - \mathbf{x}^*(r_0))$$

which shows that the fixed point will be stable provided the  $n \times n$  matrix

$$A - BK^T$$

is asymptotically stable: that is, all its eigenvalues have modulus smaller than unity. The solution to the problem of the determination of  $K^T$ , such that the eigenvalues of the matrix  $A - BK^T$  have specified values, is well known from control systems theory and is called the *pole placement technique*.

**Example.** Consider the *Hénon map* in the form

$$x_{1t+1} = a + bx_{2t} - x_{1t}^2, \quad x_{2t+1} = x_{1t}.$$

For  $b = 0.3$  and  $a = \bar{a} = 1.4$  there is an unstable saddle point contained in the chaotic attractor. This fixed point is given by

$$x_1^* = -c + \sqrt{c^2 + a}, \quad x_2^* = x_1^*$$

where  $c = (1 - b)/2$  and  $a \geq -c^2$ . We obtain

$$A = D_x \mathbf{f}(\mathbf{x}) = \begin{pmatrix} -2x_1 & b \\ 1 & 0 \end{pmatrix}, \quad B = D_r \mathbf{f}(\mathbf{x}) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

The eigenvalues and eigenvectors of  $A$  are given by

$$\lambda_s = -x_1 + \sqrt{b + x_1^2}, \quad \begin{pmatrix} \lambda_s \\ 1 \end{pmatrix}, \quad \lambda_u = -x_1 - \sqrt{b + x_1^2}, \quad \begin{pmatrix} \lambda_u \\ 1 \end{pmatrix}.$$

The  $2 \times 2$  matrix  $A - BK^T$  is given by

$$A - BK^T = \begin{pmatrix} -2x_1^* - k_1 & b - k_2 \\ 1 & 0 \end{pmatrix}. \quad \clubsuit$$

```
// Controlhenon.cpp

#include <iostream>
#include <cmath>      // for sqrt
using namespace std;

int main(void)
{
    double T = 3380;           // number of iterations
    double x = -0.760343, y = 1.40007; // initial values
    double a = 1.4, b = 0.3;   // parameter values
    double x1, y1; double a0 = a;
    // fixed point (fixed point is unstable)
    double c = (1.0-b)/2.0;
    double xf = -c+sqrt(c*c+a);
    double yf = xf;
    cout << "xf = " << xf << endl; // 0.883896
    cout << "yf = " << yf << endl; // 0.883896
    double k1 = -0.1, k2 = 1.3;
    int counter = 0;
    double eps = 0.005; // eps neighbourhood
    for(int t=0;t<T;t++)
    { x1 = x; y1 = y;
      x = a0-x1*x1+b*y1; y = x1;
      counter++;
      if((fabs(x-xf) < eps) && (fabs(y-yf) < eps))
      {
          cout << "in eps neighbourhood" << endl;
          cout << "counter = " << counter << endl;
          a0 = a-k1*(x-xf)-k2*(y-yf);
          cout << "x = " << x << endl; cout << "y = " << y << endl;
      }
    }
    cout << "x = " << x << endl; cout << "y = " << y << endl;
    return 0;
}
```

In the following C++ program we stabilize a fixed point of the Henon model using the OGY-method.

```
// henonogy.cpp

#include <iostream>
#include <cmath> // for sqrt
using namespace std;

double f(double p,double x,double y,double b)
{ return (p-x*x+b*y); }

double fp(double a,double b) // fixed point
{ return ((b-1.0)/2.0 + sqrt((1.0-b)*(1.0-b)/4.0+a)); }

double Es(double x,double b) // stable eigenvalue
{ return (-x+sqrt(x*x+b)); }

double Eu(double x,double b) // unstable eigenvalue
{ return (-x-sqrt(x*x+b)); }

int main(void)
{
    double a0 = 1.4, b = 0.3; // parameters
    double xinit = 0.6, yinit = 0.7; // initial values
    const double delta = 0.01; // maximum allowed perturbation
    double xi = xinit; double yi = yinit;
    int i = 0; // indentation
    double x_ = xinit; int k = 0;
    double at; // control parameter
    double xf = fp(a0,b); double yf = xf; // fixed point
    double lambdas = Es(xf,b); // eigenvalue stable
    double lambdau = Eu(xf,b); // eigenvalue unstable
    double fu = sqrt(lambdau*lambdau+1.0)/(lambdau-lambdas);
    double fu1 = fu;
    double fu2 = -lambdas*fu; // eigenvectors
    cout << "fixed point x,y: " << xf ;
    cout << " , " << yf << endl;
    cout << "eigenvalues: " << lambdas;
    cout << " , " << lambdau << endl;
    do
    {
        double deltap = -lambdau*(fu1*(xi-xf)+fu2*(yi-yf))/fu1;
        if(fabs(deltap) < delta)
        {
            at = a0+deltap;
            x_ = xi; xi = f(at,xi,yi,b); yi = x_;
            cout << "x" << i; cout << " : " << xi;
        }
    }
}
```

```

cout << " y" << i; cout << " : " << yi;
cout << " p  :" << at-a0 << endl;
i++; k++;
}
else
{ i++; k = 0; x_ = xi; xi = f(a0,xi,yi,b); yi = x_; }
}
while(k<=15);
return 0;
}

```

### 7.3 Time-Delayed Feedback Control

The time-delayed feedback control (or Pyragas method) ([170]) makes use of a control signal obtained from the difference between the current state and the state of the system delayed by one period of an unstable orbit, so that the signal vanishes when the target orbit is stabilized. The method does not require exact knowledge of the form of the periodic orbit or the system equations. For maps consider, as a simple example, the logistic map  $f_r(x) = rx(1-x)$ , where  $r \in [3, 4]$ . Then we have

$$x_{t+1} = rx_t(1-x_t) + K(x^* - f_r(x_t))$$

where  $x^*$  is the desired fixed point. We recall that the fixed points of the logistic map  $f_r$  are given by  $x^* = 0$  and  $x^* = 1 - 1/r$ . In the C++ program we consider the case  $r = 4$  and the desired fixed point is  $x^* = 3/4$ .

```

// DFC.cpp

#include <iostream>
using namespace std;

int main(void)
{
double x1, x0 = 0.1; double r = 4.0;
double eps = 0.02;
double K = 1.0 + 1.0/(2.0-r) + eps;
double xf = 1.0 - 1.0/r;
int T = 400;
for(int t=0;t<T;t++)
{
x1 = r*x0*(1.0-x0) + K*(xf-r*x0*(1.0-x0));
cout << "x1 = " << x1 << endl;
x0 = x1;
}
return 0;
}

```

Pyragas [169] has proposed a method for controlling chaos based on continuous-time perturbation using feedback. Later Chen and Dong [30] provided a rigorous basis for this control scheme. Consider an  $(n + m)$ -dimensional nonlinear dynamical system of the form

$$\frac{d\mathbf{u}}{dt} = \mathbf{f}(\mathbf{u}, \mathbf{v}, t), \quad \frac{d\mathbf{v}}{dt} = \mathbf{g}(\mathbf{u}, \mathbf{v}, t) + \mathbf{w}(\mathbf{u}, \mathbf{v}, t)$$

where  $\mathbf{u} \in \mathbb{R}^n$  ( $n > 0$ ) and  $\mathbf{v} \in \mathbb{R}^m$  ( $m > 0$ ) with a control input  $\mathbf{w} \in L_1(S \times \mathbb{R}^+)$  to the second subsystem with  $S \subseteq \mathbb{R}^n \times \mathbb{R}^m$ . Here  $L_1(S \times \mathbb{R}^+)$  denotes the set of all Lebesgue integrable functions on  $S \times \mathbb{R}^+$ . Suppose that the two nonlinear vector-valued functions

$$\mathbf{f} : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^+ \rightarrow \mathbb{R}^n, \quad \mathbf{g} : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^+ \rightarrow \mathbb{R}^m$$

are integrable and that the system has a unique solution trajectory  $(\mathbf{u}(t), \mathbf{v}(t))$  in  $S$  and  $t \geq t_0 \geq 0$ , for any given initial point  $(\mathbf{u}_0, \mathbf{v}_0, t_0) \in I \subset S \times \mathbb{R}^+$ . Selfcontrolling feedback then refers to a control input  $\mathbf{w}$  (satisfying the above requirements) which achieves stabilization of an existing unstable periodic trajectory of the system. Chen and Dong [30] presented some general results for dynamical systems of the above form.

**Example.** Consider the driven anharmonic oscillator

$$\frac{du_1}{dt} = u_2, \quad \frac{du_2}{dt} = -p_1 u_1 - u_1^3 - p u_2 + q \cos(\omega t)$$

which can show periodic orbits and chaotic behaviour depending on the parameters. Assume that  $(u_1^*(t), u_2^*(t))$  is a periodic solution of the system. By adding a linear correction term to the driven anharmonic oscillator we have in matrix notation

$$\begin{pmatrix} du_1/dt \\ du_2/dt \end{pmatrix} = \begin{pmatrix} u_2 \\ -p_1 u_1 - u_1^3 - p u_2 + q \cos(\omega t) \end{pmatrix} - \begin{pmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{pmatrix} \begin{pmatrix} u_1 - u_1^* \\ u_2 - u_2^* \end{pmatrix}$$

where the  $2 \times 2$  matrix on the right-hand side is called the feedback gain matrix. It follows that

$$\begin{aligned} \frac{du_1}{dt} &= -K_{11}u_1 + (1 - K_{12})u_2 + (K_{11}u_1^* + K_{12}u_2^*) \\ \frac{du_2}{dt} &= -(K_{21} + p)u_1 - u_1^3 - (K_{22} + p)u_2 + (K_{21}u_1^* + K_{22}u_2^*) + q \cos(\omega t). \end{aligned}$$

If the solution  $(u_1(t), u_2(t))$  coincides with  $(u_1^*(t), u_2^*(t))$  then the controlled driven anharmonic oscillator yields exactly the driven anharmonic oscillator, since then the control objective is satisfied and no correction is required. However, if the solution  $(u_1(t), u_2(t))$  differs from  $(u_1^*(t), u_2^*(t))$  then there is a linear correcting term added to each equation in the driven anharmonic oscillator which attempts to stabilize  $(u_1^*(t), u_2^*(t))$ . Next we show that we can find a feedback gain matrix  $K$  for the controller which stabilizes the periodic solution  $(u_1^*(t), u_2^*(t))$ . The Jacobian

matrix corresponding to the controlled driven anharmonic oscillator at the point  $(u_1^*(t), u_2^*(t))$  is given by

$$J_c(u_1^*, u_1^*) = \begin{pmatrix} -K_{11} & 1 - K_{12} \\ -(K_{21} + p_1) - 3u_1^{*2} & -(K_{22} + p) \end{pmatrix}.$$

The characteristic equation  $\det(sI_2 - J_c) = 0$  of the controlled driven anharmonic oscillator linearized about the periodic solution  $(u_1^*(t), u_2^*(t))$  is given by

$$s^2 + (p + K_{11} + K_{22})s + (K_{11}(p + K_{22}) + (1 - K_{12})(K_{21} + p_1 + 3u_1^{*2})) = 0$$

which is required to have all its roots located in the open left-hand  $s$ -plane in order for the controlled system to be stable. A necessary and sufficient condition for this to be the case for a second order system is that all coefficients of the polynomial  $\det(sI_2 - J_c)$  have the same sign

$$\begin{aligned} p + K_{11} + K_{22} &> 0 \\ K_{11}(p + K_{22}) + (1 - K_{12})(K_{21} + p_1 + 3u_1^{*2}) &> 0. \end{aligned}$$

Consider the special case  $K_{11} = K_{22} = 0$ . Since  $p > 0$  the first inequality is satisfied and the second inequality takes the form

$$(1 - K_{12})(K_{21} + p_1 + 3u_1^{*2}) > 0.$$

If we also assume that  $K_{12} = 0$  then we have

$$K_{21} > -p_1 - 3u_1^{*2}, \quad t \geq 0.$$

Next we have to linearize the driven anharmonic oscillator about  $(u_1^*(t), u_2^*(t))$  and then add the correction term. We obtain

$$\begin{pmatrix} d\tilde{u}_1/dt \\ d\tilde{u}_2/dt \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -p_1 - 3u_1^{*2} & -p \end{pmatrix} \begin{pmatrix} \tilde{u}_1 \\ \tilde{u}_2 \end{pmatrix} + \begin{pmatrix} 0 \\ w \end{pmatrix}$$

which is completely controllable. Therefore the controlled driven anharmonic oscillator is locally controllable by the feedback

$$w(t) = -K_{21}(u_1(t) - u_1^*(t)).$$

In the linearized controlled driven anharmonic oscillator we have

$$\tilde{u}_1 := u_1 - u_1^*, \quad \tilde{u}_2 := u_2 - u_2^*.$$

## 7.4 Small Periodic Perturbation

Control of chaos of dissipative systems with a control parameter can be realized by coupling weak periodic oscillations to the control parameter. We consider the

*Lorenz equations*

$$\begin{aligned}\frac{du_1}{dt} &= \sigma(u_2 - u_1) \\ \frac{du_2}{dt} &= ru_1 - u_1u_2 - u_2 \\ \frac{du_3}{dt} &= u_1u_2 - bu_3.\end{aligned}$$

One chooses the parameters of Lorenz equations where the system shows chaotic behaviour, for example  $\sigma = 10$ ,  $b = 0.4$  and  $r = 80$ . To couple the periodic oscillations to the control parameter, we replace the parameter  $r$  in the Lorenz system with

$$r \rightarrow r + k \cos(\Omega t)$$

where  $\Omega$  is the modulation frequency and  $k$  is the modulation amplitude. Thus we consider the time-dependent system

$$\begin{aligned}\frac{du_1}{dt} &= \sigma(u_2 - u_1) \\ \frac{du_2}{dt} &= (r + k \cos(\Omega t))u_1 - u_1u_2 - u_2 \\ \frac{du_3}{dt} &= u_1u_2 - bu_3.\end{aligned}$$

After controlling with the frequency  $\Omega = 6.28318$ , we obtained period one, period two and period four orbits for the values  $k = 2.11$ ,  $k = 3$  and  $k = 36.8$ , respectively.

```
// ControlLorenz.java

import java.awt.*;
import java.awt.event.*;
import java.awt.Graphics;

public class ControlLorenz extends Frame
{
    public ControlLorenz()
    {
        setSize(600,500);
        addWindowListener(new WindowAdapter()
        { public void windowClosing(WindowEvent event) { System.exit(0); } }); }

    public void fsystem(double h,double t,double u[],double hf[])
    {
        double sigma = 10.0, r = 80.0, b = 0.4;
        double Omega = 6.28318; double k = 2.0;
        hf[0] = h*sigma*(u[1]-u[0]);
        hf[1] = h*(-u[0]*u[2]+(r+k*Math.cos(Omega*t))*u[0]-u[1]);
        hf[2] = h*(u[0]*u[1]-b*u[2]);
    }
}
```

```

}

public void map(double u[],int steps,double h,double t,int N)
{
double uk[] = new double[N];
double tk;
double a[] = { 0.0, 1.0/4.0, 3.0/8.0, 12.0/13.0, 1.0, 1.0/2.0 };
double c[] = { 16.0/135.0, 0.0, 6656.0/12825.0, 28561.0/56430.0,
              -9.0/50.0, 2.0/55.0 };
double b[][] = new double[6][5];
b[0][0] = b[0][1]= b[0][2] = b[0][3] = b[0][4] = 0.0;
b[1][0] = 1.0/4.0; b[1][1] = 0.0; b[1][2] = 0.0; b[1][3] = 0.0;
b[1][4] = 0.0;
b[2][0] = 3.0/32.0; b[2][1] = 9.0/32.0;
b[2][2] = 0.0; b[2][3] = 0.0; b[2][4] = 0.0;
b[3][0] = 1932.0/2197.0; b[3][1] = -7200.0/2197.0;
b[3][2] = 7296.0/2197.0; b[3][3] = b[3][4] = 0.0;
b[4][0] = 439.0/216.0; b[4][1] = -8.0;
b[4][2] = 3680.0/513.0; b[4][3] = -845.0/4104.0; b[4][4] = 0.0;
b[5][0] = -8.0/27.0; b[5][1] = 2.0; b[5][2] = -3544.0/2565.0;
b[5][3] = 1859.0/4104.0; b[5][4] = -11.0/40.0;
double f[][] = new double[6][N];
for(int i=0;i<steps;i++)
{ fsystem(h,t,u,f[0]);
for(int k=1;k<=5;k++)
{ tk = t + a[k]*h;
for(int l=0;l<N;l++)
{ uk[l] = u[l];
for(int j=0;j<=k-1;j++) uk[l] += b[k][j]*f[j][l];
}
fsystem(h,tk,uk,f[k]);
}
for(int l=0;l<N;l++)
for(int k=0;k<6;k++) u[l] += c[k]*f[k][l];
}
}

public void paint(Graphics g)
{
g.drawRect(40,40,600,400);
int steps = 1; int N = 3;
double h = 0.005; double t = 0.0;
double u[] = { 0.8, 0.8, 0.8 }; // initial conditions
// wait for transients to decay
for(int i=0;i<4000;i++) { t += h; map(u,steps,h,t,N); }
t = 0.0;
for(int i=0;i<20000;i++)
{ t += h; map(u,steps,h,t,N);

```

```

int m = (int)(10.0*u[2]-400); int n = (int)(10.0*u[0]+200);
g.drawLine(m,n,m,n);
}
}

public static void main(String[] args)
{ Frame f = new ControlLorenz(); f.setVisible(true); }
}

```

## 7.5 Resonant Perturbation and Control

Another method to control a nonlinear dynamical system is to follow a prescribed goal dynamics (Hübler [101]). If the experimental dynamics is described by

$$\frac{d\mathbf{u}}{dt} = \mathbf{f}(\mathbf{u}, \mathbf{p}(t) + \mathbf{F}(t))$$

where both the set of parameters  $\mathbf{p}$  and the driving force  $\mathbf{F}$  depend only on time  $t$ , then the limiting behaviour of  $\mathbf{u}(t)$  can be made equal to a given goal dynamics  $\mathbf{y}(t)$ , i.e.,

$$|\mathbf{u}(t) - \mathbf{y}(t)| \rightarrow 0 \quad \text{as } t \rightarrow \infty$$

where

$$\frac{d\mathbf{y}}{dt} = \mathbf{g}(\mathbf{y}(t), t)$$

by an appropriate choice of  $\mathbf{F}$ . Complete entrainment occurs if both sets of flows are made equal, i.e.,

$$\mathbf{f}(\mathbf{y}, \mathbf{p}(t) + \mathbf{F}(t)) = \mathbf{g}(\mathbf{y}(t), t)$$

and if the special solution  $\mathbf{u}(t) = \mathbf{y}(t)$  is stable. Whether convergence occurs, i.e.  $|\mathbf{u}(t) - \mathbf{y}(t)| \rightarrow 0$  as  $t \rightarrow \infty$ , depends on the particular choice of  $\mathbf{F}$  and the initial conditions, i.e.,  $\mathbf{u}(0)$ . The regions of the  $\mathbf{u}$  state space such that convergence occur are called entrainment regions. The applied controls are not typically small and convergence to the goal is not assured.