California State University Long Beach

College Of Engineering

Computer Engineering and Computer Science Department

CECS 463 – Digital Signal Processing

Instructor: Haney W. Williams

Project-Assignment – 2

Names:

Sunny Pham

Chris Dong-Jae Shin

Lena Mourad

Due Date: 04/24/2019

Table of Contents

Part-1 **6**

Subpart 1.1a & 1.1b 6

Problem Description 6

Theory Used 6

Solution 6

Results 7

Subpart 1.1c 8

Problem Description 8

Theory Used 8

Solution 8

Results 8

Subpart 1.1d 9

Problem Description 9

Theory Used 9

Solution 9

Results 10

Subpart 1.2 11

Problem Description: 11

Theory Used 11

Solution 11

Subpart 1.3 13

Problem Description 13

Theory Used 13

Solution 13

Results 14

Subpart 1.4 14

Problem Description 14

Theory Used 14

Solution 15

Results 16

Subpart 1.5 17

Problem Description 17

Theory Used 17

Solution 17

Results 18

Part 2 **20**
Subpart 2.1 20
Problem Description 20
Theory Used 20
Solution 20
Results 20
Problem Description 21
Theory Used 21
Solution 21

Part 3 **25**
Subpart 3.1 25
Problem Description 25
Theory Used 25
Solution 25
Results 26
Subpart 3.2 27
Problem Description 27
Theory Used 27
Solution 27
Results 28
Subpart 3.3 30
Problem Description 30
Theory Used 30
Solution 30
Subpart 3.4 32
Problem Description 32
Theory Used 32
Solution 32
Results 32
Subpart 3.5 33
Problem Description 33
Theory Used 33
Solution 33
Results 34
Subpart 3.6 35
Problem Description 35

Theory Used 35
Solution 35
Results 35

Part 4 **35**
Subpart 4.1 35
Problem Description 35
Theory Used 35
Solution 36
Subpart 4.2 37
Problem Description 37
Theory Used 37
Solution 37
Results 38
Subpart 4.3 39
Problem Description 39
Theory Used 39
Solution 39

Part 5 **40**
Subpart 5.1a 40
Problem Description 40
Theory Used 40
Results 40
Subpart 5.1b 41
Problem Description 41
Theory Used 41
Results 42
Subpart 5.1.c 43
Problem Description 43
Theory Used 43
Solution 43
Results 43
Subpart 5.1.d 44
Problem Description 44
Theory 44
Solution 44
Results 45

Subpart 5.1e 46
Problem Description 46
Theory Used 46
Solution 46
Results 47
Subpart 5.1f 47
Comments 47

Encountered Problems: 48

Part-1

Subpart 1.1a & 1.1b

Problem Description

After reading the pdf file, find the expression for the complex Fourier coefficients c_n for the square wave signal f(t) from these two formulas. g(t) is the reconstruction of f(t) using the coefficients c_n:

$$c_n = \frac{1}{T}\int_0^T f(t)e^{-j\frac{2\pi nt}{T}}dt \qquad g(t) = \sum_{n=-\infty}^{\infty} c_n e^{j\frac{2\pi nt}{T}}$$

Note that c_0 is just the average value of the function taken over its period.
Once found, reconstruction of the waveform $g_N(t)$ using the complex coefficients c_n is done in Matlab by:

```
Wn=exp( 1j*2*pi/T * t'*n ) ;     % T is period, t is time vector, n is coefficient index vector.
W_n=conj(Wn);                    % The conjugate of Wn
gN=c0 + Wn*cn .' + W_n*c_n .';   % Careful not to conjugate transpose the row vector cn
```

Using the coefficients c_n, plot the square wave in blue and its 10 term ($n=\pm1, \pm3 \pm5, \pm7, \pm9$) representation g(t) in red on the same plot using the "hold on" command. Note: In Matlab, a unit square wave of period T=1 over interval −1.1 to 1.1 can be constructed in simple fashion by the commands:

```
T=1;  t = -1.1:0.01:1.1;  f = 0.5*sign(sin(2*pi*t/T));  plot(t,f);
axis([-1.1, 1.1, -1.1, 1.1]);  title('Periodic Square Wave');  xlabel('time t');  ylabel('f(t)');
```

Theory Used

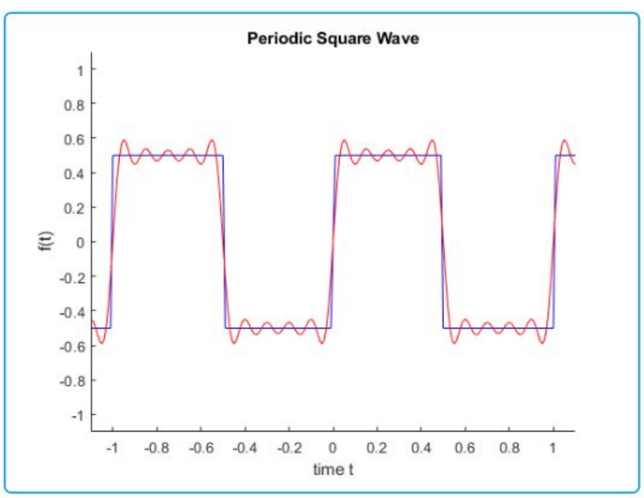
Discrete Time Fourier Transforms

Solution

```
t = -1.1:.01:1.1;

T = 2*pi; % integrate from 0 to 2pi

n = 0:10;

c0 = 0; %average energy is 0

cn = [0 -j*2/1/pi 0 -j*2/3/pi 0 -j*2/5/pi 0 -j*2/7/pi 0 -j*2/9/pi 0];

cn = cn ./2;

% y = 2/pi*sin(t)+2/3/pi*sin(3*t)+2/5/pi*sin(5*t)+2/7/pi*sin(7*t)+2/9/pi*sin(9*t);
```

```
% c0 is the average of the whole signal, which is 0.
% cn = 1/(2*n*pi)*cos(n*t);
c_n = conj(cn);

Wn = exp(1j * 2 * pi / T *2 * pi * t'*n); %Wn modified to fit the x axis required
W_n = conj(Wn);
gN = c0 + Wn * cn.' + W_n*c_n.';

gN_1 = gN;

f = 0.5*sign(sin(2*pi*t));
f_1 = f;
f = f.';
hold on
plot(t, f, 'blue')
plot(t, gN, 'red')
axis([-1.1, 1.1, -1.1, 1.1]); title('Periodic Square Wave'); xlabel('time t'); ylabel('f(t)');
hold off
```

Results



Subpart 1.1c

Problem Description

Plot the absolute value of the error between the points in f(t) and its estimate $g_N(t)$. Also calculate the rmse between the two and put it in the title of the plot using Matlab's *sprintf* command like so:

rmse = sqrt(sum(abs(f-gN).^2)/length(f)); plot(t,rmse);
str = sprintf ('RMS Error | |x(t)–g_N(t)| | with RMSE= %6.4f ', rmse);
title(str); xlabel('time t'); ylabel('Root Mean Square Error');

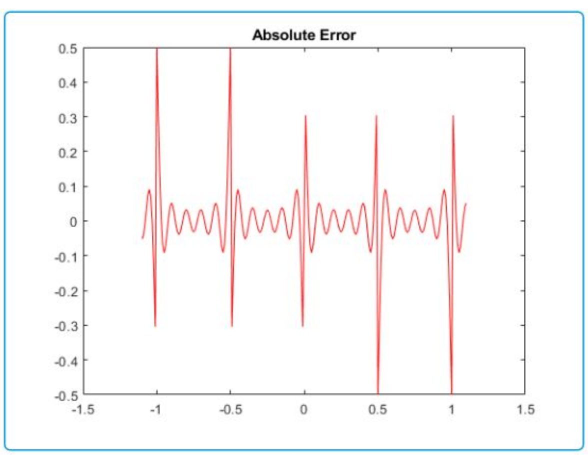
Theory Used

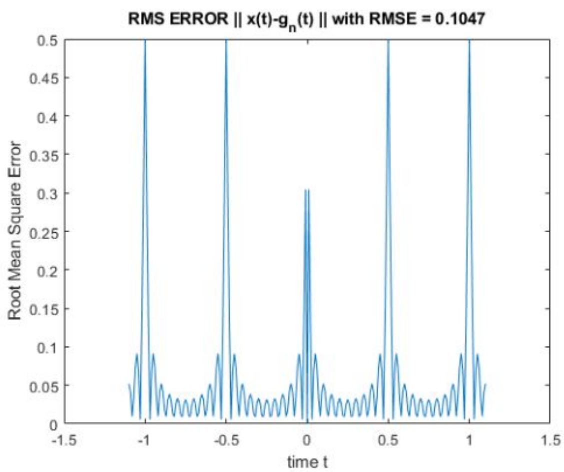
RMS Error, Discrete Time Fourier Transform

Solution

```
abs_error = f - gN;
plot (t, abs_error, 'red');
title('Absolute Error')
rmse = sqrt(sum(abs(f-gN).^2)/length(f));
plot(t, sqrt((f - gN).^2));
str = sprintf('RMS ERROR || x(t)-g_n(t) || with RMSE = %6.4f', rmse);
title(str);
xlabel('time t'); ylabel('Root Mean Square Error');
```

Results





Subpart 1.1d

Problem Description

Determine how many coefficients are required for the rmse to fall just below a value of 0.075.

Theory Used

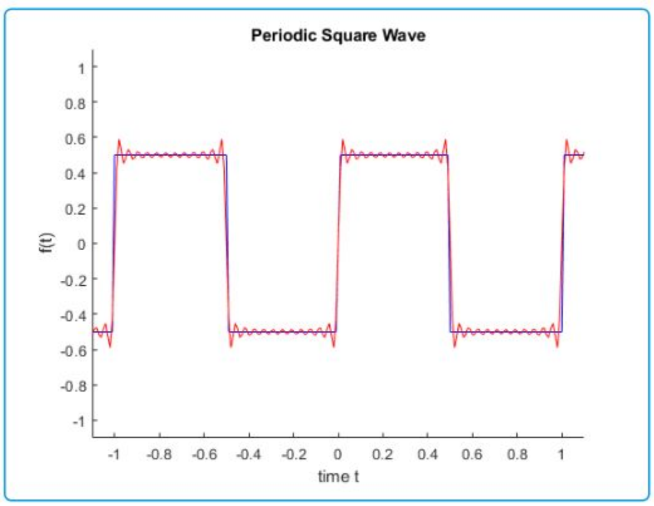
N/A

Solution

```
figure()

t = -1.1:.01:1.1;

T = 2*pi; % integrate from 0 to 2pi

n = 0:23;

c0 = 0; %average energy is 0

cn = [0 -j*2/1/pi 0 -j*2/3/pi 0 -j*2/5/pi 0 -j*2/7/pi 0 -j*2/9/pi 0 -j*2/11/pi 0 -j*2/13/pi
0 -j*2/15/pi 0 -j*2/17/pi 0 -j*2/19/pi 0 -j*2/21/pi 0 -j*2/23/pi];

cn = cn ./2;

c_n = conj(cn);


Wn = exp(1j * 2 * pi / T *2 * pi * t'*n); %Wn modified to fit the x axis required

W_n = conj(Wn);

gN = c0 + Wn * cn.' + W_n*c_n.';
```

```
f = 0.5*sign(sin(2*pi*t));
f = f.';

hold on
plot(t, f, 'blue')
plot(t, gN, 'red')
axis([-1.1, 1.1, -1.1, 1.1]); title('Periodic Square Wave'); xlabel('time t'); ylabel('f(t)');
hold off
rmse = sqrt(sum(abs(f-gN).^2)/length(f))
%24 terms are required before the RMSE falls below 0.075
```

Results




```
rmse = 0.0739
```

Subpart 1.2

Problem Description:

Repeat the process of (1), but this time use a periodic triangular waveform defined by x=t on t∈[0,1] and x=2 – t on t∈[1,2]. You can make use of the integrals in the pdf for the saw-tooth wave to find the c_n.

a) For which signal does the Fourier series converge more rapidly, the periodic square wave signal or the periodic triangular signal? On what grounds could you base your answer? Justify your response by finding the number of coefficients of each that is required to have the rmse fall just below the rmse threshold of 0.05.

Theory Used

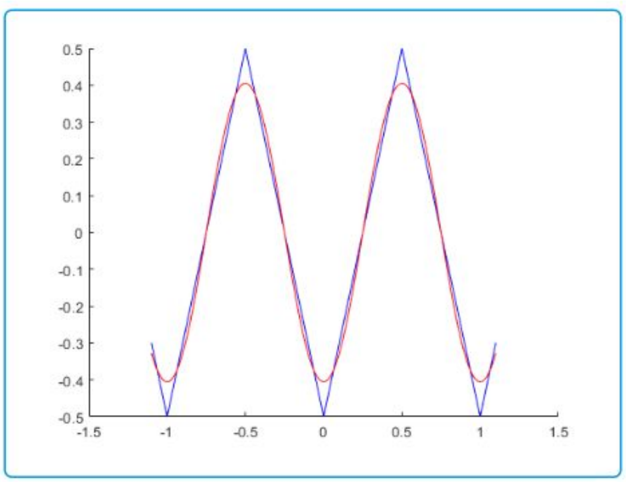
Discrete Time Fourier Transform

Solution

```
figure()

t = -1.1:.01:1.1;

T = 2*pi; % integrate from 0 to 2pi

n = 0:10;

c0 = 0; %average energy is -0.25

cn = zeros([1 11]);

for k = 1:2:10

cn(k+1) = -1*(1 - (-1).^k)/pi.^2 / k.^2;

End


% set new n and cn to find the minimum number of terms for rmse to be below

%0.05

cn = cn(1:2);

n = 0:1;

c_n = conj(cn);


Wn = exp(1j * 2 * pi / T *2 * pi * t'*n); %Wn modified to fit the x axis required

W_n = conj(Wn);

gN = c0 + Wn * cn.' + W_n*c_n.';
```

```
f = 0.5*sawtooth(2*pi*t,1/2);
f = f.';
hold on
plot(t, f, 'blue')
plot(t, gN, 'red')
%axis([-1.1, 1.1, -1.1, 1.1]); title('Periodic Triangle Wave'); xlabel('time t'); ylabel('f(t)');
hold off
rmse = sqrt(sum(abs(f-gN).^2)/length(f))
%obviously the periodic triangular signal converges more rapidly, as a sine
%wave is already close to a triangle wave. Even with one term, the RMSE is
%below .05.
```

Results:



Subpart 1.3

Problem Description

Suppose you have a periodic waveform on $t \in [0,5]$ such as:

$$x(t) = t^2 (5-|t|)e^{(-1+\cos(\sin(\pi t)))}/(2+e^{-|t-2.5|})$$

In this case the analytic integration formula for the Fourier series coefficients is difficult or impossible to find. When this occurs, the coefficients can be obtained numerically. Remember that an integral can be approximated by a sum (area under the curve):

$$\int_0^T f(t)dt \approx \Delta T \sum_m f(m\Delta T)$$

For example the Fourier coefficients

$$c_n = \frac{1}{T}\int_0^T f(t)e^{-jn\omega_0 t}\,dt$$

can be approximated by a sum

$$c_n \approx \frac{\Delta T}{T}\sum_{m=0}^{M-1} f(m\Delta T)e^{-jn\omega_0 m(\Delta T)}$$

Using numerical integration Matlab program outlined in the pdf, find the n=±10 coefficients c_n in the complex FS expansion of x(t). Plot the functions and error as done in (1). Also find the number of terms required to make the rmse fall just below the 0.075 threshold value.

Theory Used

Discrete Fourier Transform

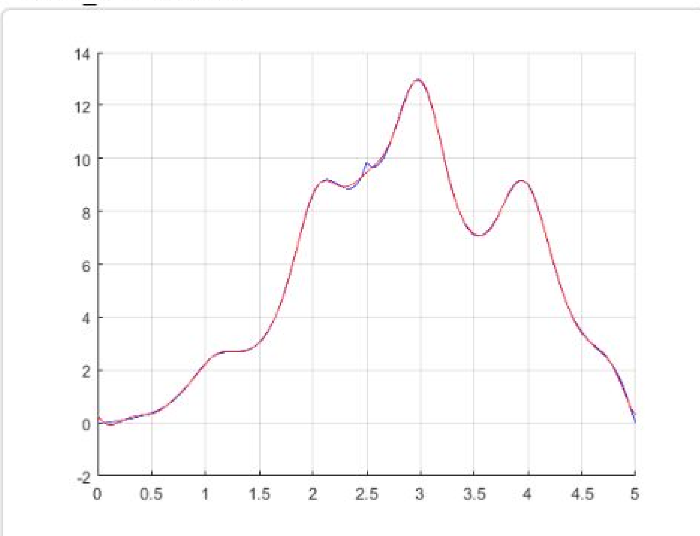
Solution

```
figure()
T=5; delT=T/1000; t=0:delT:T;
%t.^2 .* (5 - abs(t)).*exp(-1+cos(sin(pi.*t)))./(2-exp(-abs(t-2.5)));
f = t.^2 .* (5 - abs(t)).*exp(-1+cos(sin(pi.*t)))./(2-exp(-abs(t-2.5)));

c0=1/T * sum(f*delT); %getting the average
N=11; %number of terms
for n=1:N
cn(n) = delT/T * sum(f.*exp(-1j*2*pi*n*t/T));
c_n(n) = delT/T * sum(f.*exp(+1j*2*pi*n*t/T));
end
n=1:N;
Wn=exp(1j*2*pi/T*t'*n);
W_n=conj(Wn); gN = ( c0 + Wn*cn.' + W_n*c_n.' )';
rmse_1=sqrt(sum(abs(f-gN).^2)/length(f));

rmse_1 %display rmse
```

```
hold on
grid on
plot(t, f, 'b')
plot(t, gN, 'r')
hold off

% 11 terms are required to bring the RMSE below 0.075.
```

Results

```
rmse_1 = 0.0578
```



Subpart 1.4

Problem Description

The time-shift property states that if a periodic signal is shifted in time, then the Fourier coefficients of that signal are shifted in frequency according to the following rule:

$$\textit{If } x(t) \leftrightarrow c_n \quad \textit{then} \quad x(t-t_o) \leftrightarrow \exp(-j\, n\, \omega_o\, t_o)\, c_n$$

Verify this property numerically using Matlab, by letting g(t) = f (t−0.25) of the square wave of (1). Then find the Fourier coefficients of g(t) by first finding the Fourier coefficients of f (t) of problem (1) and modifying those coefficients according to the rule. Use the new coefficients to synthesize the periodic signal $g_N(t)$ with N=20 by summing the complex exponentials. Confirm the time-shift property by a 3x1 subplot of the results of f(t), g(t) and its FS approximation $g_N(t)$ and noting their similarity, and then better still, finding the root mean squared error between g(t) and $g_N(t)$ and putting it in the title of the subplot of $g_N(t)$.

Theory Used

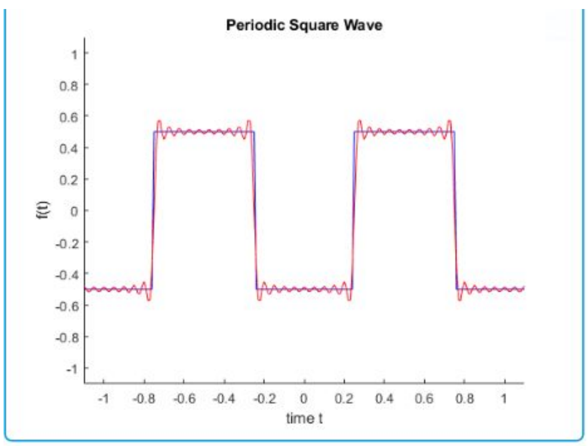
Discrete Time Fourier Transform

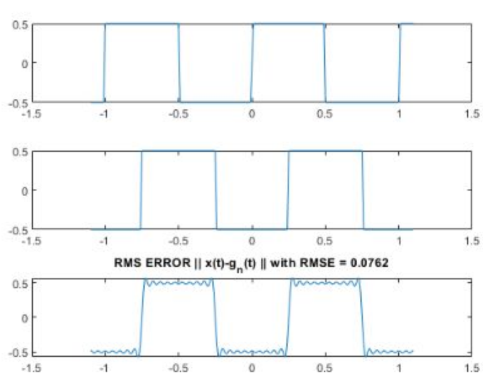
Solution

```
figure()

t = -1.1:.01:1.1;

T = 2*pi; % integrate from 0 to 2pi

n = 0:20;

c0 = 0; %average energy is 0

cn = zeros([1 21]);

for k = 1:2:20

cn(k+1) = -(1/pi/k)*((-1).^((k-1)/2));

end



c_n = conj(cn);


Wn = exp(1j * 2 * pi / T *2 * pi * (t)'*n); %Wn modified to fit the x axis required

W_n = conj(Wn);

gN = c0 + Wn * cn.' + W_n*c_n.';


f = 0.5*sign(sin(2*pi*(t-.25)));

f = f.';

hold on

plot(t, f, 'blue')

plot(t, gN, 'red')

axis([-1.1, 1.1, -1.1, 1.1]); title('Periodic Square Wave'); xlabel('time t'); ylabel('f(t)');

hold off
```

```
subplot(3,1,1)
plot(t, f_1)
subplot(3,1,2)
plot(t, f)
subplot(3,1,3)
plot(t, gN)
rmse = sqrt(sum(abs(f-gN).^2)/length(f));
str = sprintf('RMS ERROR || x(t)-g_n(t) || with RMSE = %6.4f', rmse);
title(str);
```

Results

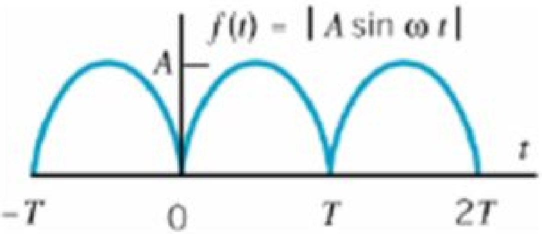




Subpart 1.5

Problem Description

Find the Fourier series coefficients for the periodic signal ***f(t) = |cos(π t)|*** , t∈[−1/2,1/2). Repeat parts (1a) and (1b) only. To help you out a bit, look at this expansion:



Full wave rectified sine wave: $\omega_0 = \dfrac{2\pi}{T}$

$$f(t) = \frac{2A}{\pi} - \frac{4A}{\pi}\sum_{n=1}^{\infty}\frac{\cos(n\,\omega_0 t)}{4n^2-1}$$

Theory Used

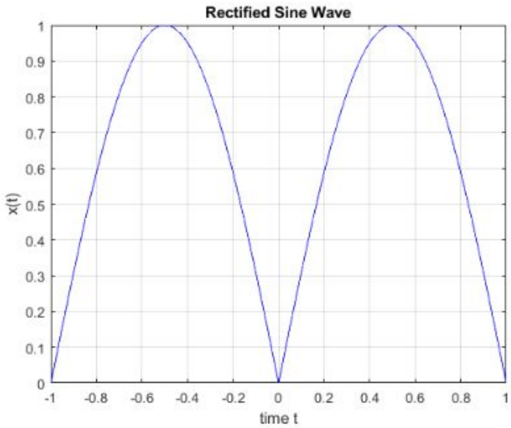
Discrete Fourier Transform, Time Shift of LTI System

Solution

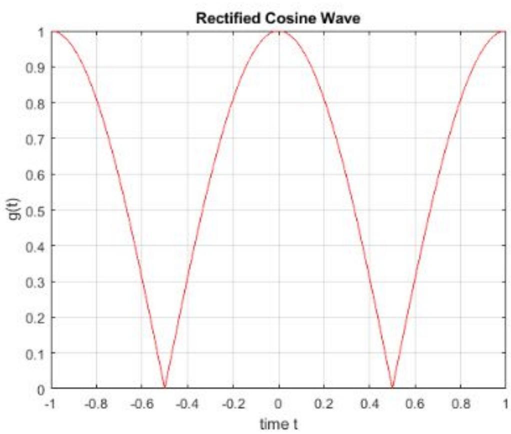
```
figure()
T=2;
delta_T=T/1000;
t=-T/2:delta_T:T/2;

f = abs(sin(2*pi/T*t));
plot(t , f, 'b');
grid on;
title("Rectified Sine Wave");
xlabel('time t');
ylabel('x(t)');

t_0 = -0.25*T;

g = abs(sin(2*(pi/T*(t-t_0))));

figure()

plot(t, g, 'r')
grid on;
title("Rectified Cosine Wave");
xlabel('time t');
```

```
ylabel('g(t)');

T = 1;
t_0 = -T/2;
N = 9;
c0 = (.5*T)*sum(g*delta_T);
cn = zeros(1,N);
c_n = cn;

for k = 1:N/2:2+1
cn(k) = -(2/pi)/(4*k*k-1)* exp(-1j*k*2*pi/T*t_0);
%Change phase of each coeff
c_n(k) = -(2/pi)/(4*k*k-1)* exp(-1j*(-k)*2*pi/T*t_0);
%Change phase of each coeff
end

n = 1:length(cn);
Wn =exp(+1j* (2*pi/T) * t'*n );
W_n=exp(-1j* (2*pi/T) * t'*n );
%reconstruct waveform
gN = ( c0 + Wn*cn.' + W_n*c_n.' )';

figure()
hold on
plot(t,g,'r');
plot(t,gN,'b');
grid on;
xlabel('time t');
ylabel('g(t) in red, g_N(t) in blue');
title("Fourier Approximation of a rectified Cosine wave")
hold off
```

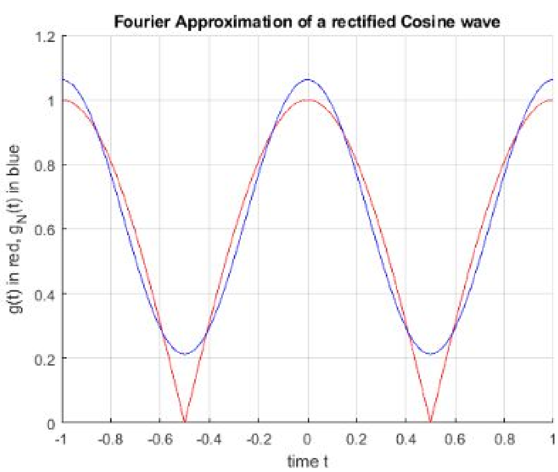
Results



Rectified Sine Wave
x(t)
time t



Rectified Cosine Wave
g(t)
time t



Fourier Approximation of a rectified Cosine wave
g(t) in red, g_N(t) in blue
time t

Part 2

Subpart 2.1

Problem Description

Let x=[1,2,3,4] and find the output for the following LTI difference equations:

(a) y[n]=0.5x[n]+x[n-1]+2x[n-2]
(b) y[n]=0.8y[n-1]+2x[n]
(c) y[n]-0.8y[n-1]=2x[n-1]

Theory Used

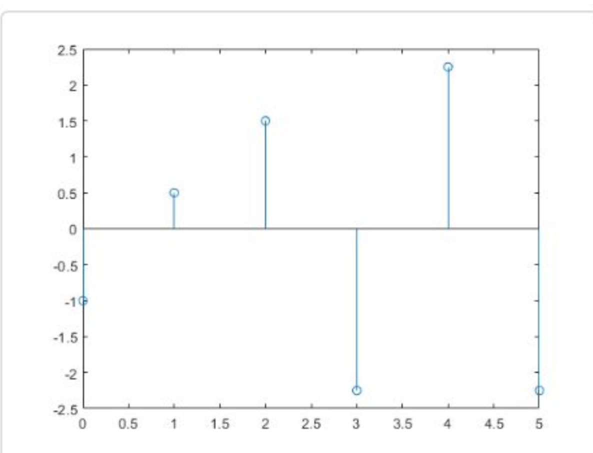
N/A

Solution

```
figure()
% indices less than 1 are all zeros
a = LTI_a()
b = LTI_b()
c = LTI_c()
```

Results

```
a = 1x4
      0.5000    2.0000    5.5000 ...

b = 1x4
      2.0000    5.6000   10.4800 ...

c = 1x4
           0    2.0000    5.6000 ...
```



Subpart 2.2

Problem Description

Let x[n]=(-1).^n over interval $0 \le n \le 5$ and let y[n]=x[n]+0.5x[n-1]-2x[n-2]+0.75x[n-3].

(a) Find the vector b for the LTI equation for y[n]. Set h=b and use the filter function y=filter(h,1,x) to find y[n] over the interval ny=0:5. Plot the results using stem(ny,y). Note: vector a is just the vector [1].

(b) If ***filter*** is to generate the same output as *conv*, then the input x[n] to the filter function must be of length 9 (=6+4-1), not 5 as before. Append 3 zeros to x to make x1[n] of length 9, and find y1=filter(h,1,x1). Compare this result to the result ***yconv=conv(h,x).*** Is the difference ***y1-yconv*** identically zero at all points?

(c) The filter function can implement a non-causal impulse response. Let h(n)=n for $0 \le n \le 5$. Generate the impulse response h1[n]=h[n+5] over $-5 \le n \le 0$. Let x[n]=(-1).^n over $0 \le n \le 5$. The output y1[n] using h1[n] can be shown to be just an advanced version of the output y[n] using h[n], namely y1[n]=y[n+5]. Stem plot using subplot the impulse responses h[n] and h1[n] and the outputs of y[n] and y1[n] to verify this result.

Theory Used

Filtering

Solution

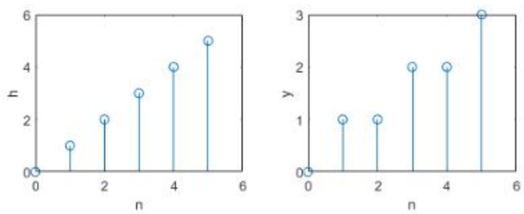
```
% (a)

n = [1:1:6];

ny = 0:5;

x(n)= (-1).^n;


y(1) = x(1);

y(2) = x(2) + 0.5*x(1);

y(3) = x(3) + 0.5*x(2) - 2*x(1);


for n = 4:6

y(n) = x(n) + 0.5*x(n-1) - 2*x(n-2) + 0.75*x(n-3);

end

h = [1,0.5,-2,0.75];

y = filter(h,1,x);

stem(ny,y);
```

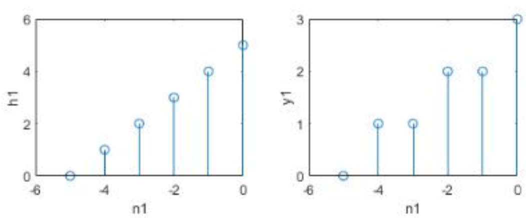
```
%(b)

x1 = zeros(1,9);
for n = 1:length(x)
x1(n) = x(n);
end

y1 = filter(h,1,x1);
yconv = conv(h,x);

% difference between the y1 - yconv is all zeros
y1-yconv

%(c)
n = [0:1:5];
h = n;
x = (-1).^n;

y = filter(h, 1, x);

figure();
hold on;
subplot(2,2,1);
stem(n,h);
xlabel('n'); ylabel('h');

subplot(2,2,2);
```

```
stem(n, y);
xlabel('n'); ylabel('y');
hold off;

n1 = [-5:1:0];



% h1 needs to be shifted n-5, but arrays cannot have negative or zero index
h1 = h;

%y1 is same as y... for n1, which is from -5 to 0. It's just shifted to the
%left...

figure();

hold on;
subplot(2,2,1);
stem(n1,h1);
xlabel('n1'); ylabel('h1');

subplot(2,2,2);
stem(n1, y);
xlabel('n1'); ylabel('y1');
hold off;
```

Results

```
ans = 1×9
     0     0     0     0     0     0 ...
```





Part 3

Subpart 3.1

Problem Description

Load the file ***mtlb*** into Matlab by executing the command: ***load mtlb.mat***. Now enter "who" and see what variables are present. You should see the variable ***mtlb*** and ***Fs***. Set x=mtlb and play the signal variable x by execution the command "sound(x,Fs)". You should hear the word "matlab." The echo signal y[n] which is represented by the vector y, is of the form y[n]=x[n] + α x[n-D], where x[n] is the uncorrupted speech signal which has been delayed by D samples and added back in with its amplitude multiplied by the gain factor α < 1. This is a reasonable model of an echo reflecting off an absorbing barrier such as a wall. Here assume the value of α =0.9. Find the length of x, and append that many zeroes to the vector x to increase the length of the sound vector to allow some room for the echo. Let the length of x be Nx and the number of delay samples be D=2750. Define the filter vectors ***a*** and ***b*** from the difference equation and generate the echo signal using ***y=filter(b,a,x)***. Play the echo signal back using ***sound(y,Fs)*** to hear the echo. Subplot on a 3x1 figure(1) the signals x and y.

Theory Used

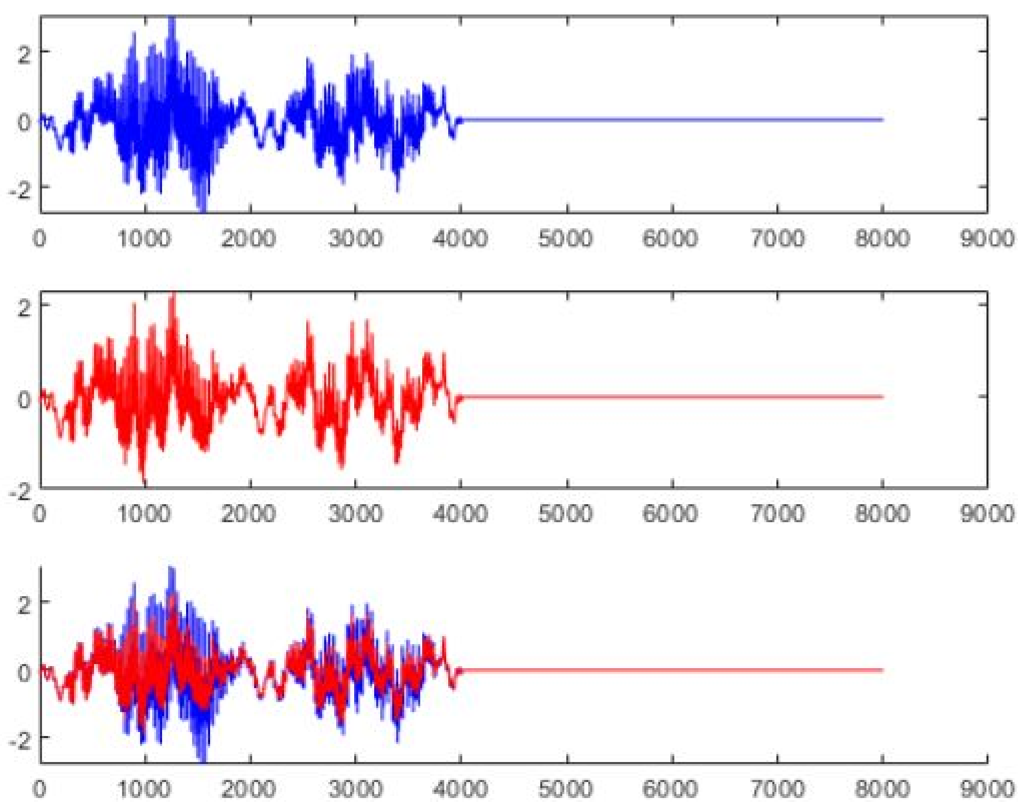
N/A

Solution

```
figure()

load mtlb.mat

x = mtlb;

%sound(x,Fs);


x = zeros(8002,1);


for n = 1:4001

x(n) = mtlb(n);

end


Nx = 1:8002;

D = 2750;


windowSize = 5;
```

```
y = filter(ones(1,windowSize)/windowSize,1,x);

subplot(3,1,1)
plot(Nx, x, "blue")
subplot(3,1,2)
plot(Nx, y, "red")
subplot(3,1,3)
hold on
plot(Nx, x, "blue")
plot(Nx, y, "red")
hold off
```

Results



Subpart 3.2

Problem Description

Since the echo signal can be represented by a linear system of the form mentioned, determine the impulse response of this echo system. Store the impulse response in a vector ***he*** for $0 \leq n \leq Nx$.

Theory Used

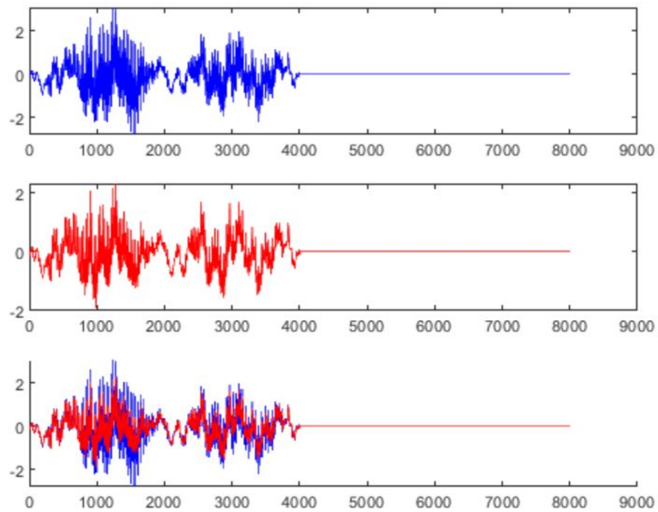
Impulse Functions

Solution

```
% form

% y is the echo signal

% x is the uncorrupted speech signal

% d is the delay

% alpha 0.9

% length of x is Nx

% delay sample D = 2750

load mtlb.mat

x = mtlb;

%sound(x,Fs);


x = zeros(8002,1);


for n = 1:4001

x(n) = mtlb(n);

end


Nx = 1:8002;

D = 2750;

a = 0.9;
```

```
for n = 1:8002
if(n > 2750)
he(n) = x(n) + a*x(n-D);
else
he(n) = x(n);
end
end

subplot(3,1,1);
plot(Nx, x, "red");
title('Regular signal');
xlabel('Nx');
ylabel('x');
subplot(3,1,2);
plot(Nx, he, "blue");
title('Echo signal');
xlabel('Nx');
ylabel('he');
subplot(3,1,3);
hold on
plot(Nx, x, "red");
plot(Nx, he, "blue");
title('Combined signal');
xlabel('Nx');
ylabel('he or x');
hold off
```

Results



2
0
-2
0 1000 2000 3000 4000 5000 6000 7000 8000 9000
2
0
-2
0 1000 2000 3000 4000 5000 6000 7000 8000 9000
2
0
-2
0 1000 2000 3000 4000 5000 6000 7000 8000 9000

Subpart 3.3

Problem Description

Consider an echo removal system described by the LSI difference equation $z[n] + \alpha z[n-D] = y[n]$, where $y[n]$ is the input and $z[n]$ is the output which has the echo removed. Show that this equation is indeed an inverse of the first equation by deriving the overall difference equation relating $z[n]$ to $x[n]$. Is $z[n]=x[n]$ a valid solution to the overall difference equation?

Theory Used

LSI Difference Equation

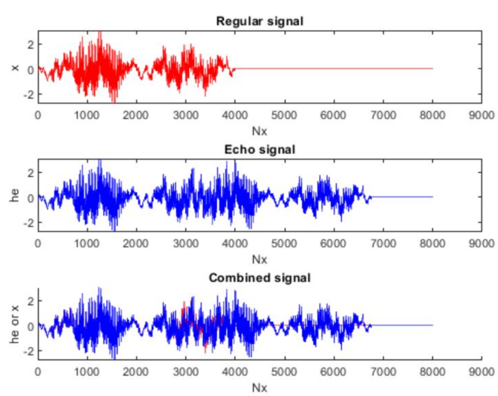
Solution

```
% Echo removal system

% LSI difference equation z[n] + a z[n-D] = y[n]

% y is the input

% z is the output


% show that this equation is the inverse of the first equation


% Echo made from the previous problem.

load mtlb.mat

x = mtlb;


x = zeros(8002,1);


for n = 1:4001

x(n) = mtlb(n);

end


Nx = 1:8002;

D = 2750;

a = 0.9;
```

```
for n = 1:8002
if(n > 2750)
he(n) = x(n) + a*x(n-D);
else
he(n) = x(n);
end
end


% LSI difference equation: z[n] = y[n] - a *z[n-d]
% y[n] is the echoed x.

for n = 1:8002
if(n > 2750)
z(n) = he(n) - a*he(n-D);
else
z(n) = he(n);
end
end

subplot(3,1,1);
plot(Nx, x, "red");
title('Regular signal');
xlabel('Nx');
ylabel('x');
subplot(3,1,2);
plot(Nx, he, "blue");
```

```
title('Echo signal');
xlabel('Nx');
ylabel('he');
subplot(3,1,3);
plot(Nx, z, "blue");
title('Echo system removal');
xlabel('Nx');
ylabel('z');
sound(z,Fs);
% z[n] is an inverse of [x] because on the echo system removal does remove
% the signal from Nx: 4000-5600 ish, because on the removal system the
% signal cancels out. However z[n] = x[n] is not a valid solution to the
% overall difference equation, because the after delay x 3 Nx, there would
% be an inverse of the signal of 0 - delay. Which means that the signal
% would be repeating afterwards.
```

Results



Subpart 3.4

Problem Description

The inverse filter echo removal system of (3) will have an infinite impulse response. Assuming the previous values of D and α, compute the impulse response using ***filter*** with an impulsive input d[n] given by the vector ***d***=[1,50000]. Store this approximation to the infinite impulse response in the vector ***her***.

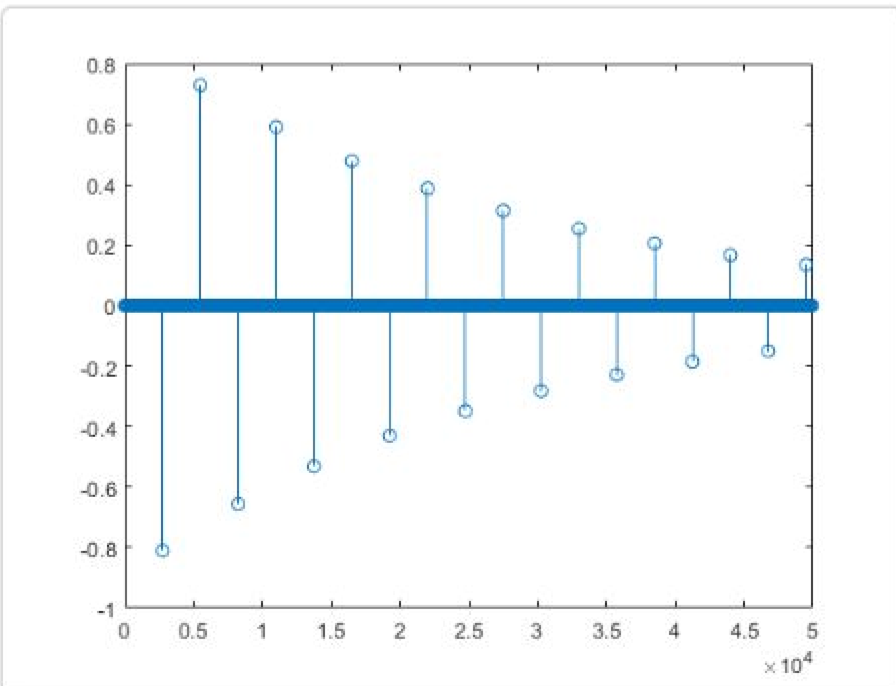
Theory Used

N/A

Solution

```
d = [1:50000];

a = 0.9;
b = [1,zeros(1,2750-1), 0.9];
her = impz(a,b,d);
figure()
title('her');
stem(her);
```

Results



Subpart 3.5

Problem Description

Implement the echo removal system using ***w=filter(1,a,y)*** where ***a*** is the appropriate coefficient vector derived from the equation in (3). Plot the output using plot as the third graph in figure(1). Also listen to the output using ***sound(w)***. The echo should not be present.

Theory Used

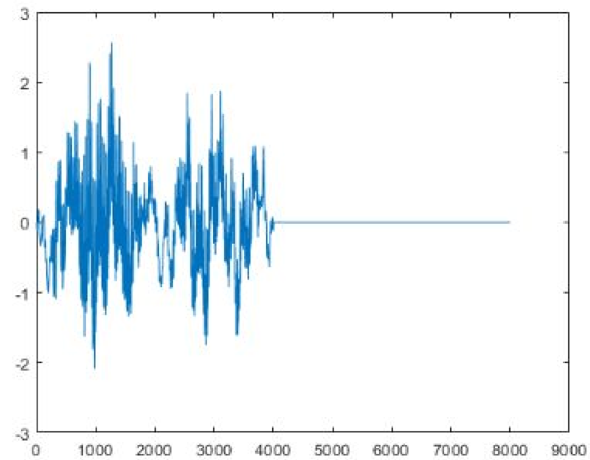
N/A

Solution

```
x = mtlb;
```

```
%sound(x,Fs);

x = zeros(8002,1);

for n = 1:4001
x(n) = mtlb(n);
end

for n = 1:8002
if(n > 2750)
he(n) = x(n) + a*x(n-D);
else
he(n) = x(n);
end
end

Nx = 1:8002;
D = 2750;

windowSize = 5;
y = filter(ones(1,windowSize)/windowSize,1,x);

a = 0.9;

w = filter(1,a,y);

figure()
title('w');
plot(w);
sound(w,Fs);
```

Results



Subpart 3.6

Problem Description

Calculate the overall impulse response of the cascaded echo system (equation in (2)) and the echo removal system (equation in (3)), by convolving **he** with *her* and store the results in *hoa*. In figure(2), plot on a 3x1 subplot the vectors *he*, *her* and *hoa*. The resulting plot of **hoa** is not a unit impulse as you would expect. Why is this the case?

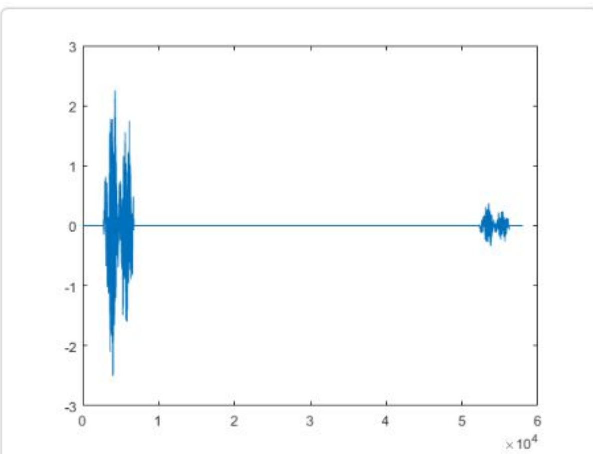
Theory Used

Convolutions

Solution

```
hoa = conv(he, her);
figure()
title('hoa');
plot(hoa);
```

Results



Part 4

Subpart 4.1

Problem Description

Create a vector h[n]=u[n]-u[n-11] where u[n] is the unit step function. For N=100, use the DTFT equation to find $H(e^{j\omega})$. Plot the results in figure(1). Remember to plot the magnitude and phase (in degrees) of a complex quantity, normally using a 2x1 subplot with the frequency axis in units of ω/π.

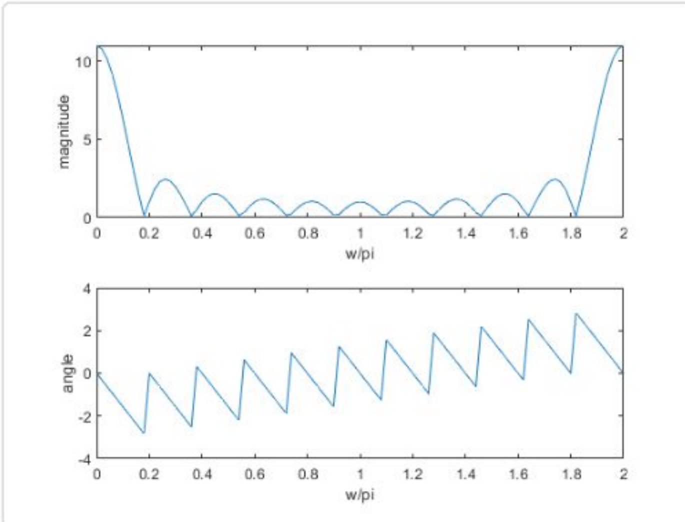
Theory Used

Unit Step Functions, DTFT

Solution

```
n = [0:100];
u = (n >= 0);
u_11 = (n - 11 >= 0);
h = u - u_11;
w = 2* pi* n / 100;
%DTFT function
x = h*exp (-1j*n'*w);
mag = abs(x); ang = angle(x);
figure()
hold on
subplot(2,1,1);
plot(w/pi,mag);
xlabel('w/pi');
ylabel('magnitude');
subplot(2,1,2);
plot(w/pi,ang);
xlabel('w/pi');
ylabel('angle');
hold off
```

Results



Subpart 4.2

Problem Description

The DTFT equation can be accomplished more easily by using the matrix capabilities of Mablab. Define a vector k=[0:N] where N is the number of frequency samples required over [0,2π] and a vector n=[n1:n2] where n1 is the initial sample index of the sequence x[n] and n2 is the final sample index. Then in Matlab the expression X=x*(exp(-1j*2*pi/N)).^(n' * k) gives the vector representing the continuous function $X(e^{j\omega})$ over the interval [0,2π] with w = k*2*pi/N. Find the DTFT of $x(n) = e^{-0.2|n|}$ over n=-10:10 with N=1000 using this vector approach to finding $X(e^{j\omega})$ and plotting the results in figure(2).

Theory Used

DTFT

Solution

```
N = 1000;

n1 = -10;

n2 = 10;


% frequency samples required over [0 ,2pi]

k = [0:N];


% n1: initial sample index

% n2: final sample index
```

```
n = [n1:n2];

x = exp(-0.2*abs(n));

% vector representing the continuous function X(e^jw)
% over interval [0, 2pi]
X = x*(exp(-1j*2*pi/N)).^(n'*k);
w = k*2*pi/N;

mag = abs(X);
ang = angle(X);

figure()
hold on
subplot(2,1,1);
plot(w/pi,mag);
xlabel('w/pi');
ylabel('magnitude');
subplot(2,1,2);
plot(w/pi,ang);
xlabel('w/pi');
ylabel('angle');
hold off
```

Results

Subpart 4.3

Problem Description

Repeat (2) for the impulse response sequence $h[n]= (0.81)^n [u(n)-u(n-101)]$ and plot the DTFT of h[n] in figure(3). If such a system has an sinusoidal input of $x[n]= \cos(0.2\pi n)$, what would be the output based on the plot of $H(e^{j\omega})$? What is the output's delay in terms of number of samples?

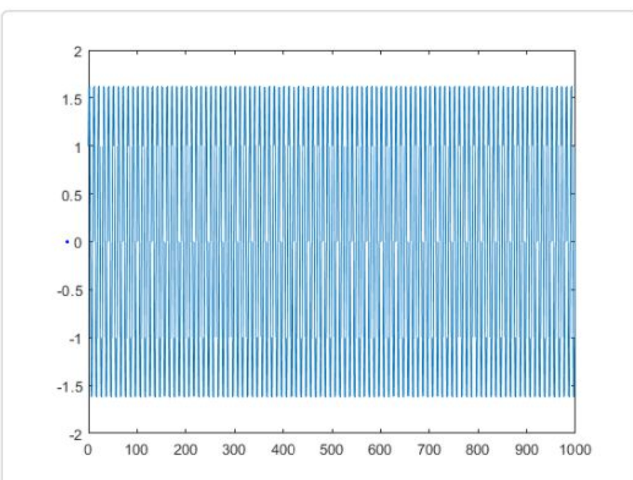
Theory Used

DTFT

Solution

```
n = 0:1000;

u = n >= 0;

u_101 = n >= 101;

h = (u-u_101).*(0.81).^n;

x = cos(0.2*pi*n);

y = conv(x, h);

y = y(1:length(x));

figure()

plot(n,y);
```

Results



Part 5

Subpart 5.1a

Problem Description

For each Ts, plot the resulting $x_k(n)$ for k=1,2,3 using a 3x1 stem subplot.

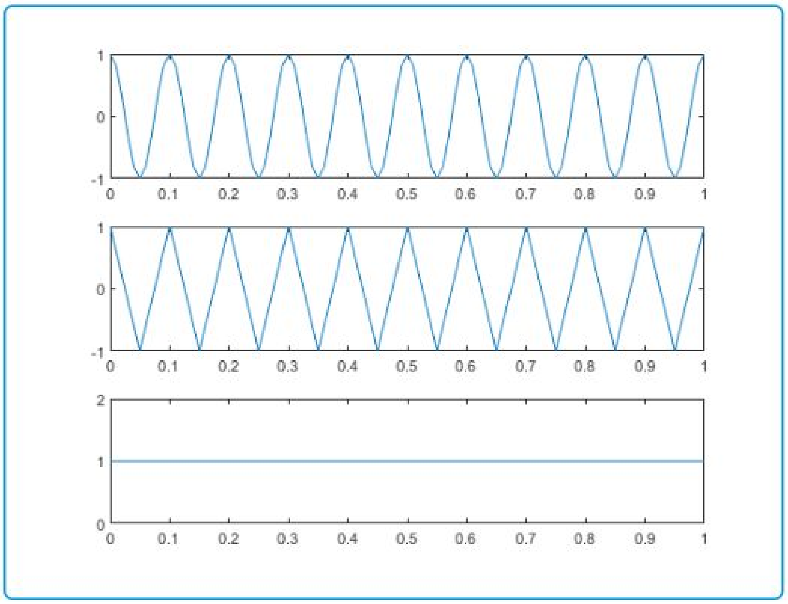
Theory Used

N/A

Solution

```
figure()
t1 = 0:1/100:1;
t2 = 0:0.05:1;
t3 = 0:0.1:1;

x_1 = cos(20*pi*t1);
x_2 = cos(20*pi*t2);
x_3 = cos(20*pi*t3);

subplot(3,1,1);
plot(t1, x_1)
subplot(3,1,2);
plot(t2, x_2)
subplot(3,1,3);
plot(t3, x_3)

% frequency is obviously 10hz
```

Results



Subpart 5.1b

Problem Description

Reconstruct the analog signal $y_c(t)$ from the samples x(n) using the ***zero-order hold*** interpolation. See page 90. Using subplot, plot $y_c(t)$ in each case. Determine the frequency of $y_c(t)$ from the your plots, ignoring end effects. An analog post-filter would be used to smooth the corners of the first staircase signal to yield a sine-like waveform.

Theory Used

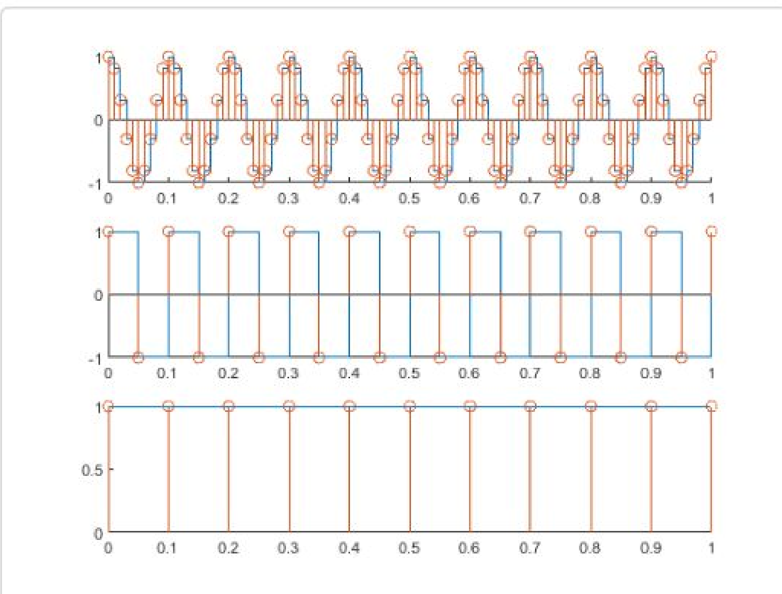
Zero-Order Hold

Solution

```
figure()
subplot(3,1,1);
hold on
stairs(t1, x_1);
stem(t1, x_1);
hold off
```

```
subplot(3,1,2);
hold on
stairs(t2, x_2);
stem(t2, x_2);
hold off
subplot(3,1,3);
hold on
stairs(t3, x_3);
stem(t3, x_3);
hold off
% frequency is obviously 10hz
```

Results



Subpart 5.1.c

Problem Description

Reconstruct the analog signal $y_c(t)$ from the samples x(n) using the ***first-order hold*** interpolation. See page 91. Using subplot, plot $y_c(t)$ in each case. The plotting function connects adjacent points with a straight line. Determine the frequency of $y_c(t)$ from the your plots, ignoring end effects. An analog post-filter would be used to smooth the corners of the straight-line plots of the signal to yield a smoother sine-like waveform.

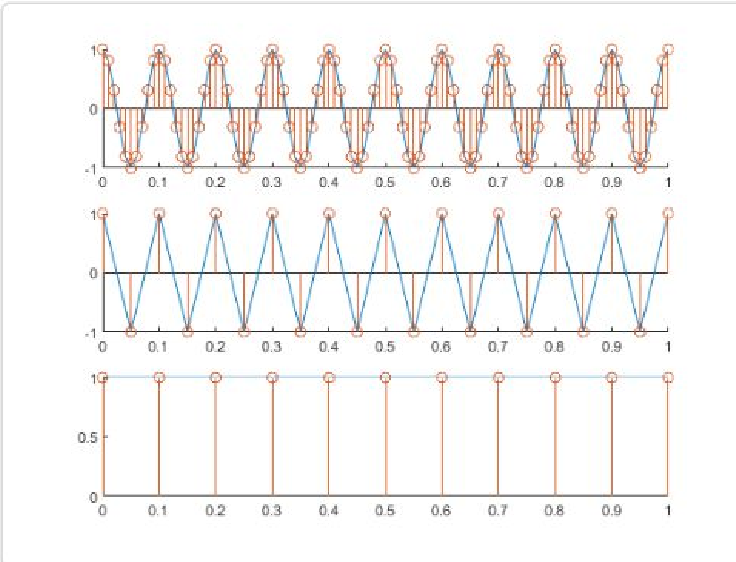
Theory Used

First-Order Hold

Solution

```
figure()


subplot(3,1,1);
hold on
plot(t1, x_1);
stem(t1, x_1);
hold off
subplot(3,1,2);
hold on
plot(t2, x_2);
stem(t2, x_2);
hold off
subplot(3,1,3);
hold on
plot(t3, x_3);
stem(t3, x_3);
hold off


% again, frequency is obviously 10hz
```

Results



Subpart 5.1.d

Problem Description

Reconstruct the analog signal $y_c(t)$ from the samples x(n) using the ***cubic spline*** interpolation. See page 96. Using subplot, plot $y_c(t)$ in each case. Determine the frequency of $y_c(t)$ from the your plots, ignoring end effects.

Theory

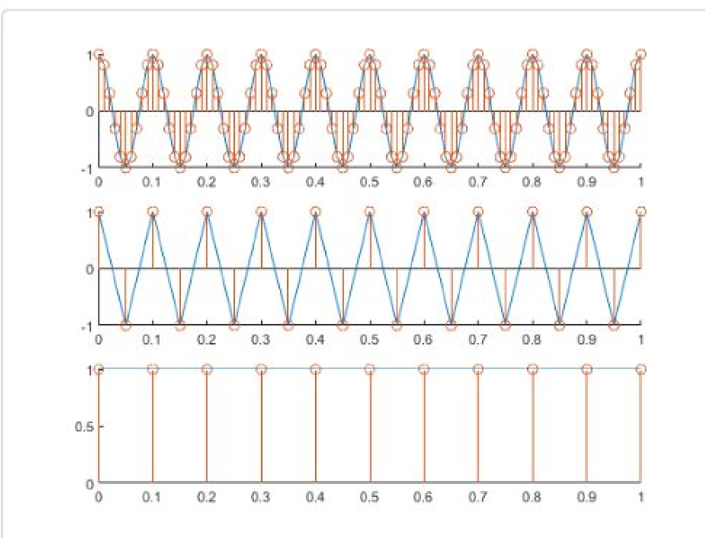
Cubic Spline

Solution

```
figure()


t = 0:0.01:1;

xa = cos(20*pi*t);

x_1_spl = spline(t1,x_1, t1);

x_2_spl = spline(t2,x_2, t2);

x_3_spl = spline(t3,x_3, t3);


subplot(3,1,1);
```

```
hold on
plot(t1,x_1);
stem(t1,x_1_spl);
hold off
subplot(3,1,2);
hold on
plot(t2,x_2);
stem(t2,x_2_spl);
hold off
subplot(3,1,3);
hold on
plot(t3,x_3);
stem(t3,x_3_spl);
hold off
% again, frequency is obviously 10hz
```

Results



Subpart 5.1e

Problem Description

Reconstruct the analog signal $y_s(t)$ from each set of samples $x_s(n)$ using the *sinc* interpolation ($\Delta t=0.001$). See page 93 of the text. Using a 3x1 subplot, plot $y_s(t)$ in each case. Determine the frequency of $y_s(t)$ from your plots, ignoring end effects.

Theory Used

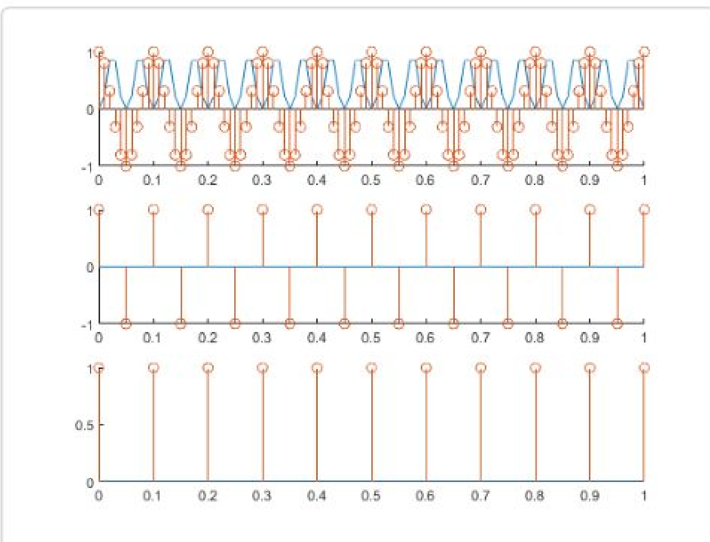
Sinc interpolation

Solution

```
figure()

y_1 = sinc(x_1);
y_2 = sinc(x_2);
y_3 = sinc(x_3);

subplot(3,1,1);
hold on
plot(t1,y_1);
stem(t1,x_1);
hold off

subplot(3,1,2);
hold on
plot(t2,y_2);
stem(t2,x_2_spl);
hold off

subplot(3,1,3);
```

```
hold on
plot(t3,y_3);
stem(t3,x_3);
hold off

% again, frequency is obviously 10hz
```

Results



Subpart 5.1f

Comments

%It appears that cubic spline and first order hold produce very similar
%results. Zero order hold creates a stair pattern. Since is odd, with the
%signals x2 and x3 creating a straight line.

Encountered Problems:

Most of the problems that we have encountered came from our lack of knowledge of matlab. other than that, some of it came from converting phasors into a straight line, as we have never done this before and had trouble figuring out how it was done. Really the biggest issue for us was the amount of time that this project required. Although it was not that difficult, the problems took a large amount of time to implement and to solve. Each part of the project took roughly 2-3 hours. An issue with matlab that arose is that sometimes in order for us to rerun the .mlx, we have to clear the workspace, else it produces an error with the graph.