Louis Rosenblum
Final Exam Part B
Page 1 of 14

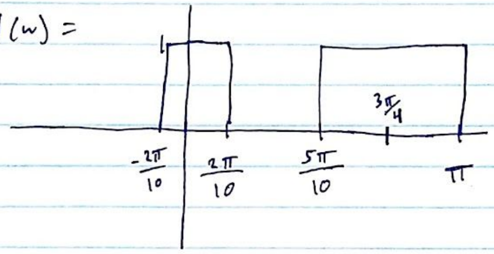
Problem 1)
1a)

1) FIR band-stop $L=8$ $F_s = 16\text{Khz}$

a) stop-band: 1600-4000 hz

$$w = \frac{2\pi F}{F_s} \rightarrow \frac{2\pi}{10} \quad \frac{5\pi}{10}$$

$H(w) =$



$$h(n) = \frac{1}{2\pi}\int e^{jwn}\,dw = \frac{1}{2\pi}\left(\int_{-\pi/5}^{\pi/5} e^{jwn}\,dw + \int_{\pi/2}^{\pi} e^{jwn}\,dw\right)$$

$$= \frac{1}{2\pi}\left(\frac{e^{jwn}}{jn}\Big|_{-\pi/5}^{\pi/5} + \frac{e^{jwn}}{jn}\Big|_{\pi/2}^{\pi}\right)$$

$$= \frac{1}{2\pi}\left(\frac{e^{j\pi/5 n}}{jn} - \frac{e^{-j\pi/5 n}}{jn} + \frac{e^{j\pi n}}{jn} - \frac{e^{j\pi/2 n}}{jn}\right)$$

$$= \frac{1}{\cancel{2}\pi}\left(\frac{\cancel{2}\sin(\pi/5\, n)}{n} + \frac{\cancel{2}\sin(\pi/2\, n)}{n} e^{j3\pi/4 n}\right)$$

$$= \frac{1}{\pi}\cdot\left(\text{sinc}(\pi/5\, n) + e^{j3\pi/4 n}\,\text{sinc}(\pi/2\, n)\right)$$

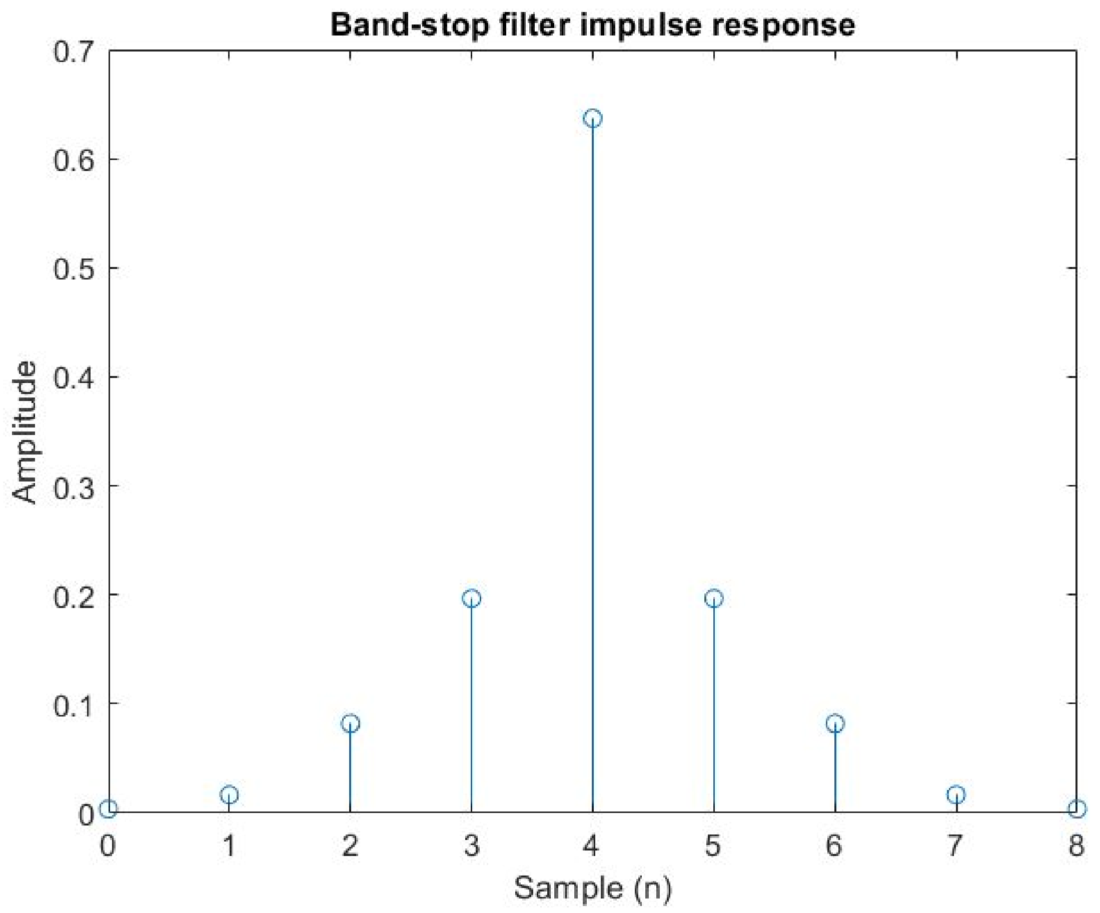
$n = -4$ to 4

hamming window: ~~[0.08 0.25 0.64 0.95 0.95 0.64 0.25 0.08]~~
[0.08 0.21 0.54 0.86 1 0.86 0.54 0.21 0.08]

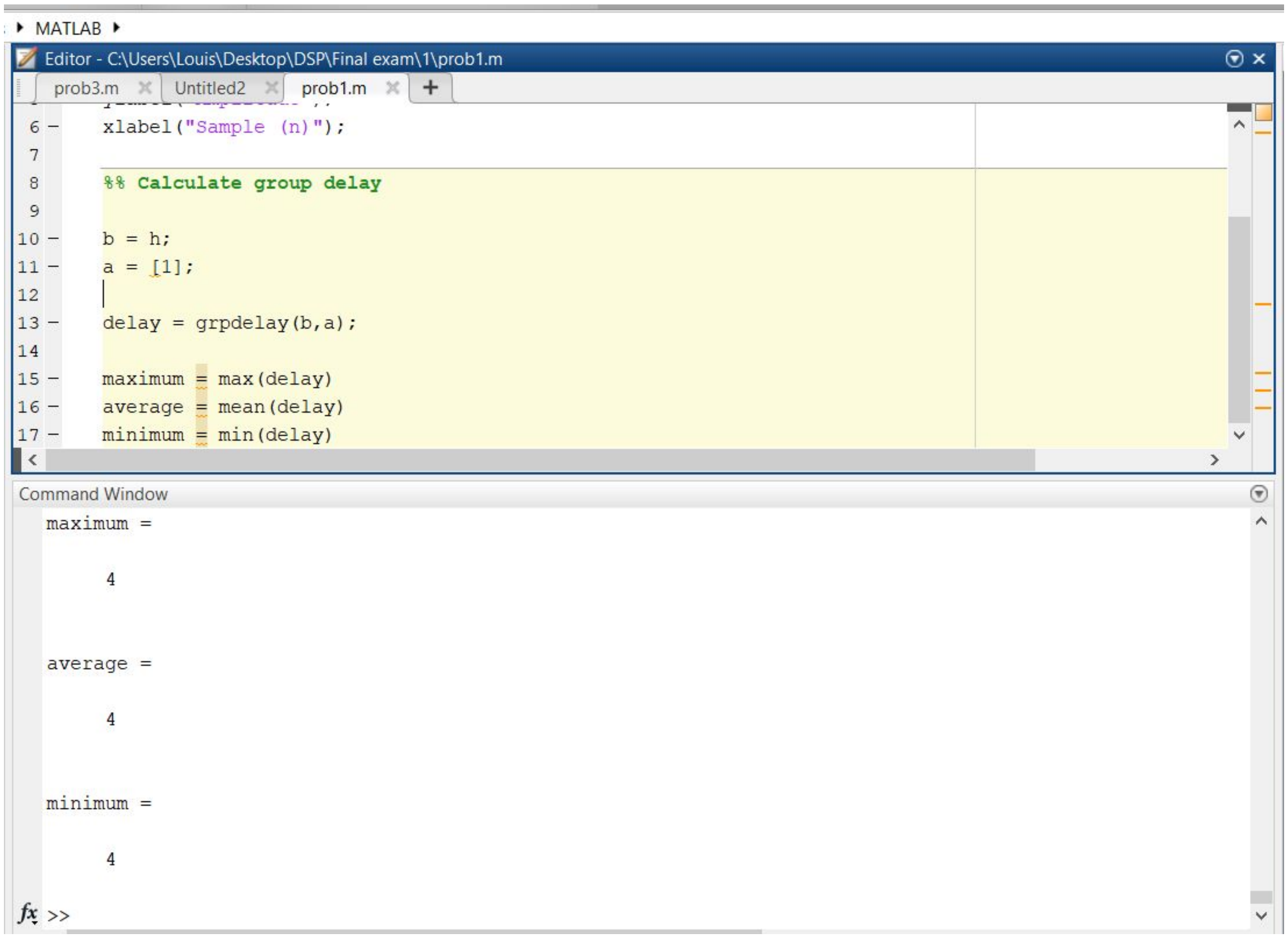
$h[n] = [0.0468 \quad (0.0259+0.075j) \quad 0.1514 \quad (-0.038-0.2251j)$
$2/\pi \quad (-0.038+0.2251j) \quad 0.1514 \quad (0.0259-0.075j) \quad 0.0468]$

$h[n] = [0.0037 \quad (0.0054+0.0158j) \quad 0.082 \quad (-0.033-0.194j)$
$0.637 \quad (-0.033+0.194j) \quad 0.082 \quad (0.0054-0.0158j) \quad 0.0037]$

1b)


Band-stop filter impulse response
Amplitude
Sample (n)
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
1
2
3
4
5
6
7
8

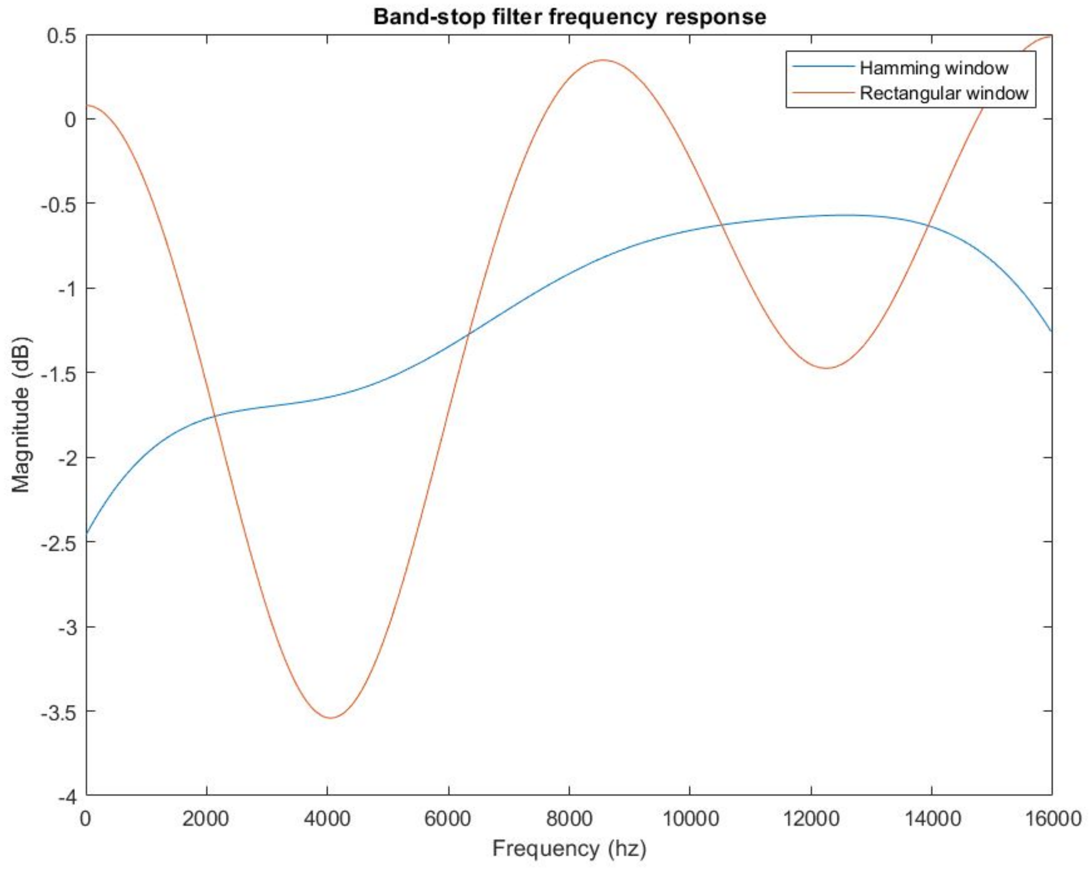
1c)



```
xlabel("Sample (n)");

%% Calculate group delay

b = h;
a = [1];

delay = grpdelay(b,a);

maximum = max(delay)
average = mean(delay)
minimum = min(delay)
```

```
maximum =

     4


average =

     4


minimum =

     4

>>
```

1d)


Band-stop filter frequency response
Hamming window
Rectangular window
Magnitude (dB)
Frequency (hz)
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
-3.5
-4
0
2000
4000
6000
8000
10000
12000
14000
16000

MATLAB Code

```
h1 = [0.0037 (0.0054+0.0158*j) (0.082) (-0.033-0.194*j) 0.637 (-0.033+0.194*j)
0.082 (0.0054 - 0.0158*j) 0.0037];
h2 = [0.0468 (0.0259 + 0.075*j) 0.1514 (-0.038 - 0.2251*j) 0.637 (-0.038 +
0.2251*j) 0.1514 (0.0259-0.075*j) 0.0468];

x = 0:8;
stem(x,abs(h))
title("Band-stop filter impulse response");
ylabel("Amplitude");
xlabel("Sample (n)");

%% Calculate group delay

b = h1;
a = [1];

delay = grpdelay(b,a);

maximum = max(delay)
average = mean(delay)
minimum = min(delay)

%% Plot hamming versus rectangular window

figure()

b = h1;
a = [1];

[h,w] = freqz(b,a);
w = w.*16000;

plot(w/pi,20*log10(abs(h)))
xlabel('Frequency (hz)')
ylabel('Magnitude (dB)')
hold on;

b = h2;
a = [1];

[h,w] = freqz(b,a);
w = w.*16000;
plot(w/pi,20*log10(abs(h)))

title("Band-stop filter frequency response")
```

```
legend("Hamming window","Rectangular window")
```

Problem 2)

2) $y(n) = 0.1\, x(n) - 0.9 y(n-1)$

$h(n) = 0.1\, d(n) - 0.9 h(n-1)$

n	h
0	0.1
1	-0.09
2	0.081
3	-0.0729
4	0.06561

$y(n) = 0.1 \; (-0.9)^{n} \; u(n)$

$$H(z) = \frac{0.1}{1 + 0.9 z^{-1}}$$

a) $r_{yy}(m) = 0.1 \sum_{k=0}^{\infty} \sum_{i=0}^{\infty} (-0.9)^{k+i} \, d(m-k+i)$

$r_{yy}(m) = 0.1 \sum_{i=0}^{\infty} (-0.9)^{m+2i} = 0.1 (-0.9)^{m} \sum_{i=0}^{\infty} (0.81)^{i}$, $m \geq 0$

$r_{yy}(m) = r_{yy}(-m) = 0.1 \cdot (-0.9)^{|m|} \cdot 5.26$

$\boxed{r_{yy}(m) = 0.526 \; (-0.9)^{|m|}}$

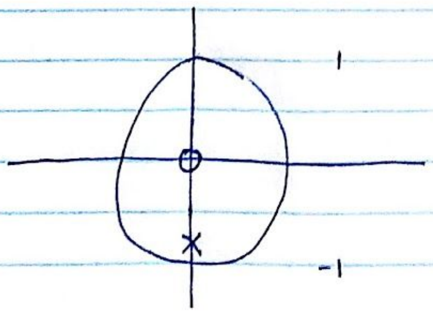
b) $\sigma_y^2 = \sum_{k=0}^{\infty} h^2(k) \sigma_x^2 = \sum_{k=0}^{\infty} (0.81)^k = \frac{1}{1-0.81} = 5.26$

$\boxed{\sigma_y^2 = 5.26}$

c) $R_{yy}(e^{jw}) = \frac{1}{|1 + 0.9 e^{-jw}|^2}$ $\qquad \frac{1}{|1 + 0.9 z^{-1}|^2} \cdot \frac{z^2}{z^2}$

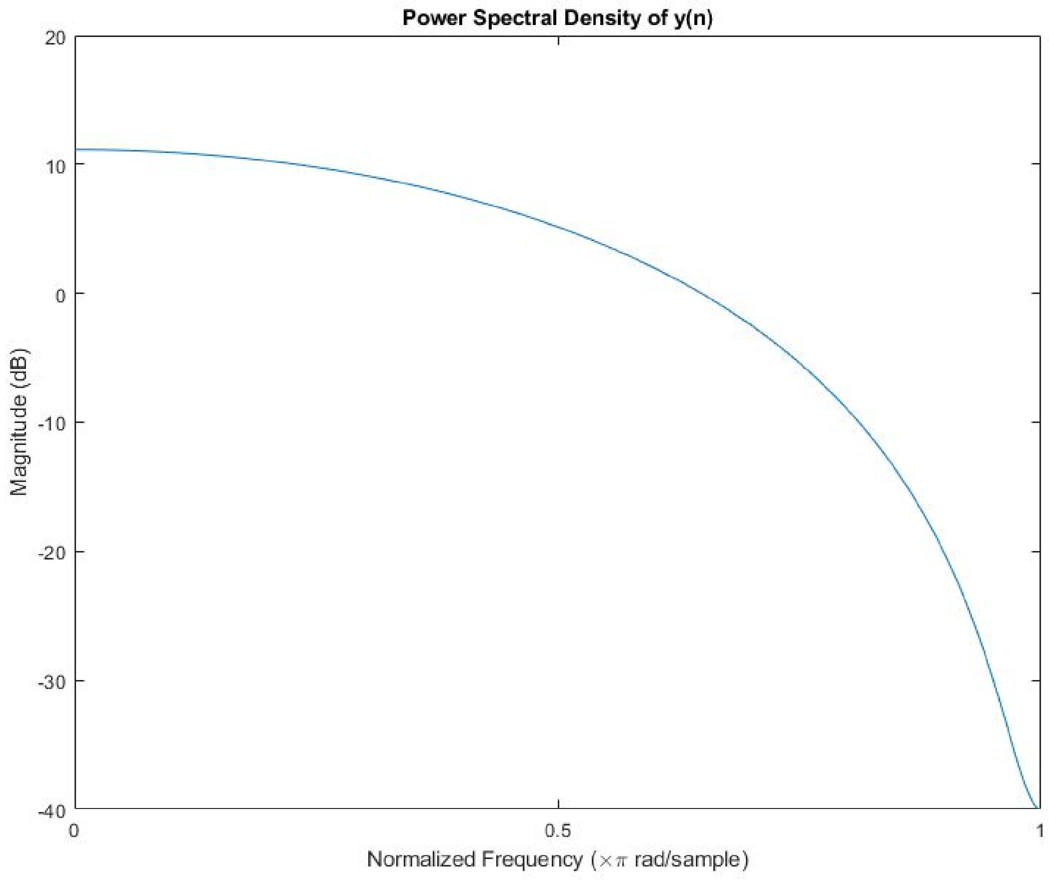
$$R_{yy}(e^{jw}) = \frac{z^2}{(z+0.9)(z+0.9)}$$

zeros: $z=0$, $z=0$ $\qquad$ poles: $z=-0.9$, $z=-0.9$



O	two zeros
X	two poles

2d)

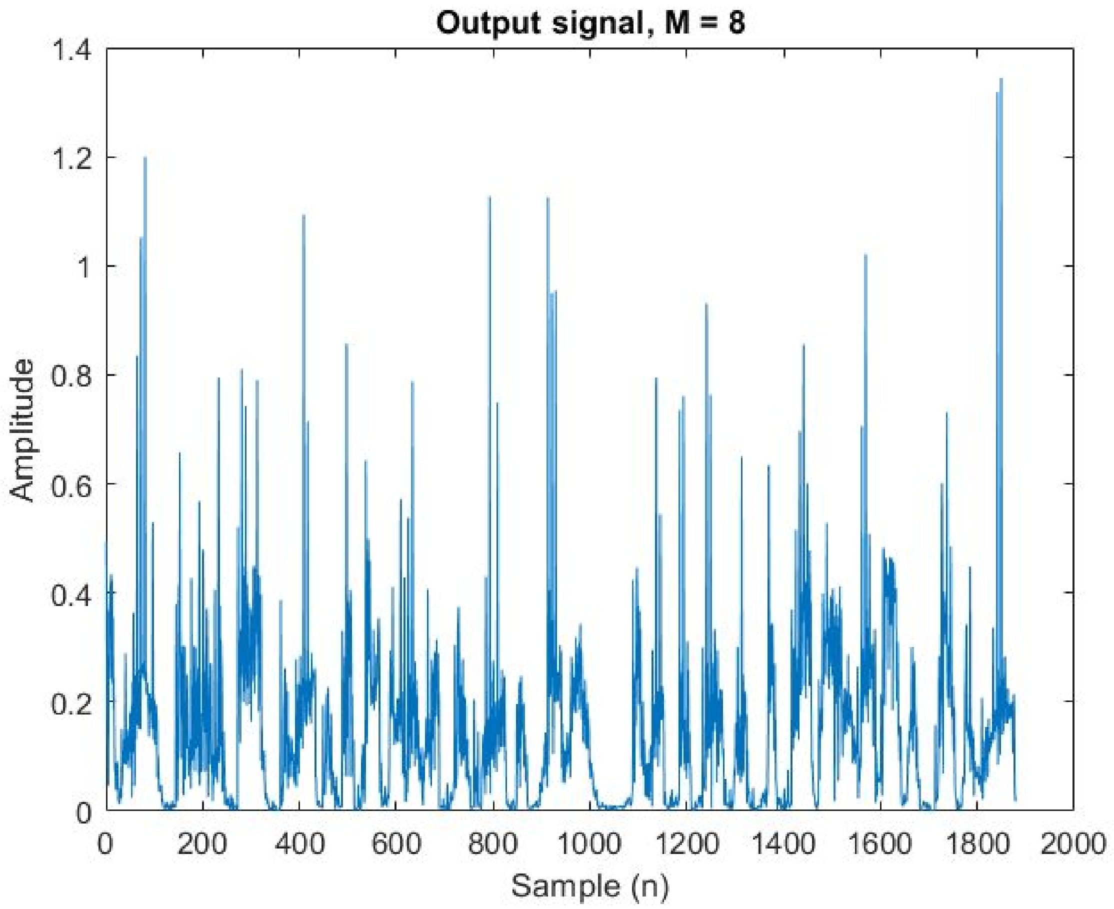


MATLAB code:

```
% Louis Rosenblum
% EEE509 - ASU
% 06/25/2020
% Final exam part B

%% Plot hamming versus rectangular window

figure()

b = [1 0.9];
a = [1];

[h,w] = freqz(b,a);

plot(w/pi,20*log10(abs(h).^2))
ax = gca;
```

```
ax.XTick = 0:.5:2;
title("Power Spectral Density of y(n)")
xlabel('Normalized Frequency (\times\pi rad/sample)')
ylabel('Magnitude (dB)')
```

Problem 3)



M	SNR (dB)	FFT Compute Time	DFT Compute Time
8	-3.71	0.296 s	2.22 s
32	-5.56	0.302 s	2.61 s
128	-7.06	0.307 s	8.17 s

Remarks:

The SNR is changing alongside different values of M because of the spectral components that are selected from the fourier transform. Of the 256-point fourier transform only the first M frequency domain coefficients are considered and the rest are removed. The reason including more spectral components (higher M) is lowering the signal to noise ratio is because the lower frequencies of the transform contain elements of human speech. Some of the higher frequencies are just noise components, so keeping them inside the signal leads to more noise being present after performing an inverse fourier transform.

The difference in CPU function between the FFT and DFT can be explained by the differences in the computations themselves. The FFT operation breaks the input data into a cascade of butterfly operations and quickly makes efficient approximations using parallel processing. Meanwhile the DFT operation is operating in the form of multiple nested for loops and this increases the compute time compared to vectorization. In addition the DFT is also iterating through all values of k to create a more accurate representation which also increases compute time.

MATLAB Code:

```
% Louis Rosenblum
% EEE509-ASU
% 06/25/2020
% Final exam part B

%% Problem 3
% Initialization

clear all
close all

cd 'C:\Users\Louis\Desktop\DSP\Final exam'

% Use FFT?
use_fft = 0;

% Start timing

tic

%% Read audio from file

[input, Fs1] = audioread('New_clean_male.wav');

%% Create frames from vector

len = length(input);
nFrames = ceil(len/256);

[dataFrames, len1] = createFrames(input,nFrames);
```

```
%% Take FFT of data frames

N = 256;

if (use_fft)
    framesFFT = fft(dataFrames,N);
else
    framesFFT = dft(dataFrames);
end

%% Select first M components

M = 32;

framesFFT_reduced = framesFFT(1:M,:);

%% Take IFFT

if (use_fft)
    output_frames = ifft(framesFFT_reduced);
else
    output_frames = idft(framesFFT_reduced);
end



%% Convert frames to vector

output = createVector(output_frames);

%% Create plots

plot(input)

figure()

plot(abs(output))

%% Calculate SNR

snr_result = snr(abs(output))

%% Stop timing

toc

%% Function definitions
```

```
function output = dft(input)
    [rows,columns] = size(input);

    len = rows;

    output = zeros(rows,columns);

    for i = 1:columns
        for k = 0:len-1
            for n = 0:len-1
                output(k+1,i) = output(k+1,i) + input(n+1,i)*exp(-j*pi/2*n*k);
            end
        end
    end
end

function output = idft(input)
    [rows,columns] = size(input);

    len = rows;

    for i = 1:columns
        for k = 0:len-1
            for n = 0:len-1
                output(k+1,i) = (exp(j*2*pi*k'*n)/len) * input(k+1,i)/len;
            end
        end
    end
end

function [data, len] = createFrames(audio,nFrames)
    len = length(audio);
      frameSize = ceil(len/nFrames);
    total = frameSize*nFrames;
    z = total - len;
    pad = [audio;zeros(z,1)];
    data = reshape(pad,frameSize,nFrames);
end

function data = createVector(audio)
    [m,n] = size(audio);
    data = [];

    for i = 1:n
        data = [data rot90(audio(:,i))];
    end
```

```
end
```