ELEC9733 Real Time Computing & Control
Optimal Control based on NMSS

Tim Hesketh

April, 2011

1 Introduction

This paper will demonstrate how a controller for a simple system may be developed based on an NMSS model. The controller will be a servomechanism, i.e. it will be designed to follow a setpoint. It should be noted that simpler designs (regulators) are possible, when only constant set point achievement is required.

The development will be undertaken by example for a simple system.

2 Example System

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.

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

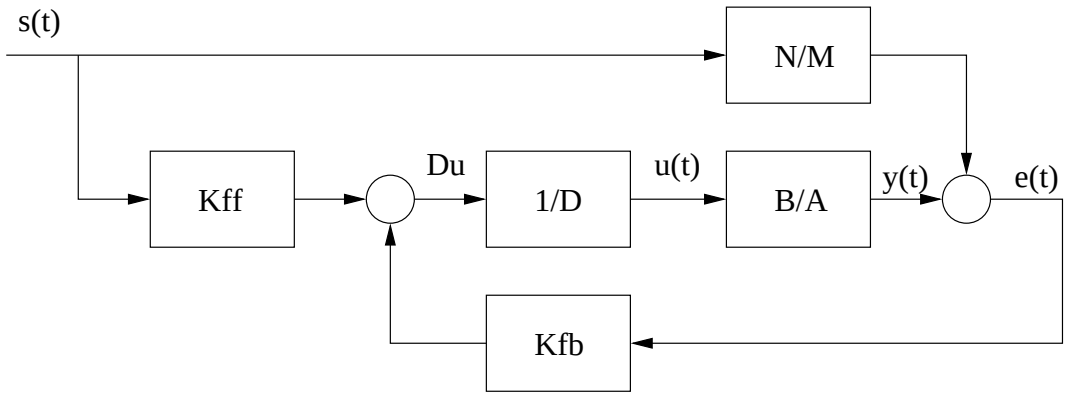
The term $\zeta(s)$ usually represents slowly time-varying disturbances. It would typically incorporate a constant offset, possibly a trend and infrequent load disturbance changes. Occasionally it might include some other systematic disturbance, such as a sinusoid.

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:

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

3 State-Space and Control Structure



From this structure we can write:

$$e(t) = \frac{B}{A}u(t) - \frac{N}{M}s(t)$$

from which we can obtain

$$\Delta AMe(t) = MB\Delta u(t) - AN\Delta s(t)$$

3.1 Signal Filtering

Just a single filter is required to remove the "disturbance", assumed to be a slowly time-varying offset, 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. This filter, $\Delta = (1-q^{-1})$, has been incorporated into the above model for $e(t)$.

In addition, the setpoint $s(t)$ is filtered to "shape" the final response, so that $y(t)$ is required to follow $\frac{N}{M}s(t)$. This filter has the potential to greatly improve closed loop robustness, as the controller is not asked to try to make the output follow set point changes which it cannot match. For the purposes of this design, we select

$$\frac{N}{M} = \frac{0.5q^{-2}}{1-0.5q^{-1}}$$

Importantly the time delay in this filter matches (or even surpasses) the time delay in the process, as the controller cannot be called on to eliminate time delay.

3.2 The State Space Model

The error model is:

$$\begin{aligned}(1. - 3.38q^{-1} + 4.81q^{-2} - 3.785q^{-3} \quad & +1.66q^{-4} - 0.305q^{-5})e(t) = \\ & (-0.18q^{-2} + 0.86q^{-3} + 0.105q^{-4} - 0.245q^{-5})\Delta u(t) - \\ & (0.5q^{-3} - 0.94q^{-4} + 0.745q^{-5} - 0.305q^{-6})\Delta s(t)\end{aligned}$$

The states will be [1]

$$x(t) = \left[\begin{array}{cccccccc} e(t) & \dots & e(t-4) & \Delta u(t-1) & \dots & \Delta u(t-4) & \Delta s(t) & \dots & \Delta s(t-4) \end{array} \right]^\top$$

and the state space system is

$$\left[\begin{array}{cccccccccccccc|c} 3.38 & -4.81 & 3.79 & -1.66 & 0.31 & -0.18 & 0.86 & 0.11 & -0.25 & 0 & -0.5 & 0.94 & -0.75 & 0.31 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right]$$

3.3 The Controller

The controller is

$$\Delta u(t) = -Kx(t)$$

where K minimises the cost function

[1] Note how $s(t)$ is actually time advanced compared to $\Delta u(t)$. The earlier knowledge of set point changes means that the feedforward action can be more effective.

$$C = \sum_{t=1}^{N} x_t^\top P x_t + \Delta u_t^\top Q \Delta u_t$$

We can select $Q = 1$ and $P = C^\top \mathcal{P} C$ so that the weighting $\mathcal{P}$ may be considered to apply to $e(t)$. The resultant controller for $\mathcal{P} = 1$ is:

$$K = \begin{bmatrix} 4.91 & -10.78 & 10.68 & -5.63 & -1.19 & 1.67 & 3.26 & \dots \\ \dots & -0.26 & -0.95 & -0.33 & -0.35 & 2.10 & -2.07 & 1.19 \end{bmatrix}$$

The controller itself is implemented as

$$\begin{aligned} \Delta u(t) &= -Kx(t) \\ u(t) &= u(t-1) + \Delta u(t) \end{aligned}$$

4 Supporting Software

The following SCILAB code may be used to compute the above results.

design.sci

This function controls the overall computation.

```
a = [1 -1.88 1.49 -0.61];
b = [0 0 -0.18 0.77 0.49];
m = [1 -0.5];
n = [0 0 0.5];
d = [1 -1];
aa = conv(conv(a,m),d); [j,k1] = size(aa);
bb = conv(m,b); [j,k2] = size(bb);
cc = conv(a,n); [j,k3] = size(cc);
k4 = k1+k2+k3 - 4;
A = [-aa(2:k1) bb(3:k2) -cc(2:k3);
eye(k4-1,k4-1) zeros(k4-1,1)];
A(k1,:) = zeros(1,k4);
A(k1+k2-2,:) = zeros(1,k4);
B = zeros(k4,1); B(1) = bb(2); B(k1) = 1;
C = zeros(1,k4); C(1) = 1;
K = dlqro(A,B,C,1);
```

conv.sci

This function performs convolution of polynomials.

```
function [c] = conv(a,b)
[m,n] = size(a);
[m,nn] = size(b);
c = zeros(1,n+nn-1);
for i = 1:n
  for j = 1:nn
    k = i+j-1;
    c(k) = c(k) + a(i)*b(j);
  end;
end;
```

dlqro.sci

This function iterates the Ricatti equation to steady state. In fact, it uses a square-root algorithm, which is numerically superior.

```
function k = dlqro(a,b,c,qy)

[n,m] = size(b);
r = eye(m,m);
nn = m+n;
p = c*qy;
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);
```

Simulation

The following program can be used to simulate the closed loop system:

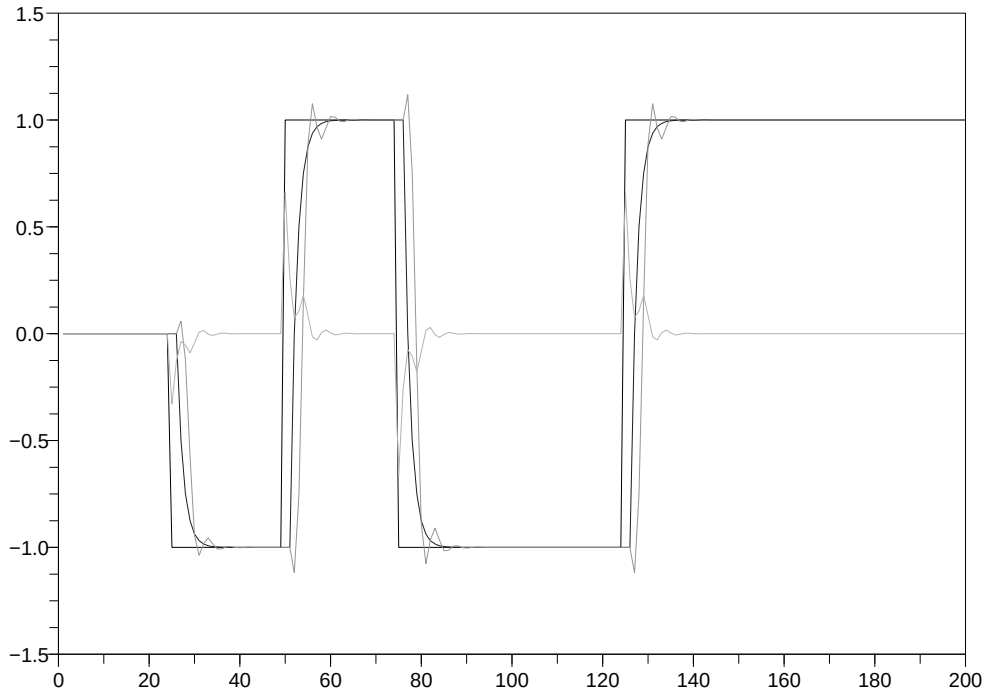
```
// Initialise vectors
[m,n]=size(K);
x=zeros(n,1);
y=zeros(1,40);
y1=y;s=y;u=y;du=y;e=y;

N=200;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)=0.5*s(t-2) + 0.5*y1(t-1);

//Run process model
y(t)= -0.18*u(t-2)+0.78*u(t-3)+0.49*u(t-4)+1.88*y(t-1)-1.49*y(t-2)+0.61*y(t-3);

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

//Form state vector
m = 1;
for j = 1:k1-1  x(m) = e(t-j+1); m = m+1; end;
for j = 1:k2-2  x(m) = du(t-j); m = m+1; end;
for j = 1:k3-1  x(m) = s(t-j+1)-s(t-j); m = m+1; end;

//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);
e(t-20)=e(t);
t=t+1;
if (t>40) then t=21;end;
end;
```

5 Simulation Result

The result of the simulation is shown in the following graph, which depicts fours signals: $s(t)$ - the setpoint; $y1(t)$ - the filtered setpoint; $u(t)$ - the actuation; and $y(t)$ - the measurement.



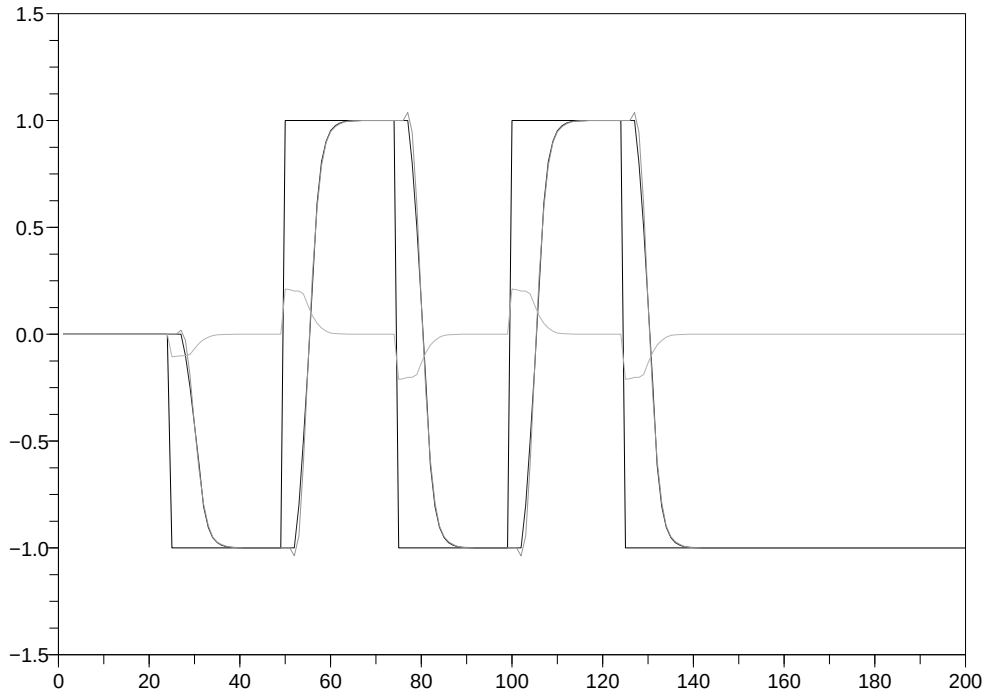
It is important to note the following:

- The delay and non-minimum phase behaviour are still apparent.
- The filtered setpoint is followed reasonably well.

- The slight oscillatory behaviour is the result of inability to follow the filtered setpoint exactly because of delay and non-minimum phase characteristic. If we chose a setpoint filter which produced a less-demanding characteristic, then the following could be almost exact.

To illustrate this contention, here is the result of control with a setpoint filter where the numberator generates a ramp (see below). It is useful to consider some of the points made above, in particular those relating to time delay and non-minimum phase.

$$\frac{y1}{s} = \frac{0.1q^{-3} + 0.1q^{-4} + 0.1q^{-5} + 0.1q^{-6} + 0.1q^{-7}}{1 - 0.5q^{-1}}$$



References

[1] Bitmead, R.R., Gevers, M. and Wertz, V., (1990), *Adaptive Optimal Control; The Thinking Man's GPC*, Prentice Hall International.

[2] Hesketh, T., (1982), 'A State space pole placing self tuning regulator using input output values', *IEE Proceedings*, Vol 129, Pt D, No. 4, 123-128.

[3] Hesketh, T. and Sandoz, D.J, (1987), 'Application of a multivariable adaptive controller', *ACC Proceedings*, 1301 - 1306.