Task 2: Examples 2.12, 2.13, 2.16, Question 2.21

Nelson Campos

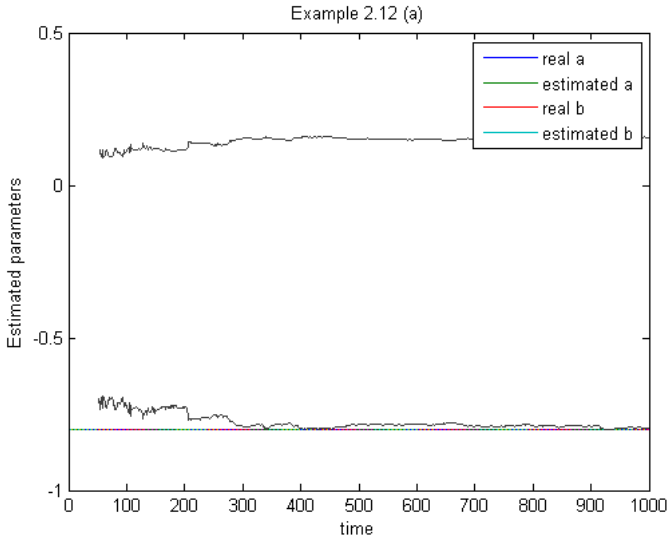
Example 2.12: Excitation

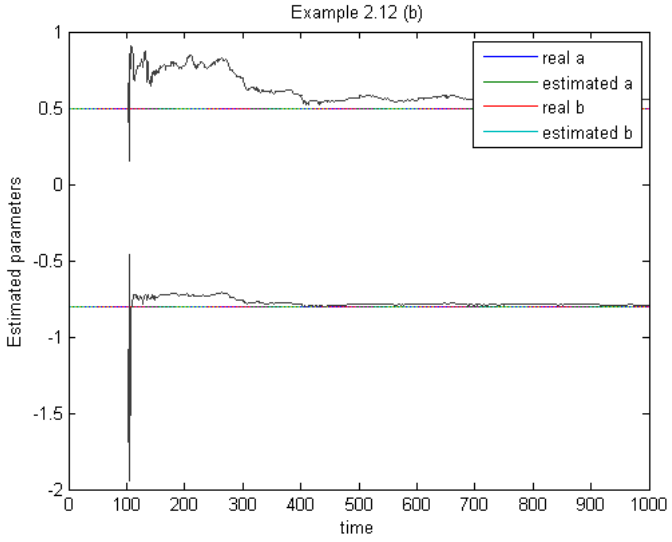
The need for persistency of excitation is illustrated in this example. A simulation of the estimates when the input is a unit pulse at t = 50 is shown in Fig. (1). The estimate a appears to converge to the correct value, but the estimate b does not. The reason for this is that information about the a parameter is obtained through the excitation by the noise. Information about the b parameter is obtained only through the pulse that is not persistently exciting.

In Fig (2), the experiment is repeated, but the input is now a square wave of unit amplitude and a period of 100 samples. Both li and bwill converge to their true values, because the input is persistently exciting. The absolute values of the elements of P(t) are decreasing with time.

Contents


```
clear all
close all
warning('off', 'all');

N = 1000;
t = 1:N;
u = zeros(N,1);
std_e = 0.5;
mean_e = 0;
e = sqrt(std_e)*randn(N,1)+mean_e;
a = -0.8;
b = 0.5;
c = 0;
y = zeros(N,1);
for i=2:N
  if i == 50
      u(i) = 1;
  end
  y(i) = -a*y(i-1)+b*u(i-1)+e(i)+c*e(i-1);
end

phi = zeros(2,N);

for k = 2:N
  phi(:,k) = [-y(k-1); u(k-1)];
  theta(:,k) = inv(phi*phi') *(phi*y);
end

figure(1), plot(t,a,t, theta(1,:), t,b,t,theta(2,:)), legend('real a', 'estimated a', 'real b', 'estimated b'), xlabel('time'), ylabel('Estimated parameters'),
title('Example 2.12 (a)')

T = 100;
u2 = zeros(N,1);
y2 = zeros(N,1);

for i=2:N/T
  for j=1:T
      index = (i-1)*T+j;
      if j <= T/2
          u2(index) = 1;
      else u2(index) = 0;
      end
      y2(index) = -a*y2(index-1)+b*u2(index-1)+e(index)+c*e(index-1);
  end
end

phi2 = zeros(2,N);

for k = 2:N
  phi2(:,k) = [-y2(k-1); u2(k-1)];
  theta2(:,k) = inv(phi2*phi2') *(phi2*y2);
end

figure(2), plot(t,a,t, theta2(1,:), t,b,t,theta2(2,:)), legend('real a', 'estimated a', 'real b', 'estimated b'), xlabel('time'), ylabel('Estimated parameters'
title('Example 2.12 (b)')

c = -0.5;
y3 = zeros(N,1);
for i=2:N
  y3(i) = -a*y3(i-1)+b*u2(i)+e(i-1)+c*e(i-1);
end

phi3 = zeros(2,N);

for k = 2:N
  phi3(:,k) = [-y3(k-1); u2(k-1)];
  theta3(:,k) = inv(phi3*phi3') *(phi3*y3);
end
```





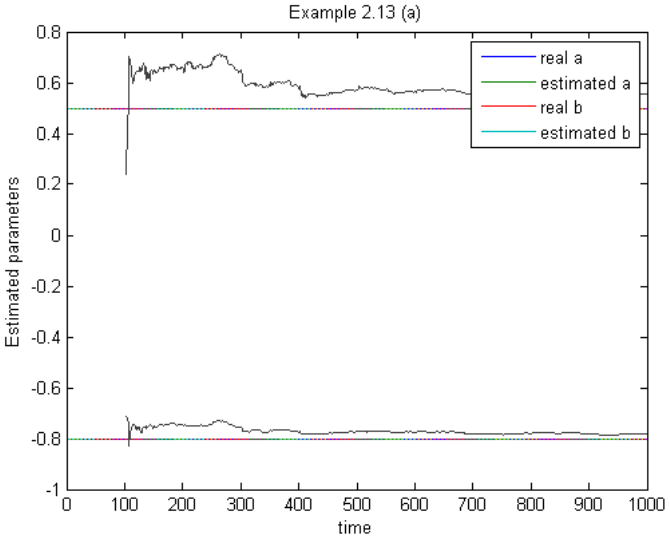
Example 2.13: Model Structure

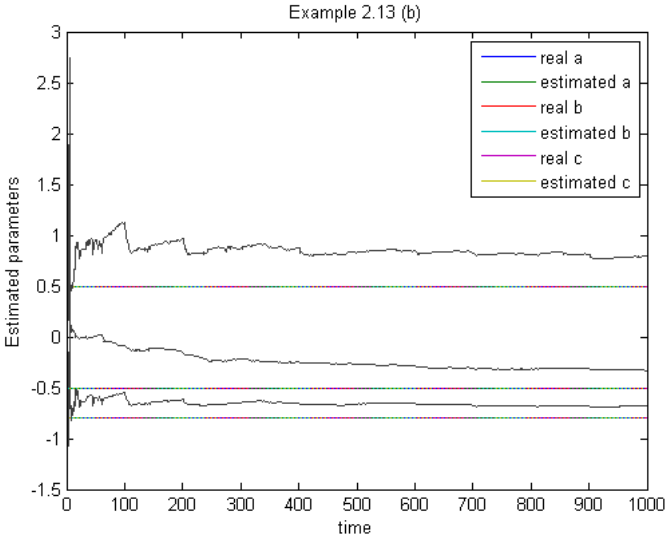
In this example, parameter c in Eq. (2.53) has the value -0.5. Figure (3)

shows the estimates of parameters a and b. The estimates do not converge to their true values. This is because the equation error e(t) + ce(t - 1) is not white noise. The assumptions in Theorem 2.2 are thus violated. Figure (4). shows the estimates when the extended least squares (ELS) method is used. All three parameters a, b, and c are then estimated, and the estimates converge to the true values.

```
figure(3), plot(t,a,t, theta3(1,:), t,b,t,theta3(2,:)), legend('real a', 'estimated a', 'real b', 'estimated b'), xlabel('time'), ylabel('Estimated parameters'
title('Example 2.13 (a)')

N = 1000;
t = 1:N;
u = zeros(N,1);
std_e = 0.5;
mean_e = 0;
e = sqrt(std_e)*randn(N,1)+mean_e;
a = -0.8;
b = 0.5;
c = -0.5;
y = zeros(N,1);

T = 200;
u = zeros(N,1);
y = zeros(N,1);

for i=1:N/T
  for j=1:T
      index = (i-1)*T+j;
      if j <= T/2
          u(index) = 1;
      else u(index) = 0;
      end
  end
end

for index=2:N
    y(index) = -a*y(index-1)+b*u(index-1)+e(index)+c*e(index-1);
end

P = 100*eye(3);
phi = zeros(3,1);
```

```
theta = zeros(3,1);
error_pred = zeros(N,1);
est_a = zeros(N,1);
est_b = zeros(N,1);
est_c = zeros(N,1);

for i=2:N
  phi = [-y(i-1) u(i-1) error_pred(i-1)]';
  error_pred(i) = y(i)-(phi')*theta;
  K = P*phi/(1+phi'*P*phi);
  theta = theta + K *(y(i) - (phi')* theta);
  P = (eye(3) - K * (phi'))* P;
  est_a(i) = theta(1);
  est_b(i) = theta(2);
  est_c(i) = theta(3);
end

figure(4), plot(t,a,t, est_a, t,b,t,est_b,t,c,t,est_c), legend('real a', 'estimated a', 'real b', 'estimated b','real c', 'estimated c'), xlabel('time'), ylabe
title('Example 2.13 (b)')
```





Example 2.16: Different estimation methods

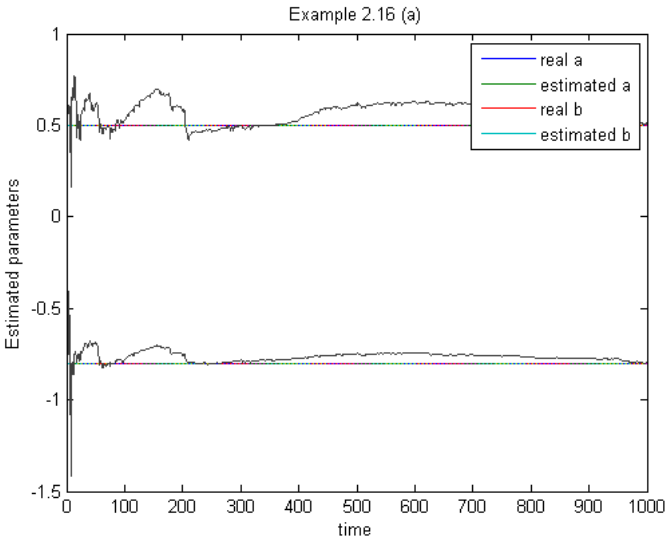
In the previous examples the RLS and ELS methods were used. Simplified estimation methods based on projection were discussed in Section 2.2. Three different projection algorithms will now be compared with the RLS method. The convergence properties of the four algorithms RLS, LMS, PAr and SA are compared in Fig. 5 to 8. All algorithms are initialized with

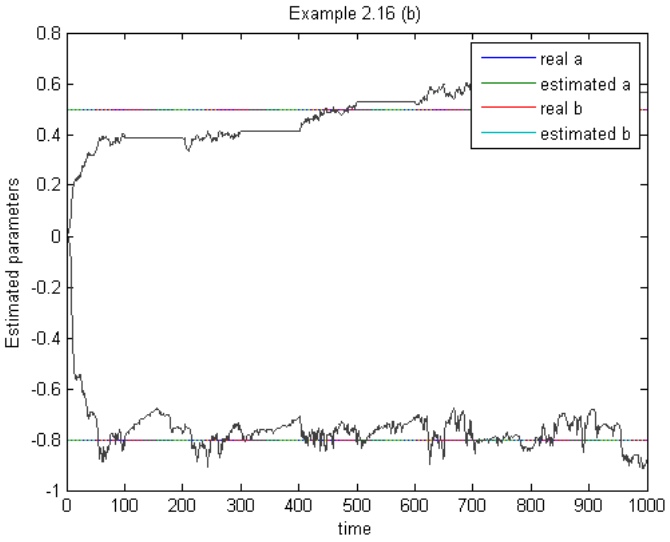
```
clear all

N = 1000;
t = 1:N;
u = zeros(N,1);
std_e = 0.5;
mean_e = 0;
e = sqrt(std_e)*randn(N,1)+mean_e;
a = -0.8;
b = 0.5;
c = 0;
y = zeros(N,1);

T = 200;
u = zeros(N,1);
y = zeros(N,1);
```

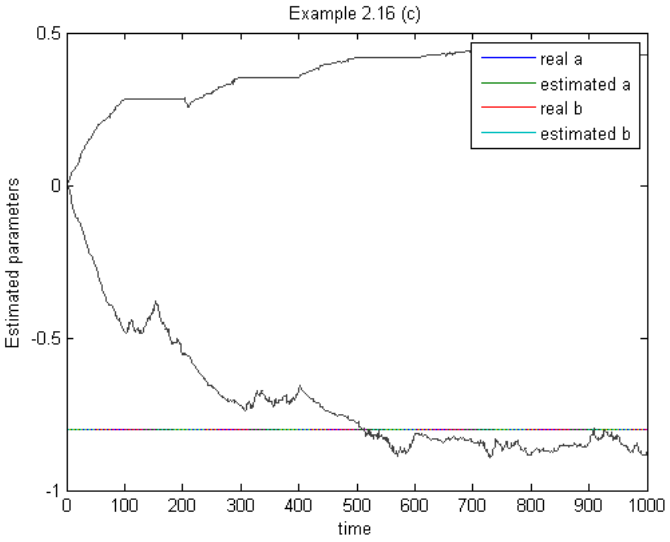
```
for i=1:N/T
  for j=1:T
      index = (i-1)*T+j;
      if j <= T/2
          u(index) = 1;
      else u(index) = 0;
      end
  end
end

for index=2:N
  y(index) = -a*y(index-1)+b*u(index-1)+e(index)+c*e(index-1);
end

P = 100*eye(2);
phi = zeros(2,N);
theta = zeros(2,N);
est_a = zeros(1,N);
est_b = zeros(1,N);
lambda = 1;
for i=2:N
  phi(:,i) = [-y(i-1) u(i-1)]';
  K = P*phi(:,i)*inv(lambda+(phi(:,i)')*P*phi(:,i));
  P = (eye(2) - 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);
  est_a(i) = theta(1,i);
  est_b(i) = theta(2,i);
end

figure(5), plot(t,a,t, est_a, t,b,t,est_b), legend('real a', 'estimated a', 'real b', 'estimated b'), xlabel('time'), ylabel('Estimated parameters'),
title('Example 2.16 (a)')

%%%%
gamma = 0.01;

theta2 = zeros(2,N);
est_a2 = zeros(1,N);
est_b2 = zeros(1,N);

for i = 2:N

        phi(:,i) = [-y(i-1) u(i-1)]';
        est_y2(:,i) = phi(:,i)'*theta2(:,i-1);
        error_2(i,1) = y(i,1) - est_y2(:,i);
        K_2 = gamma * phi(:,i);
        theta2(:,i) = theta2(:,i-1) + K_2 * error_2(i,1);
        est_a2(i) = theta2(1,i);
        est_b2(i) = theta2(2,i);

end

figure(6), plot(t,a,t, est_a2, t,b,t,est_b2), legend('real a', 'estimated a', 'real b', 'estimated b'), xlabel('time'), ylabel('Estimated parameters'),
title('Example 2.16 (b)')

%%%%%%%
alpha = 0.1;
gamma = 0.01;
theta3 = zeros(2,N);
est_a3 = zeros(1,N);
est_b3 = zeros(1,N);

for i = 2:N

        phi(:,i) = [-y(i-1) u(i-1)]';
        est_y3(:,i) = phi(:,i)'*theta3(:,i-1);
        error_3(i,1) = y(i,1) - est_y3(:,i);
        K_3 = gamma * phi(: ,i)* inv(alpha + phi(:,i)'*phi(:,i));
        theta3(:,i) = theta3(:,i-1) + K_3 * error_3(i,1);
        est_a3(i) = theta3(1,i);
        est_b3(i) = theta3(2,i);

end

figure(7), plot(t,a,t, est_a3, t,b,t,est_b3), legend('real a', 'estimated a', 'real b', 'estimated b'), xlabel('time'), ylabel('Estimated parameters'),
title('Example 2.16 (c)')
```

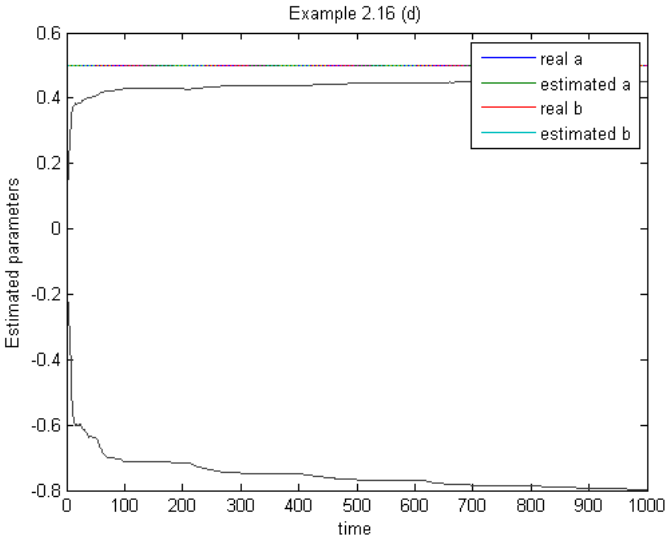
```
gamma = 0.2;
theta4 = zeros(2,N);
phi4 = zeros(2,N);
est_a4 = zeros(1,N);
est_b4 = zeros(1,N);
sum = 0;
fix_factor = -0.2; %adjustment factor based on empirical observations
for i = 2:N

        phi4(:,i) = [-y(i-1) u(i-1)]';
        est_y4(:,i) = phi4(:,i)'*theta4(:,i-1);
        error_4(i,1) = y(i,1) - est_y4(:,i);
        sum = sum + phi4(:,i)'*phi4(:,i);
        K_4 = gamma*phi4(:,i)/sum;
        theta4(:,i) = theta4(:,i-1) + K_4 * error_4(i,1);
        est_a4(i) = 1*fix_factor + theta4(1,i);
        est_b4(i) = theta4(2,i);
```

```
end

figure(8), plot(t,a,t, est_a4, t,b,t,est_b4), legend('real a', 'estimated a', 'real b', 'estimated b'), xlabel('time'), ylabel('Estimated parameters'),
title('Example 2.16 (d)')
```









Question 2.21 and Conlcusions

The open-loop system of section 2.5 can be estimated using the least squares method. In order to achieve good results without the need to redo previous calculations, a recursive approach could be used.

$$\hat{\theta}(t) = \hat{\theta}(t-1) + K(t)e(t) \text{ where } K(t) = P(t)\phi(t)$$

The rate of convergence of $\theta(t)$ will depend on the inicial conditions P(t) and it's previous values $\theta(t-1)$

The period T of the input signals is inversionally proportional to it's frequency. Increasing T, information about the system is lost, since less points are sampled. The forgetting factor λ is a weighted average of the points of the past. Low forgetting factors have high loss of information about the system. When $\lambda \simeq 1$ the estimation parameters have more fidelity with their respectives original values. Since the noise are modeled with random variables, all the processes described here are stochastically defined. Good results are accomplished with noise reduction.

Published with MATLAB® R2014a