ELEC9733 Real Time Computing and Control

Discrete Non-Minimal State Space Systems

1 Introduction

The purpose of a non-minimal state-space representation is to avoid the introduction of states which cannot be measured, and hence to provide a means to design output feedback controllers which do not require state estimation. The discrete non-minimal representation relies on present and past values of the input and output signals in place of the states. Under appropriate conditions, state feedback design methods can be followed in designing the controller.

2 NMSS Representation

Consider a system described by a discrete state space model:

$$\begin{aligned} x(t+1) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t) \end{aligned}$$

A method for obtaining the state vector x from the measurements y and actuations u is required. Consider the following sequence of calculations:

$$\begin{aligned} y(t) &= Cx(t) + Du(t) \\ x(t+1) &= Ax(t) + Bu(t) \\ y(t+1) &= Cx(t+1) + Du(t+1) \\ &= CAx(t) + CBu(t) + Du(t+1) \\ y(t+2) &= CAx(t+1) + CBu(t+1) + Du(t+2) \\ &= CA^2x(t) + CABu(t) + CBu(t+1) + Du(t+2) \\ \vdots &= \vdots \end{aligned}$$

It is apparent that

$$\begin{bmatrix} y(t) \\ y(t+1) \\ y(t+2) \\ \vdots \\ y(t+n-1) \end{bmatrix} = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{(n-1)} \end{bmatrix} x(t) + \begin{bmatrix} D & 0 & \dots & & \\ CB & D & 0 & \dots & \\ CAB & CB & D & 0 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ CA^{(n-2)}B & \dots & CAB & CB & D \end{bmatrix} \begin{bmatrix} u(t) \\ u(t+1) \\ u(t+2) \\ \vdots \\ u(t+n-1) \end{bmatrix}$$

With suitable definition of variables, this may be rewritten

$$Y = Wx + VU$$

As long as the process is observable, this provides a mechanism for calculation of $x(t)$

$$x = W^{-1}(Y - VU)$$

and it may also be noted that

$$y(t+n) = CA^n x(t) + \begin{bmatrix} CA^{(n-1)}B & \dots & CB \end{bmatrix} U + Du(t+n) = CA^n x(t) + V_n U + Du(t+n)$$

Defining

$$D_n = \begin{bmatrix} 0_m & I_m & \dots & 0_m \\ \vdots & \vdots & \vdots & \vdots \\ 0_m & 0_m & \dots & I_m \\ 0_m & 0_m & \dots & 0_m \end{bmatrix}$$

and

$$e_n = [0_m 0_m \dots I_m]^\top$$

permits the process finally to be described in non-minimal state-space form

$$\begin{bmatrix} Y(t+n) \\ U(t+n) \end{bmatrix} = \begin{bmatrix} D_n + e_n CA^n W^{-1} & 0_n + e_n(V_n - CA^n W^{-1} V) \\ 0_n & D_n \end{bmatrix} \begin{bmatrix} Y(t+n-1) \\ U(t+n-1) \end{bmatrix} + \begin{bmatrix} 0_m \\ \vdots \\ D \\ \vdots \\ I_m \end{bmatrix} u(t+n)$$

It may be noted that:

1. The state vector contains only measurements and their derivatives, and actuations and their derivatives.

2. The same state vector may be used for both the process and the dynamic controller.

3. The feedback regulator can be written in terms of polynomials which can finally be reduced to a series of transfer functions.

We will now proceed to illustrate the use of a discrete non-minimal state space system by means of an example.

3 Controlling a Basic System (Regulator Design)

Predictive controller designs are suitable for systems possessing a variety of characteristics which might be regarded as intractable for simpler designs. The following process exhibits many of these characteristics: non-minimum phase; time delay; integrating factor; and oscillatory behaviour.

$$Y(s) = e^{-4s}\frac{-s+1.5}{s(s^2+0.5s+1)}U(s) + \zeta(s) \tag{1}$$

The term $\zeta(s)$ usually represents slowly time-varying disturbances. It would typically incorporate a constant offset, possibly a trend and infrequent load disturbance changes. Occasionally it might include some other systematic disturbance, such as a sinusoid.

It is important to stress that models with very large gains should not occur. Actuation and measurement signals should be appropriately scaled, so that the full range of A/D and D/A converters are utilised. Further scaling may be necessary within the computer if subranges of the signals are used. This should mean that all models have steady state gains (after the removal of integrating or differentiating factors) which are close to unity. Failure to perform adequate scaling results in loss of precision in input/output conversion, and numerical procedures will be poorly conditioned.

A further consequence of scaling is that measurements should almost never exceed upper and lower bound constraints. Actuators, however, may be required to operate near constraint limits to achieve optimum performance.

Discrete process models adequately describe only a relatively narrow band of the entire process spectrum of frequencies (usually 2-3 decades). Toward each end of this frequency band, the model can be less reliable, particularly if it was obtained as the result of some identification procedure. It is prudent to select a sampling rate which will place those frequencies which determine the dominant process characteristics in the centre of the model spectrum. It may also be necessary to compromise this placement if there are other frequencies which are regarded as particularly important. For many processes where steady state set-point attainment is a control objective, this means providing an adequate description of the steady state (or lower frequency) behaviour, a requirement which results in the selection of a sampling rate that is slower than many would expect. For the given process, the natural frequency is $\omega = 1$ radian sec^{-1}. A control interval of 1 second is appropriate, resulting in 2π samples per cycle at the natural frequency. The process may be described by a discrete step-invariant model, which is appropriate for computer implementations of controllers. The "exact" discrete model is:

$$
\begin{aligned}
(1+\bar{A})y(t) &= Bu(t)+\eta(t) && (2)\\
(1+\bar{A}) &= 1-1.8828q^{-1}+1.4893q^{-2}-0.6065q^{-3} && (3)\\
B &= -0.1817q^{-4}+0.7726q^{-5}+0.4947q^{-6} && (4)
\end{aligned}
$$

3.1 Signal Filtering

Just a single filter is required for this process. Filtering is performed to remove the term $\eta(t)$. This is a special filter because it must be incorporated into the model so that the disturbance is cancelled by the controller. For this simple process, $\eta(t)$ represents only a constant offset, so selection of $\Delta = (1-q^{-1})$ is adequate.

$$
\begin{aligned}
\Delta(1+\bar{A})y(t) &= (1+A)y(t) = B\Delta u(t) && (5)\\
(1+A) &= 1-2.8828q^{-1}+3.3721q^{-2}-2.0958q^{-3}+0.6065q^{-4} && (6)
\end{aligned}
$$

The controller will calculate the filtered signal $\Delta u(t)$, and the actuation for the process is recovered by inverse filtering, which for cancellation of constant disturbance terms is equivalent to incorporating a simple integrator into the loop.

$$u(t) = \Delta^{-1}(\Delta u(t)) \tag{7}$$

3.2 Forming a State-Space Model

A state-space model may be constructed as described in the appendix (Equations **??** to **??**), so that the state vector contains past values of the process signals.

$$
\begin{aligned}
z(t) &= \mathcal{A}z(t-1)+\mathcal{B}\Delta u(t-1) && (8)\\
y(t) &= \mathcal{C}z(t) && (9)\\
\bar{s} &= s(t) && (10)
\end{aligned}
$$

$$
z(t) = \begin{bmatrix} \Delta u(t-1)\\ \Delta u(t-2)\\ \Delta u(t-3)\\ \Delta u(t-4)\\ \Delta u(t-5)\\ y(t)-s(t)\\ y(t-1)-s(t)\\ y(t-2)-s(t)\\ y(t-3)-s(t) \end{bmatrix} = \begin{bmatrix} \Delta u(t-1)\\ \Delta u(t-2)\\ \Delta u(t-3)\\ \Delta u(t-4)\\ \Delta u(t-5)\\ y(t)-\bar{s}\\ y(t-1)-\bar{s}\\ y(t-2)-\bar{s}\\ y(t-3)-\bar{s} \end{bmatrix} \quad \mathcal{B} = \begin{bmatrix} 1\\ 0\\ 0\\ 0\\ 0\\ 0\\ 0\\ 0\\ 0 \end{bmatrix} \tag{11}
$$

$$\mathcal{A} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -0.1817 & 0.7726 & 0.4947 & 2.8828 & -3.3721 & 2.0958 & 0.6065 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad (12)$$

$$\mathcal{C} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \quad (13)$$

Time delays increase the dimension of the state vector.

3.3 Calculating Controller Gains

The controller is required to minimise a cost function, subject to the constraint of the state space model.

$$Minimise \quad (14)$$

$$J = lim_{N\to\infty} \left\{ \sum_{t=0}^{N} \left[(Q_y\{y(t) - \bar{s}\})^2 + \Delta u(t)^2 \right] \right\} \quad (15)$$

$$Q_y = 1 \quad (16)$$

$$Constraint \quad (17)$$

$$z(t) = \mathcal{A}z(t-1) + \mathcal{B}\Delta u(t-1) \quad (18)$$

$$y(t) = \mathcal{C}z(t) \quad (19)$$

$$(20)$$

A fast, numerically efficient and computationally simple algorithm for obtaining a solution results from a square root formulation is described elsewhere.

The result of these computations is a vector of feedback gains. The same feedback gains could be obtained by any other LQ methodology, although care is required with eigenvalue methods as the system is not fully observable.

$$\Delta u(t) = -\mathcal{K}z(t) \quad (21)$$

$$u(t) = u(t-1) + \Delta u(t) \quad (22)$$

$$\mathcal{K} = \begin{bmatrix} 1.672 & 2.394 & 2.941 & 4.081 & 1.671 & 3.837 & -6.697 & 5.275 & -2.049 \end{bmatrix} \quad (23)$$

3.4 Closed Loop Performance

The performance of the adaptive controller is shown in Figure 1.

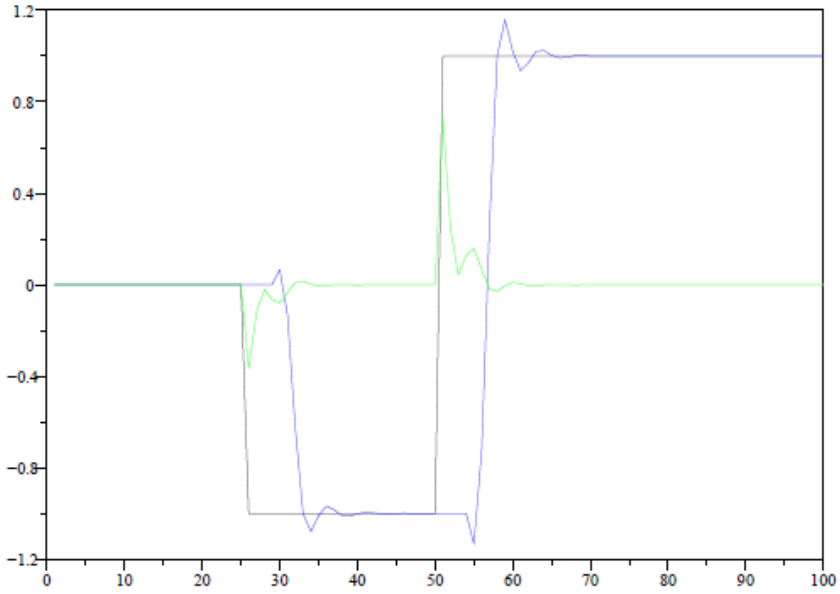

Figure 1: LQ Control of the Basic System

4 Servomechanisms

The controller designed for the basic system was a regulator, relying on integral control action to ensure attainment of a constant setpoint. Conventional servomechanism loops include some feedforward of set point changes to promote closer tracking of the setpoint signal. It is possible to exploit the properties of a discrete model to achieve this in different ways.

4.1 Simple Servomechanism Design

The usual method of achieving a servo design is to augment the state vector with present and past values of the setpoint signal.

$$\begin{bmatrix} z(t) \\ s(t) \\ s(t-1) \\ \vdots \\ s(t-n) \end{bmatrix} = \begin{bmatrix} \mathcal{A} & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 \end{bmatrix} \begin{bmatrix} z(t-1) \\ s(t-1) \\ s(t-2) \\ \vdots \\ s(t-n-1) \end{bmatrix} + \begin{bmatrix} \mathcal{B} \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \Delta u(t-1) + \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} s(t) \quad (24)$$

$$J = lim_{N\to\infty} \left\{ \sum_{t=0}^{N} \left[(y(t) - s(t))^2 + \Delta u(t)^2 \right] \right\} \quad (25)$$

$$\Delta u(t) \;=\; -\begin{bmatrix} \mathcal{K}_{fb} & \mathcal{K}_{sp} \end{bmatrix} \begin{bmatrix} z(t) \\ s(t) \\ s(t-1) \\ \vdots \\ s(t-n) \end{bmatrix} \tag{26}$$

The methods described above for obtaining the LQ solution may be used if the problem is re-defined trivially.

$$J \;=\; lim_{N\to\infty} \left\{ \sum_{t=0}^{N} \left[\bar{y}(t)^2 + \Delta u(t)^2 \right] \right\} \tag{27}$$

$$\bar{y}(t) \;=\; y(t) - s(t) \tag{28}$$

$$\;=\; \bar{\mathcal{C}} z(t) \tag{29}$$

It is also possible to incorporate into the state equations a more complex model of setpoint behaviour should this be known, by modifications to the $\bar{\mathcal{C}}$ vector, and even by defining dynamics affecting the state vector and the evolution of the set point.

This method requires that $\mathcal{K}_{fb}$ and $\mathcal{K}_{sp}$ be computed with some accuracy if there is to be no steady-state offset from setpoint. For slightly pathological systems, such accuracy is numerically difficult to achieve, and the servo design will fail at times when the regulator design is still adequate. Several re-formulations of the model are possible which produce outputs which have to be regulated to zero if set-point following is to be achieved, and which recover the numerical robustness of the regulator design.

4.2 Numerically Robust Servomechanism Design

The loop configuration depicted in Figure 2 requires a regulator design for the feedback controller, provides facilities for feedforward of setpoint changes, and also possesses other interesting properties.

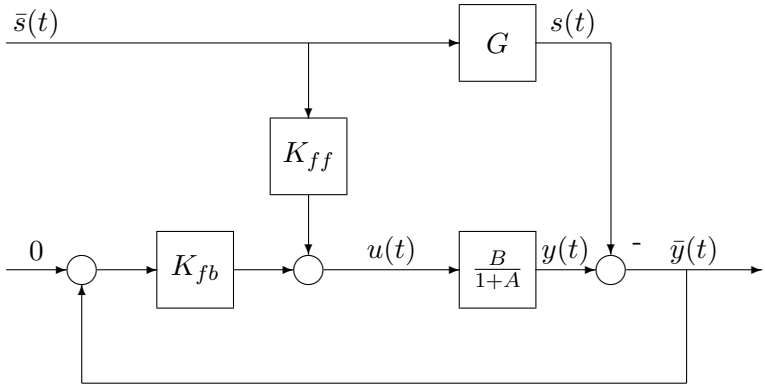

Figure 2: Discrete Servomechanism Loop Structure

A regulator is applied to maintain $\bar{y}(t)$ at 0, which will ensure that $y(t)$ will follow the required setpoint $s(t)$. The feedforward of $\bar{s}(t)$ affects only the transient response of the process. A transfer function G relates $s(t)$ to $\bar{s}(t)$. For the purpose of designing the required regulator, the process may be represented by

$$\begin{aligned} \bar{y}(t) &= \frac{B}{1+\bar{A}}u(t) - G\bar{s}(t) && (30)\\ (1+\bar{A})\bar{y}(t) &= Bu(t) - (1+\bar{A})G\bar{s}(t) && (31)\\ (1+A)\bar{y}(t) &= B\Delta u(t) - (1+\bar{A})G\Delta\bar{s}(t) && (32) \end{aligned}$$

The following points may be made regarding the system defined in Equations 31 to 32.

- In its simplest form, G may be just a time delay. In that case, $\bar{s}(t)$ is a preview of the required setpoint change.
- $\bar{s}(t)$ acts as a disturbance to the process. Feedforward action ensures that any deviation of $y(t)$ from $s(t)$ as a result of a change in $\bar{s}(t)$ is removed as quickly as possible. As a result, the controller is a two-degree-of-freedom mixed feedback/feedforward system.
- G may also be defined as either a rational transfer function or a moving average filter (defining an impulse response). In such an event, the closed loop system exhibits a form of model following. G defines the response of the model to a change in $\bar{s}(t)$ (i.e. $s(t)$), and $y(t)$ follows this response in a least squares sense.
- As G is known, only the parameters of the process (B and $(1+A)$) need to be identified. The model given above must then be used for designing the feedback and feedforward gains for the controller.
- The block diagram given in Figure 2 and the equations derived from it are important for determining the degree of the polynomials in the servo design, relying as they do on the choice of G.

4.3 Servomechanism Simulations

Figure 3 defines a system simulated to illustrate the servomechanism approach to tracking a changing setpoint. Figure 3 defines both the process model and the control loop structure used for the simulations.

The model equation may be written so that the setpoint preview $\bar{s}(t)$ is treated as a disturbance signal, for the purposes of performing a feedforward controller design.

$$\begin{aligned} s(t) &= (0.1q^{-1} + 0.2q^{-2}\bar{s}(t) + 0.4q^{-3} + 0.2q^{-4} + 0.1q^{-5})\bar{s}(t) && (33)\\ \Delta(1 - q^{-1} + .4q^{-2})y(t) &= (.3q^{-1} + .2q^{-2})\Delta u(t) && (34)\\ &\quad -(0.1q^{-1} + 0.1q^{-2} + 0.24q^{-3} - 0.12q^{-4} && (35)\\ &\quad +0.06q^{-5} - 0.02q^{-6} + 0.04q^{-7})\Delta\bar{s}(t) && (36) \end{aligned}$$

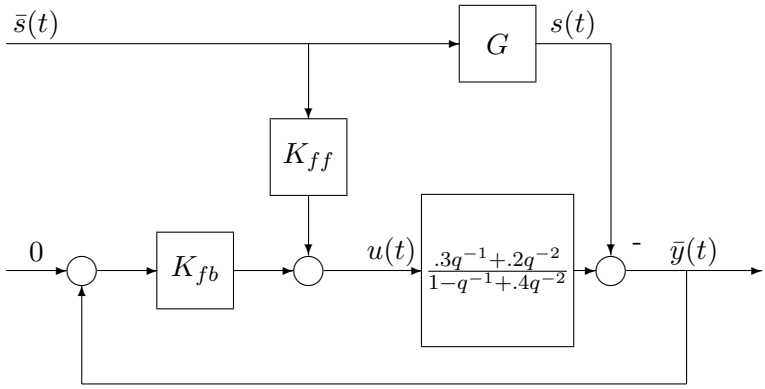

Figure 3: System to Demonstrate Servomechanism Behaviour

The setpoint filter has been selected in this instance as a moving average filter to shape the setpoint response to the form of a sigmoid curve. The model is subjected to filtering for offset removal, and is expressed in state space form, as described above for the basic system example (Equations 9 to 11).

For the purposes of designing the controller gains the state space model is constructed with a particular state vector in mind.

$$z(t) = \mathcal{A}z(t-1) + \mathcal{B}\Delta u(t-1) + \mathcal{D}\Delta\bar{s}(t) \tag{37}$$
$$\bar{y}(t) = \mathcal{C}z(t) \tag{38}$$

$$z(t) = \begin{bmatrix} \Delta u(t-1) \\ \bar{y}(t) \\ \bar{y}(t-1) \\ \bar{y}(t-2) \\ \Delta\bar{s}(t) \\ \Delta\bar{s}(t-1) \\ \Delta\bar{s}(t-2) \\ \Delta\bar{s}(t-3) \\ \Delta\bar{s}(t-4) \\ \Delta\bar{s}(t-5) \\ \Delta\bar{s}(t-6) \end{bmatrix} \tag{39}$$

The matrices $\mathcal{A}$, $\lfloor$ and $\mathcal{C}$ are formed to be consistent with the state vector. From the block diagram it is apparent that $\bar{y}(t) = y(t) - s(t)$. For this process, $\Delta = (1 - q^{-1}$, so that $u(t) = u(t-1) + \Delta u(t)$. For $Q_y = 2$ the controller gains are:

$$\mathcal{K} = \begin{bmatrix} 0.609 & 2.611 & -2.400 & 0.812 & -0.382 & -0.434 & -0.368 & 0.187 & -0.110 & -0.004 & -0.081 \end{bmatrix} \tag{40}$$

The closed loop performance of this controller is shown in Figure 4.

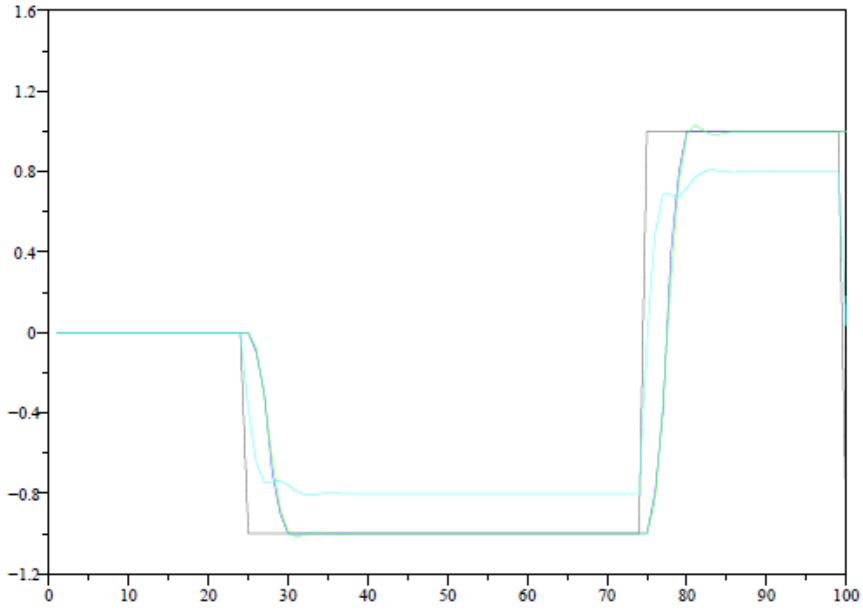

Figure 4: LQ Servomechanism Performance - Setpoint Following

5 Appendix

5.1 Program for Calculating Controller Gains

The following SCILAB program is suitable for calculating the controller gains. It utilises a square-root algorithm for the purpose, (introduce4d via the QR factorisation) and solves the Riccatti equation rather than using an eigenvalue method. Eigenvalue methods may be employed as long as steps are taken to avoid problems with unobservable modes (largely due to the introduction of an integrator, and the direct introduction of the setpoint into the state vector).

```
function k = dlqro(a,b,c,qy)
q = c'*c*qy;

[n,m] = size(b);
r = eye(m,m);
nn = m+n;
p = c*q;
u = [r zeros(m,n)];
for i = 1:100
    d = [u(1:m,:);p*b p*a];
    [t,u] = qr(d);
    p = u(m+1:2*m,m+1:nn);
end;
k = u(1:m,1:m)\u(1:m,m+1:nn);
```

5.2 Program for performing simulations

The following SCILAB program is suitable for performing the simulations which produced the figures in these notes. Specifically, this program is correct for the servomechanism example. It requires prior calculation of the feedback gain vector $\mathcal{K}$, the graphical figure is reproduced with the command *plot2d(result)* and the code is offered here without further explanation.

```
// Initialise vectors
[m,n] = size(k);
x = zeros(n,1);
y = zeros(1,40);
y1 = y; s = y; u = y; du = y; e = y;

N = 100; Nsp = 25; ndisp = 4;
result = zeros(N,ndisp);
j = Nsp; ss = 0;

// Setpoint sequence
for i = 1:N
    j = j - 1;
    if j <= 0
        t = rand();
        if t < 0.5 then ss = -1; else ss = 1; end;
        j = Nsp;
    end;
    result(i,1) = ss;
end;

// Main loop
t = 21;
for i = 1:N

// Determine setpoint
s(t) = result(i,1);

// Run SP filter
y1(t) = 0.1*s(t-1)+0.2*s(t-2)+0.4*s(t-3)+0.2*s(t-4)+0.1*s(t-5);

// Run Process model
y(t) = 0.2*u(t-1)+0.3*u(t-2)+y(t-1)-0.4*y(t-2);

// Error y-Gs
e(t) = y(t) - y1(t);

// Form state vector
x(1) = du(t-1);
x(2) = e(t); x(3) = e(t-1); x(4) = e(t-2);
```

```
x(5) = s(t)-s(t-1); x(6) = s(t-1)-s(t-2);
x(7) = s(t-2)-s(t-3); x(8) = s(t-3)-s(t-4);
x(9) = s(t-4)-s(t-5); x(10) = s(t-5)-s(t-6);
x(11) = s(t-6)-s(t-7);

// Control calculation
du(t) = -k*x;
u(t) = u(t-1) + du(t);

// Display
result(i,2) = y1(t);
result(i,3) = y(t);
result(i,4) = u(t);

// Time update
s(t-20) = s(t);
y1(t-20) = y1(t);
du(t-20) = du(t);
u(t-20) = u(t);
y(t-20) = y(t);
t = t+1;
if (t > 40) then t = 21; end;

end;
```