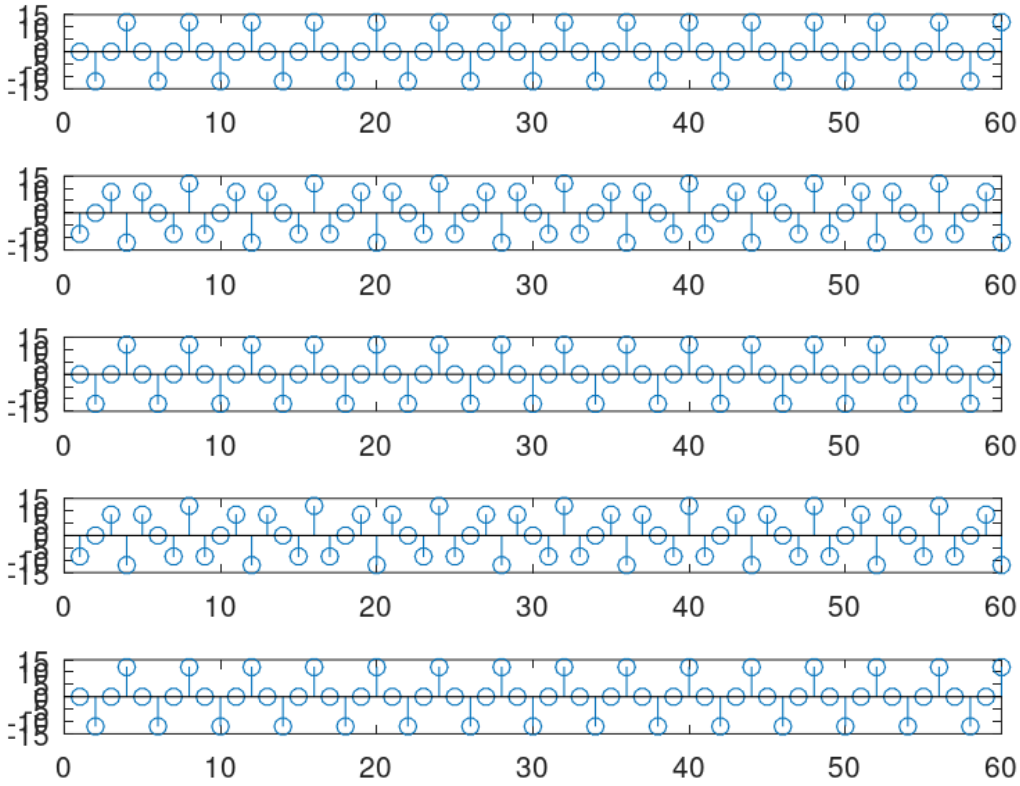
Aliasing_01

```
% Aliasing
clc
clear
A = 12;
F1 = 10; % Unit : Hz
F2 = 15; % Unit : Hz
F3 = 50; % Unit : Hz
F4 = 95; % Unit : Hz
F5 = 12010; % Unit : Hz
Fs = 40; % Unit : Hz
for n = 1:1:60
  x1(n) = A * cos(2*pi*F1*n/Fs);
  x2(n) = A * cos(2*pi*F2*n/Fs);
  x3(n) = A * cos(2*pi*F3*n/Fs);
  x4(n) = A * cos(2*pi*F4*n/Fs);
  x5(n) = A * cos(2*pi*F5*n/Fs);
endfor
x1;
%figure (1), stem(x1)
%figure (2), stem(x2)
%figure (3), stem(x3)
%figure (4), stem(x4)
%figure (5), stem(x5)

subplot(5,1,1), stem(x1)
subplot(5,1,2), stem(x2)
subplot(5,1,3), stem(x3)
subplot(5,1,4), stem(x4)
subplot(5,1,5), stem(x5)
```

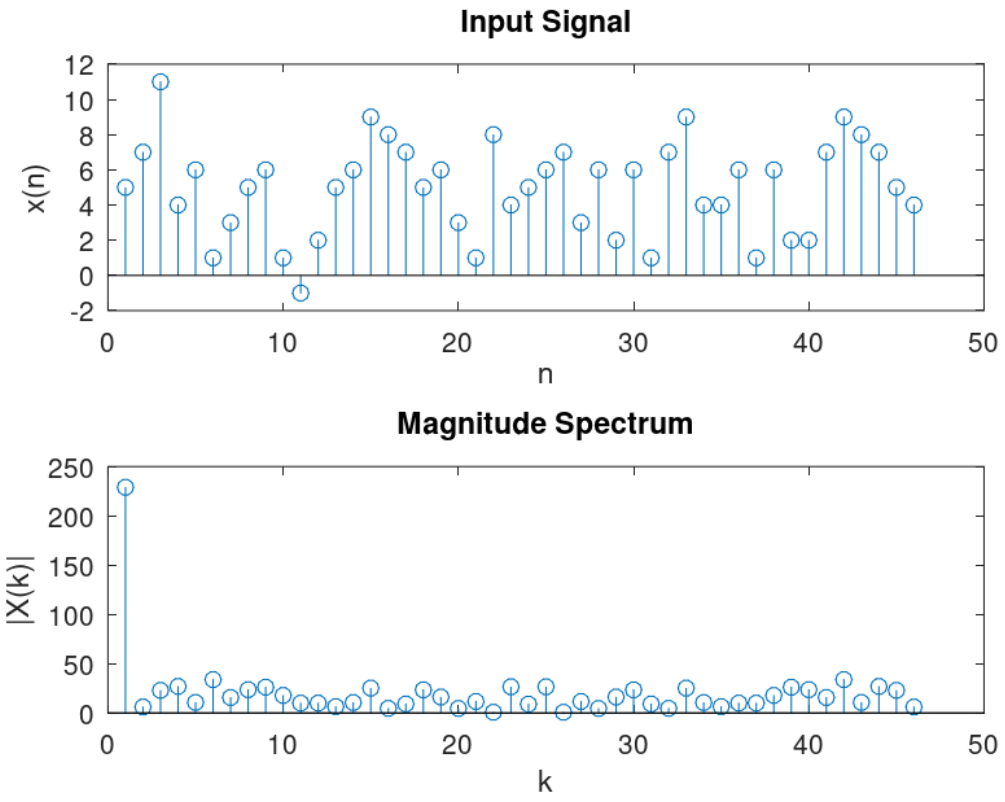


Published with GNU Octave 9.3.0

DFT_02

```
%DFT Code
clc
clear
x = [5 7 11 4 6 1 3 5 6 1 -1 2 5 6 9 8 7 5 6 3 1 8 4 5 6 7 3 6 2 6 1 7 9 4 4 6 1 6 2 2 7 9 8 7 5 4 ];

N = length (x)

for n = 0:1:N-1
  for k = 0:1:N-1
    W(n+1,k+1) = exp(-j*2*pi*n*k/N);
  endfor
end
W;
X = W*x';
Xmag = abs(X);
subplot(2,1,1) , stem(x) , xlabel("n") , ylabel("x(n)"), title("Input Signal")
subplot(2,1,2) , stem(Xmag) , xlabel("k") , ylabel("|X(k)|") , title("Magnitude Spectrum")
```

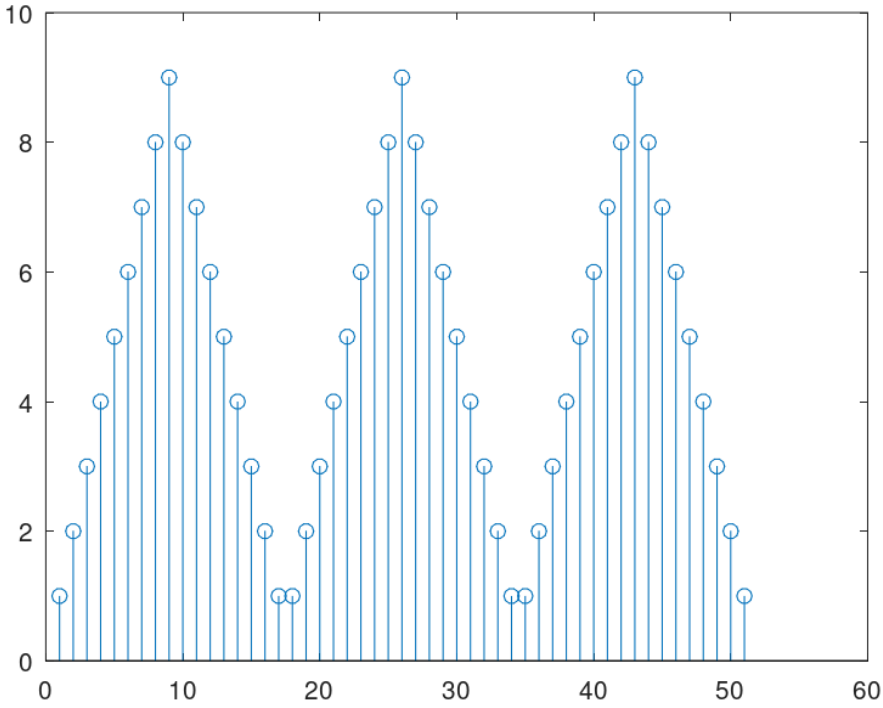
```
N = 46
```

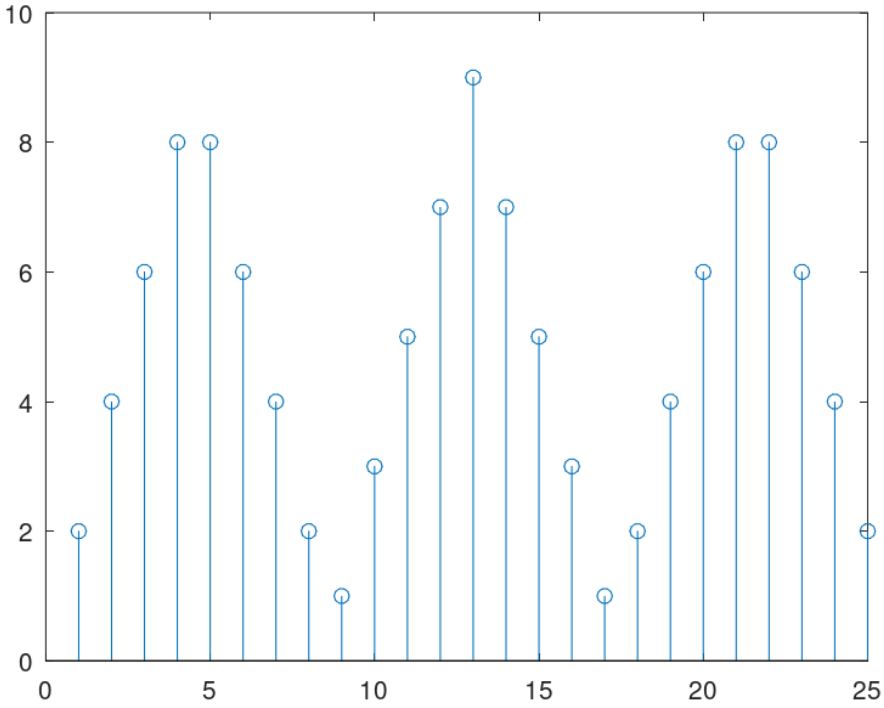


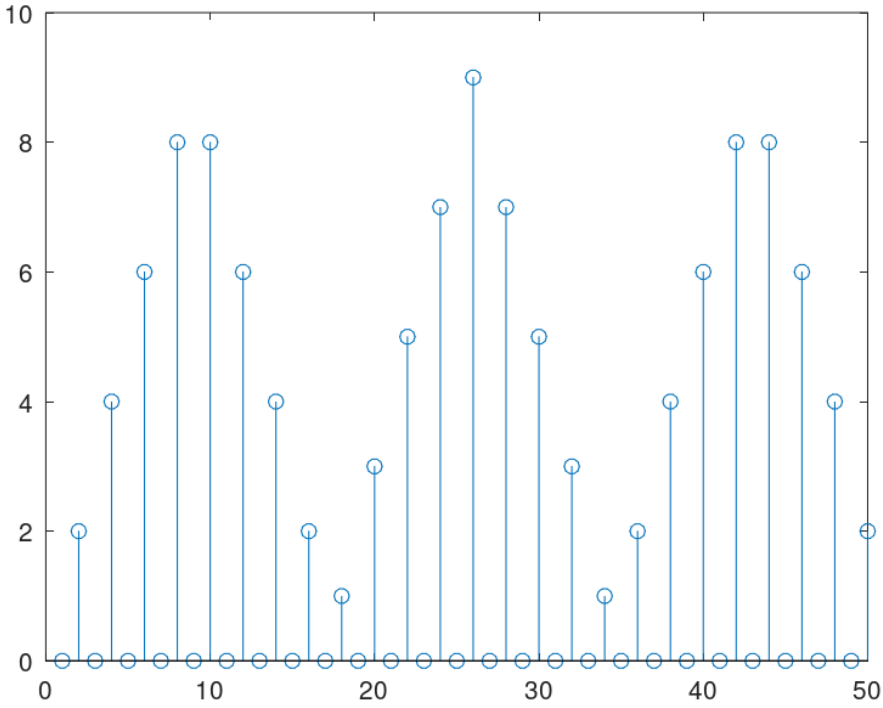
Published with GNU Octave 9.3.0

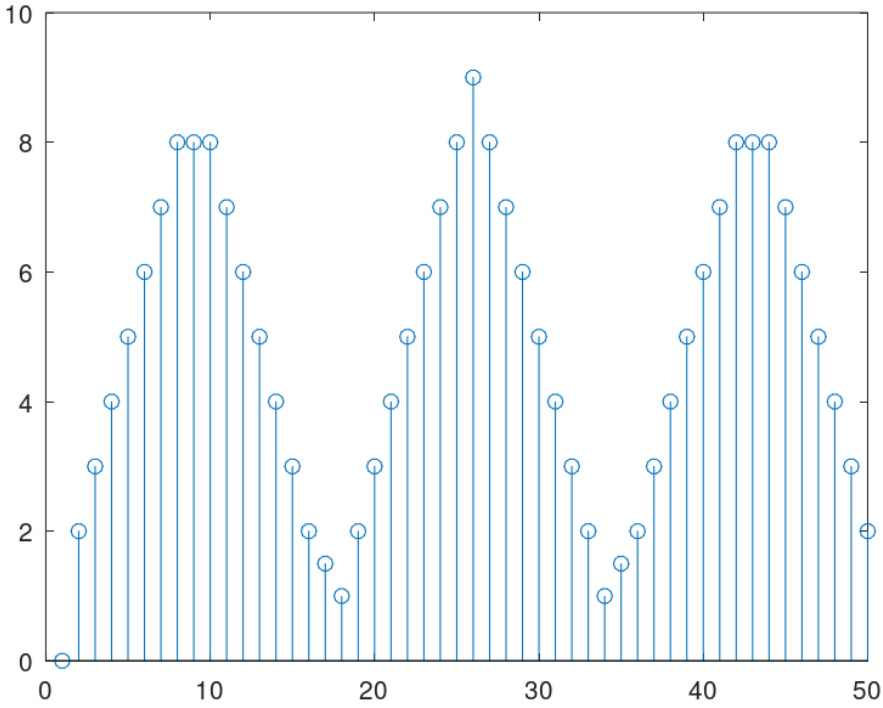
down_up_sample_03

```
clc
clear
x = [ 1 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 1 1 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 1 1 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 1];
N = length(x);
for n = 1:1:(N/2)
  y(n) = x(2*n);
endfor
y;
figure(1), stem(x)
figure(2), stem(y)
N1 = length(y);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for  n = 2:2:(2*N1)
  z(n) = y(n/2);
endfor
z;
N2 = length(z);
figure(3), stem(z)
z1 = z;

for n = 3:2:N2
  z1(n) = (z(n-1) + z(n+1))/2;
endfor
z1;
figure(4), stem(z1)
```


10
8
6
4
2
0
0
10
20
30
40
50
60


10
8
6
4
2
0
0
5
10
15
20
25

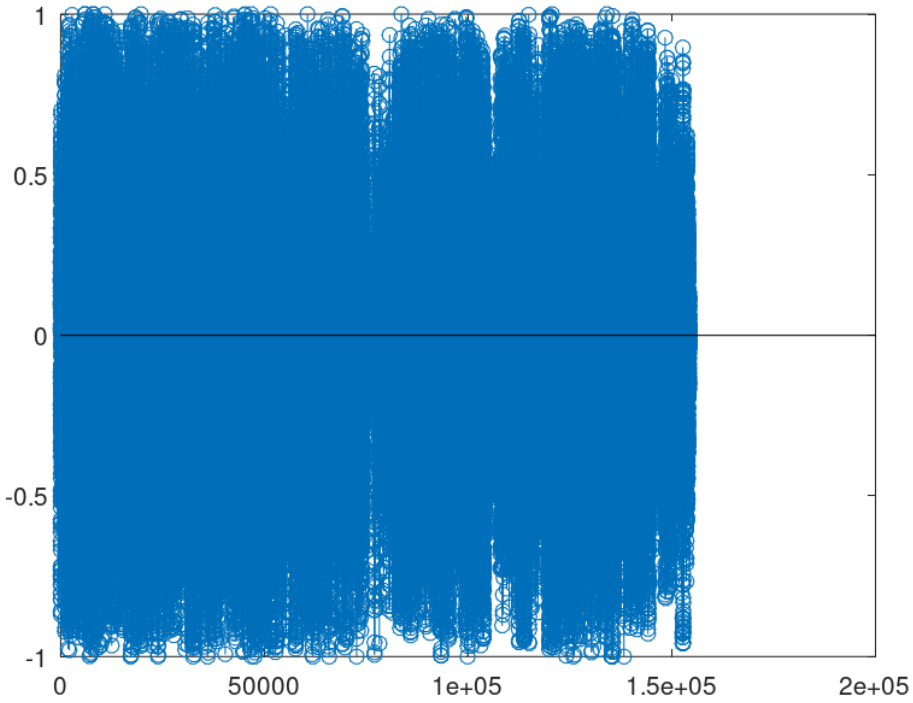


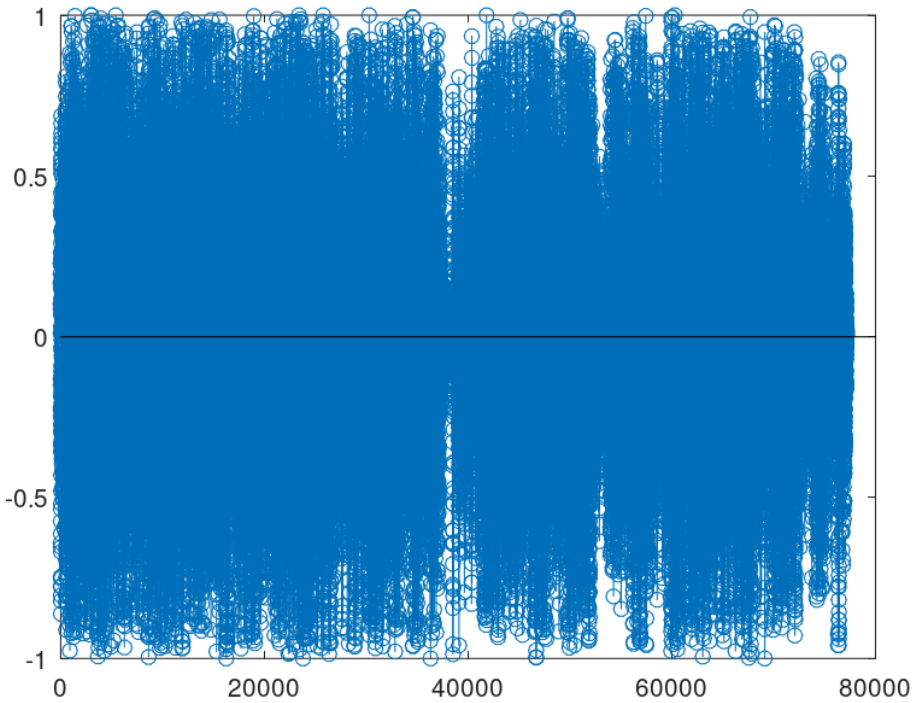


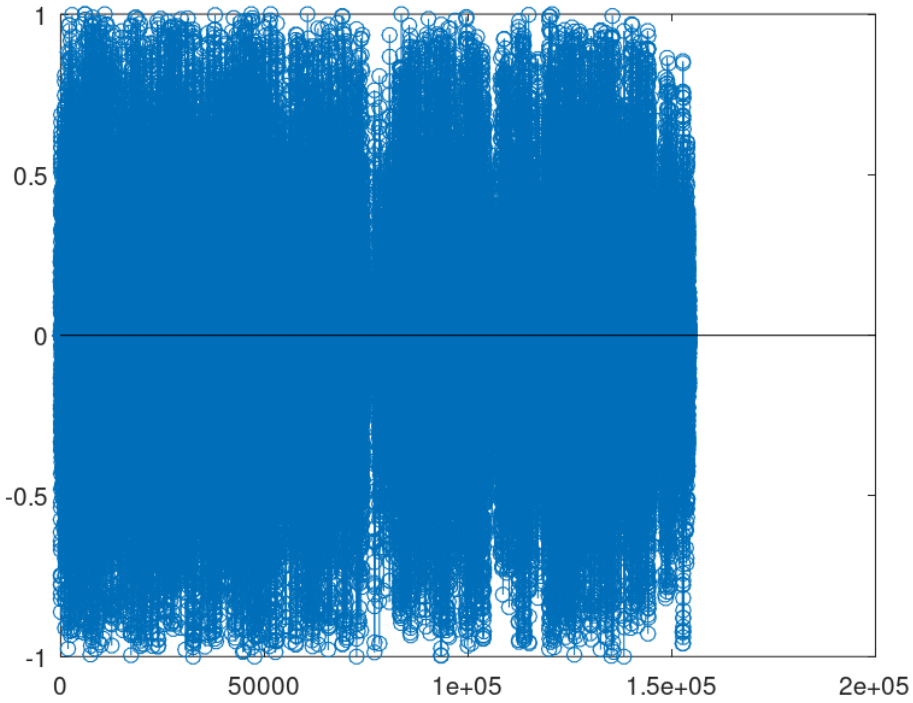
Published with GNU Octave 9.3.0

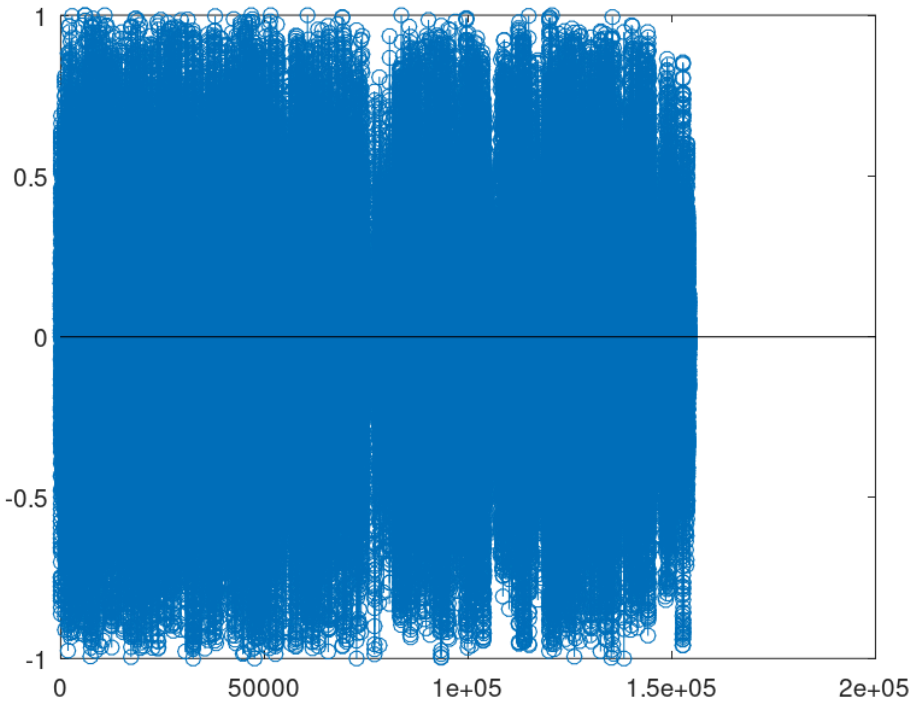
down_sampling_audio_03

```
clc
clear
%x = [ 1 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 1 1 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 1 1 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 1];
[x1, FS] = audioread('Tauba_2.wav');
x = x1(:,1)';
N = length(x);
for n = 1:1:(N/2)
  y(n) = x(2*n);
endfor
y;
figure(1), stem(x)
figure(2), stem(y)
N1 = length(y);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for  n = 2:2:(2*N1)
  z(n) = y(n/2);
endfor
z;
N2 = length(z);
figure(3), stem(z)
z1 = z;

for n = 3:2:N2
  z1(n) = (z(n-1) + z(n+1))/2;
endfor
z1;
figure(4), stem(z1)
```


1
0.5
0
-0.5
-1
0
50000
1e+05
1.5e+05
2e+05


1
0.5
0
-0.5
-1
0
20000
40000
60000
80000





Published with GNU Octave 9.3.0

Circular_Convolution_04

```
clc
clear
x1 = [6 3 2 1 8];
x2 = [4 5 1];
N1 = length(x1);
N2 = length(x2);
%appending zeros
if (N1 < N2)
  x1 = [x1 zeros(1,N2-N1)];
else
  x2 = [x2 zeros(1, N1-N2)];
end
x1
x2
%circular matrix
N2 = length(x2);
c = x2';
b = x2;
for n = 1:1:(N2-1)
  b = [b(N2) b(1:(N2-1))];
  %pause()
  c = [c  b'];
end
c
%circular convolution
y = c * x1'
```

```
x1 =

   6   3   2   1   8

x2 =

   4   5   1   0   0

c =

   4   0   0   1   5
   5   4   0   0   1
   1   5   4   0   0
   0   1   5   4   0
   0   0   1   5   4

y =

   65
   50
   29
   17
   39
```

Published with GNU Octave 9.3.0

Linear_Convolution_04

```
clc
clear
x1 = [21 42 41];
x2 = [1 0 4 -2 8];
N1 = length(x1);
N2 = length(x2);
N3 = N1+N2-1
%appending zeros
x1 = [x1 zeros(1 , N3-N1)];
x2 = [x2 zeros(1 , N3-N2)];
x1
x2
%circular matrix
c = x2';
b = x2;
for n = 1:1:(N3-1)
  b = [b(N3) b(1:(N3-1))];
  %pause()
  c = [c  b'];
end
c
%linear convolution
y = c * x1'
```

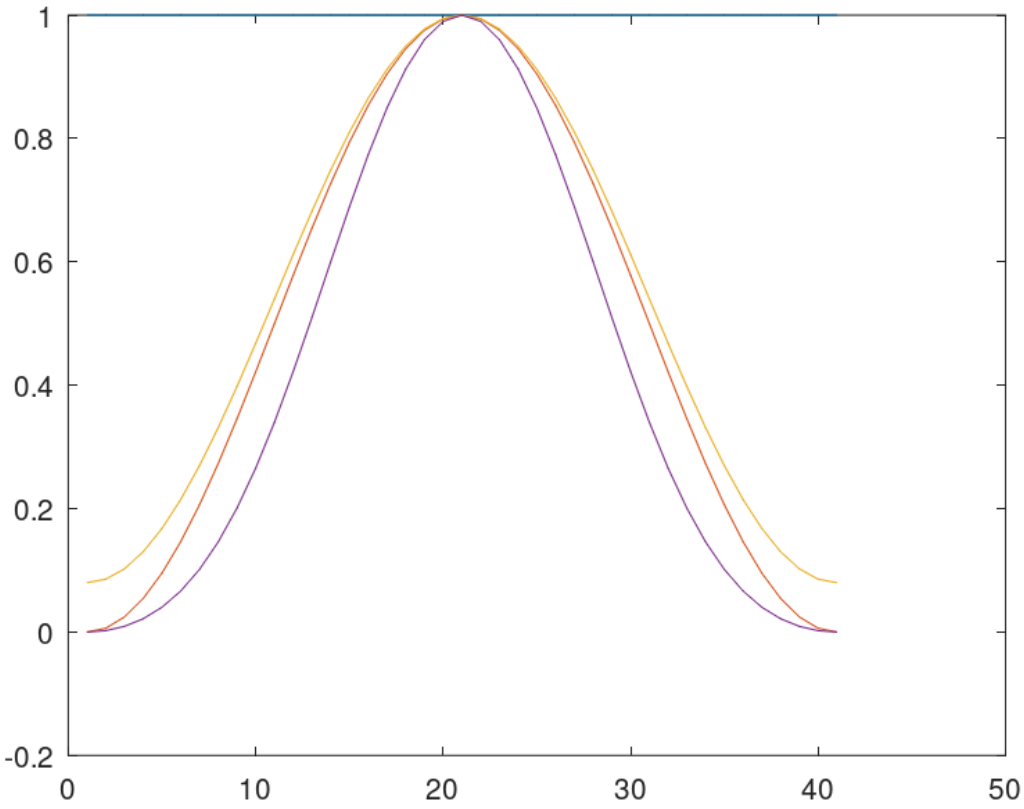
```
N3 = 7
x1 =

   21   42   41    0    0    0    0

x2 =

   1   0   4  -2   8   0   0

c =

   1   0   0   8  -2   4   0
   0   1   0   0   8  -2   4
   4   0   1   0   0   8  -2
  -2   4   0   1   0   0   8
   8  -2   4   0   1   0   0
   0   8  -2   4   0   1   0
   0   0   8  -2   4   0   1

y =

    21
    42
   125
   126
   248
   254
   328
```

Published with GNU Octave 9.3.0

Windows_1

```
clc
clear
M = 41
for n = 0 : 1 : M-1
  wr(n+1) = 1;
  wh(n+1) = 0.5 - 0.5*cos((2*pi*n) / (M-1));
  whm(n+1) = 0.54 - 0.46*cos((2*pi*n) / (M-1));
  bm(n+1) = 0.42 - 0.5*cos((2*pi*n) / (M-1)) + 0.08*cos((4*pi*n) / (M-1));
end
plot(wr)
hold on
plot(wh)
hold on
plot(whm)
hold on
plot(bm)
```

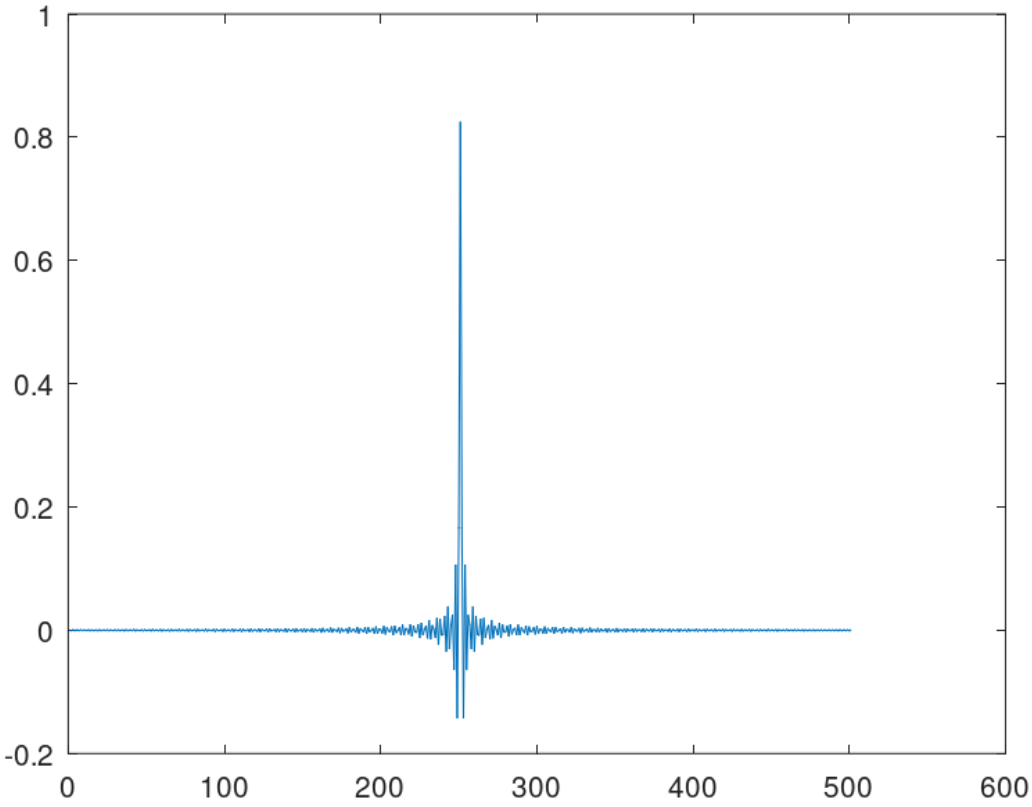
```
M = 41
```

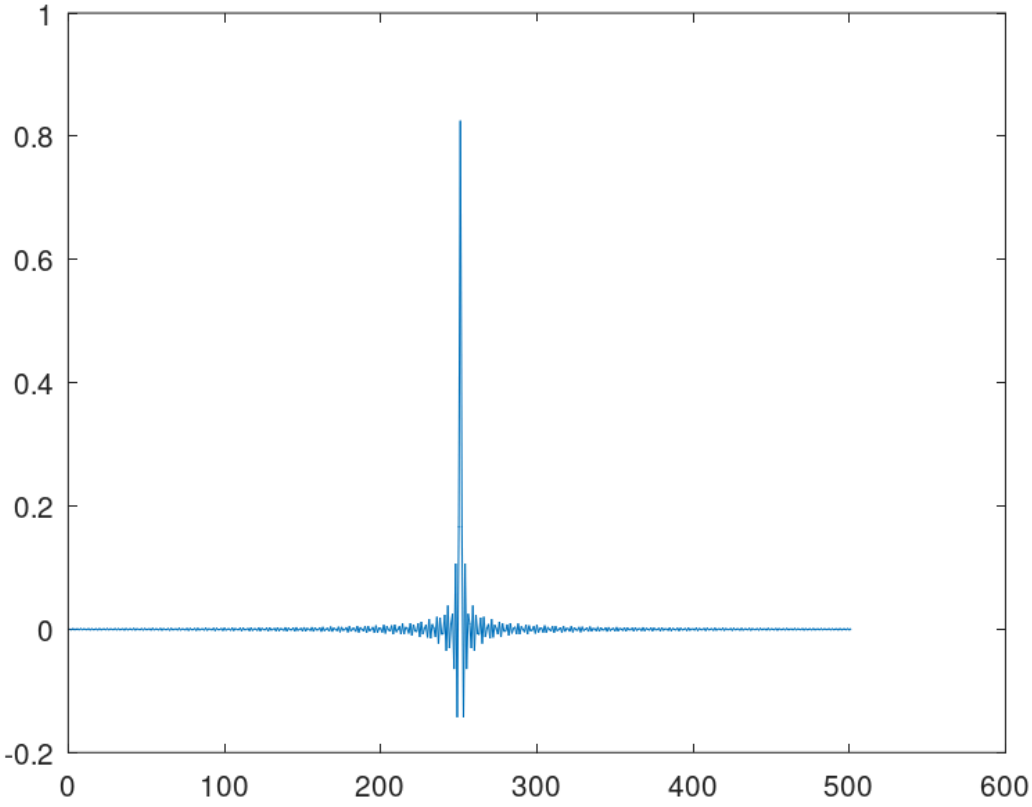


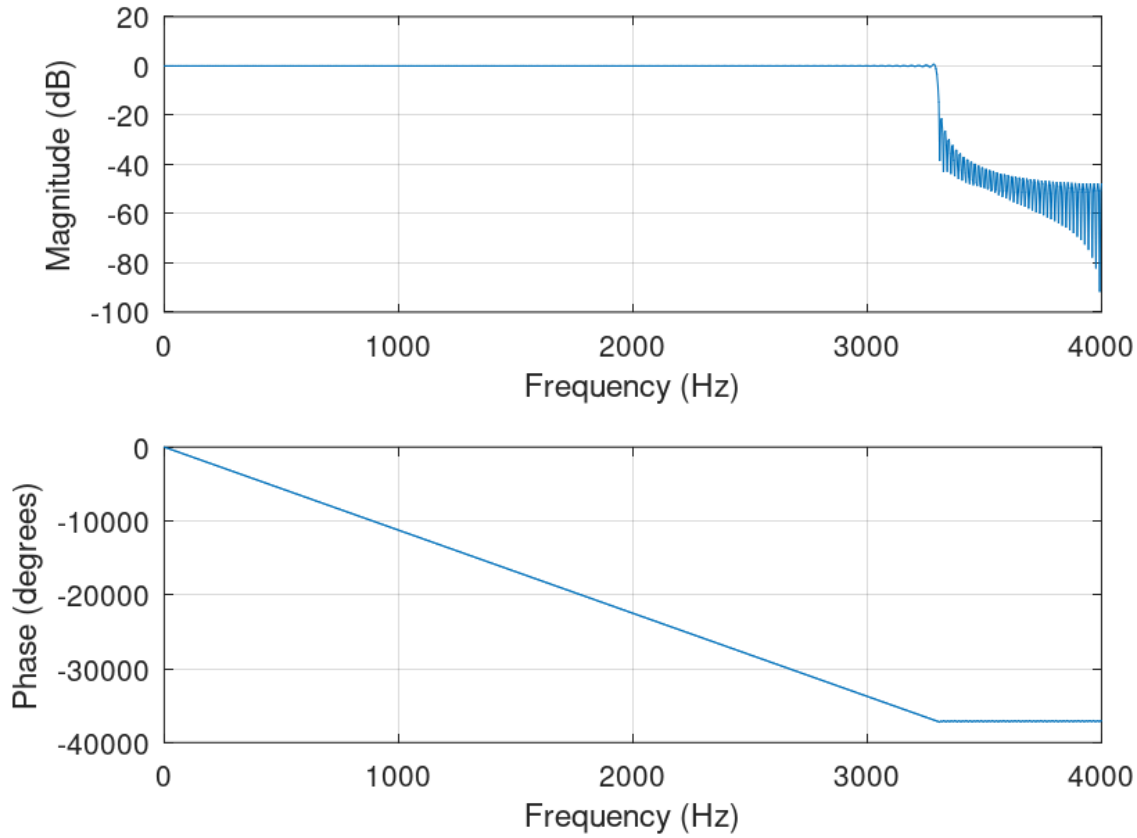
Published with GNU Octave 9.3.0

Filterimplementation

```
clc
clear

#Impulse Response

M = 501;
shift = (M-1) / 2;

for n = 0 : 1 : M-1
  hds(n+1) = sin(0.825*pi*(n-shift))/(pi*(n-shift));
end
hds(shift+1) = 0.825;
figure(1)
plot(hds)

#Rectangular Window

for n = 0 : 1 : M-1
  wr(n+1) = 1;
end

#Multiply Impulse Response with window

h = hds .* wr;
figure(2)
plot(h)

#Checking the filter

figure(3)
freqz(h,1,M,8000)

#Implementing
```


1
0.8
0.6
0.4
0.2
0
-0.2
0
100
200
300
400
500
600

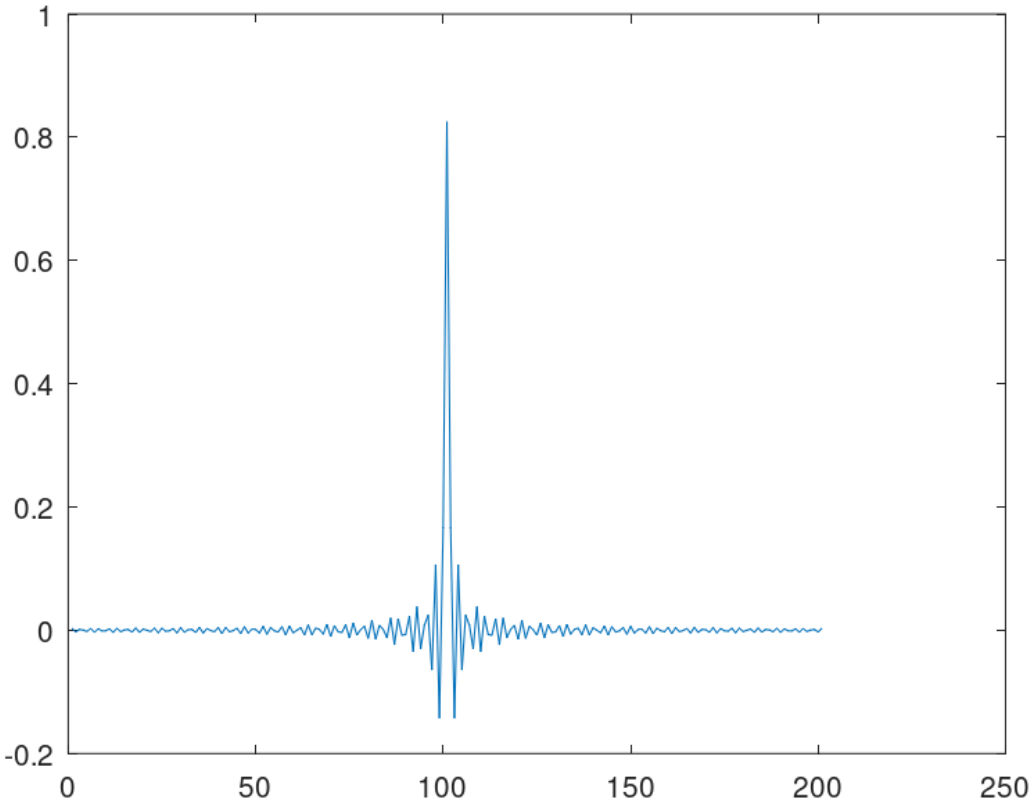

1
0.8
0.6
0.4
0.2
0
-0.2
0
100
200
300
400
500
600

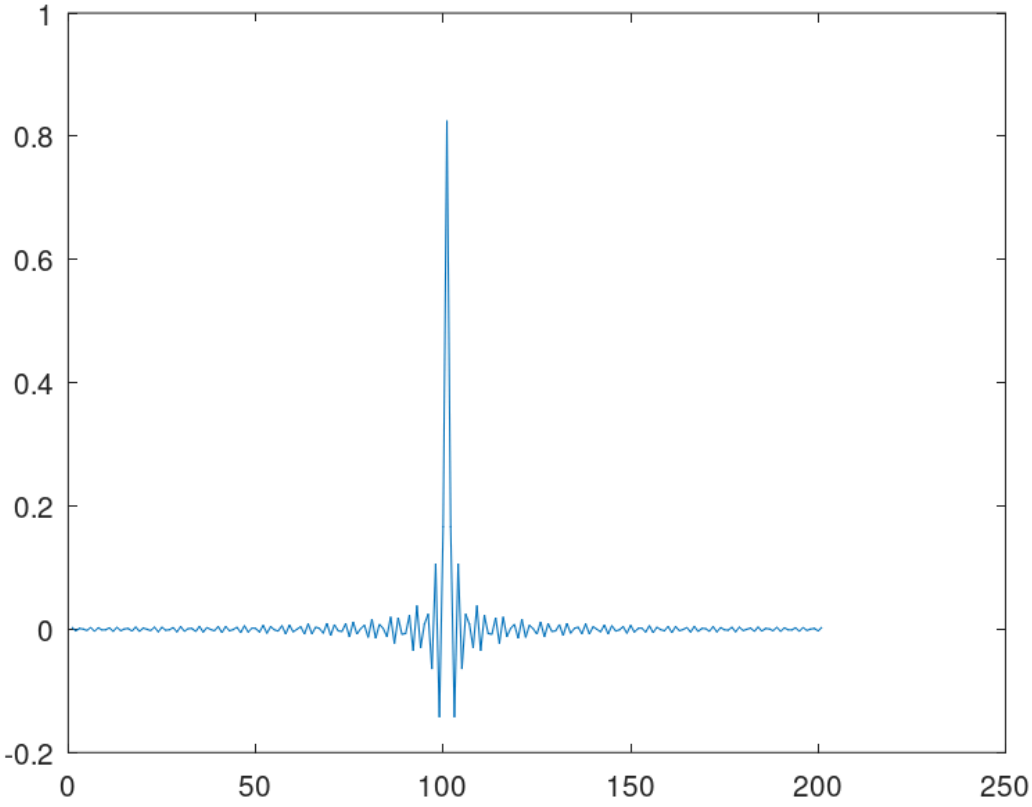


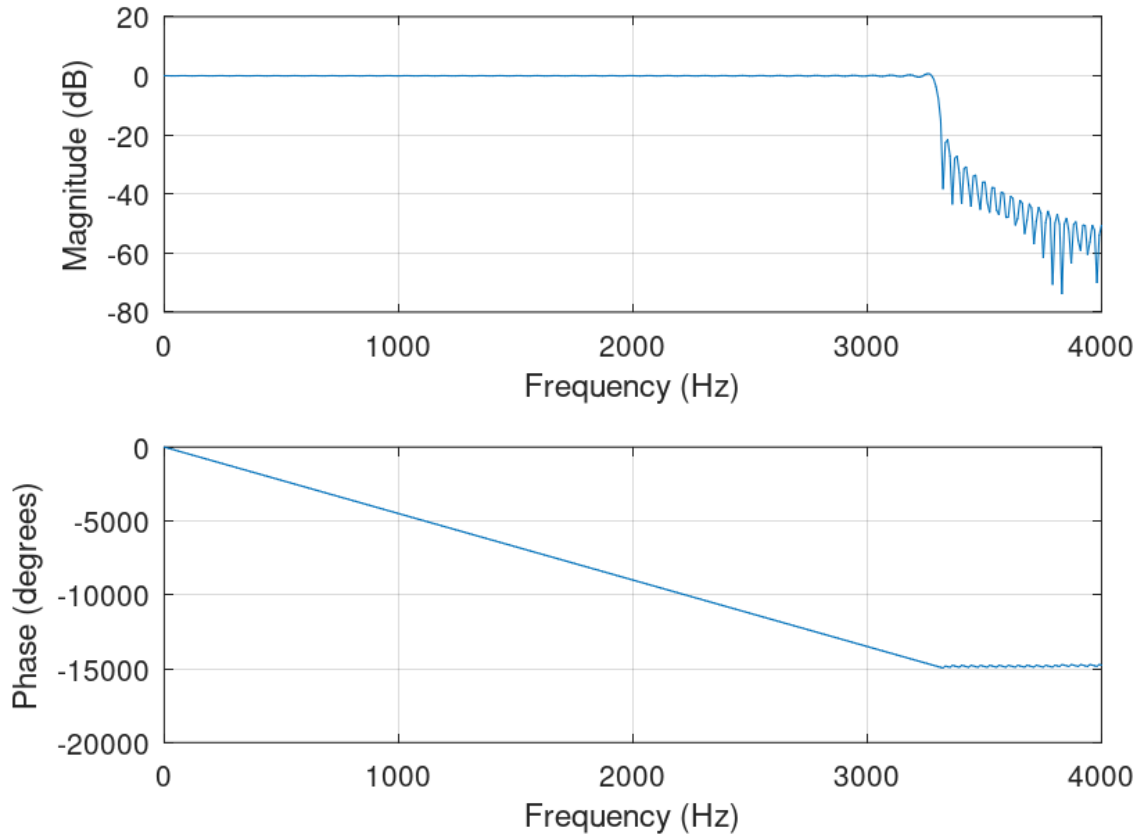
Published with GNU Octave 9.3.0

FIR_LPF_2

```
clc
clear

#Impulse Response

M = 201;
shift = (M-1) / 2;

for n = 0 : 1 : M-1
  hds(n+1) = sin(0.825*pi*(n-shift))/(pi*(n-shift));
end
hds(shift+1) = 0.825;
figure(1)
plot(hds)

#Rectangular Window

for n = 0 : 1 : M-1
  wr(n+1) = 1;
end

#Multiply Impulse Response with window

h = hds .* wr;
figure(2)
plot(h)

#Checking the filter

figure(3)
freqz(h,1,M,8000)

#Implementing
```


1
0.8
0.6
0.4
0.2
0
-0.2
0
50
100
150
200
250

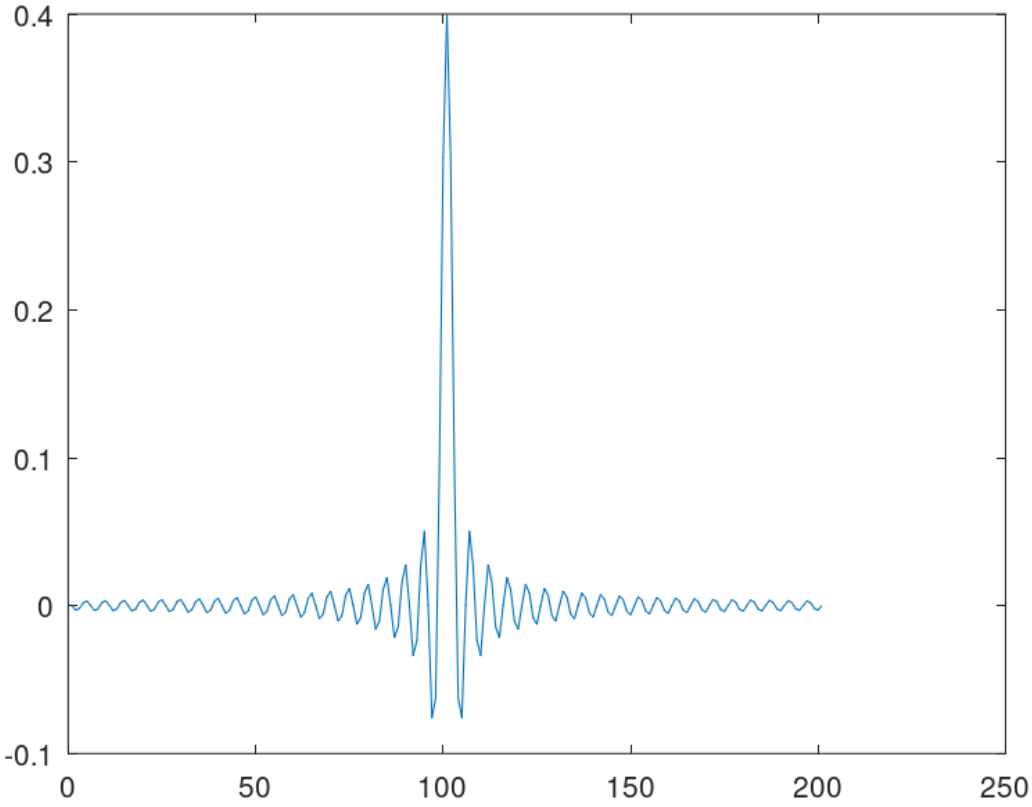

1
0.8
0.6
0.4
0.2
0
-0.2
0
50
100
150
200
250

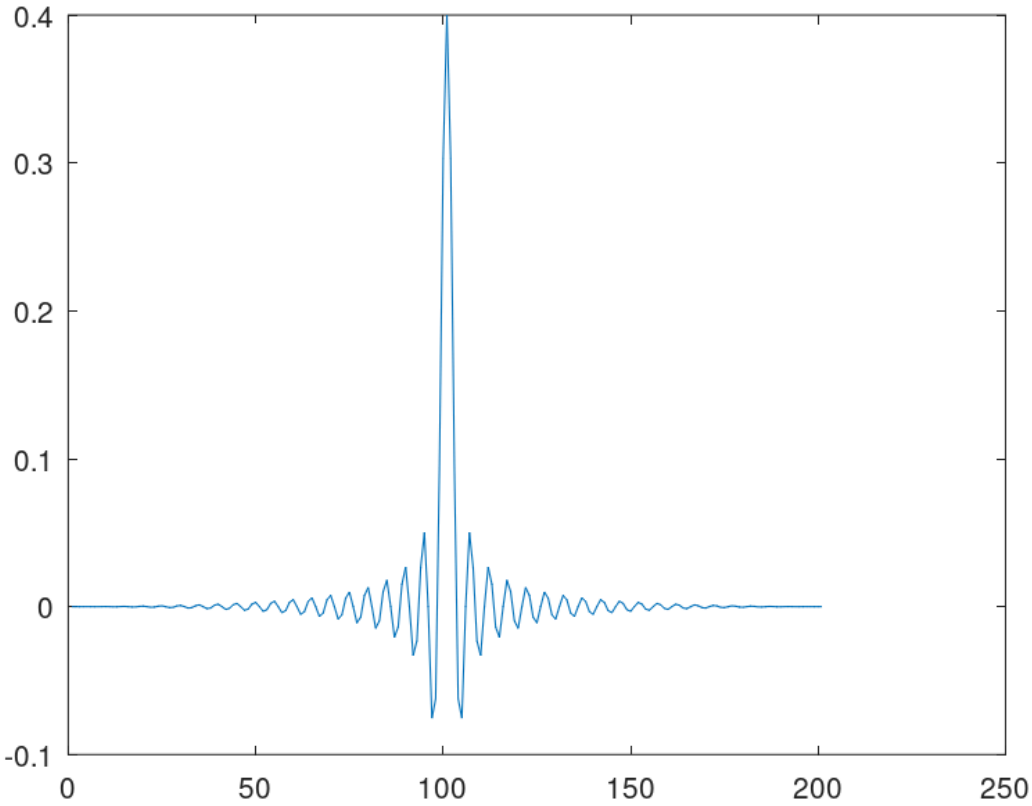


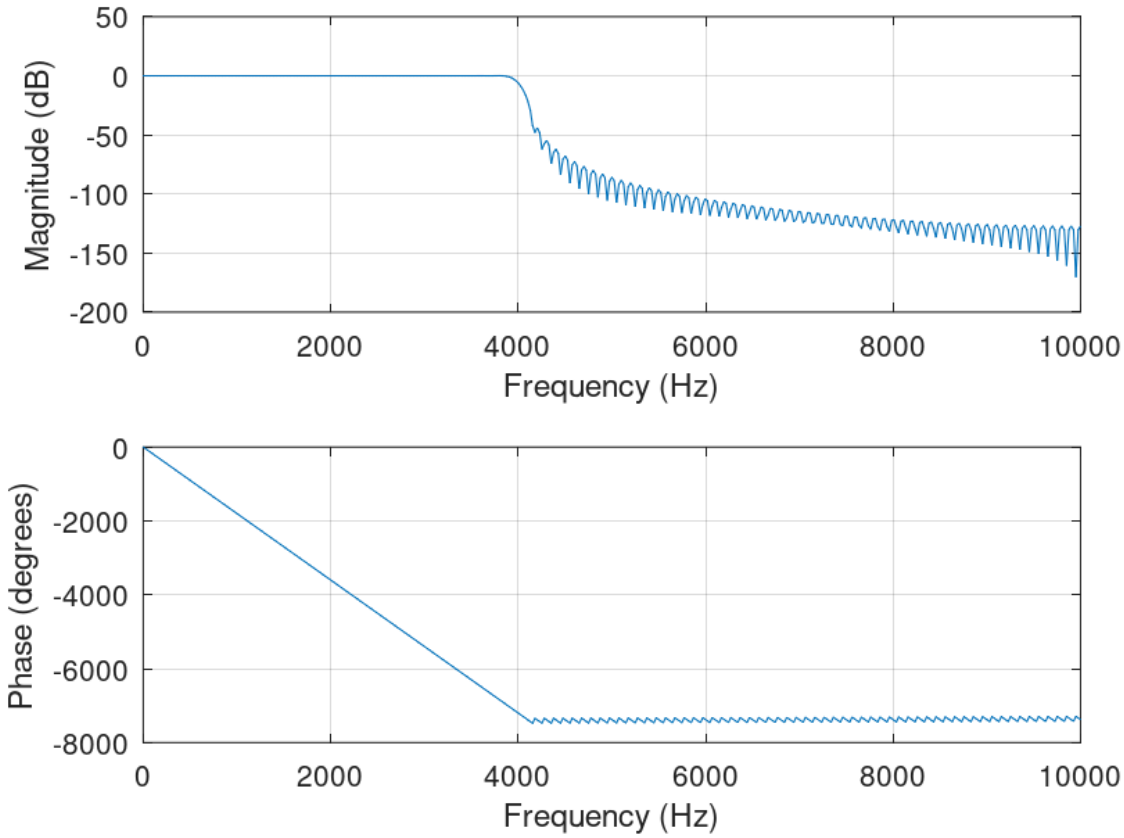
Published with GNU Octave 9.3.0

FIR_LPF_3

```
clc
clear

#Impulse Response

M = 201;
shift = (M-1) / 2;

for n = 0 : 1 : M-1
  hds(n+1) = sin(0.4*pi*(n-shift))/(pi*(n-shift));
end
hds(shift+1) = 0.4;
figure(1)
plot(hds)

#Hanning Window

for n = 0 : 1 : M-1
  wh(n+1) = 0.5 - 0.5*cos((2*pi*n) / (M-1));
end

#Multiply Impulse Response with window

h = hds .* wh;
figure(2)
plot(h)

#Checking the filter

figure(3)
freqz(h,1,M,20000)

#Implementing
```


0.4
0.3
0.2
0.1
0
-0.1
0
50
100
150
200
250

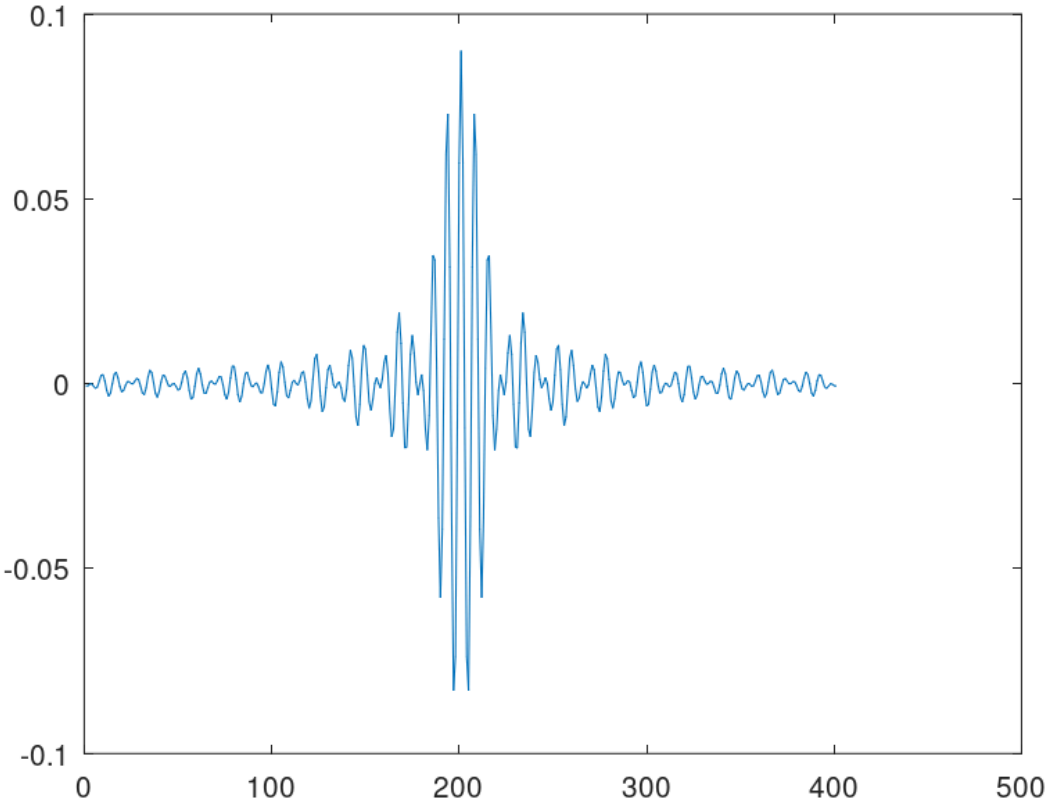

0.4
0.3
0.2
0.1
0
-0.1
0
50
100
150
200
250

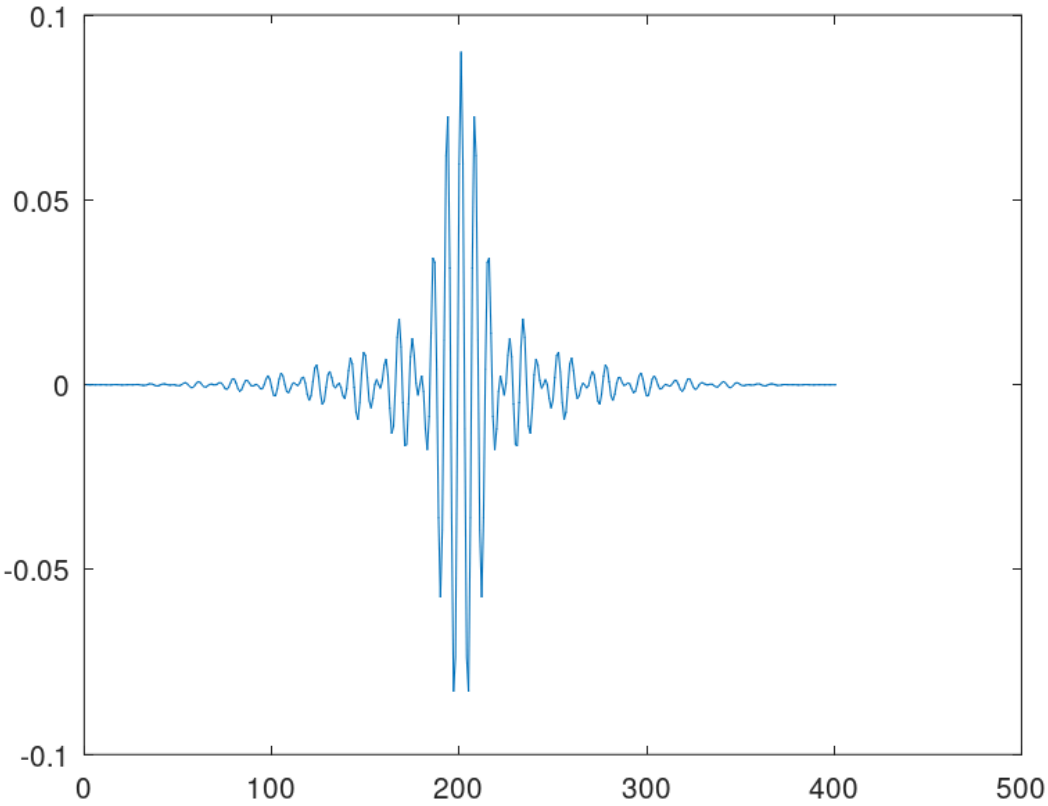


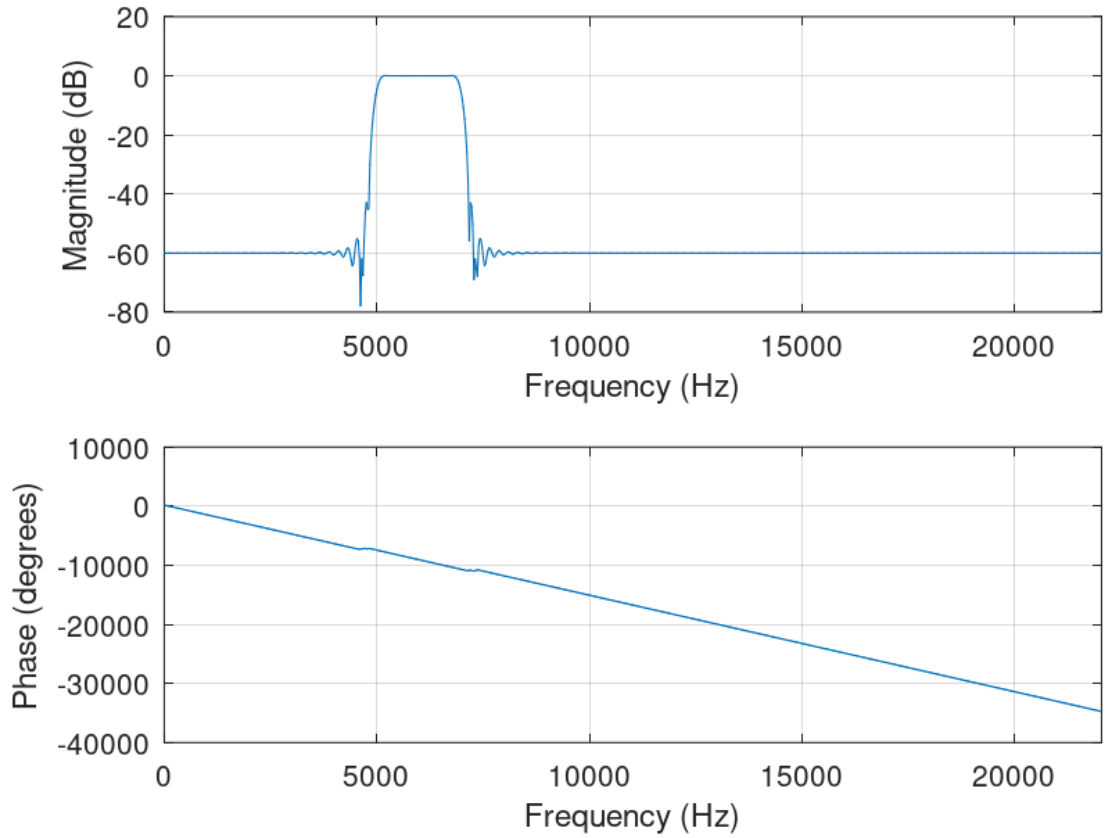
Published with GNU Octave 9.3.0

FIR_BP

```
clc
clear

#Impulse Response

M = 401;
shift = (M-1) / 2;

for n = 0 : 1 : M-1
  hds(n+1) = sin(0.317*pi*(n-shift))/(pi*(n-shift)) - sin(0.226*pi*(n-shift))/(pi*(n-shift));
end
hds(shift+1) = 0.09;
figure(1)
plot(hds)

#Hanning Window

for n = 0 : 1 : M-1
  wh(n+1) = 0.5 - 0.5*cos((2*pi*n) / (M-1));
end

#Multiply Impulse Response with window

h = hds .* wh;
figure(2)
plot(h)

#Checking the filter

figure(3)
freqz(h,1,M,44100)

#Implementing
```


0.1
0.05
0
-0.05
-0.1
0
100
200
300
400
500

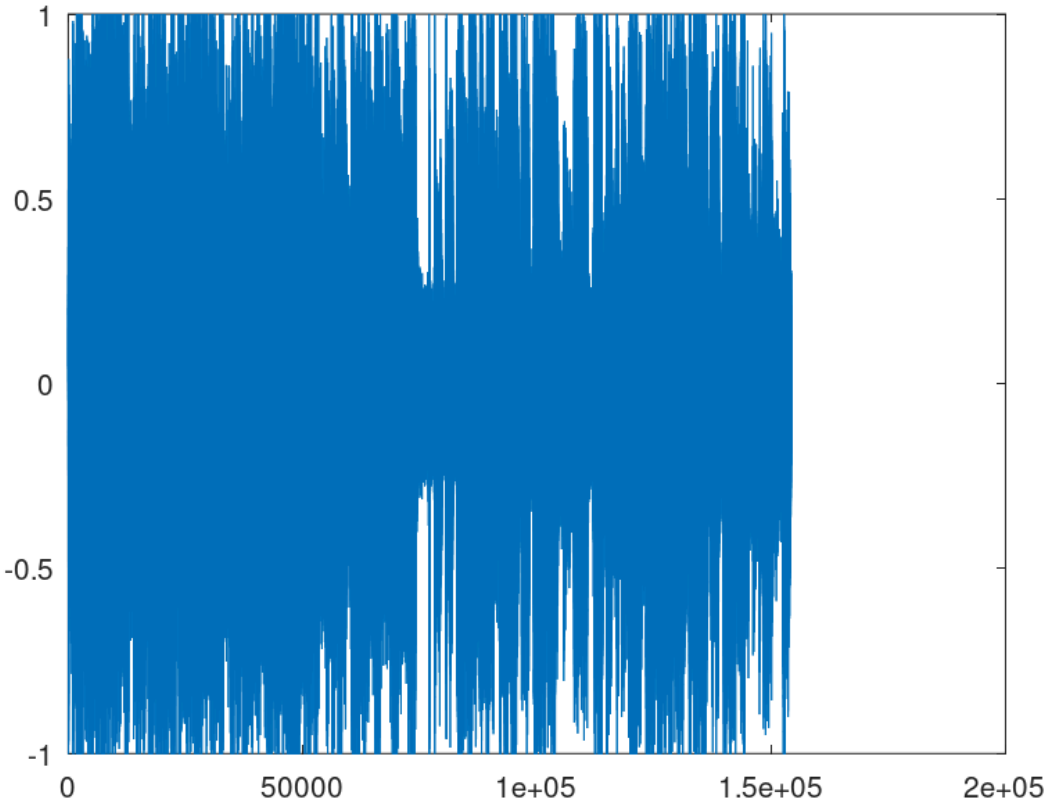

0.1
0.05
0
-0.05
-0.1
0
100
200
300
400
500

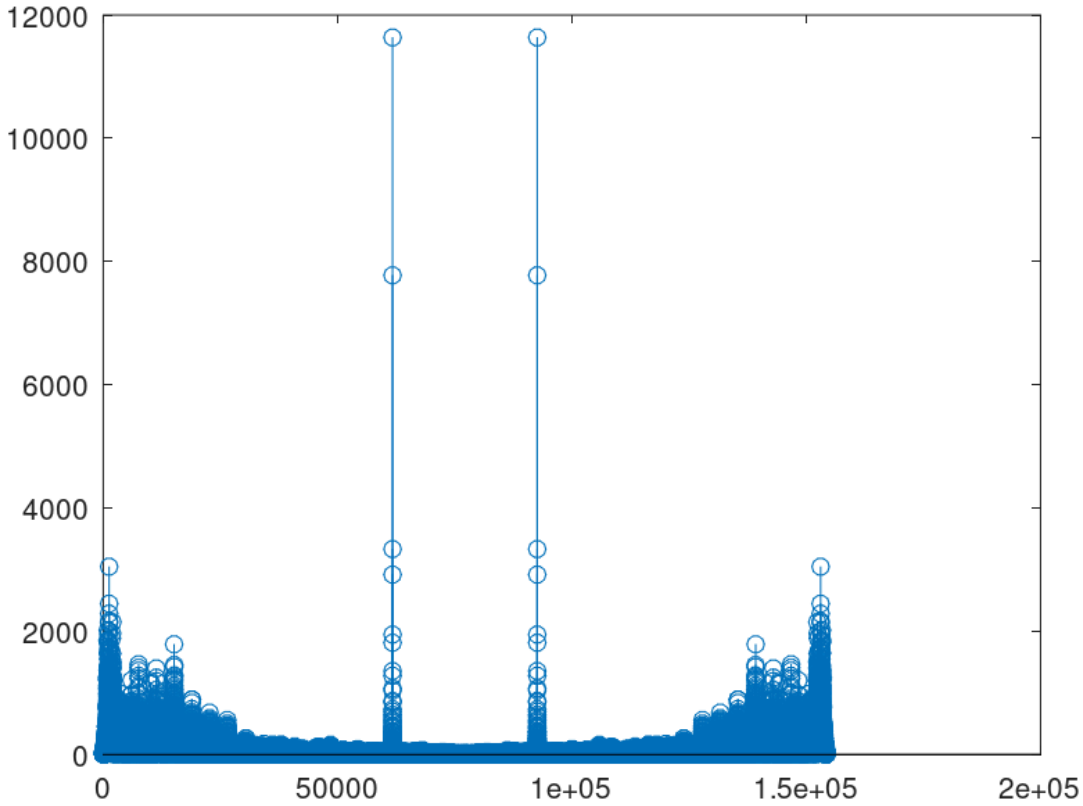


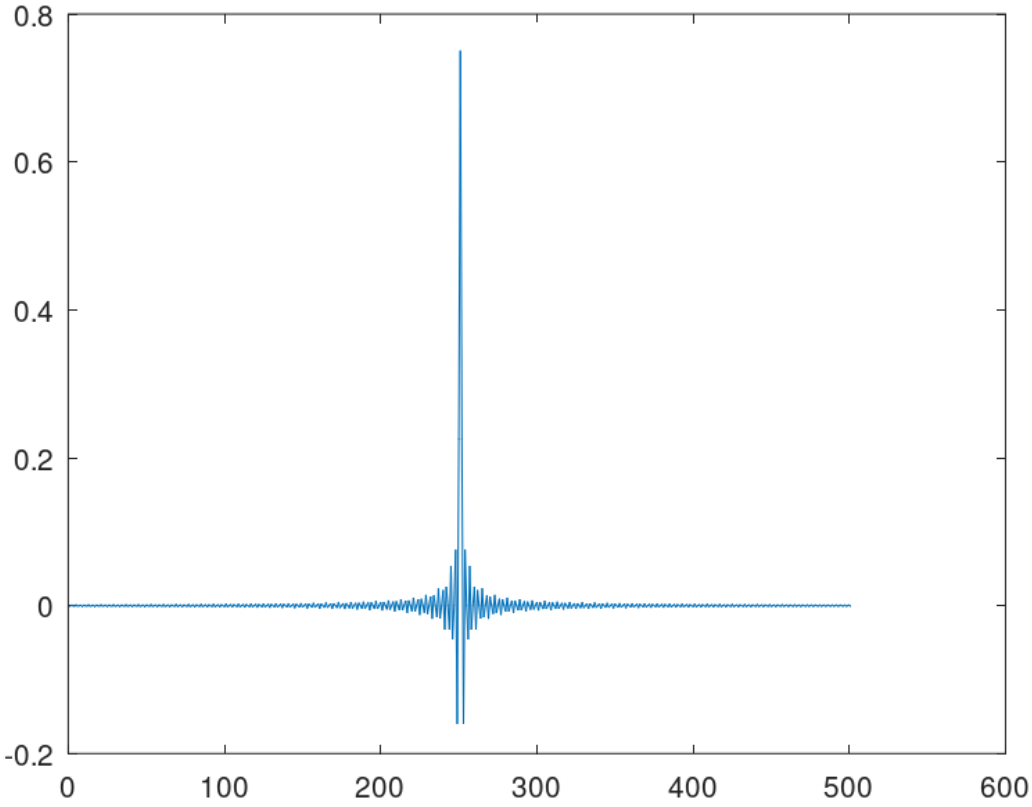
Published with GNU Octave 9.3.0

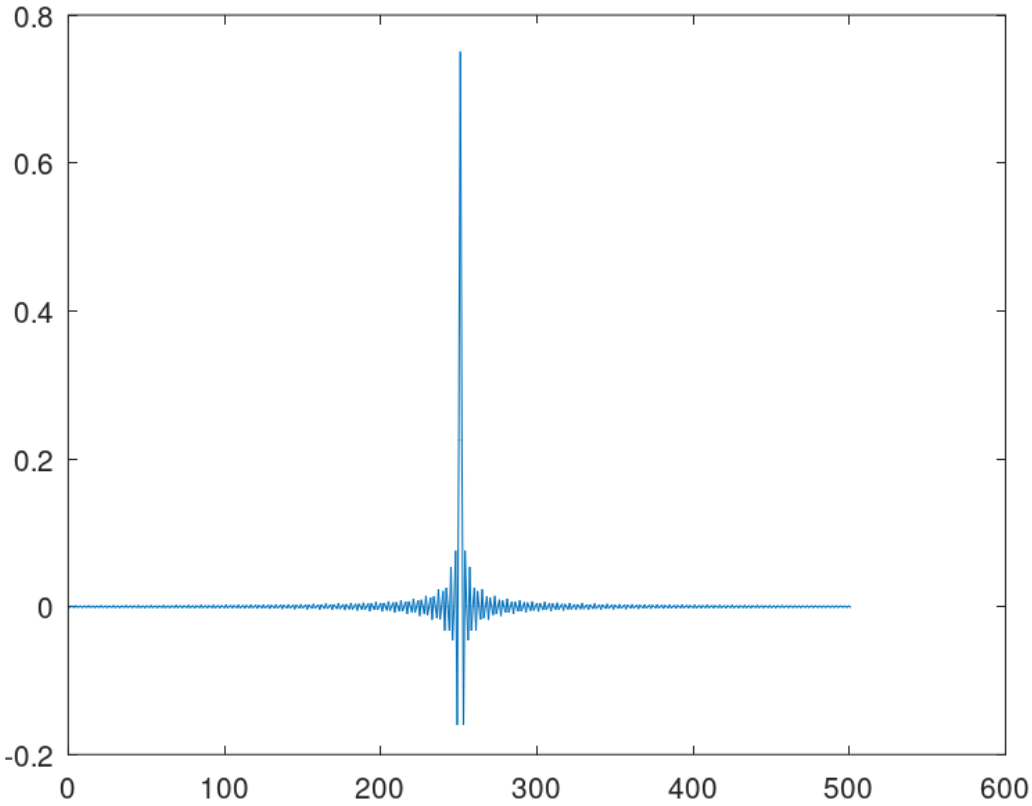
noise_removal

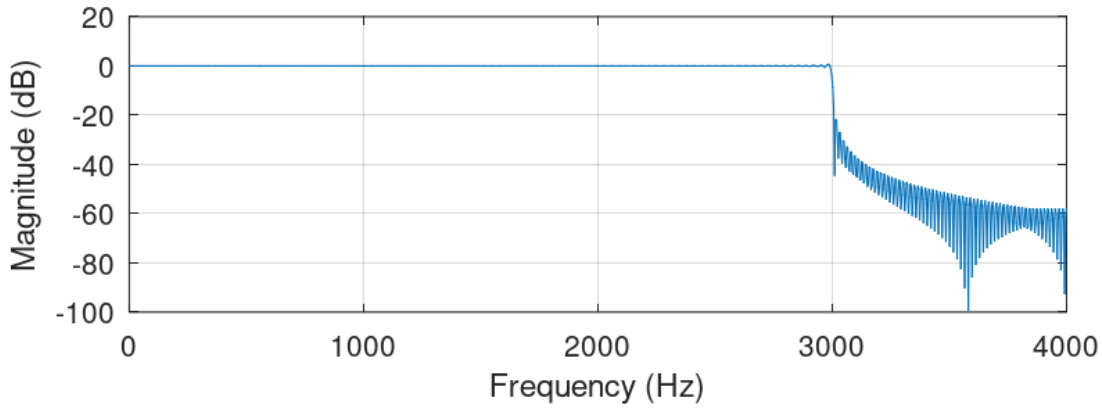
```
clc
clear
[x1 , Fs] = audioread('Tauba_noise1.wav');
x = x1(:,1)';
figure(1) , plot(x)
##################
X = abs(fft(x)); #Magnitude Spectrum
figure(2) , stem(X)
##################
#Impulse Response

M = 501;
shift = (M-1) / 2;

for n = 0 : 1 : M-1
  hds(n+1) = sin(0.75*pi*(n-shift))/(pi*(n-shift));
end
hds(shift+1) = 0.75;
figure(3)
plot(hds)

#Rectangular Window

for n = 0 : 1 : M-1
  wr(n+1) = 1;
end

#Multiply Impulse Response with window

h = hds .* wr;
figure(4)
plot(h)

#Checking the filter

figure(5)
freqz(h,1,M,Fs)

#Implementing the filter
y = conv(x,h);
# DFT to check filtering
Y = abs(fft(y));
figure(6)
stem(Y)

audiowrite('Noise_removed.wav' , y , Fs)
```

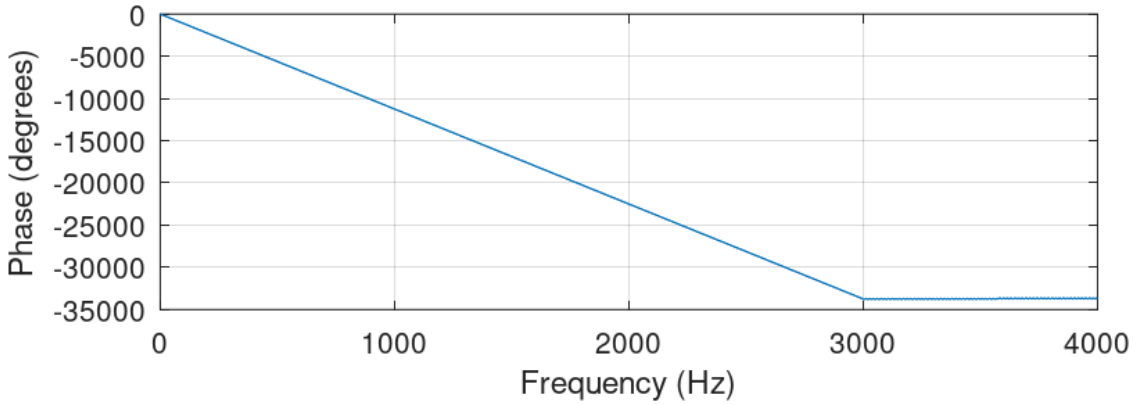

1
0.5
0
-0.5
-1
0
50000
1e+05
1.5e+05
2e+05

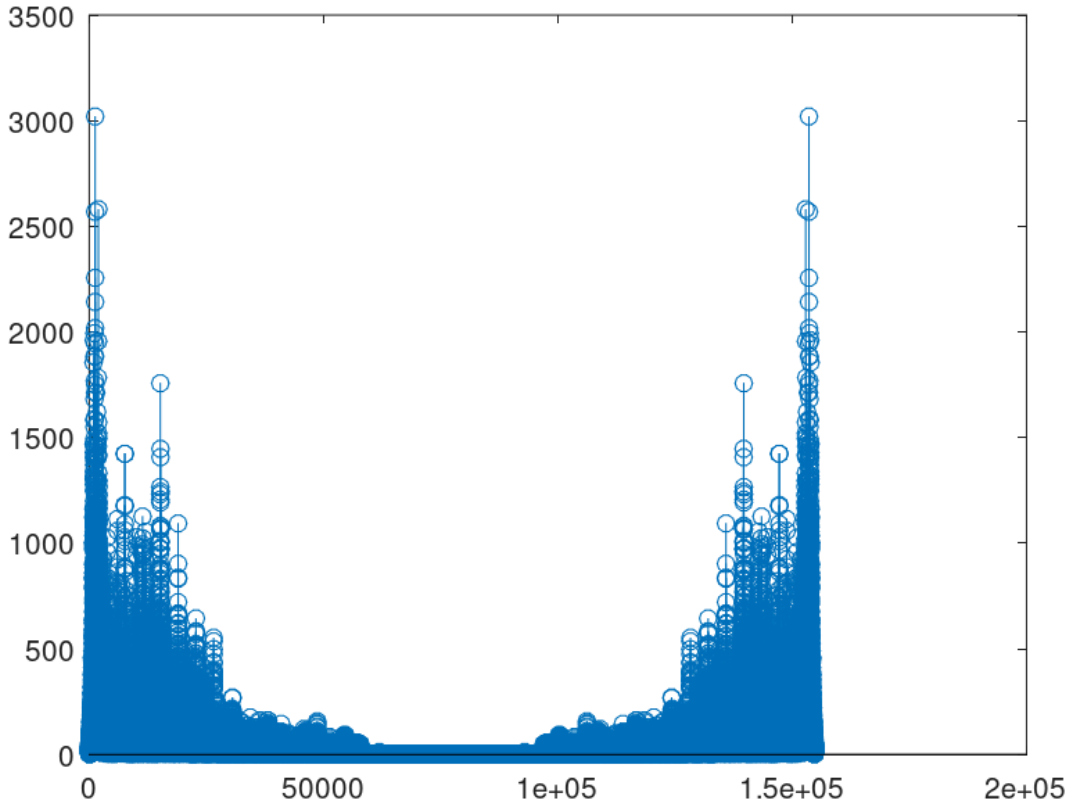

12000
10000
8000
6000
4000
2000
0
0
50000
1e+05
1.5e+05
2e+05


0.8
0.6
0.4
0.2
0
-0.2
0
100
200
300
400
500
600


0.8
0.6
0.4
0.2
0
-0.2
0
100
200
300
400
500
600







Published with GNU Octave 9.3.0