Alternative NMSS Controller Design

Tim Hesketh

September, 2009

1 Introduction

A controller design based on a non-minimal state-space model is introdced by means of example. The features of the design are as follows:

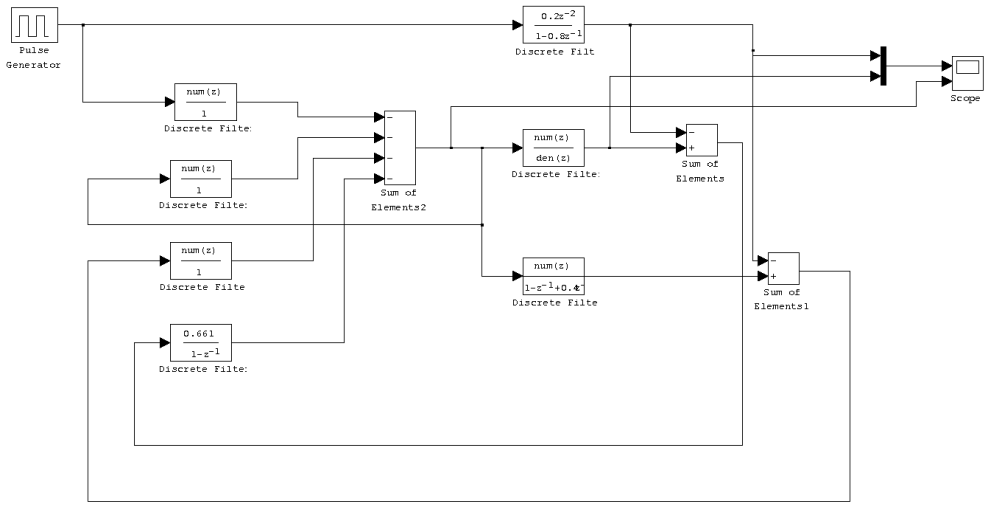
- The controller is a 2-degree of freedom controller, following a setpoint signal which is conditioned by means of a transfer function. The controller incorporates feedforward from the unconditioned setpoint signal, and feedback from an error representing the difference of the conditioned setpoint signal and the process output.
- The controller incorporates the nominal process model (e.g. the model obtained by parameter identification) as an internal model. The structure may be compared to that of a controller plus observer, or a Smith Predictor.
- The controller incorporates integral action.
- Although the example is presented for a SISO system, MIMO designs are possible, and further feedforward from measured disturbances may also be introduced.

2 Closed Loop System Structure

The closed loop system structure is shown by means of a Simulink model.

The features of this model are:

- The setpoint, in this case a square wave, is conditioned by means of the uppermost transfer function block.
- The real process (in this case with unmodelled dynamics) is represented by the mid-level transfer function.
- The lower transfer function represents the nominal process model, on which the controller design is based.
- The controller is represented by four transfer functions, three of them moving average filters:



- Feedforward from the unconditioned setpoint signal.
- Autoregressive feedback of the control signal.
- Feedback of the error between the nominal model and the conditioned setpoint.
- Feedback through an integrator (with appropriate gain) of the error between the actual process output and the conditioned setpoint signal.

3 Controller Design

The actual process is

$$y(t) = \frac{0.2q^{-1} + 0.3q^{-2} + 0.1q^{-3}}{1 - 1.1q^{-1} + 0.5q^{-2} - 0.04q^{-3}}u(t)$$

The nominal process model is

$$\hat{y}(t) = \frac{0.2q^{-1} + 0.3q^{-2}}{1 - q^{-1} + 0.4q^{-2}}u(t)$$

The actual process thus contains unmodelled dynamics. The controller is designed for the nominal model. If $s(t)$ is the unconditioned setpoint signal, and the conditioning transfer function is $M = \frac{0.2q^{-2}}{1-0.8q^{-1}}$:

$$e(t) = \frac{0.2q^{-1} + 0.3q^{-2}}{1 - q^{-1} + 0.4q^{-2}}u(t) - \frac{0.2q^{-2}}{1 - 0.8q^{-1}}s(t)$$

$$(1 - 1.8q^{-1} + 1.2q^{-2} - 0.32q^{-3})e(t) = (0.2q^{-1} + 0.14q^{-2} - 0.24q^{-3})u(t) + \ldots$$
$$\ldots + (0.2q^{-2} - 0.2q^{-3} + 0.08q^{-4})s(t)$$

Defining $esum(t) = \sum_{k=0}^{t} e(k)$ (i.e. integral of error):

$$esum(t) = esum(t-1) + e(t)$$

From the above difference equations, it is possible to write the non-minimal state-space model for the system.

$$x(t) = \begin{bmatrix} e(t) & e(t-1) & e(t-2) & u(t-1) & u(t-2) & \ldots \\ \ldots & s(t) & s(t-1) & s(t-2) & s(t-3) & esum(t) \end{bmatrix}^\top$$

$$\left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right] = \left[\begin{array}{cccccccccc|c} 1.8 & -1.2 & 0.32 & 0.14 & -0.24 & 0 & -0.2 & 0.2 & -0.08 & 0 & 0.2 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1.8 & -1.2 & 0.32 & 0.14 & -0.24 & 0 & -0.2 & 0.2 & -0.08 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{array}\right]$$

The cost function to be minimised is

$$C = \sum_{t=0}^{\infty} (x(t)^\top C^\top C x(t) + u(t)^\top R u(t))$$

Note that the states (and their weightings) to be included in the cost function are determined by C, and in this case, only $esum(t)$ is selected to contribute to the cost. For $R = 1$ (scalar for a SISO system), the controller is:

$$u(t) = -Kx(t)$$
$$K = \begin{bmatrix} 3.66 & -3.18 & 1.01 & -0.02 & -0.76 & -0.23 & -0.32 & 0.48 & -0.25 & 0.66 \end{bmatrix}$$

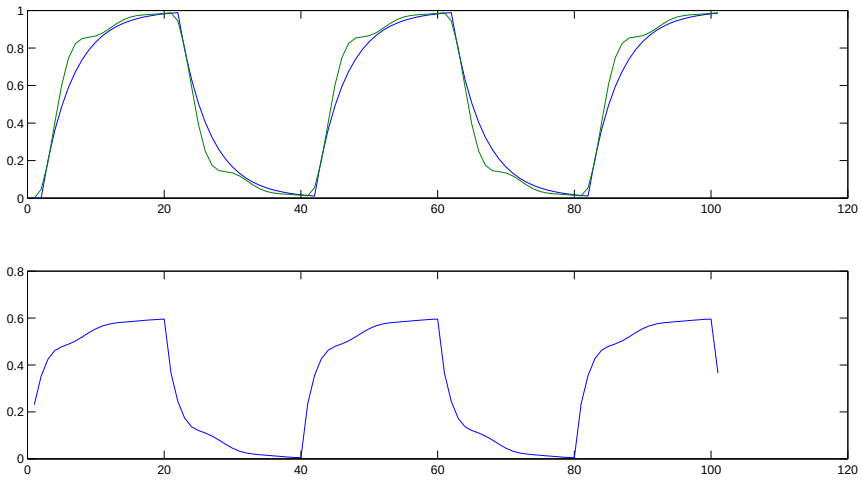
For the purposes of inclusion in the block diagram model, this controller may be expressed as:

$$u(t) = -(3.66 - 3.18q^{-1} + 1.01q^{-2})\hat{e}(t) - (-0.02q^{-1} - 0.76q^{-2})u(t) \ldots$$
$$\ldots - (-0.23 - 0.32q^{-1} + 0.48q^{-2} - 0.25q^{-3})s(t) - \frac{0.66}{1 - q^{-1}} e(t)$$

NOTE: The proportional error feedback, determining the controller dynamics, is from the nominal model output, but the integral of error feedback, determining the steady-state error and model following, is from the actual process output.

4 Closed Loop Performance

The closed loop performance was obtained from Simulink.



5 Conclusion

- The closed loop performance shows good model following and attainment of the correct steady-state setpoint.
- The controller is a compromise between error minimisation and actuation amplitude. It would be possible to weight the error more heavily and obtain closer setpoint following.
- The effect of the unmodelled dynamics on the closed loop behaviour is apparent, but the controller compensates satisfactorily.
- In the above simulation, only the integral of error and control actuation signals play any part in the cost function. By choice of the vector C, other signals may be included in the cost function. Controllers designed in this way may display greater robustness (to disturbances or model uncertainty).

6 Appendix - Design Code

The following SCILAB code may be used to design the controller and to verify the closed loop performance of the system, although the incorporation of the internal model of the process is not accommodated.

6.1 Design

```
num = [0 0.2 0.3];
den = [1 -1 0.4];
m = [1 -0.8];
n = [0 0 0.2];
aa = convol(den,m); [j,k1] = size(aa);
bb = convol(m,num); [j,k2] = size(bb);
cc = convol(den,n); [j,k3] = size(cc);
k4 = k1+k2+k3 - 3;
a1 = -aa(2:k1);
if k2>2 then a1 = [a1 bb(3:k2)]; end;
a1 = [a1 -cc(2:k3)];
A = [[a1 0]; eye(k4-1,k4-1) zeros(k4-1,1)];
if k2>2 then A(k1,:) = zeros(1,k4); end;
A(k1+k2-2,:) = zeros(1,k4);
A(k4,:) = [a1 1];
B = zeros(k4,1); B(1) = bb(2);
if k2>2 then B(k1) = 1; end;
C = zeros(1,k4); C(k4) = 1;
w = input("Cost function error weight?");
K = dlqro(A,B,C,w);

f = file('open','sim.dat','unknown');
[j,kn] = size(n);
[j,km] = size(m);
fprintf(f,'%3d',kn);
for j=1:kn fprintf(f,'%9.4f',n(j)); end;
fprintf(f,'%3d',km);
for j=1:km fprintf(f,'%9.4f',m(j)); end;
fprintf(f,'%3d %3d %3d %3d',k1-1,k2-2,k3-1,k4);
for j=1:k4 fprintf(f,'%9.4f',K(j)); end;
file('close',f);
```

6.2 Simulation

```
// Initialise vectors
[m1,n1]=size(K);
x=zeros(n1,1);
y=zeros(1,40);
y1=y;s=y;u=y;e=y;esum=0;

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;

//Model size
[j,knum] = size(num);
[j,kden] = size(den);

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

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

//Run Sp filter
y1(t) = 0;
for j = 1:kn  y1(t) = y1(t) + n(j)*s(t-j+1); end;
if km > 1 then
  for j = 2:km  y1(t) = y1(t) - m(j)*y1(t-j+1); end;
end;

//Run process model
y(t) = 0;
for j = 1:knum  y(t) = y(t) + num(j)*u(t-j+1); end;
for j = 2:kden  y(t) = y(t) - den(j)*y(t-j+1); end;

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

//Sum of error
esum = esum+e(t);

//Form state vector
m1 = 1;
for j = 1:k1-1  x(m1) = e(t-j+1); m1 = m1+1; end;
if k2 > 2 then
  for j = 1:k2-2  x(m1) = u(t-j); m1 = m1+1; end;
end;
```

```
for j = 1:k3-1  x(m1) = s(t-j+1); m1 = m1+1; end;
x(m1) = esum;

//Control calculation
u(t)=-K*x;

//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);
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;

//Plot result
subplot(211)
plot(result(:,1:3))
subplot(212)
plot(result(:,4));
```

6.3 LQ Design

```
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);
```