```
% Auto Regressive Random Process computer  exercise 4.1
%M.Tognolini 17.03.2022
% MSE statistical signal processing Lausanne
%  rev 1.1  28.03.2022  Correction of  the error  computing in FIR filter
%  design line 140  we  must give  all  the  sample  to  compute  the  xhat
%  signal and  the  error  over  all the  samples ! (Thanks M Mieville)

clear;

clc
```

Padé approximation

M.Tognolini HEIG-VD 2022 MA-StatDig

Example 4.3.2

Test Padé approximation with the signal x(n) defined below.

Test with AR(2) model, AM(2) and ARMA(1,1).

```
N = 12;
n = 0:N-1;
xn = [1, 3* (0.5).^n(2:end)]
dn = [1,zeros(1,N-1)];
```

```
N = 12;
n = 0:N-1;
An = 0.2
```

```
An = 0.2000
```

```
rng default
x = [1, 3* (0.5).^n(2:end)] % noiseless
```

```
x = 1×12
    1.0000    1.5000    0.7500    0.3750    0.1875    0.0938    0.0469    0.0234 ···
```

```
x = x(:)
```

```
x = 12×1
    1.0000
    1.5000
    0.7500
    0.3750
    0.1875
    0.0938
    0.0469
    0.0234
    0.0117
    0.0059
```

⋮

```
xn = [1, 3* (0.5).^n(2:end)]+ An*randn(1,N) % noisy signal
```

```
xn = 1x12
    1.1075    1.8668    0.2982    0.5474    0.2513   -0.1678   -0.0398    0.0920 ···
```

```
dn = [1,zeros(1,N-1)];
```

a) Test with AR(2) : p=2 and q = 0

```
p=2
q=0
xn = xn(:)
```

```
p=2
```

```
p = 2
```

```
q=0
```

```
q = 0
```

```
xn = xn(:)
```

```
xn = 12x1
    1.1075
    1.8668
    0.2982
    0.5474
    0.2513
   -0.1678
   -0.0398
    0.0920
    0.7274
    0.5597
      ⋮
```

Compute the Xq matrix using the Matlab X= convmtx(xn,...) function and chose the right index to extract the Xq as in theory eq (4.13)

```
X = convmtx(xn,p+1)
```

```
X = convmtx(xn,p+1)
```

```
X = 14x3
    1.1075         0         0
    1.8668    1.1075         0
    0.2982    1.8668    1.1075
    0.5474    0.2982    1.8668
    0.2513    0.5474    0.2982
   -0.1678    0.2513    0.5474
   -0.0398   -0.1678    0.2513
    0.0920   -0.0398   -0.1678
```

```
    0.7274    0.0920   -0.0398
    0.5597    0.7274    0.0920
      ⋮
```

```
Xq = X(q+2:q+p+1, 2:p+1)
```

```
Xq = 2×2
    1.1075         0
    1.8668    1.1075
```

Compute the vector x_{q+1} as well:

```
xq_1 = xn(q+2:q+p+1)
```

```
xq_1 = 2×1
    1.8668
    0.2982
```

```
%Xq(1,2) = 1.0  for  test only
%Xq(2,1) = 1.0
```

And now compute the ap coefficients :

```
if (det(Xq)) %if det = 0  singular  matrix
    ap = inv(Xq)*(-xq_1);  % pinv is  used  in case
    ap = [1;ap]
else
    disp('Xq is  a singular matrix !')
end
```

```
ap = 3×1
    1.0000
   -1.6855
    2.5717
```

Compute the numerator coefficienzs here is obvous that bq(0) = xn(0)

```
X0 = X(1: q+1, 1:p+1 )
```

```
X0 = 1×3
    1.1075         0         0
```

```
bq = X0*ap
```

```
bq = 1.1075
```

Now compute the impulse response that is $\hat{x}(n) = h(n)$

```
xhat = filter(bq',ap',dn')
```

```
xhat = 12×1
    1.1075
    1.8668
    0.2982
   -4.2982
   -8.0116
```

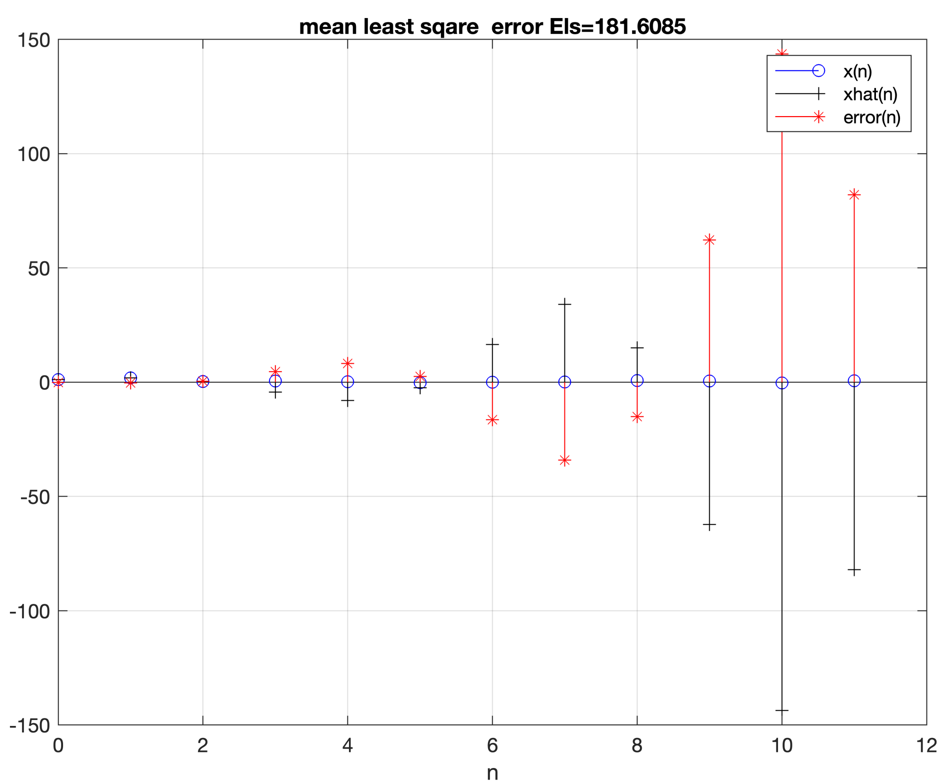
```
  -2.4501
  16.4740
  34.0684
  15.0566
 -62.2361
    ⋮
```

Compute the error of t he approximation and the least square error:

```
ep = x - xhat;
Els = norm(ep)
```

```
Els = 181.6085
```

```
figure(1)
stem(n,xn,'ob')
hold on
stem(n,xhat,'+k')
stem(n,ep,'*r')
hold off
xlabel('n')
legend('x(n)','xhat(n)','error(n)')
grid;
title(['mean least sqare  error Els=',num2str(Els)])
```



b) Test with MA(2) : p=0 and q = 2

In this case the solution for bq(n) is obvous because h(n) = bq(n) so bq(n) = x(n) :

```
q= 2
```

```
q = 2
```

```
p= 0
```

```
p = 0
```

```

bqma = xn(1:q+1)
```

```
bqma = 3×1
    1.1075
    1.8668
    0.2982
```

```
apma = 1
```

```
apma = 1
```

```

xhatma = filter(bqma',apma',dn')
```

```
xhatma = 12×1
    1.1075
    1.8668
    0.2982
         0
         0
         0
         0
         0
         0
         0
    ⋮
```

```
epma = x - xhatma;
Elsma = norm(epma)
```

```
Elsma = 0.7333
```

c) Test with ARMA(1,1) : p=1 and q = 1

In this case we need to solve with 2 step algorithm first we compute the ap coeff as in point a)

```
q= 1
```

```
q = 1
```

```
p= 1
```

```
p = 1
```

```
Xq = X(q+2:q+p+1, 2:p+1)
```

```
Xq = 1.8668
```

Compute the vector x_{q+1} as well:

```
xq_1 = xn(q+2:q+p+1)
```

```
xq_1 = 0.2982
```

```
if (det(Xq)) %if det = 0  singular  matrix
    ap_arma = inv(Xq)*(-xq_1);  % pinv is  used  in case
    ap_arma = [1;ap_arma]
else
    disp('Xq is  a singular matrix !')
end
```

```
ap_arma = 2×1
    1.0000
   -0.1598
```

Next we apply the step 2 and compute the bq coefficients:

```
X0 = X(1: q+1, 1:p+1 )
```

```
X0 = 2×2
    1.1075         0
    1.8668    1.1075
```

```
bq_arma = X0*ap_arma
```

```
bq_arma = 2×1
    1.1075
    1.6898
```

The transfer function H(z) is:

```
printsys(bq_arma',ap_arma','z')
```

```
num/den =

   1.1075 z + 1.6898
   -----------------
      z - 0.15976
```

```
xhat_arma = filter(bq_arma',ap_arma',dn')
```

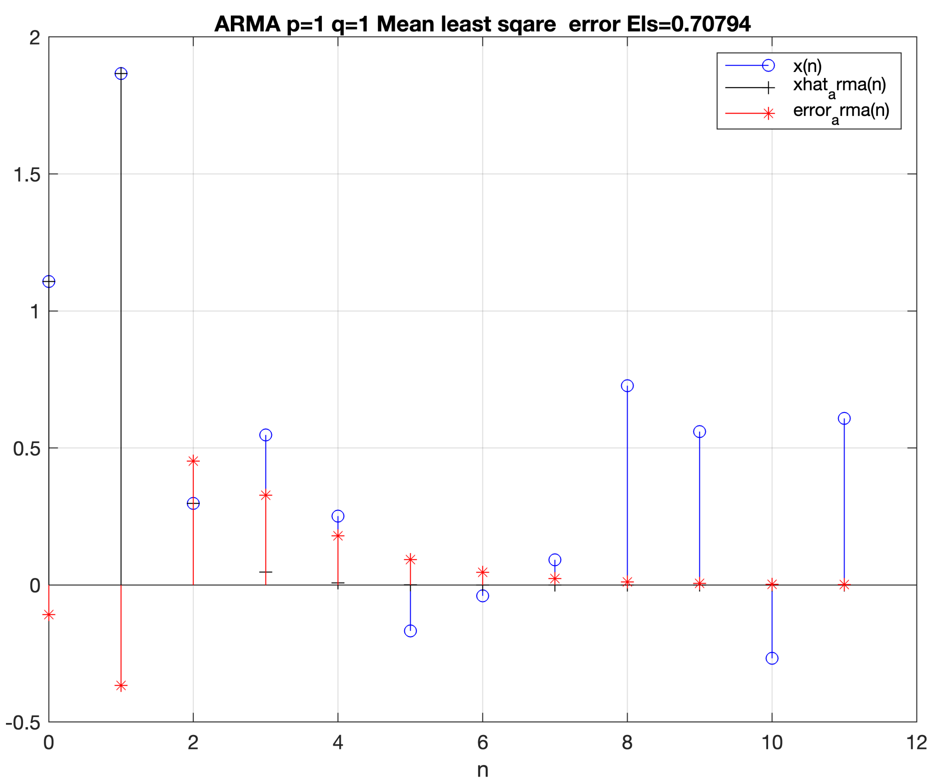
```
xhat_arma = 12×1
    1.1075
    1.8668
    0.2982
    0.0476
    0.0076
    0.0012
    0.0002
    0.0000
    0.0000
```

```
0.0000
  ⋮
```

```
ep_arma = x - xhat_arma;
Els_arma = norm(ep_arma)
```

```
Els_arma = 0.7079
```

```

figure(2)
stem(n,xn,'ob')
hold on
stem(n,xhat_arma,'+k')
stem(n,ep_arma,'*r')
hold off
xlabel('n')
legend('x(n)','xhat_arma(n)','error_arma(n)')
grid;
title(['ARMA p=',num2str(p),' q=',num2str(q),' Mean least sqare  error Els=',num2str(E
```



Build a functionto compute the coefficients ap and bq by Padé approximation

The function heade would be :

```
help pade
```

```
pade compute the pade approximation of the coeff of  ARMA  system with p
poles  and q zeros usage: [ap,bq,Els,xhat] = pade(x,p,q)
Els is optional and is the L2 norm of  the approx error x-xhat
xhat is optional  and it is the  approximated signal.
(c) M.Tognolini HEIG-VD 2022 r1.0

  Other functions named pade
```

test it with the prevous example.

```
p=2
```

```
p = 2
```

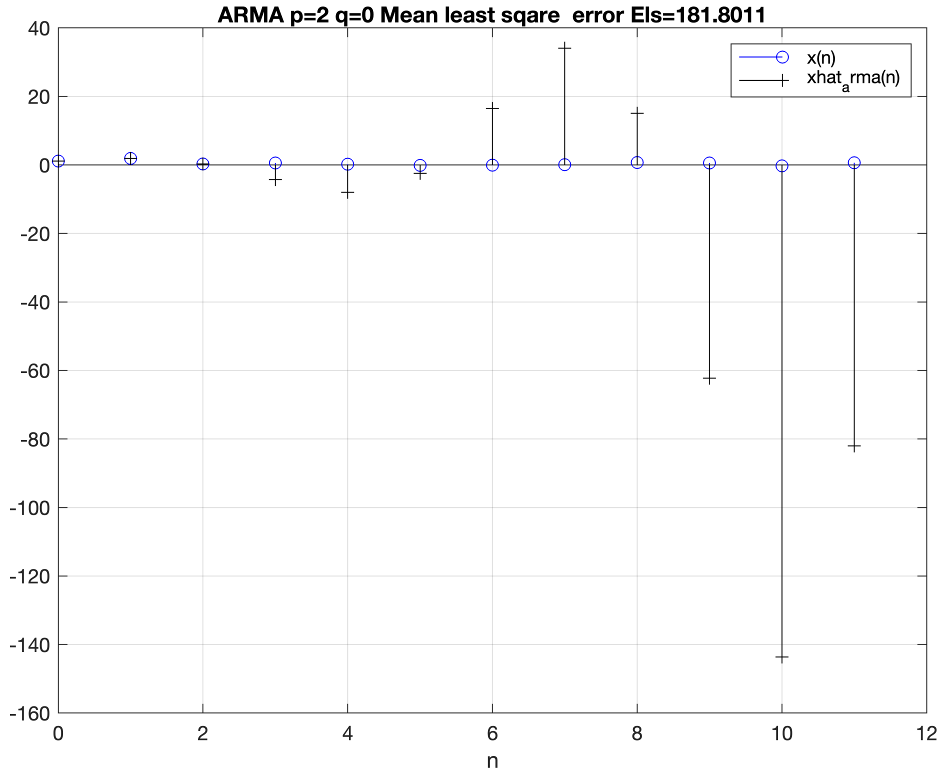
```
q=0
```

```
q = 0
```

```
[ap_test,bq_test,Els_test,xhat_test] = pade(xn,p,q)
```

```
Calculating the  squared error of  the approximation
and the approximation signal xhat
ap_test = 3×1
    1.0000
   -1.6855
    2.5717
bq_test = 1.1075
Els_test = 181.8011
xhat_test = 12×1
    1.1075
    1.8668
    0.2982
   -4.2982
   -8.0116
   -2.4501
   16.4740
   34.0684
   15.0566
  -62.2361
     ⋮
```

```
figure(3)
stem(n,xn,'ob')
hold on
stem(n,xhat_test,'+k')

hold off
xlabel('n')
legend('x(n)','xhat_arma(n)')
grid;
title(['ARMA p=',num2str(p),' q=',num2str(q),' Mean least sqare  error Els=',num2str(E
```



Example 4.3.3 : Singular exemple

Digital filter approximation

```
x2 = [1,4,2,1,3]
```

```
x2 = 1×5
     1     4     2     1     3
```

```
p=2
```

```
p = 2
```

```
q=2
```

```
q = 2
```

Compute the pade approximation with the pade function:

```
[ap_2,bq_2] = pade(x2,p,q)
```

```
Error using pade
Xq is  a singular matrix !
```

The Xq matrix is singular in this case !

That means the assumption that ap(0) = 1 is incorrect for this model so we put it to 0.

Example 4.3.4: Filter design using the Padé approximation

The proble of the filter approximation is to find the impulse response h(n) of the filter from a frequency constraints.

For this exemple to be as simple as possible we want to synthetize a filter with a frequency response $|H(jf)| = 1$ for f < Fp and 0 for Fp < f < 0.5,

aditionally we want a linear phase response then $H(jf) = e^{-jn_d 2\pi f}$ for f < Fp and 0 for Fp < f < 0.5 , the constant nd is the time delay of the filter to assure the causality of the filter.

$$H(jf) = \begin{cases} e^{jn_d 2\pi f} & ; \ |f| < 0.25 \\ 0 & ; \ \text{otherwise} \end{cases}$$

Since the frequency response is a rectangular function the inverse transform is a sinc function of the variable n, the the impulse response is :

$$h(n) = \frac{sin[(n - n_d)\pi/2]}{(n - n_d)\pi}$$

With the function sinc we write it as: hn = 1/2 * sinc((n-nd)*2*Fp)

```
N = 100
Fp = 0.25
n= 0:N-1;
nd = 5

hn = 1/2 * sinc((n-nd)*2*Fp)

H = fftshift(fft(hn));
f = -0.5:1/N:0.5-0.5/N;


figure;
subplot(2,1,1)
stem(n,hn)
axis([0,20,-0.2,0.6])
grid;
xlabel('n')
legend('h(n)')
subplot(2,1,2)
plot(f,20*log10(abs(H)))
axis([0,0.5,-40,1])
grid;
xlabel('f [-]')
legend('|H(f)|')
```

```
N = 100
```

```
N = 100
```

```
Fp = 0.25
```

```
Fp = 0.2500
```

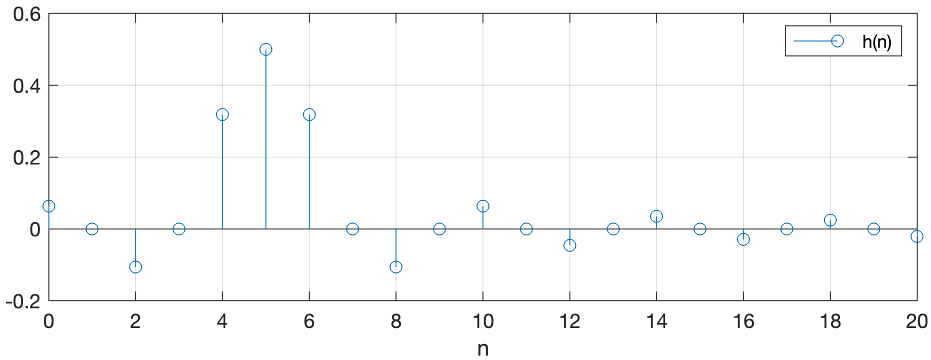
```
n= 0:N-1;
nd = 5
```

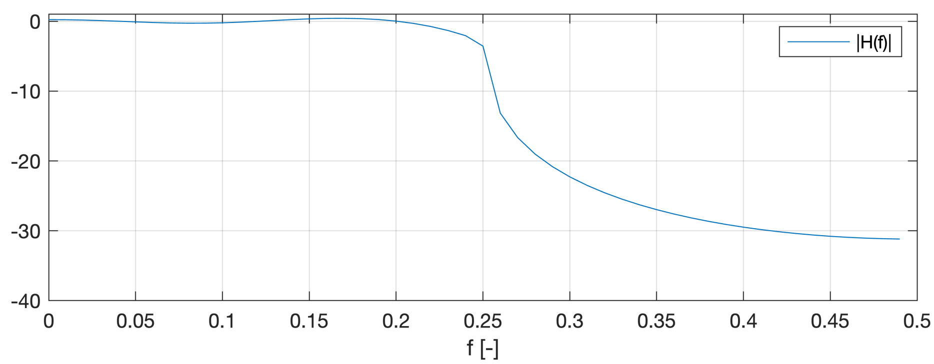
```
nd = 5
```

```


hn = 1/2 * sinc((n-nd)*2*Fp)
```

```
hn = 1×100
    0.0637   -0.0000   -0.1061    0.0000    0.3183    0.5000    0.3183    0.0000 ···
```

```
H = fftshift(fft(hn));
f = -0.5:1/N:0.5-0.5/N;
figure;
subplot(2,1,1)
stem(n,hn)
axis([0,20,-0.2,0.6])
grid;
xlabel('n')
legend('h(n)')
subplot(2,1,2)
plot(f,20*log10(abs(H)))
axis([0,0.5,-40,1])
grid;
xlabel('f [-]')
legend('|H(f)|')
```





If we design a FIR filter with q= 10 and p=0 (MA) the Padé approximation wil give the 11 first values of the h(n). In this case bq(n) = h(n) for n = 0 ..10.

```
p=0
```

```
p = 0
```

```
q=10
```

```
q = 10
```

```
bqFIR = hn(1:p+q+1)
```

```
bqFIR = 1×11
    0.0637   -0.0000   -0.1061    0.0000    0.3183    0.5000    0.3183    0.0000 ⋯
```

```
apFIR = [1,zeros(1,10)]
```

```
apFIR = 1×11
     1     0     0     0     0     0     0     0     0     0     0
```

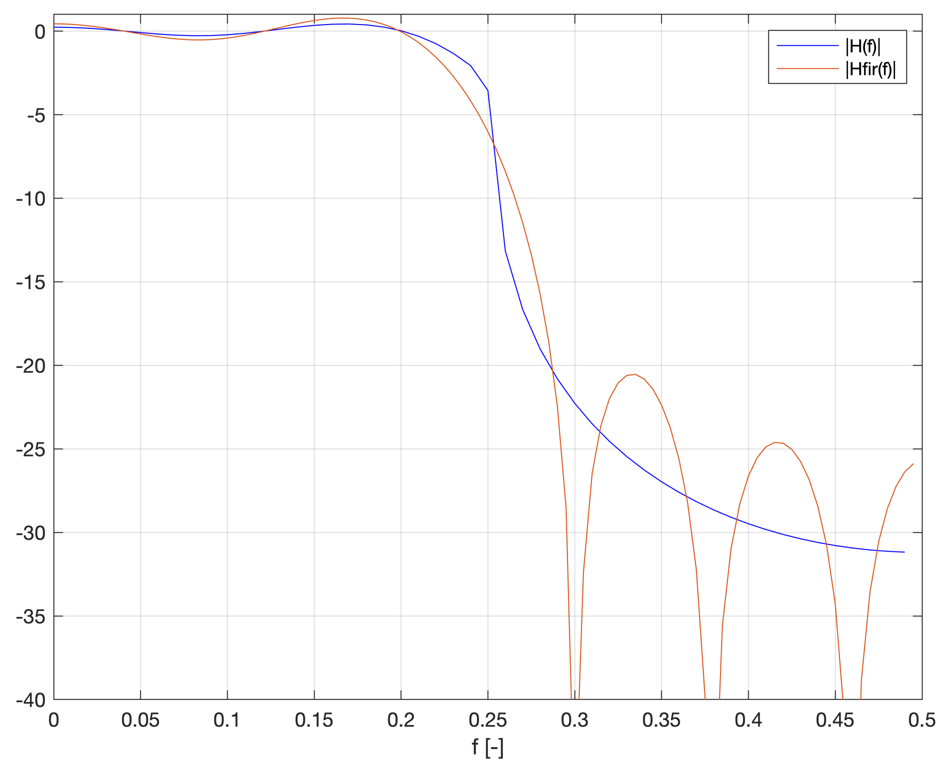
```
Hfir = tf(bqFIR,apFIR,1)
```

```
Hfir =
```

```
0.06366 z^10 - 1.949e-17 z^9 - 0.1061 z^8 + 1.949e-17 z^7 + 0.3183 z^6 + 0.5 z^5 + 0.3183 z^4 + 1.949e
                                                                                   -17 z^3 - 0.1061 z^2 - 1.949e-17 z -
-------------------------------------------------------------------------------------------------------------
                                                   z^10

Sample time: 1 seconds
Discrete-time transfer function.
```

We could plot the frequency response of this filter with the function freqz():

```
[HFIR,f2]=freqz(bqFIR,apFIR ,N,1);


figure;
plot(f,20*log10(abs(H)),'b', f2,20*log10(abs(HFIR)))
axis([0,0.5,-40,1])
grid;
xlabel('f [-]')
legend('|H(f)|','|Hfir(f)|')
```



We want to find the transfer fuction of an ARMA system with p=6 and q=6

```
p=6
```

```
p = 6
```

```
q=6
```

```
q = 6
```

```
[ap_IIR,bq_IIR,Els_IIR] = pade(hn,p,q)
```

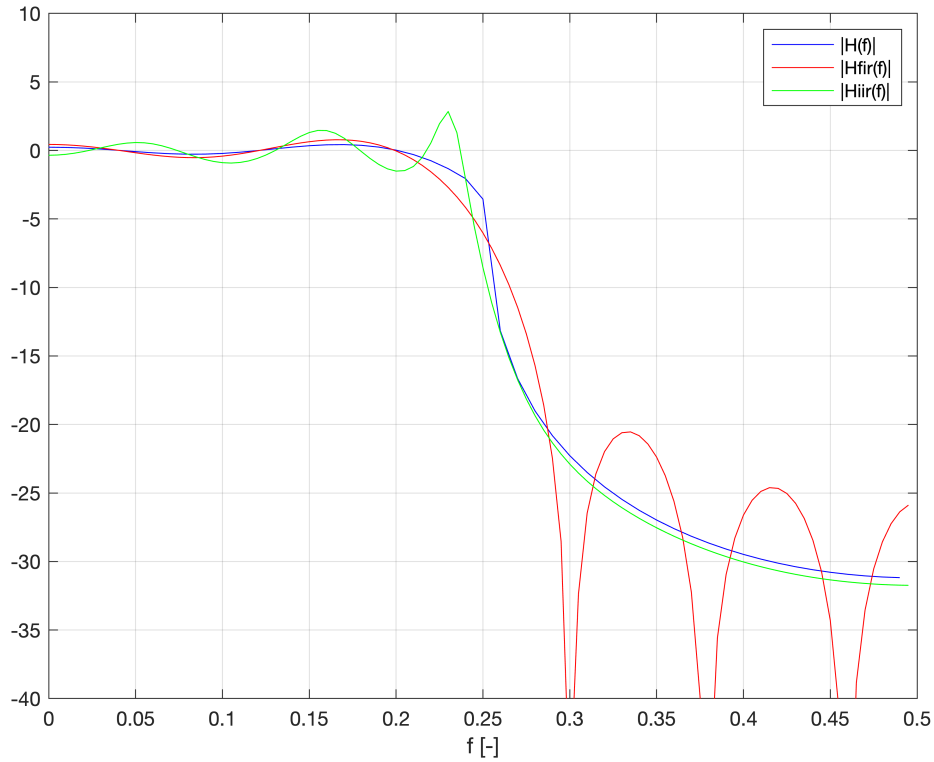
```
Calculating the  squared error of  the approximation
ap_IIR = 7×1
    1.0000
   -2.5171
    3.9055
   -4.0872
    2.9647
   -1.4088
    0.3500
bq_IIR = 7×1
    0.0637
   -0.1602
    0.1425
    0.0069
    0.0927
    0.0428
    0.0106
Els_IIR = 0.1365
```

```
Hiir = tf(bq_IIR',ap_IIR',1)
```

```
Hiir =

  0.06366 z^6 - 0.1602 z^5 + 0.1425 z^4 + 0.006869 z^3 + 0.09267 z^2 + 0.04277 z + 0.01064
  ---------------------------------------------------------------------------------------
            z^6 - 2.517 z^5 + 3.905 z^4 - 4.087 z^3 + 2.965 z^2 - 1.409 z + 0.35

Sample time: 1 seconds
Discrete-time transfer function.
```

As before we will plot the frequency response compared to the desired frequency response H(f):

```
[HIIR,f2]=freqz(bq_IIR,ap_IIR ,N,1);


figure;
plot(f,20*log10(abs(H)),'b', f2,20*log10(abs(HFIR)),'r',f2,20*log10(abs(HIIR)),'g')
axis([0,0.5,-40,10])
grid;
xlabel('f [-]')
legend('|H(f)|','|Hfir(f)|', '|Hiir(f)|')
```

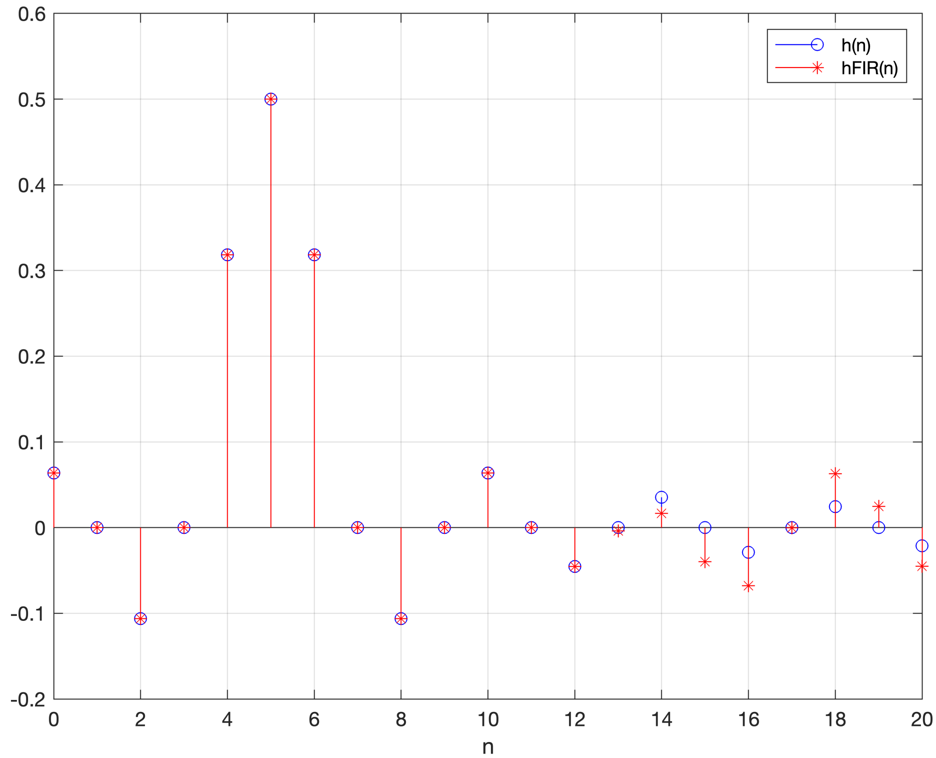


Compute the impulse response of this filter over the N samples and compare to the original impulse response h(n):

```
dn=[1,zeros(1,N-1)];
hiir = filter(bq_IIR,ap_IIR,dn)
```

```
hiir = 1×100
    0.0637   -0.0000   -0.1061         0    0.3183    0.5000    0.3183   -0.0000 ···
```

```
figure
stem(n,hn,'ob')
hold on
stem(n,hiir,'*r')
hold off
axis([0,20,-0.2,0.6])
grid;
xlabel('n')
legend('h(n)','hFIR(n)')
```



h(n)
hFIR(n)
0.6
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
0
2
4
6
8
10
12
14
16
18
20
n