Task 3: Examples 3.{4;5;7;8;9;10} and Questions 3.{5;11;12;15}

Nelson Campos

Contents

- Example 3.4: Indirect self-tuner with cancellation of process zero
- Example 3.5: Indirect self-tuner without cancellation of process zero

Example 3.4: Indirect self-tuner with cancellation of process zero

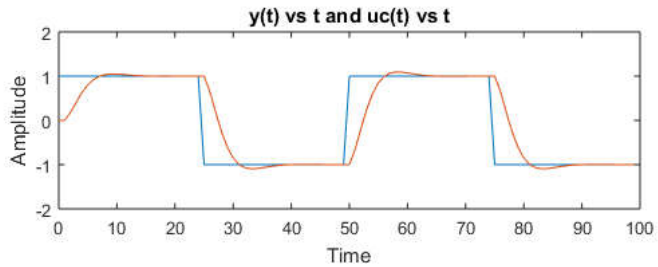
Let the process be the same as in Example 3.1 and assume that the process zero is canceled. The specifications are the same as in Example 3.1, that is, to obtain a closed-loop characteristic polynomial Am. The parameters of the model $y(t) + a_1 y(t-1) + a_2 y(t-2) = b_0 u(t-1) + b_1 u(t-2)$ which has the same structure as Eq. (3.17), are estimated by using the leastsquares algorithm. Algorithm 3.2 is used for the self-tuning regulator. The calculations, which were done in Example 3.1, give the control law

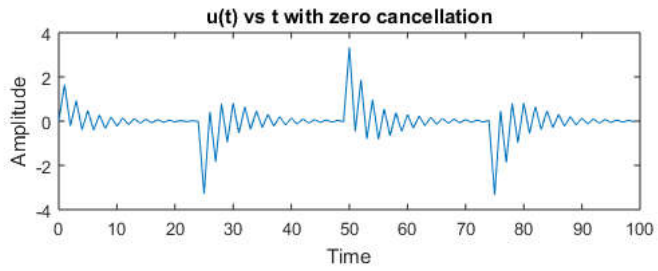
$$u(t) + r_1 u(t-1) = t_0 uc(t) - s_0 y(t) - s_1 y(t-1)$$

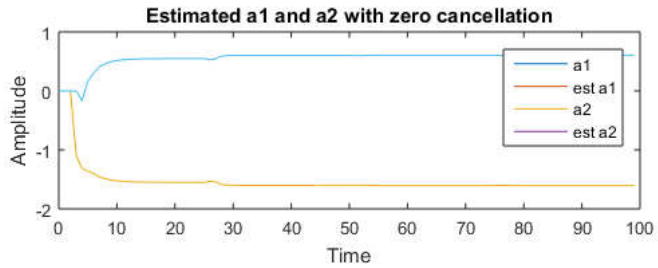
The controller parameters were expressed as functions of the model parameters and the specifications. Figure 1 shows the process output and the control signal in a simulation of the process with the self-tuner when the command signal is a square wave. Notice that the estimates converge quickly. The system in Example 3.4 behaves quite well, apart from the “ringing” control signal. This can be avoided by using a design in which the process zero is not canceled. The consequences of this are illustrated in the next example.

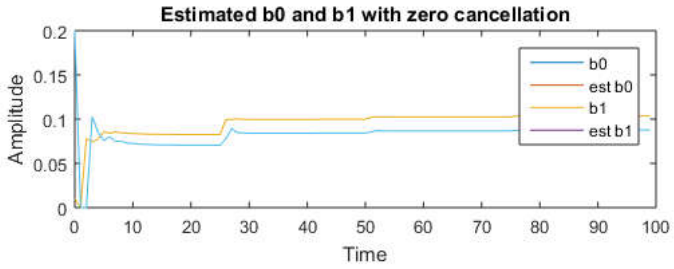
```
close all
clear all

w = warning ('off','all');

a1 = -1.6065;
a2 = 0.6065;
b0 = 0.1065;
b1 = 0.0902;
%---------------
bm0 = 0.1761;
am1 = -1.3205;
am2 = 0.4966;

t_step = 1;
t = [0:t_step:100-t_step];
uc = zeros(1,size(t,2));
y = zeros(1, size(t,2));
T = 50;

for i=1:size(t,2)/T
    for j=1:T
        index = (i-1)*T+j;
        if j <= T/2
            uc(index) = 1;
        else uc(index) = -1;
        end
    end
end

for i=3:size(t,2)
  y(i) = -am1*y(i-1)-am2*y(i-2)+bm0*uc(i-1);
end

% R(q) = B+ = q+b1/b0, S(q) = s0q+s1, T(q) = AoBm' = bm0q/b0
s0 = (am1-a1)/b0;
s1 = (am2-a2)/b0;
R = [1 b1/b0];
S = [s0 s1];
T = [bm0/b0 0];

u = zeros(1, size(t,2));

for i=2:size(t,2)
  u(i) = -R(2)*u(i-1)+T(1)*uc(i)-S(1)*y(i)-S(2)*y(i-1);
end

figure(1), subplot(2,1,1), plot(t,uc, t, y), xlabel('Time'), ylabel('Amplitude'), title('y(t) vs t and uc(t) vs t')
           subplot(2,1,2), plot(t,u),, xlabel('Time'), ylabel('Amplitude'), title('u(t) vs t with zero cancellation')

% Estimation goes here
N = 4;
theta = zeros(N, size(t,2));
phi = zeros(N, size(t,2));
error = zeros(1,size(t,2));
K = zeros(N,1);
P = zeros(N);

%The initial conditions
%theta = [a1 a2 b0 b1]
theta(1,1) = 0;
theta(2,1) = 0;
theta(3,1) = 0.01;
theta(4,1) = 0.2;
I = eye(N);
P(1,1) = 100;
```

```
P(2,2) = 100;
P(3,3) = 1;
P(4,4) = 1;

lambda =  1;

for i = 3 : size(t,2)

phi(:,i) = [-y(i-1) -y(i-2) u(i-1) u(i-2)]';
K = P * phi(:,i)* inv(lambda + phi(:,i)'* P * phi(:,i));
P = ((I - K * phi(:,i)')* P)/lambda;
est_y = phi(:,i)'*theta(:,i-1);
error(i) = y(i)-est_y;

theta(:,i) = theta(:,i-1) + K * error(i);

end

figure(2), subplot(2,1,1), plot(t,a1,t,theta(1,:),t,a2,t,theta(2,:)), xlabel('Time'), ylabel('Amplitude'), title('Estimated a1 and a2 with zero cancellation'), leg
           subplot(2,1,2), plot(t,b0,t,theta(3,:),t,b1,t,theta(4,:)), xlabel('Time'), ylabel('Amplitude'), title('Estimated b0 and b1 with zero cancellation'), leg
```









Example 3.5: Indirect self-tuner without cancellation of process zero

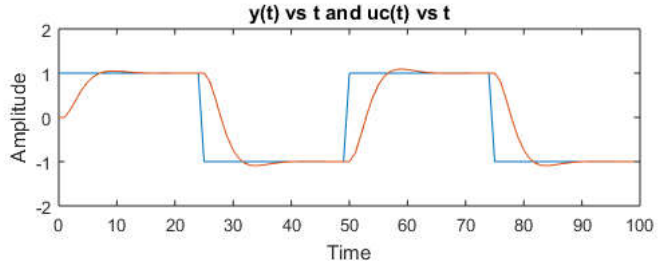
Consider the same process as in Example 3.4, but use a control design in which there is no cancellation of the process zero. The parameters are estimated in the same way as in Example 3.4, but the control law is now computed as in Example 3.2. Figure 2 shows the process output and the control signal in a simulation of the process with the indirect self-tuner when the command signal is a square wave. Notice that the behavior of the process output is quite similar to that in Fig. 1 but that there is no “ringing” in the control signal.

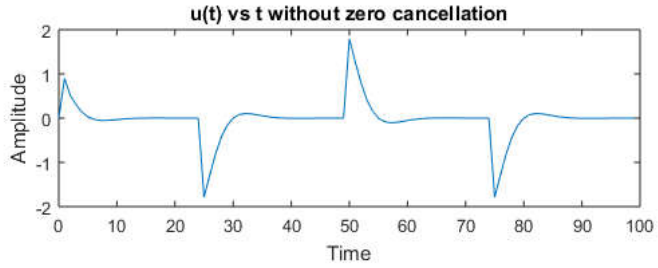
```
% Hm(q)= beta*B(q)/Am(q) = Bm(q)/Am(q), bm0 = beta*b0
% beta = (1+am1+am2)/(b0+b1)

a0 = 0; % deg(A(q)) = 1
beta = (1+am1+am2)/(b0+b1);
bm0 = beta*b0;
bm1 = beta*b1;
```

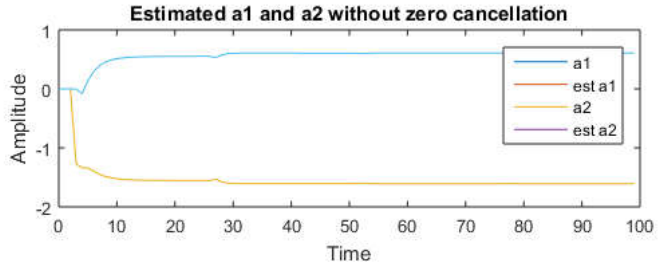
```
for i=3:size(t,2)
  y(i) = -am1*y(i-1)-am2*y(i-2)+bm0*uc(i-1)+bm1*uc(i-2);
end

r1 = b1/b0 + (b1^2-am1*b0*b1+am2*b0^2)*(-b1+a0*b0)/(b0*(b1^2-a1*b0*b1+a2*b0^2));
s0_2 = b1*(a0*am1-a2-am1*a1+a1^2+am2-a1*a0)/(b1^2-a1*b0*b1+a2*b0^2)+...
    b0*(am1*a2-a1*a2-a0*am2+a0*a2)/(b1^2-a1*b0*b1+a2*b0^2);

s1_2 = b1*(a1*a2-am1*a2+a0*am2-a0*a2)/(b1^2-a1*b0*b1+a2*b0^2)+...
     b0*(a2*am2-a2^2-a0*am2*a1+a0*a2*am1)/(b1^2-a1*b0*b1+a2*b0^2);

R2 = [1 r1];
S2 = [s0_2 s1_2];
T2 = [bm0/b0];

for i=2:size(t,2)
  u(i) = -R2(2)*u(i-1)+T2(1)*uc(i)-S2(1)*y(i)-S2(2)*y(i-1);
end

figure(3), subplot(2,1,1), plot(t,uc, t, y), xlabel('Time'), ylabel('Amplitude'), title('y(t) vs t and uc(t) vs t')
           subplot(2,1,2), plot(t,u),, xlabel('Time'), ylabel('Amplitude'), title('u(t) vs t without zero cancellation')
```

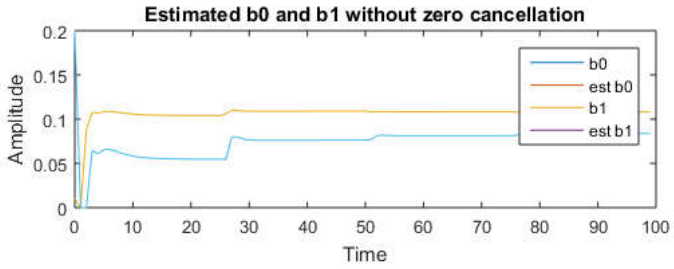
Estimation goes here

```
N = 4;
theta = zeros(N, size(t,2));
phi = zeros(N, size(t,2));
error = zeros(1,size(t,2));
K = zeros(N,1);
P = zeros(N);

%The initial conditions
%theta = [a1 a2 b0 b1]
theta(1,1) = 0;
theta(2,1) = 0;
theta(3,1) = 0.01;
theta(4,1) = 0.2;
I = eye(N);
P(1,1) = 100;
P(2,2) = 100;
P(3,3) = 1;
P(4,4) = 1;

lambda =  1;

for i = 3 : size(t,2)

phi(:,i) = [-y(i-1) -y(i-2) u(i-1) u(i-2)]';
K = P * phi(:,i)* inv(lambda + phi(:,i)'* P * phi(:,i));
P = ((I - K * phi(:,i)')* P)/lambda;
est_y = phi(:,i)'*theta(:,i-1);
error(i) = y(i)-est_y;

theta(:,i) = theta(:,i-1) + K * error(i);

end

figure(4), subplot(2,1,1), plot(t,a1,t,theta(1,:),t,a2,t,theta(2,:)), xlabel('Time'), ylabel('Amplitude'), title('Estimated a1 and a2 without zero cancellation'),
           subplot(2,1,2), plot(t,b0,t,theta(3,:),t,b1,t,theta(4,:)), xlabel('Time'), ylabel('Amplitude'), title('Estimated b0 and b1 without zero cancellation'),

% The controller obtained from the estimated parameters
est_A = [theta(1,size(t,2)) theta(2,size(t,2))];
est_B = [theta(3,size(t,2)) theta(4,size(t,2))];
h = 0.5; % The sampling time

est_Hq = tf(est_B, est_A, h, 'variable', 'q');
est_beta  = (1+am1+am2)/(est_B(1)+est_B(2));

b0 = est_B(1);
b1 = est_B(2);

est_r1 = b1/b0 + (b1^2-am1*b0*b1+am2*b0^2)*(-b1+a0*b0)/(b0*(b1^2-a1*b0*b1+a2*b0^2));
est_s0_2 = b1*(a0*am1-a2-am1*a1+a1^2+am2-a1*a0)/(b1^2-a1*b0*b1+a2*b0^2)+...
    b0*(am1*a2-a1*a2-a0*am2+a0*a2)/(b1^2-a1*b0*b1+a2*b0^2);

est_s1_2 = b1*(a1*a2-am1*a2+a0*am2-a0*a2)/(b1^2-a1*b0*b1+a2*b0^2)+...
     b0*(a2*am2-a2^2-a0*am2*a1+a0*a2*am1)/(b1^2-a1*b0*b1+a2*b0^2);

est_R2 = tf([1 est_r1], 1, h,'variable', 'q');
est_S2 = tf([est_s0_2 est_s1_2], 1, h,'variable', 'q');
est_T2 = tf([bm0/b0], 1, h,'variable', 'q');
```









Published with MATLAB® R2015a

Task 3 - Part II: Examples 7 and 8

Nelson Campos

Contents

- Example 3.7: Direct self-tuner with d0 = 1
- Example 3.8: Direct self-tuner with d0 = 2

Example 3.7: Direct self-tuner with d0 = 1

Consider the system in Example 3.1. Since deg A = 2 and deg B = 1, we have deg Am = 2 and deg Ao = 0. Hence Ao = 1, and we will choose Bm = qAm(1). Equation (3.29) in Algorithm 3.3 then gives T = qAm(1). The controller structure is given by deg R = deg S = deg T = deg A ? 1 = 1. The model given by Eq. (3.27) therefore becomes

$$y(t) = r_0 u_f(t-1) + r_1 u_f(t-2) + s_0 y_f(t-1) + s_1 y_f(t-2)$$

where:

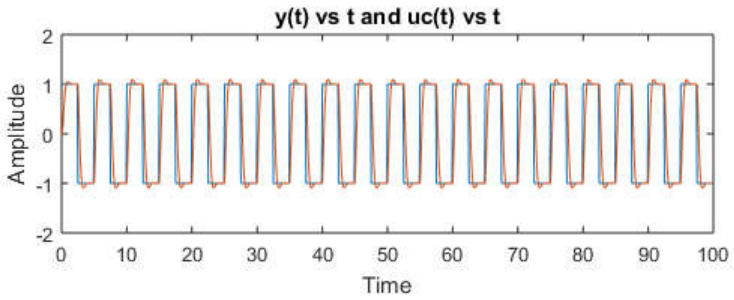
$$u_f(t) + a_{m1} uf(t-1) + a_{m1} u_f(t-2) = u(t)$$

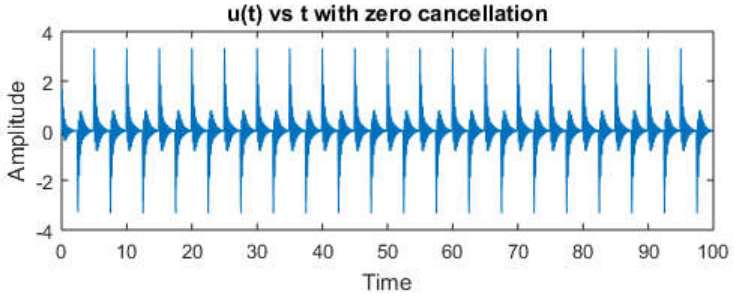
$$y_f(t) + a_{m1} y_f(t-1) + a_{m1} y_f(t-2) = y(t)$$

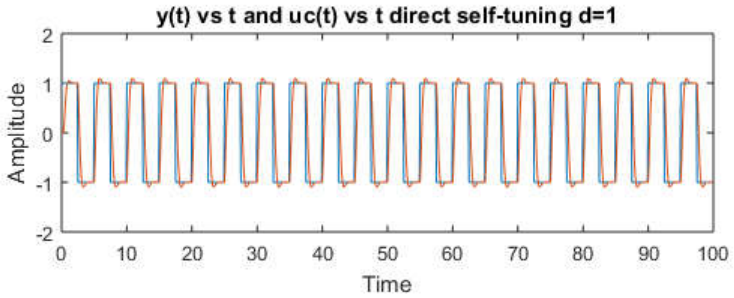
Figure 1 shows the simulation of the results. In a practical case the time delay and the order of the process that we would like to control are not known. It is therefore natural to consider these variables as design parameters that are chosen by the user. The parameter d0 is of particular importance for a direct algorithm. In the next example we show that “ringing” can be avoided simply by increasing the value of d0.

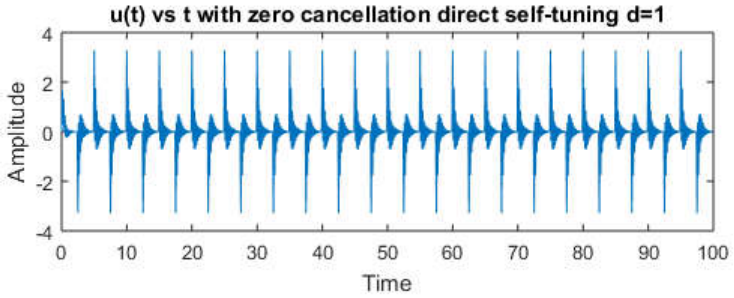
```
close all
clear all

a1 = -1.6065;
a2 = 0.6065;
b0 = 0.1065;
b1 = 0.0902;
%---------------
bm0 = 0.1761;
am1 = -1.3205;
am2 = 0.4966;

t_step = 1/10;
t = [0:t_step:100-t_step];
uc = zeros(1,size(t,2));
y = zeros(1, size(t,2));
T = 50;

for i=1:size(t,2)/T
    for j=1:T
        index = (i-1)*T+j;
        if j <= T/2
            uc(index) = 1;
        else uc(index) = -1;
        end
    end
end

for i=3:size(t,2)
  y(i) = -am1*y(i-1)-am2*y(i-2)+bm0*uc(i-1);
end

% R(q) = B+ = q+b1/b0, S(q) = s0q+s1, T(q) = AoBm' = bm0q/b0
s0 = (am1-a1)/b0;
s1 = (am2-a2)/b0;
R = [1 b1/b0];
S = [s0 s1];
T = [bm0/b0 0];

u = zeros(1, size(t,2));

for i=2:size(t,2)
  u(i) = -R(2)*u(i-1)+T(1)*uc(i)-S(1)*y(i)-S(2)*y(i-1);
end

figure(1), subplot(2,1,1), plot(t,uc, t, y), xlabel('Time'), ylabel('Amplitude'), title('y(t) vs t and uc(t) vs t')
           subplot(2,1,2), plot(t,u), xlabel('Time'), ylabel('Amplitude'), title('u(t) vs t with zero cancellation')


uf = zeros(1,size(t,2));
yf = zeros(1, size(t,2));
```

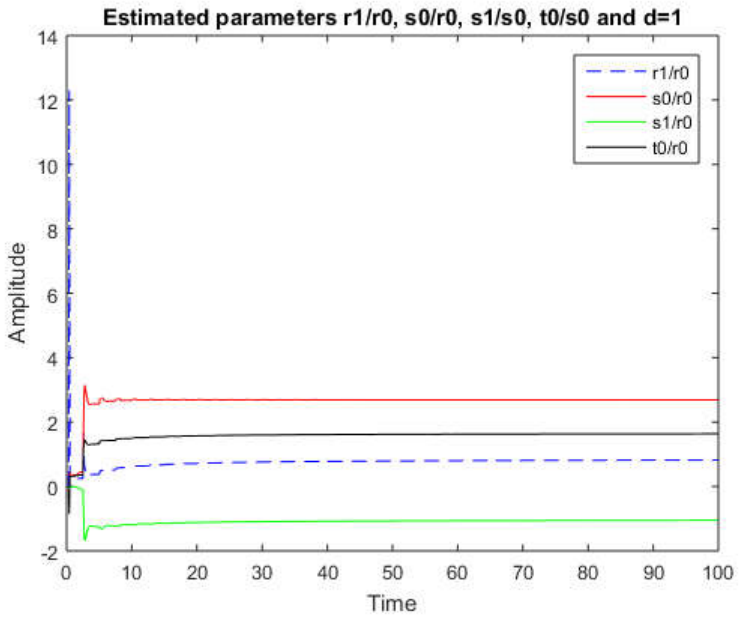
```
for i=3:size(t,2)
  uf(i) = -am1*uf(i-1)-am2*uf(i-2)+u(i);
  yf(i) = -am1*yf(i-1)-am2*yf(i-2)+y(i);
end

% Estimation goes here
N = 4;
theta = zeros(N, size(t,2));
phi = zeros(N, size(t,2));
error = zeros(1,size(t,2));
K = zeros(N,1);
P = zeros(N);

%The initial conditions
%theta = [a1 a2 b0 b1]
theta(1,1) = 0;
theta(2,1) = 0;
theta(3,1) = 0.01;
theta(4,1) = 0.2;
I = eye(N);
P(1,1) = 100;
P(2,2) = 100;
P(3,3) = 1;
P(4,4) = 1;

lambda =  1;
est_r0 = zeros(1,size(t,2));
est_r1 = zeros(1,size(t,2));
est_s0 = zeros(1,size(t,2));
est_s1 = zeros(1,size(t,2));

for i = 3 : size(t,2)

phi(:,i) = [uf(i-1) uf(i-2) yf(i-1) yf(i-2)]';
K = P * phi(:,i)* inv(lambda + phi(:,i)'* P * phi(:,i));
P = ((I - K * phi(:,i)')* P)/lambda;
est_y = phi(:,i)'*theta(:,i-1);
error(i) = y(i)-est_y;

theta(:,i) = theta(:,i-1) + K * error(i);
est_r0(i) = theta(1,i);
est_r1(i) = theta(2,i);
est_s0(i) = theta(3,i);
est_s1(i) = theta(4,i);

end

t0 = 1+am1+am2;
r0 = est_r0(size(t,2));
r1 = est_r1(size(t,2));
s0 = est_s0(size(t,2));
s1 = est_s1(size(t,2));

for i=3:size(t,2)
  y(i) = r0*uf(i-1)+r1*uf(i-2)+s0*yf(i-1)+s1*yf(i-2);
  u(i) = (-r1*u(i-1)+t0*uc(i)-s0*y(i)-s1*y(i-1))/r0;
end

figure(2), subplot(2,1,1), plot(t,uc, t, y), xlabel('Time'), ylabel('Amplitude'), title('y(t) vs t and uc(t) vs t direct self-tuning d=1')
           subplot(2,1,2), plot(t,u), xlabel('Time'), ylabel('Amplitude'), title('u(t) vs t with zero cancellation direct self-tuning d=1')

figure(3), plot(t,est_r1./est_r0, 'b--', t, est_s0./est_r0, 'r', t, est_s1./est_r0, 'g', t, t0./est_r0, 'k'),xlabel('Time'), ylabel('Amplitude'),
title('Estimated parameters r1/r0, s0/r0, s1/s0, t0/s0 and d=1'), legend('r1/r0', 's0/r0', 's1/r0', 't0/r0')
```


y(t) vs t and uc(t) vs t
Amplitude
Time


u(t) vs t with zero cancellation
Amplitude
Time


y(t) vs t and uc(t) vs t direct self-tuning d=1
Amplitude
Time


u(t) vs t with zero cancellation direct self-tuning d=1
Amplitude
Time

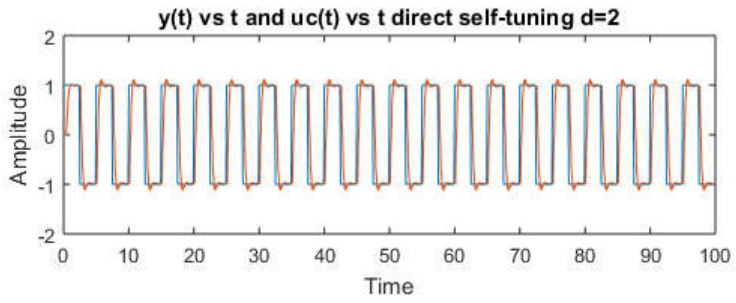


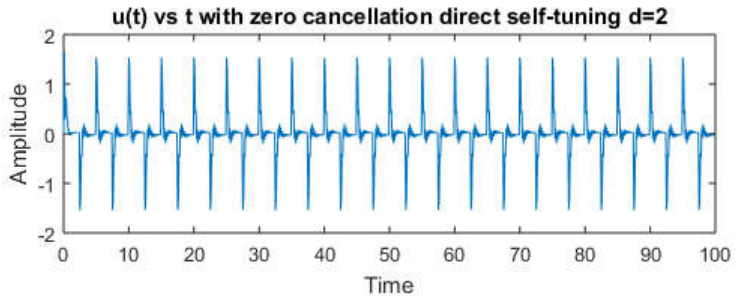
Example 3.8: Direct self-tuner with d0 = 2

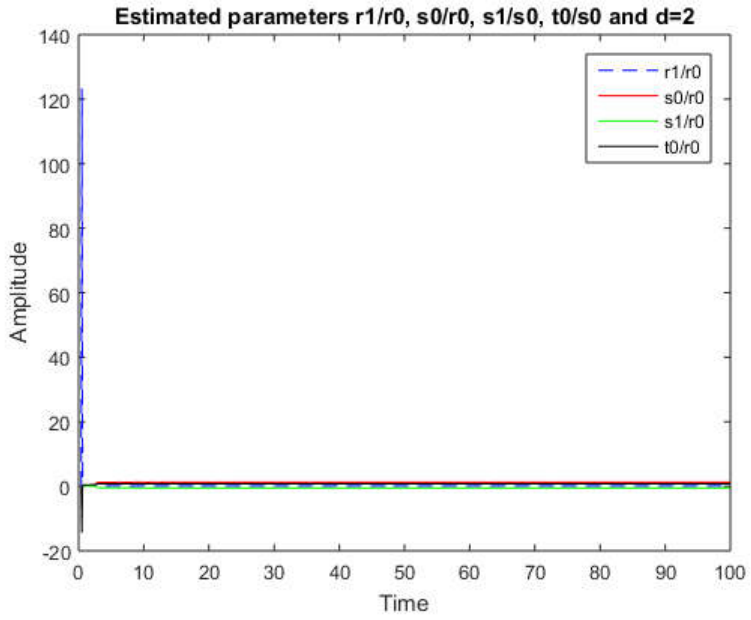
In the derivation of the direct algorithm the parameter d0 was the pole excess of the plant. Assume for a moment that we do not know the value of d0 and that we treat it as a design parameter instead. Figure 2 shows the simulation of the results, but with d0 = 2 instead of d0 = 1. We thus find the interesting and surprising result that cancellation of the process zero can be avoided by increasing the parameter d0. This observation will be explained later when we will be analyzing the algorithms.

```
% Estimation goes here
N = 4;
theta = zeros(N, size(t,2));
phi = zeros(N, size(t,2));
error = zeros(1,size(t,2));
K = zeros(N,1);
P = zeros(N);

%The initial conditions
%theta = [a1 a2 b0 b1]
theta(1,1) = 0;
theta(2,1) = 0;
theta(3,1) = 0.01;
theta(4,1) = 0.2;
I = eye(N);
P(1,1) = 100;
P(2,2) = 100;
P(3,3) = 1;
P(4,4) = 1;

lambda =  1;
est_r0 = zeros(1,size(t,2));
est_r1 = zeros(1,size(t,2));
est_s0 = zeros(1,size(t,2));
est_s1 = zeros(1,size(t,2));

%d=2
for i = 4 : size(t,2)

phi(:,i) = [uf(i-2) uf(i-3) yf(i-2) yf(i-3)]';
K = P * phi(:,i)* inv(lambda + phi(:,i)'* P * phi(:,i));
P = ((I - K * phi(:,i)')* P)/lambda;
est_y = phi(:,i)'*theta(:,i-1);
error(i) = y(i)-est_y;

theta(:,i) = theta(:,i-1) + K * error(i);
est_r0(i) = theta(1,i);
est_r1(i) = theta(2,i);
est_s0(i) = theta(3,i);
est_s1(i) = theta(4,i);

end

t0 = 1+am1+am2;
r0 = est_r0(size(t,2));
r1 = est_r1(size(t,2));
s0 = est_s0(size(t,2));
```

```
s1 = est_s1(size(t,2));

for i=4:size(t,2)
  y(i) = r0*uf(i-2)+r1*uf(i-3)+s0*yf(i-2)+s1*yf(i-3);
  u(i) = (-r1*u(i-1)+t0*uc(i)-s0*y(i)-s1*y(i-1))/r0;
end

figure(4), subplot(2,1,1), plot(t,uc, t, y), xlabel('Time'), ylabel('Amplitude'), title('y(t) vs t and uc(t) vs t direct self-tuning d=2')
           subplot(2,1,2), plot(t,u), xlabel('Time'), ylabel('Amplitude'), title('u(t) vs t with zero cancellation direct self-tuning d=2')

figure(5), plot(t,est_r1./est_r0, 'b--', t, est_s0./est_r0, 'r', t, est_s1./est_r0, 'g', t, t0./est_r0, 'k'),xlabel('Time'), ylabel('Amplitude'),
title('Estimated parameters r1/r0, s0/r0, s1/s0, t0/s0 and d=2'), legend('r1/r0', 's0/r0', 's1/r0', 't0/r0')
```







Published with MATLAB® R2015a

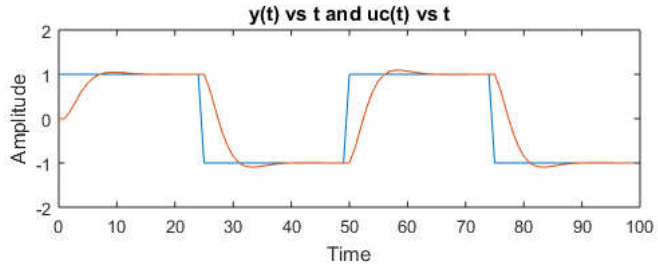
Task 3 - Part III: Example 3.9

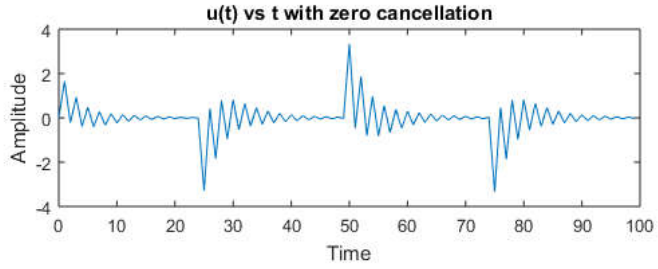
Nelson Campos

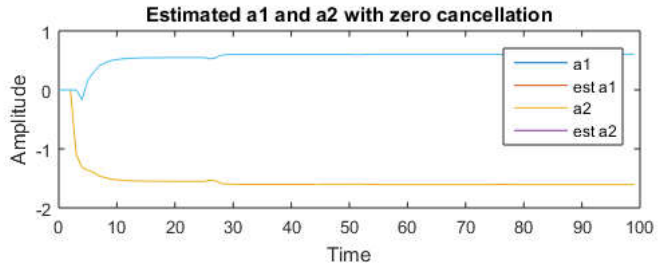
Example 3.9: Effect of Load Disturbances Consider the system in Example 3.5, that is, an indirect self-tuning regulator with no zero cancellation. We will now make a simulation that is identical to the one shown in Fig. 3.6 except that the load disturbance will be v(t) = 0.5 for $t \geq 40$. A forgetting factor $\lambda = 0.98$ has also been introduced; otherwise, the conditions are identical to those in Example 3.5. The behavior of the system is shown in Fig. 3. A load disturbance may be disastrous. Notice that the response is strongly asymmetric. The reason for this is that the controller parameters change rapidly when the control signal changes. Rapid changes of the estimates in response to command signals indicates that the model structure is not correct. The parameter estimates also change significantly at the step in the load disturbance. When the command signal is constant, the parameters appear to settle at constant values that are far from the true parameters.

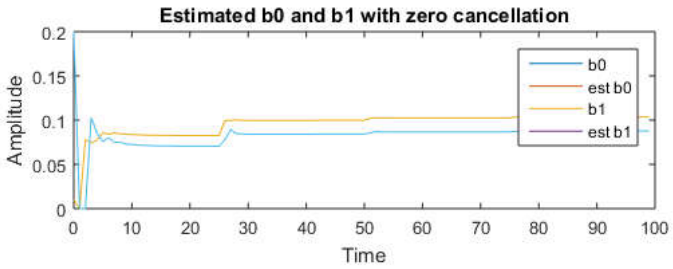
```
close all
clear all

a1 = -1.6065;
a2 = 0.6065;
b0 = 0.1065;
b1 = 0.0902;
%---------------
bm0 = 0.1761;
am1 = -1.3205;
am2 = 0.4966;

t_step = 1;
t = [0:t_step:100-t_step];
uc = zeros(1,size(t,2));
y = zeros(1, size(t,2));
T = 50;

for i=1:size(t,2)/T
    for j=1:T
        index = (i-1)*T+j;
        if j <= T/2
            uc(index) = 1;
        else uc(index) = -1;
        end
    end
end

for i=3:size(t,2)
  y(i) = -am1*y(i-1)-am2*y(i-2)+bm0*uc(i-1);
end

% R(q) = B+ = q+b1/b0, S(q) = s0q+s1, T(q) = AoBm' = bm0q/b0
s0 = (am1-a1)/b0;
s1 = (am2-a2)/b0;
R = [1 b1/b0];
S = [s0 s1];
T = [bm0/b0 0];

u = zeros(1, size(t,2));

for i=2:size(t,2)
  u(i) = -R(2)*u(i-1)+T(1)*uc(i)-S(1)*y(i)-S(2)*y(i-1);
end

figure(1), subplot(2,1,1), plot(t,uc, t, y), xlabel('Time'), ylabel('Amplitude'), title('y(t) vs t and uc(t) vs t')
           subplot(2,1,2), plot(t,u),, xlabel('Time'), ylabel('Amplitude'), title('u(t) vs t with zero cancellation')

% Estimation goes here
N = 4;
theta = zeros(N, size(t,2));
phi = zeros(N, size(t,2));
error = zeros(1,size(t,2));
K = zeros(N,1);
P = zeros(N);

%The initial conditions
%theta = [a1 a2 b0 b1]
theta(1,1) = 0;
theta(2,1) = 0;
theta(3,1) = 0.01;
theta(4,1) = 0.2;
I = eye(N);
P(1,1) = 100;
P(2,2) = 100;
P(3,3) = 1;
P(4,4) = 1;

lambda =  1;

for i = 3 : size(t,2)

phi(:,i) = [-y(i-1) -y(i-2) u(i-1) u(i-2)]';
K = P * phi(:,i)* inv(lambda + phi(:,i)'* P * phi(:,i));
P = ((I - K * phi(:,i)')* P)/lambda;
est_y = phi(:,i)'*theta(:,i-1);
error(i) = y(i)-est_y;
```

```
theta(:,i) = theta(:,i-1) + K * error(i);

end

figure(2), subplot(2,1,1), plot(t,a1,t,theta(1,:),t,a2,t,theta(2,:)), xlabel('Time'), ylabel('Amplitude'), title('Estimated a1 and a2 with zero cancellation'), leg
           subplot(2,1,2), plot(t,b0,t,theta(3,:),t,b1,t,theta(4,:)), xlabel('Time'), ylabel('Amplitude'), title('Estimated b0 and b1 with zero cancellation'), leg

%%Example 3.5
```







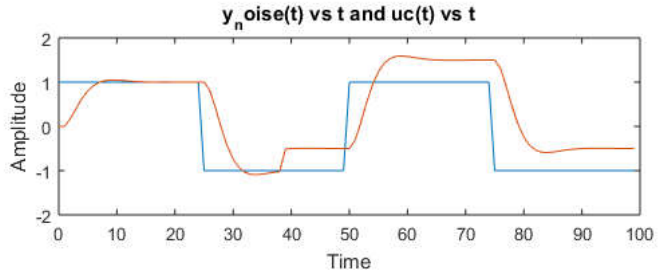


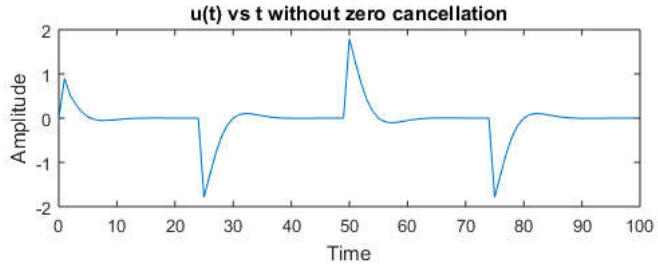

```
% Hm(q)= beta*B(q)/Am(q) = Bm(q)/Am(q), bm0 = beta*b0
% beta = (1+am1+am2)/(b0+b1)

a0 = 0; % deg(A(q)) = 1
beta = (1+am1+am2)/(b0+b1);
bm0 = beta*b0;
bm1 = beta*b1;

for i=3:size(t,2)
  y(i) = -am1*y(i-1)-am2*y(i-2)+bm0*uc(i-1)+bm1*uc(i-2);
end

y_noise = y;

for i=40:size(t,2)
  y_noise(i) = y(i)+0.5;
end

r1 = b1/b0 + (b1^2-am1*b0*b1+am2*b0^2)*(-b1+a0*b0)/(b0*(b1^2-a1*b0*b1+a2*b0^2));
s0_2 = b1*(a0*am1-a2-am1*a1+a1^2+am2-a1*a0)/(b1^2-a1*b0*b1+a2*b0^2)+...
    b0*(am1*a2-a1*a2-a0*am2+a0*a2)/(b1^2-a1*b0*b1+a2*b0^2);

s1_2 = b1*(a1*a2-am1*a2+a0*am2-a0*a2)/(b1^2-a1*b0*b1+a2*b0^2)+...
     b0*(a2*am2-a2^2-a0*am2*a1+a0*a2*am1)/(b1^2-a1*b0*b1+a2*b0^2);
```

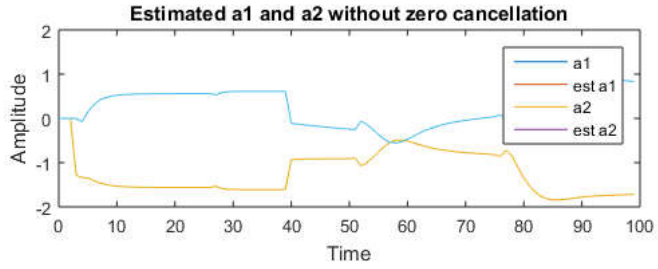
```
R2 = [1 r1];
S2 = [s0_2 s1_2];
T2 = [bm0/b0];

for i=2:size(t,2)
  u(i) = -R2(2)*u(i-1)+T2(1)*uc(i)-S2(1)*y(i)-S2(2)*y(i-1);
end

figure(3), subplot(2,1,1), plot(t,uc, t, y_noise), xlabel('Time'), ylabel('Amplitude'), title('y_noise(t) vs t and uc(t) vs t')
           subplot(2,1,2), plot(t,u),, xlabel('Time'), ylabel('Amplitude'), title('u(t) vs t without zero cancellation')
```

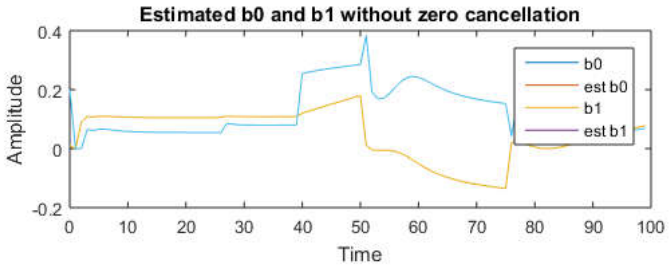
Estimation goes here

```
N = 4;
theta = zeros(N, size(t,2));
phi = zeros(N, size(t,2));
error = zeros(1,size(t,2));
K = zeros(N,1);
P = zeros(N);

%The initial conditions
%theta = [a1 a2 b0 b1]
theta(1,1) = 0;
theta(2,1) = 0;
theta(3,1) = 0.01;
theta(4,1) = 0.2;
I = eye(N);
P(1,1) = 100;
P(2,2) = 100;
P(3,3) = 1;
P(4,4) = 1;

lambda =  0.98;

for i = 3 : size(t,2)

phi(:,i) = [-y_noise(i-1) -y_noise(i-2) u(i-1) u(i-2)]';
K = P * phi(:,i)* inv(lambda + phi(:,i)'* P * phi(:,i));
P = ((I - K * phi(:,i)')* P)/lambda;
est_y = phi(:,i)'*theta(:,i-1);
error(i) = y(i)-est_y;

theta(:,i) = theta(:,i-1) + K * error(i);

end

figure(4), subplot(2,1,1), plot(t,a1,t,theta(1,:),t,a2,t,theta(2,:)), xlabel('Time'), ylabel('Amplitude'), title('Estimated a1 and a2 without zero cancellation'),
           subplot(2,1,2), plot(t,b0,t,theta(3,:),t,b1,t,theta(4,:)), xlabel('Time'), ylabel('Amplitude'), title('Estimated b0 and b1 without zero cancellation'),

% The controller obtained from the estimated parameters
est_A = [theta(1,size(t,2)) theta(2,size(t,2))];
est_B = [theta(3,size(t,2)) theta(4,size(t,2))];
h = 0.5; % The sampling time

est_Hq = tf(est_B, est_A, h, 'variable', 'q');
est_beta  = (1+am1+am2)/(est_B(1)+est_B(2));

b0 = est_B(1);
b1 = est_B(2);

est_r1 = b1/b0 + (b1^2-am1*b0*b1+am2*b0^2)*(-b1+a0*b0)/(b0*(b1^2-a1*b0*b1+a2*b0^2));
est_s0_2 = b1*(a0*am1-a2-am1*a1+a1^2+am2-a1*a0)/(b1^2-a1*b0*b1+a2*b0^2)+...
    b0*(am1*a2-a1*a2-a0*am2+a0*a2)/(b1^2-a1*b0*b1+a2*b0^2);

est_s1_2 = b1*(a1*a2-am1*a2+a0*am2-a0*a2)/(b1^2-a1*b0*b1+a2*b0^2)+...
     b0*(a2*am2-a2^2-a0*am2*a1+a0*a2*am1)/(b1^2-a1*b0*b1+a2*b0^2);

est_R2 = tf([1 est_r1], 1, h,'variable', 'q');
est_S2 = tf([est_s0_2 est_s1_2], 1, h,'variable', 'q');
est_T2 = tf([bm0/b0], 1, h,'variable', 'q');
```









Published with MATLAB® R2015a

Task 3 - Part IV: Example 3.10

Nelson Campos

Example 3.10: Load disturbances: Modified estimator and controller

We now show that the difficulties found in Example 3.9 can be avoided by modifying the estimator and the controller. We first introduce a controller that has integral action by applying the design procedure that we have just described. To do this, we consider the same system as in Example 3.5 where the controller was defined by

$$R^0 = q + r_1$$

$$S^0 = s_0 q + s_1$$

The closed-loop characteristic polynomial Ac has degree three. To obtain a controller with integral action, the order of the closed-loop system is increased by introducing an extra closed-loop pole at q = ?x0 = 0. It then follows from Eq. (3.41) that

$$y_0 = -\frac{1 + r_1}{b_0 + b_1}$$

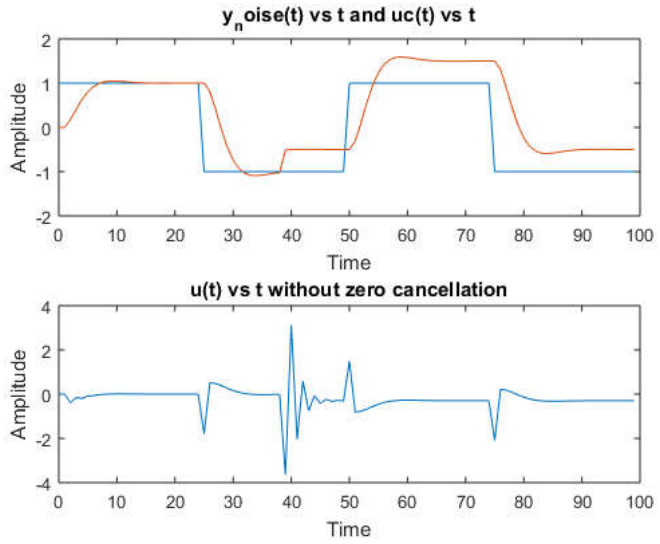
The estimates are based on the model (3.42) with Ad = q ? 1 to reduce the effects of the disturbances. Figure 2 shows a simulation corresponding to Fig. 2 with the modified self-tuning regulator. A comparison with the results of example 3.9 shows a significant improvement. The load disturbance is reduced quickly. Because of the integral action the control will decrease with a magnitude corresponding to the load disturbance shortly after t = 40. The parameters estimates are shown in Figure 2, which indicates the advantages in using the modified estimator. Notice in particular that there is a very small change in the estimates when the load disturbance occurs.

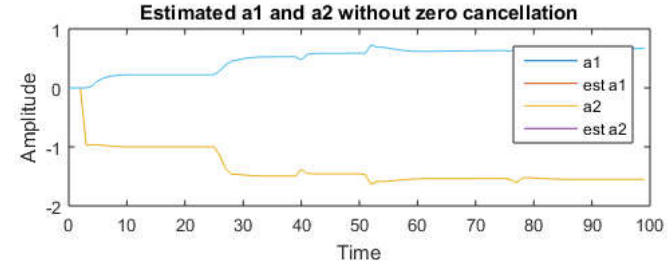
```
close all
clear all

a1 = -1.6065;
a2 = 0.6065;
b0 = 0.1065;
b1 = 0.0902;
%---------------
bm0 = 0.1761;
am1 = -1.3205;
am2 = 0.4966;

t_step = 1;
t = [0:t_step:100-t_step];
uc = zeros(1,size(t,2));
y = zeros(1, size(t,2));
T = 50;

for i=1:size(t,2)/T
    for j=1:T
        index = (i-1)*T+j;
        if j <= T/2
            uc(index) = 1;
        else uc(index) = -1;
        end
    end
end


u = zeros(1, size(t,2));

a0 = 0; % deg(A(q)) = 1
beta = (1+am1+am2)/(b0+b1);
bm0 = beta*b0;
bm1 = beta*b1;

for i=3:size(t,2)
  y(i) = -am1*y(i-1)-am2*y(i-2)+bm0*uc(i-1)+bm1*uc(i-2);
end

y_noise = y;

for i=40:size(t,2)
  y_noise(i) = y(i)+0.5;
end

yf = zeros(1,size(t,2)); % The filtered output

for i=2:size(t,2)
  yf(i) = y(i)-y(i-1);
end

r1 = b1/b0 + (b1^2-am1*b0*b1+am2*b0^2)*(-b1+a0*b0)/(b0*(b1^2-a1*b0*b1+a2*b0^2));
s0_2 = b1*(a0*am1-a2-am1*a1+a1^2+am2-a1*a0)/(b1^2-a1*b0*b1+a2*b0^2)+...
    b0*(am1*a2-a1*a2-a0*am2+a0*a2)/(b1^2-a1*b0*b1+a2*b0^2);

s1_2 = b1*(a1*a2-am1*a2+a0*am2-a0*a2)/(b1^2-a1*b0*b1+a2*b0^2)+...
     b0*(a2*am2-a2^2-a0*am2*a1+a0*a2*am1)/(b1^2-a1*b0*b1+a2*b0^2);

 y0 = -(1+r1)/(b0+b1);

% R2 = [1 r1];
% S2 = [s0_2 s1_2];
```

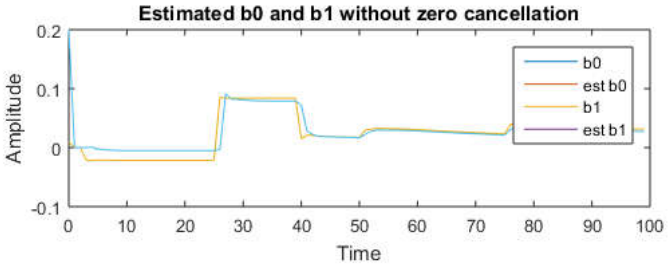
```
R2 = [1 -(y0*b1+1) y0*b1];
S2 = [(s0_2-y0) (s1_2-y0*a1) (-y0*a2)];

T2 = [bm0/b0];

uf = zeros(1,size(t,2)); % The filtered input
for i=3:size(t,2)
  u(i) = -R2(2)*u(i-1)-R2(3)*u(i-2)+T2(1)*uc(i)-S2(1)*y_noise(i)-S2(2)*y_noise(i-1)-S2(3)*y_noise(i-2);
  uf(i) = u(i)-u(i-1);
end

figure(1), subplot(2,1,1), plot(t,uc, t, y_noise), xlabel('Time'), ylabel('Amplitude'), title('y_noise(t) vs t and uc(t) vs t')
           subplot(2,1,2), plot(t,uf),, xlabel('Time'), ylabel('Amplitude'), title('u(t) vs t without zero cancellation')
```

Estimation goes here

```
N = 4;
theta = zeros(N, size(t,2));
phi = zeros(N, size(t,2));
error = zeros(1,size(t,2));
K = zeros(N,1);
P = zeros(N);

%The initial conditions
%theta = [a1 a2 b0 b1]
theta(1,1) = 0;
theta(2,1) = 0;
theta(3,1) = 0.01;
theta(4,1) = 0.2;
I = eye(N);
P(1,1) = 100;
P(2,2) = 100;
P(3,3) = 1;
P(4,4) = 1;

lambda =  0.98;

for i = 3 : size(t,2)

phi(:,i) = [-yf(i-1) -yf(i-2) uf(i-1) uf(i-2)]';
K = P * phi(:,i)* inv(lambda + phi(:,i)'* P * phi(:,i));
P = ((I - K * phi(:,i)')* P)/lambda;
est_y = phi(:,i)'*theta(:,i-1);
error(i) = yf(i)-est_y;

theta(:,i) = theta(:,i-1) + K * error(i);

end

figure(2), subplot(2,1,1), plot(t,a1,t,theta(1,:),t,a2,t,theta(2,:)), xlabel('Time'), ylabel('Amplitude'), title('Estimated a1 and a2 without zero cancellation'),
           subplot(2,1,2), plot(t,b0,t,theta(3,:),t,b1,t,theta(4,:)), xlabel('Time'), ylabel('Amplitude'), title('Estimated b0 and b1 without zero cancellation'),
```







Published with MATLAB® R2015a

Task 3 - Part V: Questions 3.5, 3.11 and 3.12

Nelson Campos

Contents

- Question 3.11
- Question 3.5
- Question 3.12
- Solution of the code of questions 3.5 and 3.12

Question 3.11

Apply the indirect self-tuning regulator in Example 3.5 to a process with the transfer function

$$G(s) = \frac{1}{(s+1)^2}$$

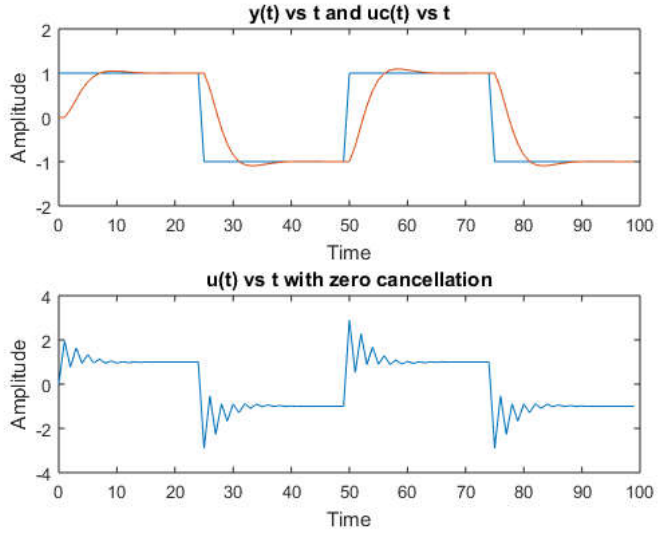
Study and explain the behavior of the error when the reference signal is a square wave.

Solution: the Fourier Transform of a Square Wave is given by

$$F(j\omega) = \frac{AT}{\frac{\omega T}{2}} sin(\frac{\omega T}{2})$$

where A and T are the amplitude and the period of the square wave, respectively. This means that at the time t = k*T/2, wher k is an integer, the error of the estimated signal will show peaks with their amplitudes decreasing with the frequency.

```
close all
clear all

w = warning ('off','all');

a1 = -1.213;
a2 = 0.3679;
b0 = 0.0902;
b1 = 0.06461;
%---------------
bm0 = 0.1761;
am1 = -1.3205;
am2 = 0.4966;

t_step = 1;
t = [0:t_step:100-t_step];
uc = zeros(1,size(t,2));
y = zeros(1, size(t,2));
T = 50;

for i=1:size(t,2)/T
    for j=1:T
        index = (i-1)*T+j;
        if j <= T/2
            uc(index) = 1;
        else uc(index) = -1;
        end
    end
end

for i=3:size(t,2)
  y(i) = -am1*y(i-1)-am2*y(i-2)+bm0*uc(i-1);
end

% R(q) = B+ = q+b1/b0, S(q) = s0q+s1, T(q) = AoBm' = bm0q/b0
s0 = (am1-a1)/b0;
s1 = (am2-a2)/b0;
R = [1 b1/b0];
S = [s0 s1];
T = [bm0/b0 0];

u = zeros(1, size(t,2));

for i=2:size(t,2)
  u(i) = -R(2)*u(i-1)+T(1)*uc(i)-S(1)*y(i)-S(2)*y(i-1);
end

figure(1), subplot(2,1,1), plot(t,uc, t, y), xlabel('Time'), ylabel('Amplitude'), title('y(t) vs t and uc(t) vs t')
           subplot(2,1,2), plot(t,u),, xlabel('Time'), ylabel('Amplitude'), title('u(t) vs t with zero cancellation')

sys = tf('s');
G = 1/(sys+1)^2;
Gd = c2d(G, 0.5);

%%Example 3.5
```

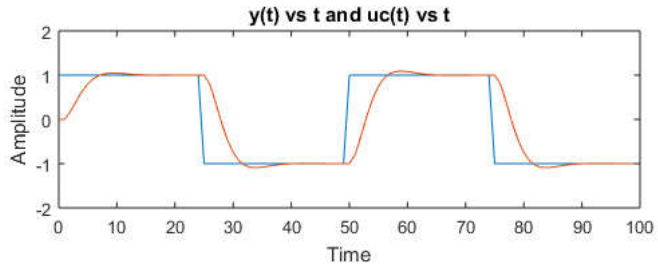


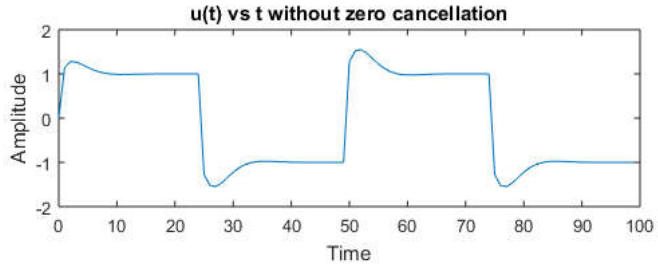

```
% Hm(q)= beta*B(q)/Am(q) = Bm(q)/Am(q), bm0 = beta*b0
% beta = (1+am1+am2)/(b0+b1)

a0 = 0; % deg(A(q)) = 1
beta = (1+am1+am2)/(b0+b1);
bm0 = beta*b0;
bm1 = beta*b1;

for i=3:size(t,2)
  y(i) = -am1*y(i-1)-am2*y(i-2)+bm0*uc(i-1)+bm1*uc(i-2);
end

r1 = b1/b0 + (b1^2-am1*b0*b1+am2*b0^2)*(-b1+a0*b0)/(b0*(b1^2-a1*b0*b1+a2*b0^2));
s0_2 = b1*(a0*am1-a2-am1*a1+a1^2+am2-a1*a0)/(b1^2-a1*b0*b1+a2*b0^2)+...
    b0*(am1*a2-a1*a2-a0*am2+a0*a2)/(b1^2-a1*b0*b1+a2*b0^2);

s1_2 = b1*(a1*a2-am1*a2+a0*am2-a0*a2)/(b1^2-a1*b0*b1+a2*b0^2)+...
     b0*(a2*am2-a2^2-a0*am2*a1+a0*a2*am1)/(b1^2-a1*b0*b1+a2*b0^2);

R2 = [1 r1];
S2 = [s0_2 s1_2];
T2 = [bm0/b0];

for i=2:size(t,2)
  u(i) = -R2(2)*u(i-1)+T2(1)*uc(i)-S2(1)*y(i)-S2(2)*y(i-1);
end

figure(3), subplot(2,1,1), plot(t,uc, t, y), xlabel('Time'), ylabel('Amplitude'), title('y(t) vs t and uc(t) vs t')
           subplot(2,1,2), plot(t,u),, xlabel('Time'), ylabel('Amplitude'), title('u(t) vs t without zero cancellation')
```

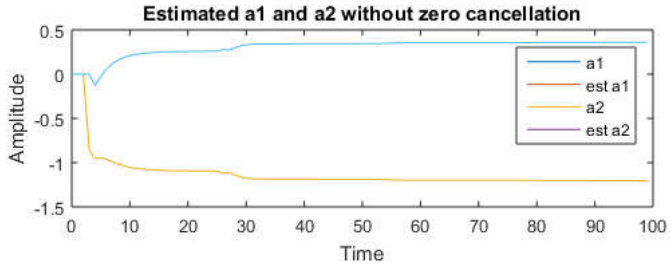
Estimation goes here

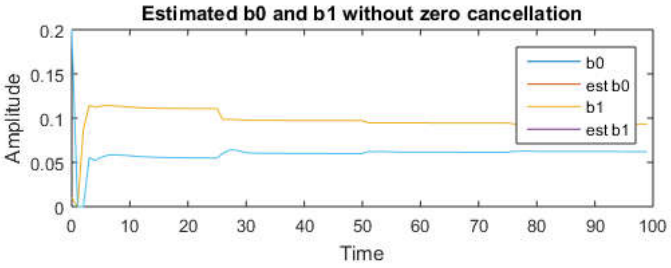
```
N = 4;
theta = zeros(N, size(t,2));
phi = zeros(N, size(t,2));
error = zeros(1,size(t,2));
K = zeros(N,1);
P = zeros(N);

%The initial conditions
%theta = [a1 a2 b0 b1]
theta(1,1) = 0;
theta(2,1) = 0;
theta(3,1) = 0.01;
theta(4,1) = 0.2;
I = eye(N);
P(1,1) = 100;
P(2,2) = 100;
P(3,3) = 1;
P(4,4) = 1;

lambda =  1;

for i = 3 : size(t,2)

phi(:,i) = [-y(i-1) -y(i-2) u(i-1) u(i-2)]';
K = P * phi(:,i)* inv(lambda + phi(:,i)'* P * phi(:,i));
P = ((I - K * phi(:,i)')* P)/lambda;
est_y = phi(:,i)'*theta(:,i-1);
error(i) = y(i)-est_y;
```

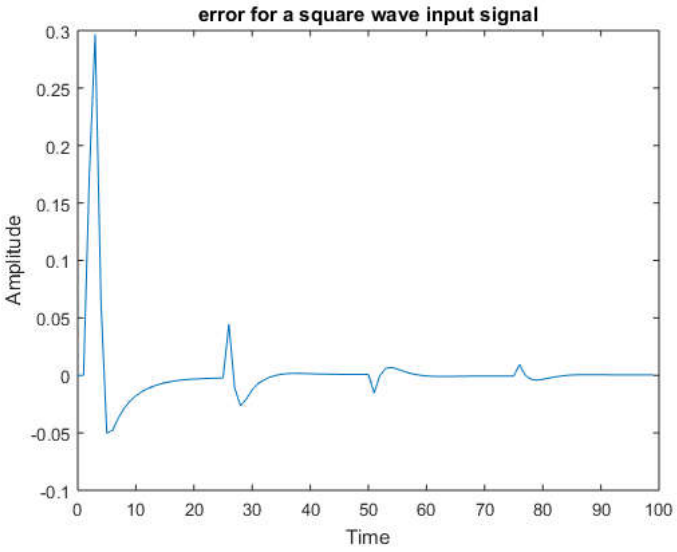
```
theta(:,i) = theta(:,i-1) + K * error(i);

end

figure(4), subplot(2,1,1), plot(t,a1,t,theta(1,:),t,a2,t,theta(2,:)), xlabel('Time'), ylabel('Amplitude'), title('Estimated a1 and a2 without zero cancellation'),
           subplot(2,1,2), plot(t,b0,t,theta(3,:),t,b1,t,theta(4,:)), xlabel('Time'), ylabel('Amplitude'), title('Estimated b0 and b1 without zero cancellation'),


figure(5), plot(t, error), xlabel('Time'), ylabel('Amplitude'), title('error for a square wave input signal')
```











Question 3.5

The code for simulating Examples 3.4 and 3.5 is listed below. Study the code and try to understand the details.

```
% 1.     DISCRETE SYSTEM reg
% 2.     "Indirect Self-Tuning Regulator based on the model
% 3.     " H(q)=(b0*q+b1)/(q^2+a1*q+a2)
% 4.     "using standard RLS estimation and pole placement design
% 5.     "Polynomial B is canceled if cancel>0.5
% 6.     INPUT ysp y "set point and process output
% 7.     OUTPUT u "control variable
% 8.     STATE ysp1 y1 u1 v1 "controller states
% 9.     STATE th1 th2 th3 th4 "parameter estimates
% 10.    STATE f1 f2 f3 f4 "regression variables
% 11.    STATE p11 p12 p13 p14 "covariance matrix
% 12.    STATE p22 p23 p24
% 13.    STATE p33 p34
% 14.    STATE p44
% 15.    NEW nysp1 ny1 nu1 nv1
% 16.    NEW nth1 nth2 nth3 nth4
% 17.    NEW nf1 nf2 nf3 nf4
% 18.    NEW n11 n12 n13 n14 n22 n23 n24 n33 n34 n44
% 19.    TIME t
% 20.    TSAMP ts
% 21.    INITIAL
% 22.    "Compute sampled Am and Ao
% 23.    a=exp(-z*w*h)
% 24.    am1=-2*a*cos(w*h*sqrt(1-z*z))
% 25.
% 26.    am2=a*a
% 27.    aop=IF w*To>100 THEN 0 ELSE -exp(-h/To)
% 28.    ao=IF cancel>0.5 THEN 0 ELSE -aop
% 29.    SORT
% 30.    "1.0 Parameter Estimation
% 31.    "1.1 Computation of P*f and estimator gain k
% 32.    pf1=p11*f1+p12*f2+p13*f3+p14*f4
% 33.    pf2=p12*f1+p22*f2+p23*f3+p24*f4
% 34.    pf3=p13*f1+p23*f2+p33*f3+p34*f4
% 35.    pf4=p14*f1+p24*f2+p34*f3+p44*f4
% 36.    denom=lambda+f1*pf1+f2*pf2+f3*pf3+f4*pf4
% 37.    k1=pf1/denom
% 38.    k2=pf2/denom
% 39.    k3=pf3/denom
% 40.    k4=pf4/denom
% 41.    "1.2 Update estimates and covariances
% 42.    eps=y-f1*th1-f2*th2-f3*th3-f4*th4
% 43.    nth1=th1+k1*eps
% 44.    nth2=th2+k2*eps
% 45.    nth3=th3+k3*eps
% 46.    nth4=th4+k4*eps
% 47.    n11=(p11-pf1*k1)/lambda
% 48.    n12=(p12-pf1*k2)/lambda
% 49.    n13=(p13-pf1*k3)/lambda
% 50.    n14=(p14-pf1*k4)/lambda
% 51.    n22=(p22-pf2*k2)/lambda
% 52.    n23=(p23-pf2*k3)/lambda
% 53.    n24=(p24-pf2*k4)/lambda
% 54.    n33=(p33-pf3*k3)/lambda
% 55.    n34=(p34-pf3*k4)/lambda
% 56.    n44=(p44-pf4*k4)/lambda
% 57.
% 58.    "1.3 Update and filter regression vector
% 59.    nf1=-y
```

```
% 60.    nf2=f1
% 61.    nf3=u
% 62.    nf4=f3
% 63.    "2.0 Control desig
% 64.    b0=nth3
% 65.    b1=nth4
% 66.    "2.2 Solve the polynomial identity AR+BS=AoAm
% 67.    n=b1*b1-a1*b0*b1+a2*b0*b0
% 68.    r10=(ao*am2*b0^2+(a2-am2-ao*am1)*b0*b1+(ao+am1-a1)*b1^2)/n
% 69.    w1=(a2*am1+a2*ao-a1*a2-am2*ao)*b0
% 70.    s00=(w1+(-a1*am1-a1*ao-a2+a1^2+am2+am1*ao)*b1)/n
% 71.    w2=(-a1*am2*ao+a2*am2+a2*am1*ao-a2^2)*b0
% 72.    s10=(w2+(-a2*am1-a2*ao+a1*a2+am2*ao)*b1)/n
% 73.    "2.3 Compute polynomial T=Ao*Am(1)/B(1)
% 74.    bs=b0+b1
% 75.    as=1+am1+am2
% 76.    bm0=as/bs
% 77.    "2.4 Choose control algorithm
% 78.    r1=IF cancel>0.5 THEN b1/b0 ELSE r10
% 79.    s0=IF cancel>0.5 THEN (am1-a1)/b0 ELSE s00
% 80.    s1=IF cancel>0.5 THEN (am2-a2)/b0 ELSE s10
% 81.    t0=IF cancel>0.5 THEN as/b0 ELSE bm0
% 82.    t1=IF cancel>0.5 THEN 0 ELSE bm0*ao
% 83.    "3.0 Control law with anti-windup
% 84.    v=-ao*v1+t0*ysp+t1*ysp1-s0*y-s1*y1+(ao-r1)*u1
% 85.    u=IF v<-ulim THEN -ulim ELSE IF v<ulim THEN v ELSE ulim
% 86.    "3.1 Update controller state
% 87.    ny1=y
% 88.    nu1=u
% 89.    nv1=v
% 90.    nysp1=ysp
% 91.    "4.0 Update sampling time
% 92.    ts=t+h
% 93.    "Parameters
% 94.    lambda:1 "forgetting factor
% 95.    To:200 "observer time constant
% 96.    z:0.7 "desired closed loop damping
% 97.    w:1 "desired closed loop natural frequency
% 98.    h:1 "sampling period
% 99.    ulim:1 "limit of control signal
% 100. cancel:1 "switch for cancellation
% 101. th1:-2 "initial estimates
% 102.
% 103. th2:1
% 104. th3:0.01
% 105. th4:0.01
% 106. p11:100 "initial covariances
% 107. p22:100
% 108. p33:100
% 109. p44:100
% 110. END
```

Question 3.12

The code for simulating Example 3.6 is listed below. Study the code and try to understand all the details.

```
% 1.    CONTINUOUS SYSTEM reg
% 2.    "Continuous time STR for the system b/[s(s+a)]
% 3.    "Desired response given by am2/(s^2+am1*s+am2)
% 4.    "Observer polynomial s+ao
% 5.    INPUT y ysp
% 6.    OUTPUT u
% 7.    STATE yf yf1 uf uf1 xu
% 8.    STATE th1 th2
% 9.    STATE p11 p12 p22
% 10.   DER dyf dyf1 duf duf1 dxu
% 11.   DER dth1 dth2
% 12.   DER dp11 dp12 dp22
% 13.   "Filter input and output
% 14.   dyf=yf1
% 15.   dyf1=-am1*yf1+am2*(y-yf)
% 16.   duf=uf1
% 17.   duf1=-am1*uf1+am2*(u-uf)
% 18.   "Update parameter estimate
% 19.   f1=-yf1
% 20.   f2=uf
% 21.   e=dyf1-f1*th1-f2*th2
% 22.   pf1=p11*f1+p12*f2
% 23.   pf2=p12*f1+p22*f2
% 24.   dth1=pf1*e
% 25.   dth2=pf2*e
% 26.   "Update covariance matrix
% 27.   dp11=alpha*p11-pf1*pf1
% 28.   dp12=alpha*p12-pf1*pf2
% 29.   dp22=alpha*p22-pf2*pf2
% 30.   det=p11*p22-p12*p12
% 31.   "Control design
% 32.   a=th1
% 33.   b=th2
% 34.   r1=am1+ao-a
```

```
% 35.   s0=(am2+am1*ao-a*r1)/b
% 36.   s1=am2*ao/b
% 37.   t0=am2/b
% 38.   "Control signal computation
% 39.   dxu=-ao*xu-(s1-ao*s0)*y+(ao-r1)*u
% 40.   v=t0*ysp-s0*y+xu
% 41.   u=if v<-ulim then -ulim else if v>ulim then ulim else v
% 42.   "Parameters
% 43.   am1:1.4
% 44.   am2:1
% 45.   alpha:0
% 46.   ao:2
% 47.   ulim:4
% 48.   END
```

Solution of the code of questions 3.5 and 3.12

The descripition of the code is given as follows: Line 1 defines a DISCRETE TIME regulator. All the text after " is a comment. Inputs and outputs are defined with the reserved words INPUT and OUTPUT. All the variables between input and output are state variables with the reserved word STATE. From lines 7 to 14, all the variables are being defined. All the parameters that is being to be estimated are defined as with the type NEW. These parameters are defined from lines 15 to 20. The program execution takes place between INITIAL and END. Inside theses blocks, the variables are initialized and the estiamated parameters also. From lines 43 to 56, the covariance matrix is being updated recursively. Note the similarities withe these pseudo-code (?) with the implementaiton of the examples 3.4 and 3.5. Regarding the the question 3.12, the logic is the same of the example 3.6. Is important to note that the derivative filters are have the reserved word DER, but they are just difference equations, once df(t)/dt ~= (f(t+h)-f(t))/h.

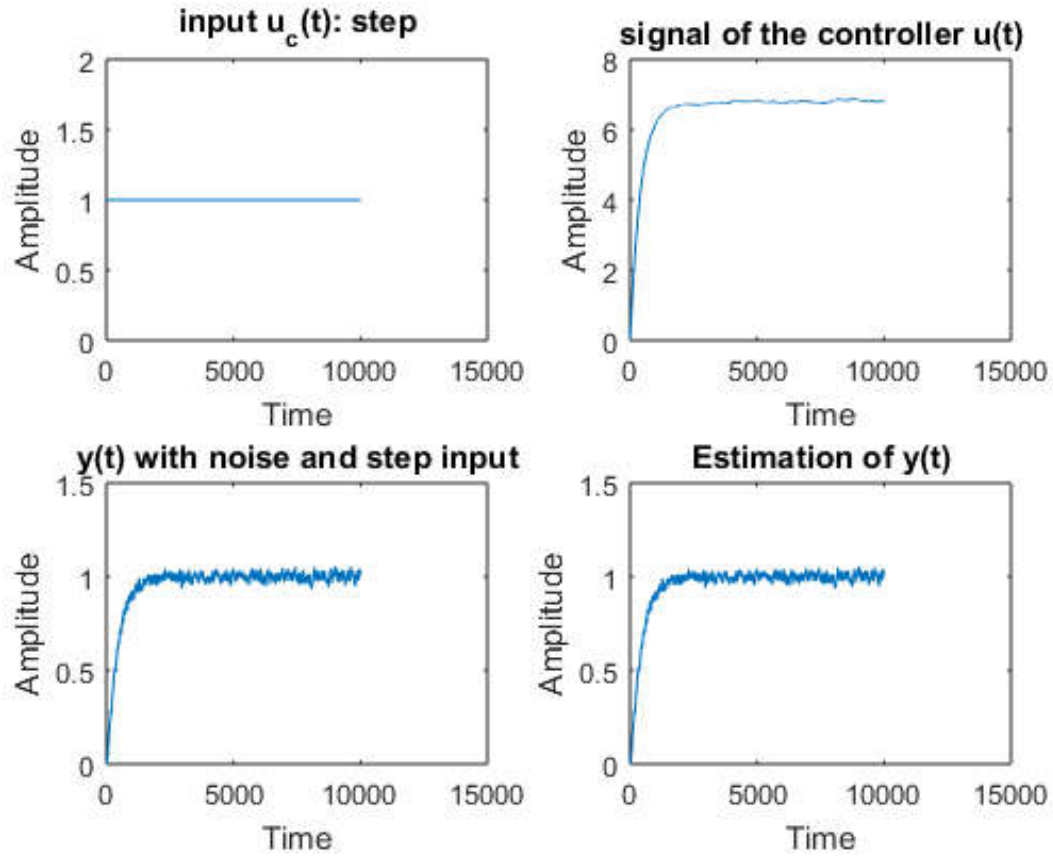
Published with MATLAB® R2015a

Question 3.15

Consider the system in Problem 1.9. (a) Sample the system, and determine a discrete-time controller for the known nominal system such that the specifications are satisfied. (b) Use a direct self-tuning controller, and study the transient for different initial conditions and different values of the variable parameters of the system. (c) Assume that e = 0 and that uc is a square wave. Simulate a selftuning controller for different prediction horizons. (d) Investigate the behavior when the disturbance d is a step. What happens when the controller does not have an integrator?

```
close all
clear all

Ki = 1.5;
K0 = 1;
deltaK = -0.5;
K = K0+deltaK;
a0 = 1.4;
delta_a = 2;
a = 1;
a = a0+delta_a;

sim('q315a.mdl')

uc = uc.signals.values;
u = u.signals.values;
y = y.signals.values;
t = 1:size(uc,1);

% Estimation goes here
N = 4;
theta = zeros(N, size(t,2));
phi = zeros(N, size(t,2));
error = zeros(1,size(t,2));
K = zeros(N,1);
P = zeros(N);

%The initial conditions
%theta = [a1 a2 b0 b1]
theta(1,1) = 0;
theta(2,1) = 0;
theta(3,1) = 0.01;
theta(4,1) = 0.2;
I = eye(N);
P(1,1) = 100;
P(2,2) = 100;
P(3,3) = 1;
P(4,4) = 1;

lambda =  1;
est_y = zeros(1, size(t,2));

for i = 3 : size(t,2)

phi(:,i) = [-y(i-1) -y(i-2) u(i-1) u(i-2)]';
K = P * phi(:,i)* inv(lambda + phi(:,i)'* P * phi(:,i));
```

```
P = ((I - K * phi(:,i)')* P)/lambda;
est_y(i) = phi(:,i)'*theta(:,i-1);
error(i) = y(i)-est_y(i);

theta(:,i) = theta(:,i-1) + K * error(i);

end

figure(1), subplot(2,2,1), plot(uc), xlabel('Time'), ylabel('Amplitude'), title('input u_c(t): step')
subplot(2,2,2), plot(u), xlabel('Time'), ylabel('Amplitude'), title('signal of the controller u(t)')
subplot(2,2,3),plot(y), xlabel('Time'), ylabel('Amplitude'), title('y(t) with noise and step input')
subplot(2,2,4),plot(est_y), xlabel('Time'), ylabel('Amplitude'), title('Estimation of y(t)')
```



Published with MATLAB® R2015a

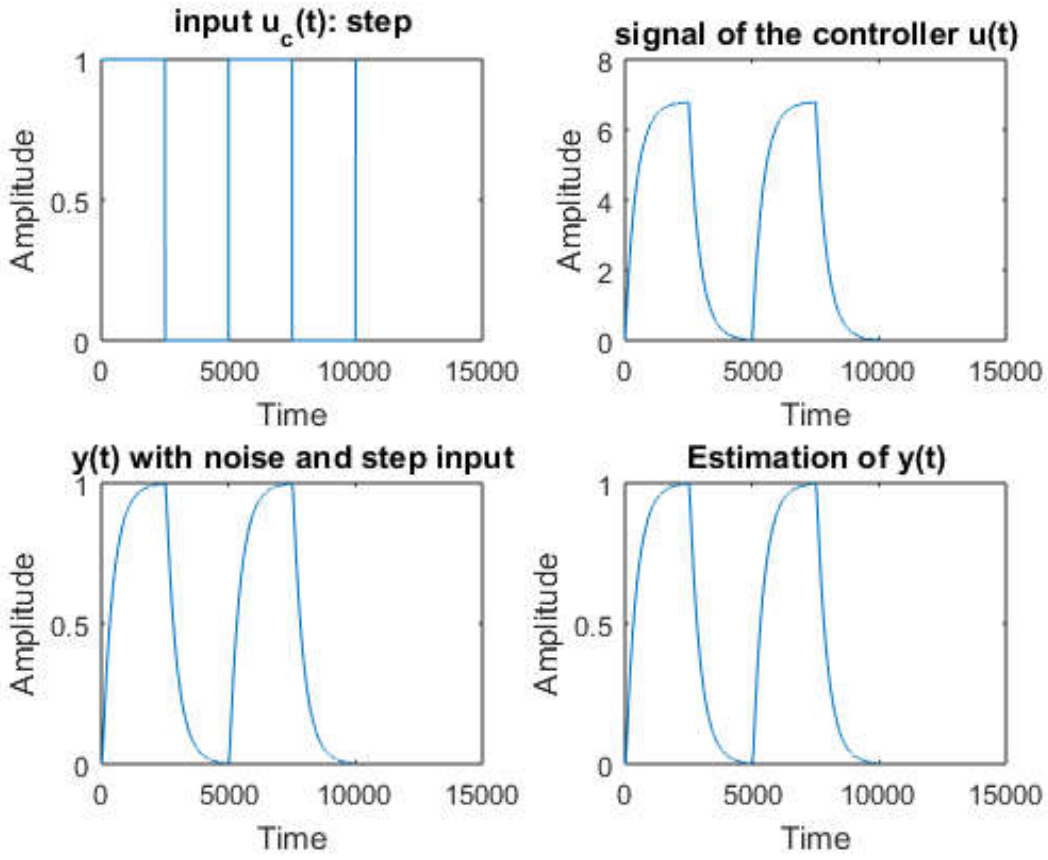
Question 3.15 part II

Contents

- References


```
close all
clear all

Ki = 1.5;
K0 = 1;
deltaK = -0.5;
K = K0+deltaK;
a0 = 1.4;
delta_a = 2;
a = 1;
a = a0+delta_a;

sim('q315b.mdl')

uc = uc.signals.values;
u = u.signals.values;
y = y.signals.values;
t = 1:size(uc,1);

% Estimation goes here
N = 4;
theta = zeros(N, size(t,2));
phi = zeros(N, size(t,2));
error = zeros(1,size(t,2));
K = zeros(N,1);
P = zeros(N);

%The initial conditions
%theta = [a1 a2 b0 b1]
theta(1,1) = 0;
theta(2,1) = 0;
theta(3,1) = 0.01;
theta(4,1) = 0.2;
I = eye(N);
P(1,1) = 100;
P(2,2) = 100;
P(3,3) = 1;
P(4,4) = 1;

lambda =  1;
est_y = zeros(1, size(t,2));

for i = 3 : size(t,2)

phi(:,i) = [-y(i-1) -y(i-2) u(i-1) u(i-2)]';
K = P * phi(:,i)* inv(lambda + phi(:,i)'* P * phi(:,i));
P = ((I - K * phi(:,i)')* P)/lambda;
est_y(i) = phi(:,i)'*theta(:,i-1);
error(i) = y(i)-est_y(i);
```

```
theta(:,i) = theta(:,i-1) + K * error(i);

end

figure(1), subplot(2,2,1), plot(uc), xlabel('Time'), ylabel('Amplitude'), title('input u_c(t): step')
subplot(2,2,2), plot(u), xlabel('Time'), ylabel('Amplitude'), title('signal of the controller u(t)')
subplot(2,2,3),plot(y), xlabel('Time'), ylabel('Amplitude'), title('y(t) with noise and step input')
subplot(2,2,4),plot(est_y), xlabel('Time'), ylabel('Amplitude'), title('Estimation of y(t)')
```



References

[1] Adaptive Control 2nd Edition Astrom & Wittenmark

Published with MATLAB® R2015a