Digital Signal Processing Lab

TECHNISCHE UNIVERSITÄT DARMSTADT

Lab 2: Discrete-Time Signals and Systems
Group: 04
Report by Rong ZHI and Yunlong SONG

SIGNAL PROCESSING GROUP

1. Preparation

a). What is MATLAB

MATLAB stands for both a high-performance language for technical computer and integrated numerical computing environment. Typically, we can use MATLAB to do math and compution, algorithm development, data analysis and engineering graphics, etc.

b). Data Type

The three special cases are: 1> Scalars: A scalar is an element of a field which is used to define a vector space, which can be consider as a 1x1 matrix. 2> Vectors: A vectors can be considered as an 1xn matrix, which has one column and n rows. 3> Matrix:a matrix is a rectangular array of numbers, symbols, or expressions, arranged in rows and columns.

c). for loop and if-else statement

```
v =

Columns 1 through 13

-1.0000    -0.9000    -0.8000    -0.7000    -0.6000    -0.5000    -0.4000    -0.3000    -0.2000    -0.1

Columns 14 through 21

0.3000     0.4000     0.5000     0.6000     0.7000     0.8000     0.9000     1.0000

ans =

-1    -1    -1    -1    -1    -1    -1    -1    -1    -1     0     1     1     1     1     1

s =

-1    -1    -1    -1    -1    -1    -1    -1    -1    -1     0     1     1     1     1     1
```

Code:

```
%%
clear
clear all
v = -1:0.1:1;
sign(v);
v(1);
m = 0;
for l = -1:0.1:1
    m = m + 1;
    if v(m) > 0
        s(m) = 1;
% % % % always be careful about the "=" and "=="
    elseif v(m) == 0
        s(m) = 0;
    elseif v(m) < 0
        s(m) = -1;
    end
end
v
sign(v)
s
```

d). Matix

```
G =

0.6000    1.5000    2.3000   -0.5000
8.2000    0.5000   -0.1000   -2.0000
5.7000    8.2000    9.0000    1.5000
0.5000    0.5000    2.4000    0.5000
1.2000   -2.3000   -4.5000    0.5000



ans =

5     4

G(2 2) = 0.5
G(4 1) = 0.5
G(4 2) = 0.5
G(4 4) = 0.5
G(5 4) = 0.5
G(1 4) < 0
G(2 3) < 0
G(2 4) < 0
G(5 2) < 0
G(5 3) < 0
```

Code:

```
%%

% % % % % % % % Preparation d)

G =...
[0.6, 1.5, 2.3, -0.5;...
8.2, 0.5, -0.1, -2.0;...
5.7, 8.2, 9.0, 1.5;...
0.5, 0.5, 2.4, 0.5;...
1.2, -2.3, -4.5, 0.5]

size(G)

for m = 1:5
    for n = 1:4
        if G(m, n) == 0.5
            fprintf('G(%d %d) = 0.5\n', m, n);
        end
    end
end



for m = 1:5
    for n = 1:4
        if G(m, n) < 0
            fprintf('G(%d %d) < 0 \n', m, n);
        end
    end
end
```

f).Functions and Scripts

Program files can be scripts that simply execute a series of MATLAB® statements, or they can be functions that also accept input arguments and produce output. Both scripts and functions contain MATLAB code, and both are stored in text files with a .m extension. However, functions are more flexible and more easily extensible.

2. Experiments & Results

Problems 2.1 Magic Matrics

M is a magic function, which returns a 5-by-5 matrix constructed from the integers 1 through 25 with equal row and column sums.

```
>> M = magic(5);

M =

17    24     1     8    15
23     5     7    14    16
```

```
4      6      13     20     22
10     12     19     21      3
11     18     25      2      9

>>sum(M)

ans =

65     65     65     65     65

>>sum(M?)

ans =

65     65     65     65     65

The first row is:

R1 =

17     24      1      8     15

The third column is:

ans =

1
7
13
19
25

Column 1 to 3 of rows2 to the end of M:

M_sub =

23      5      7
4      6     13
10     12     19
11     18     25

M(1, 1) = 17 > 10
M(1, 2) = 24 > 10
M(1, 5) = 15 > 10
M(2, 1) = 23 > 10
M(2, 4) = 14 > 10
```

```
M(2, 5) = 16 > 10
M(3, 3) = 13 > 10
M(3, 4) = 20 > 10
M(3, 5) = 22 > 10
M(4, 2) = 12 > 10
M(4, 3) = 19 > 10
M(4, 4) = 21 > 10
M(5, 1) = 11 > 10
M(5, 2) = 18 > 10
M(5, 3) = 25 > 10
M(1, 3) = 1 < 4
M(4, 5) = 3 < 4
M(5, 4) = 2 < 4
```

```
%%
clear
clear all
M = magic(5)
sum(M)
sum(M')
 for row = 1:5
     for column = 1:5
         if row == 1
             R1(column) = M(row,column);
         end

         if column == 3
             C3(row) = M(row, column);
         end

         if (row >= 2) && (column <=3)
             M_sub(row-1,column) = M(row,column);
         end

     end
 end

 fprintf('The first row is: ');
 R1
 fprintf('The third column is: ');
 C3'
 fprintf('Column 1 to 3 of rows2 to the end of M: ');
 M_sub
for row = 1:5
    for column = 1:5
        if M(row,column) > 10
            fprintf('M(%d, %d) = %d > 10 \n',row,column, M(row,column))
        end
    end
end

for row = 1:5
    for column = 1:5
        if M(row,column) < 4
            fprintf('M(%d, %d) = %d < 4 \n',row,column, M(row,column))
        end
    end
end
```

2.2 Magic Matrices

```
The first 12 Fiboncacci numbers:

1    1    2    3    5    8    13    21    34    55    89    144    233
```

The approximation compared with golden ratio:

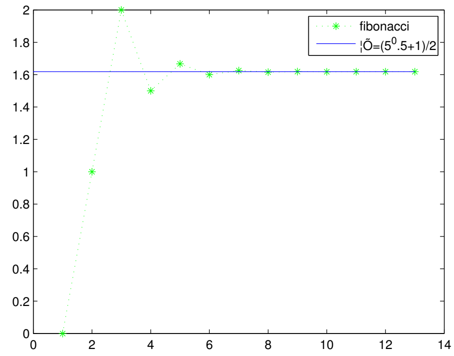

Figure 1: The approximation compared with golden ratio.

Code:

```
fibonacci(13)
for n=2:13
f=fibonacci(n);
r(n)=f(n)/f(n-1);
end
plot(r,'g:*');
t=0:0.01:13;
a=((5)^0.5+1)/2;
hold on;
plot(t,a,'b');
legend('fibonacci','¦Õ=(5^0.5+1)/2');
```

2.3 Statistical Measurements

```
Minimum value of  x:  0.9986
Maximum value of  x: 8.0862e-04
Mean of  x:  0.5071
Standard deviation of  x: 0.2862
Mean of y:  0.0283
Standard deviation of y: 1.1448
```

We can get the mean and standard deviation of y by calculating those values of x, and substituting them into y=4x-2.
The histogram of x shows in Figure 2, as we can see from it, it is a Gaussian distribution.

Code:

```
clear;
x=rand(1000,1);
max(x)
min(x)
r=mean(x)
s=std(x)
y=4*x-2;
```

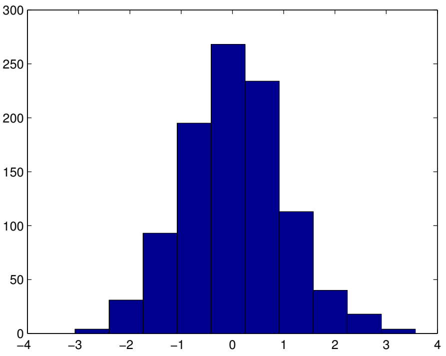

Figure 2: The Histogram of x.

```
m=mean(y)
n=std(y)
fprintf('4*%f-2=%f',r,m);
clear;
x=randn(1000,1);
hist(x)
```

2.4 An Optimization Example

Code:

```
clear;
r=[0.5:0.01:10];
for m=1:951
   h(m)=330/(pi*(r(m)^2));
end
A=2*pi*(r.^2)+2*pi*r.*h;
a=find(A==min(A));
r(a)
h(a)
figure(1);
plot(r,A,'g');grid on;
hold on;
plot(r(a),min(A),'r:*');
text(r(a),min(A)+100,'minimum');
fprintf('%f,%f',r(a),h(a));
figure(2);
u=[0:pi/60:2*pi];
v=[0:0.1:h(a)];
[U,V]=meshgrid(u,v);
X=r(a)*cos(U);
Y=r(a)*sin(U);
Z=V;
mesh(X,Y,Z);
axis equal;
```

Plot A(r) can we see in Figure 3

The minimum value of A(r) can we see from Figure 4, with r=3.7400,h=7.5097.

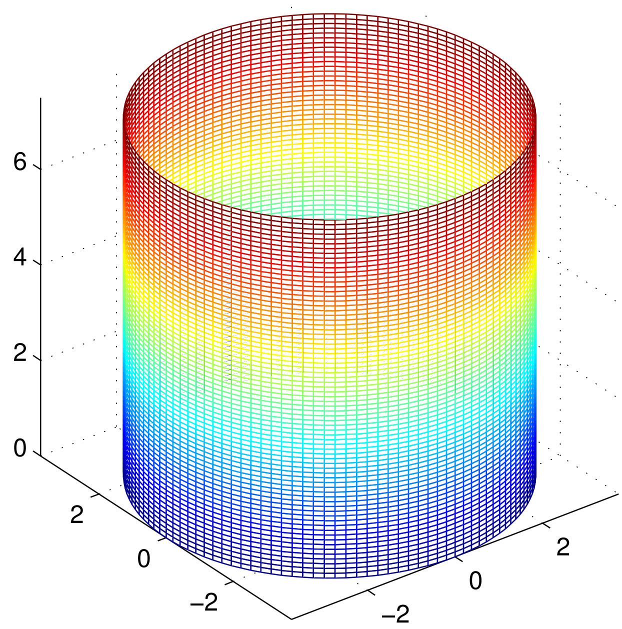

Figure 3: The surface graphic of A(r).

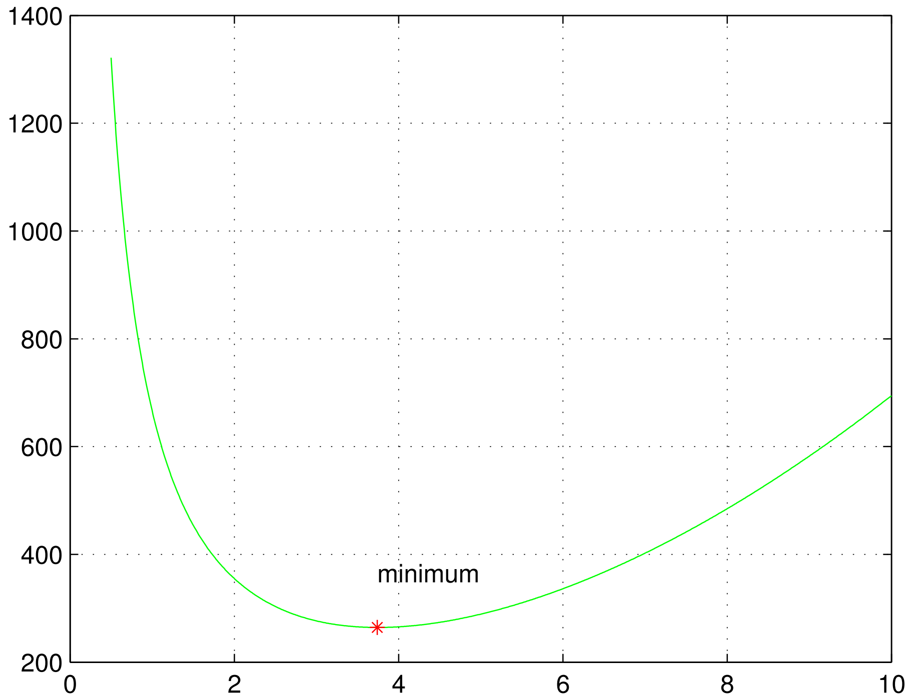

Figure 4: The minimum value of A(r).

Problem 2.5

As we can observe from this picture, as m increases, the more flatter the average result is.

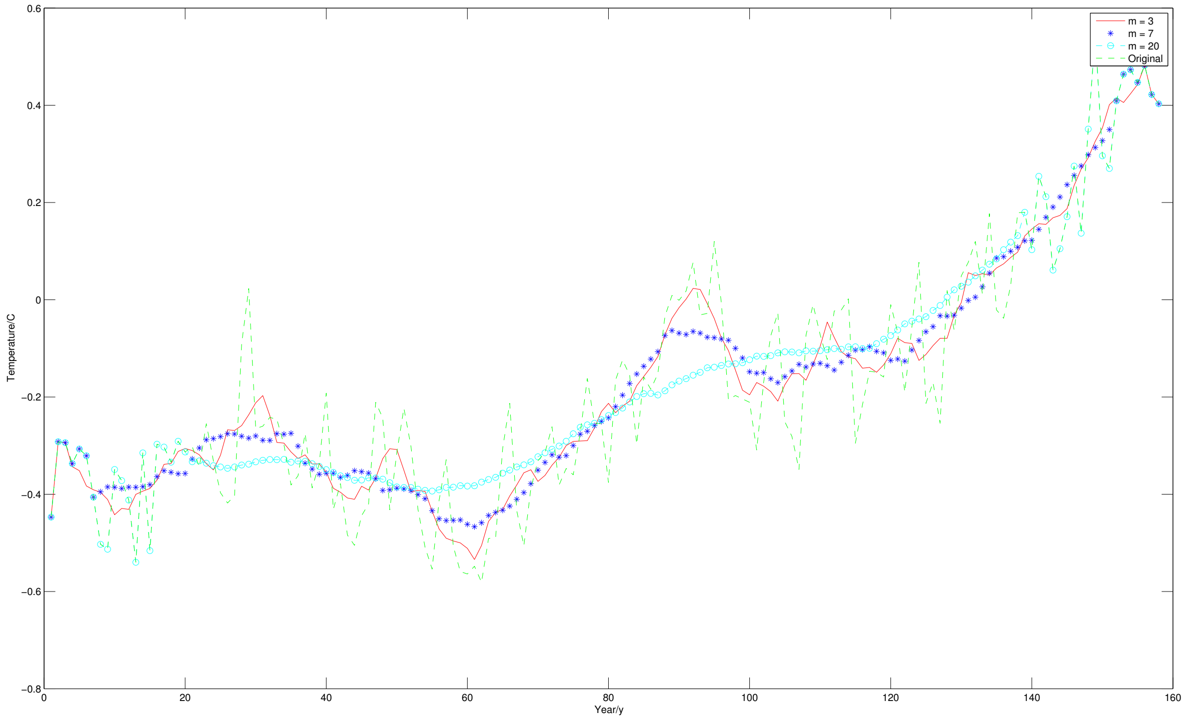

Figure 5: Climate Data with different average parameters m.

Code:

```
%%
clear
clear all
load glob_warm.mat
x_head = zeros(158,1);
x_head = moving_average(158,3);
plot(x_head,'r-');
xlabel('Year/y');
ylabel('Temperature/C');
hold on
plot(moving_average(158,7),'b*')
hold on
plot(moving_average(158,20),'c--o')
plot(Ta,'g--');
legend('m = 3','m = 7','m = 20','Original')
```

Function(`moving_averaeg.m`):

```
function  x_head = moving_average( n, m )
%MOVING_AVERAGE Summary of this function goes here
%   Detailed explanation goes here
load glob_warm.mat;

```

```
x_head = zeros(158,1);

for l =1:n
    if l < (m+1);
        x_head(l) = Ta(l);
    elseif l > (n-m);
        x_head(l) = Ta(l);
    elseif l>=(m+1) && l <=(n-m)
        for k = -m:m
            x_head(l) = x_head(l) + Ta(l+k);
        end
            x_head(l) = x_head(l)/(2*m+1);
    end
end
end
```

Problem 2.6

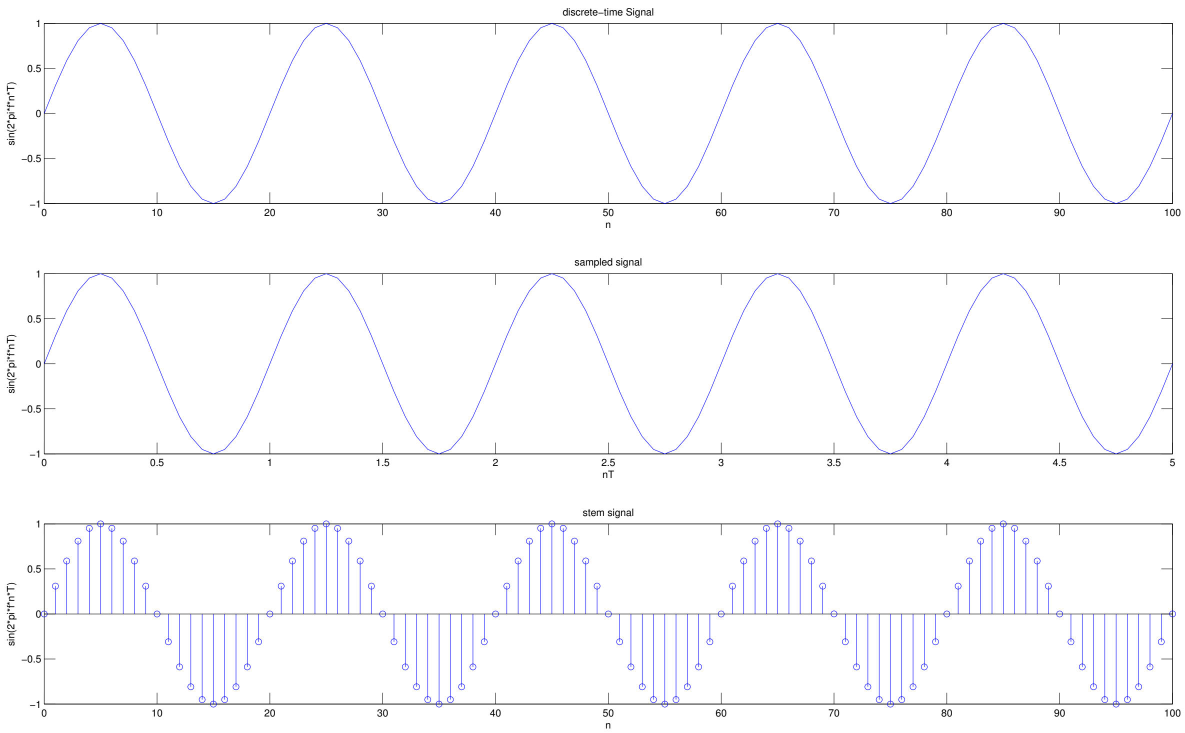

Figure 6: Polt of the original signal and sample signal.

Code:

```
%%
clear
clear all
n = 0:100;
F = 1;
T = 0.05;
s = sin(2*pi*F*n*T);
figure
```

```
subplot(3,1,1);
plot(n,s);
xlabel(' n ');
ylabel(' sin(2*pi*f*n*T) ');
title('discrete-time Signal');

subplot(3,1,2);
plot(n*T,s)
xlabel(' nT ');
ylabel(' sin(2*pi*f*nT) ');
title('sampled signal');

subplot(3,1,3);
stem(n,s);
xlabel(' n ');
ylabel(' sin(2*pi*f*n*T) ');
title('stem signal');
```

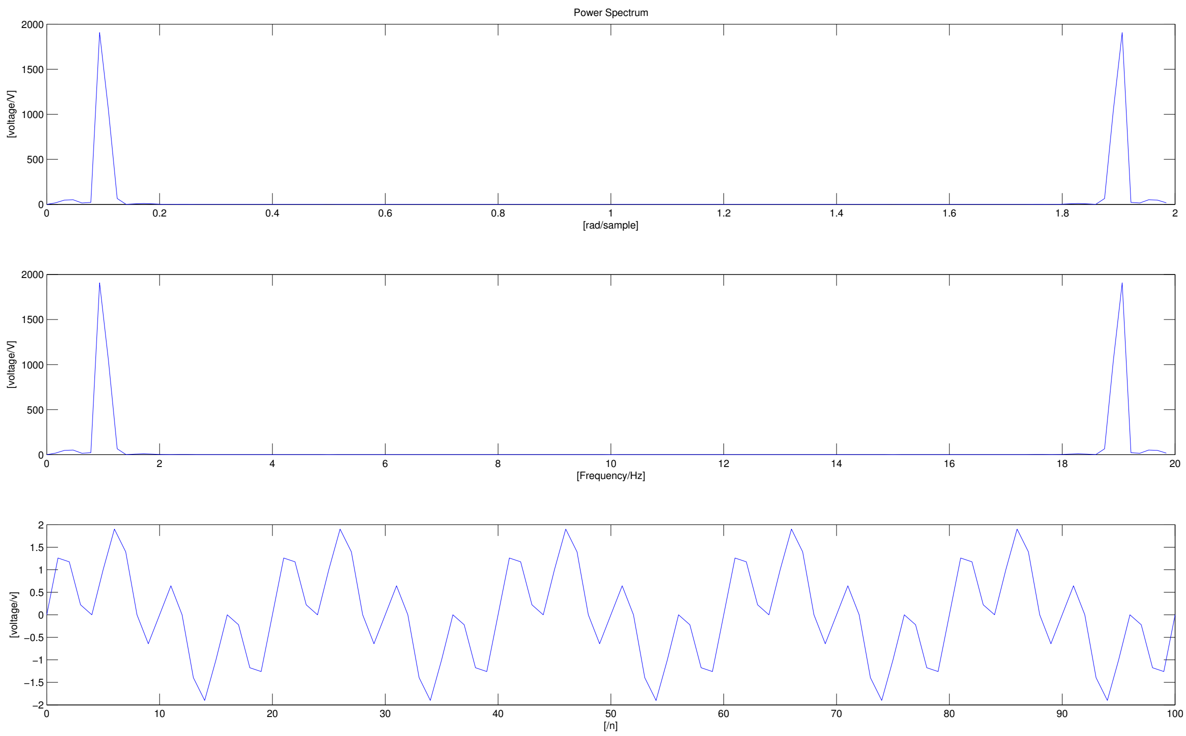

Figure 7: Spectrum of the signal (rad/sample vs. freq/Hz) and the distrubanced signal.

Code:

```
S = fft(s,128);
P = S.*conj(S);
w = (0:127)/128;
figure
subplot(3,1,1);
plot(2*w,P);
xlabel('[rad/sample]');
ylabel(' [voltage/V] ');
title('Power Spectrum');
```

```

subplot(3,1,2);
plot(w/T,P);
xlabel('[Frequency/Hz]');
ylabel(' [voltage/V] ');

s2 = s + sin(2*pi*4*n*T);
subplot(3,1,3);
plot(n,s2);
xlabel('[/n]');
ylabel(' [voltage/v] ')
```

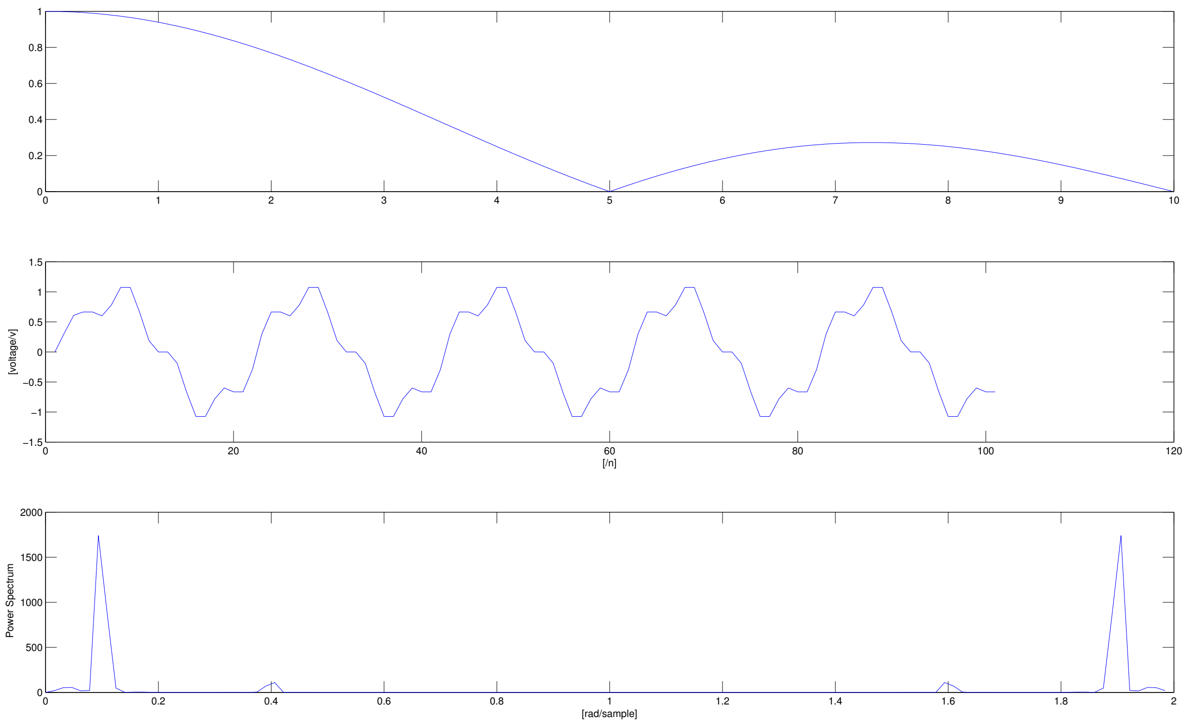

Figure 8: Frequency response of the filter and the recovered signal(magnitude vs. frequency/Hz).

Code:

```
b = [1 1 1 1]/4;
a = 1;
[H,W1] = freqz(b,a);

figure
subplot(3,1,1);
plot(W1/(2*pi*T),abs(H));
sf = filter(b,a,s2);

subplot(3,1,2);
plot(sf);
xlabel('[/n]');
ylabel(' [voltage/v] ');

SF    = fft(sf,128);
```

```
P_SF = SF.*conj(SF);
subplot(3,1,3);
plot(2*w,P_SF);
xlabel('[rad/sample]');
ylabel(' Power Spectrum');
```