SIGNALS PROJECT

Presented for ELC2030 Project
Presented to:
Dr. Michael Melek
T.A: Alaa Khairalla

TEAM MEMBERS (2nd Year Electronics and Electrical Communication Engineers)	
Magdy Ahmed Abbas Abdelhamid	Section: 3 / I.D: 9210899

Image Compression

Task 1

Explanation

- (a): first: we read the image file ("image1.bmp") using "imread" function. then we print the image using "imshow" function as shown in Fig. (a.1). second: we use the function "imsplit" to extract the three components of the image which are RGB components, then we display them in grayscale & RGB Scale using "imshow" as shown in from Fig. (a.2) to Fig (a.7) & each of them is labeled by its name using "title" function.

- (b): **first:** we define 2 functions one is DCT & the other is inverse DCT to call them when needed.
  second: we decompose each color component image into 8x8 blocks using "blockproc" function which at the same time it applies the above defined DCT function to each 8x8 block of each of the 3 components then store the coefficients of the basis images in matrices called R_DCT, G_DCT, B_DCT.
  (Each 8x8 block of them will have 64 coefficients)
  third: we define a temporary matrix of zeros of size (m x m) called "mask" which we add to it ones at specific positions at the top left according to the value of m, then we multiply it by the DCT coefficients of needed size at different values of $m = 1,2,3,4$ using a function called temporary & repeat this with each 8x8 block of each of the 3 components then store the result in matrices called R_NEW, G_NEW , B_NEW (each 8x8 block of them will contain coefficient matrix of size m x m in the top left of each block & the other cells outside size m x m will be zero) As Shown in Fig (b & c.8) to Fig (b & c.11).

- (c): As Shown in Fig (b & c.8) to Fig (b & c.11).
  Size of Compressed Image at $m = 1$ is 8 times smaller than the original image size.
  Size of Compressed Image at $m = 2$ is 4 times smaller than the original image size.
  Size of Compressed Image at $m = 3$ is $\frac{3}{8}$ times smaller than the original image size.
  Size of Compressed Image at $m = 4$ is 2 times smaller than the original image size.

- (d): **first:** to return to the original image we apply inverse DCT function on each 8x8 block of the matrices containing the coefficients which are R_NEW, G_NEW , B_NEW & store the result in R_INV, G_INV , B_INV , then we concatenate them using function "cat" and store the result in matrix called "decompression" which will contain the approximate 8x8 blocks of the original image at different values of m.
  second: we display the compressed image at $m = 1,2,3,4$ as shown in Fig (b & c.8) to Fig (b & c.11) & display the decompressed image at $m = 1,2,3,4$ as shown in Fig (d & c.12) to Fig (d & c.15).

- (e): to get PSNR we need first to calculate Mean square error (MSE) between the original image and the decompressed image which is obtained by subtracting the corresponding pixel values of two images and obtaining the average of the square of all the differences (the peak value is considered to be 255)

- (f): we plot the PSNR graph for each value of m as shown in Fig (e & f.16) & Fig (e & f.17) we notice that PSNR value increase with increasing value of m which means that the quality of image becomes better as m increase since we take greater number of DCT coefficients.

- Content In ZIP File in Task 1 (Image Compression): -
  - ✓ Figures: In this Folder you Will Find Figures From (Fig 1) to (Fig 17).
  - ✓ MATLAB.m Codes:
    In this Folder you Will Find .m file (image_compression. m)

- Content of This File Respectively:
  - ✓ One uncompressed pdf project report containing:
    a. Explanation of your work.
    b. All the required results and answers to questions.
    c. All the required figures. Label your figures properly.
    d. All the codes, included at the end.

QUESTIONS & ANSWERS

(a): Read the image file 'image1.bmp'.
Extract and display each one of its three-color components.
(Gray and RGB) Scales

Ans: -

- Plotting: As Shown in from Fig. (a.2) to Fig (a.7)

(b): Compress the image by applying 2D DCT to each block.
at $m = 1, 2, 3, 4$ and knowing its Sizes.

Ans: -

- Plotting: As Shown in from Fig (b & c.8) to Fig (b & c.11).

(c): Compare the size of the original and compressed images.

Ans: -

- Sizes: As Shown in from Fig (b & c.8) to Fig (b & c.11).

(d): Decompress the image by applying Inverse 2D DCT to each block.
at $m = 1, 2, 3, 4$ and knowing its Sizes.

Ans: -

- Plotting: As Shown in from Fig (d & c.12) to Fig (d & c.15).

(e): Obtain the PSNR for each value of m.

Ans: - As Shown in from Fig (e & f.16) to Fig (e & f.17).

m	1	2	3	4
PSNR	19.8635	21.9069	23.7734	25.9427

(f): Plot a curve displaying the PSNR (on the vertical axis) against m (on the horizontal axis). Comment on the resulting graph and quality of images.

Ans: -

- As DCT coefficients increase, the quality of the image gets better. hence the PSNR value was expected to increase
  The higher the PSNR, the better the quality of the compressed, or reconstructed image.

Figures & Labels

Size:	5.93 MB (6,220,854 bytes)
Size on disk:	5.93 MB (6,221,824 bytes)

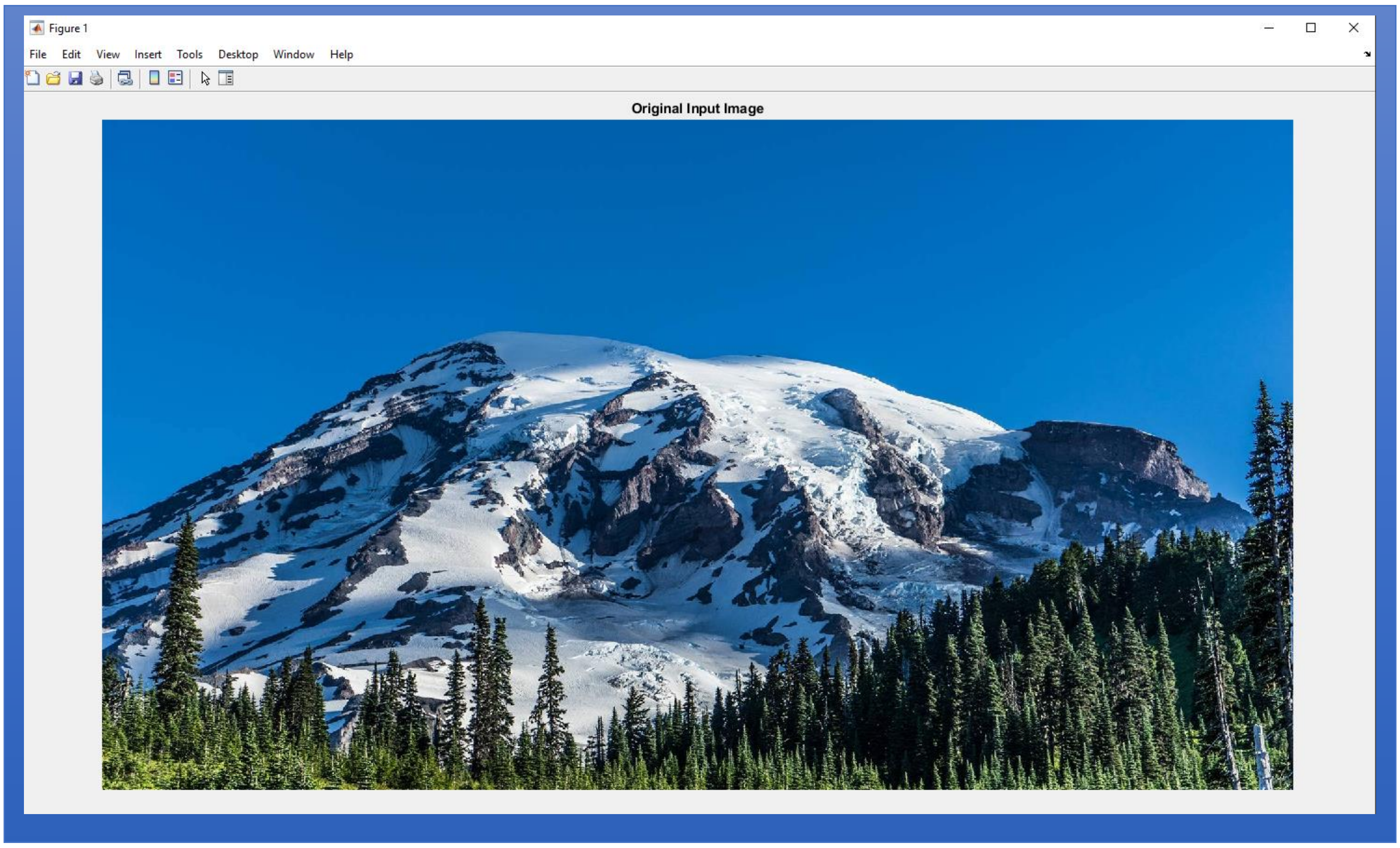


Fig (a & c.1): Orignal Image & its Size

Size:	1.97 MB (2,074,678 bytes)
Size on disk:	1.98 MB (2,076,672 bytes)

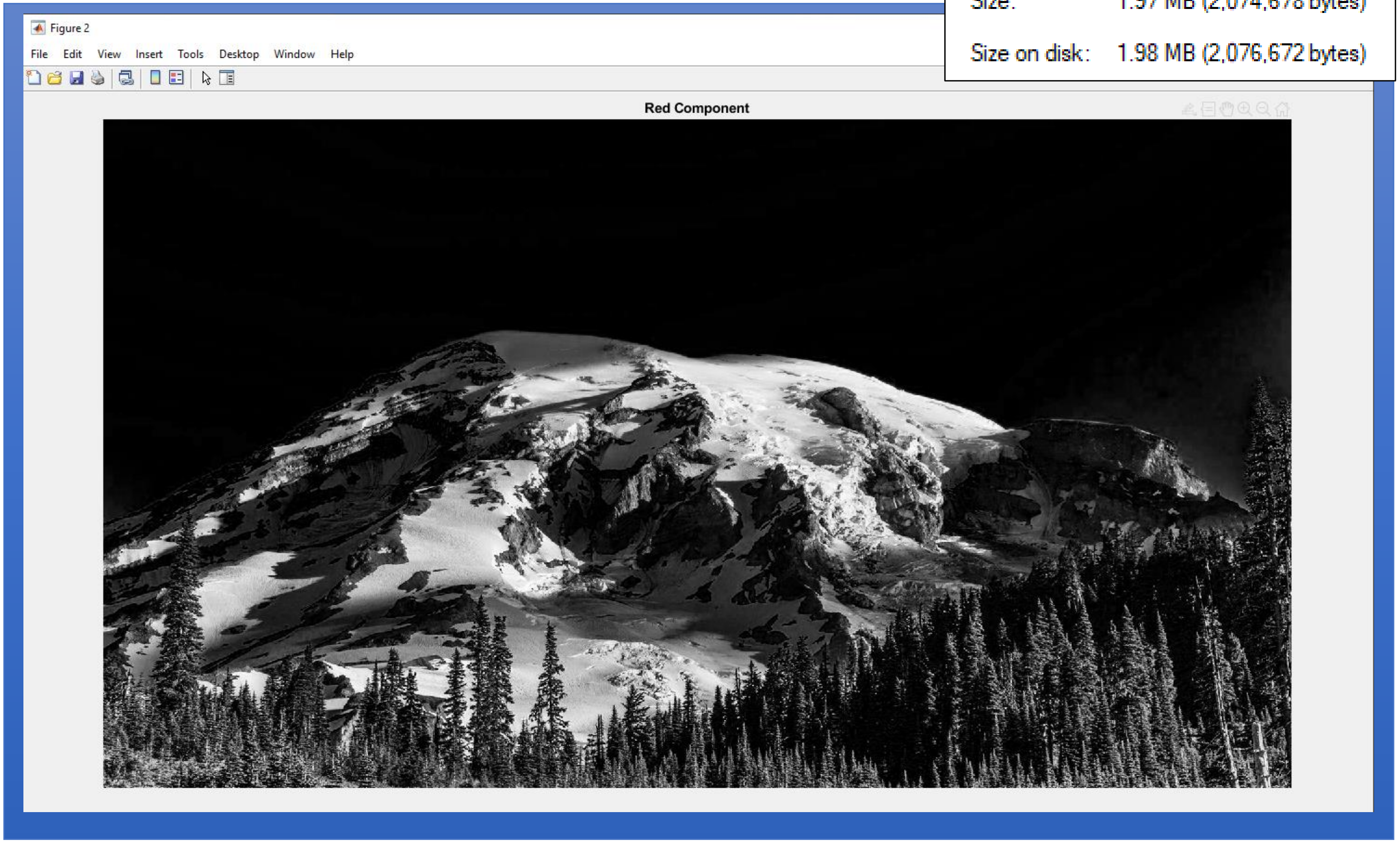
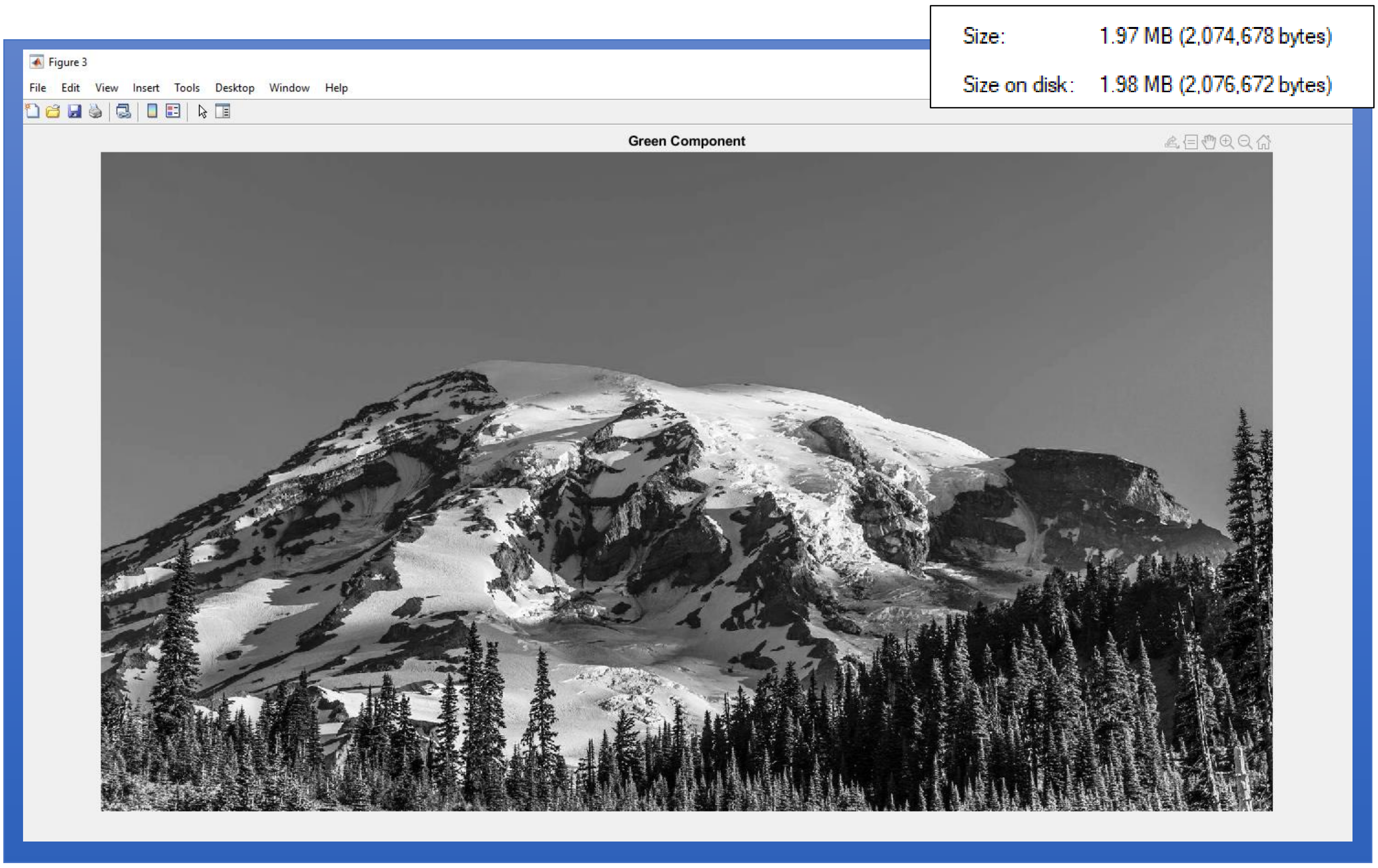
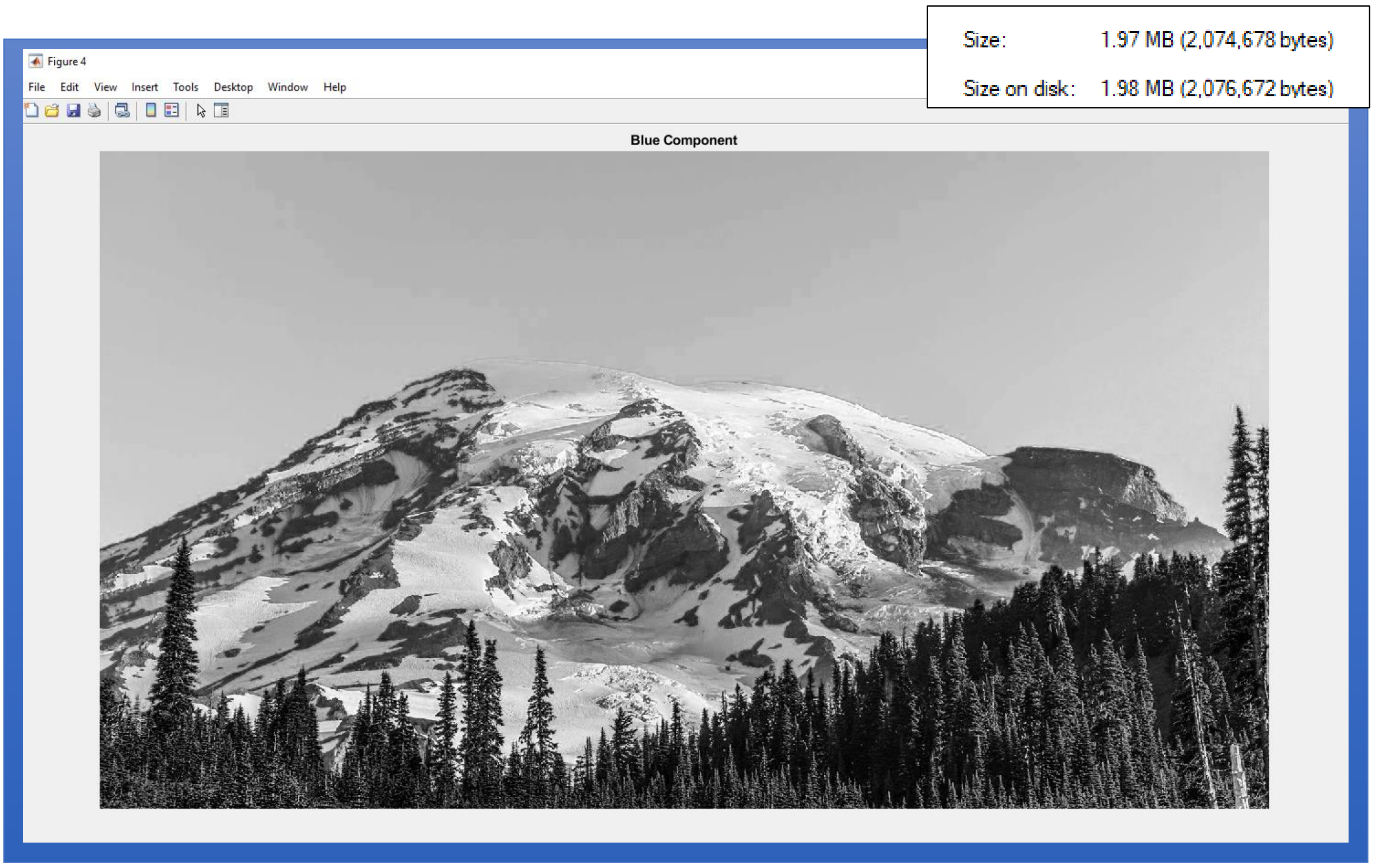


Fig (a.2): Red Component Image in Gray Scale



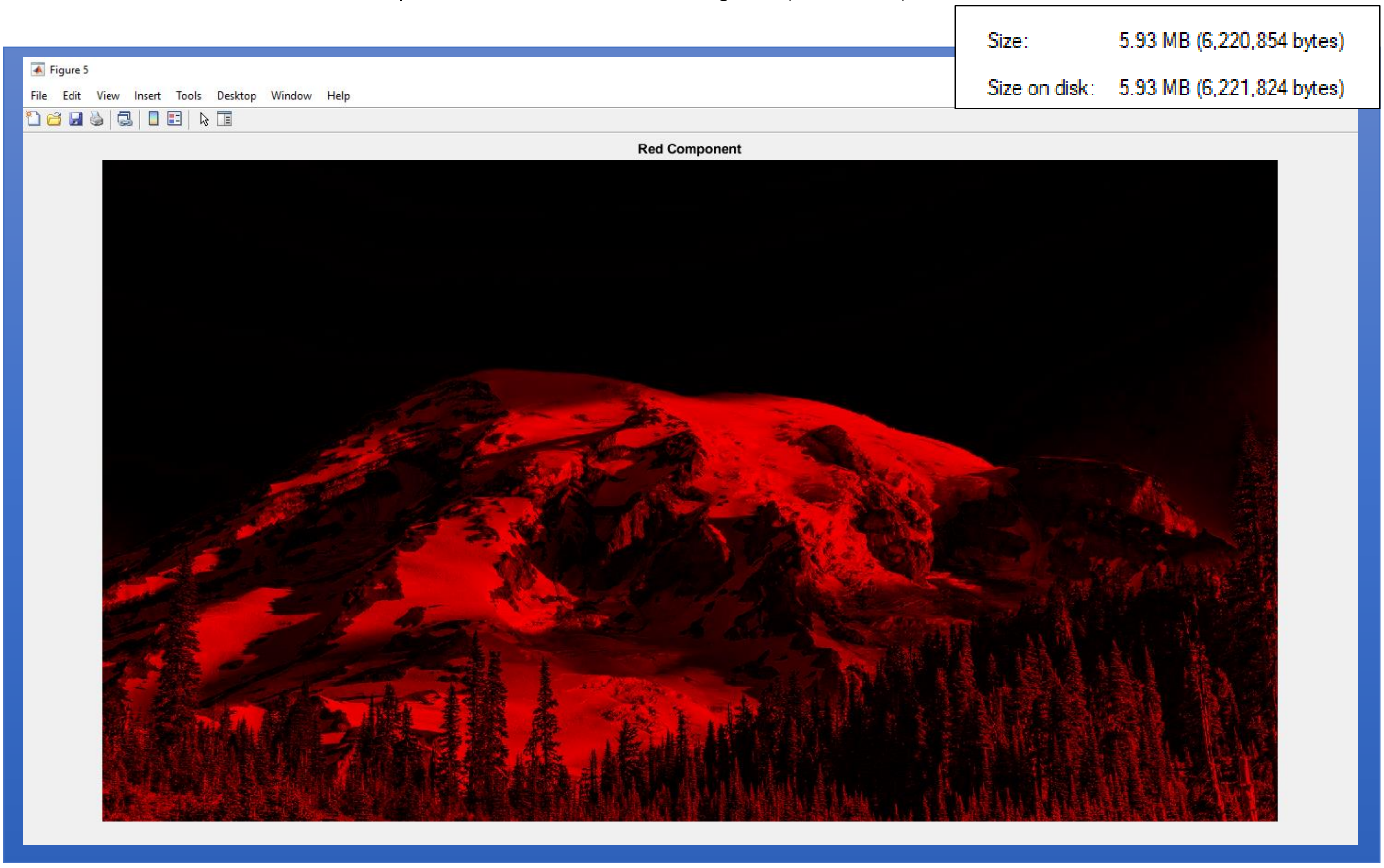
Size:	1.97 MB (2,074,678 bytes)
Size on disk:	1.98 MB (2,076,672 bytes)

Fig (a.3): Green Component Image in Gray Scale



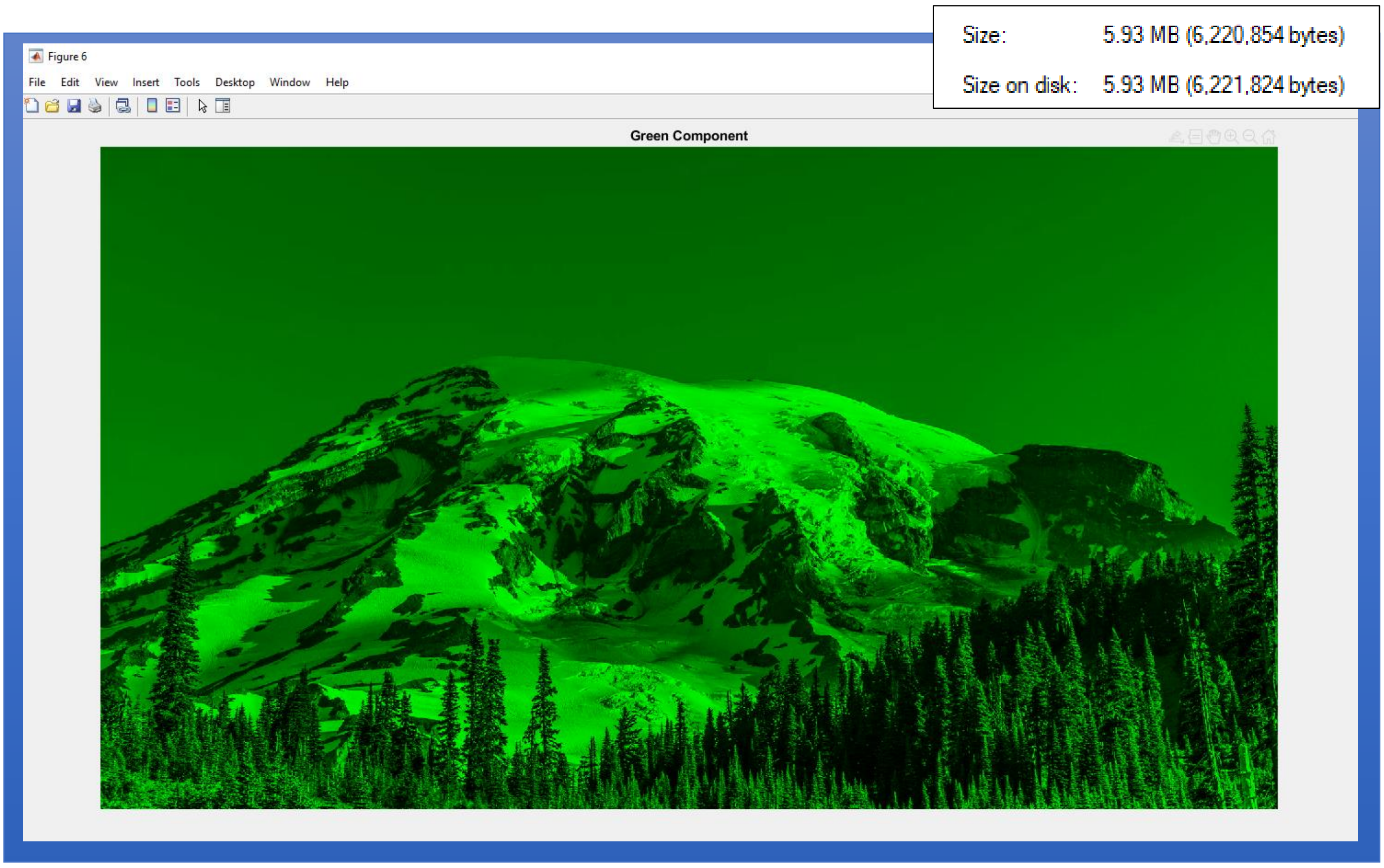
Size:	1.97 MB (2,074,678 bytes)
Size on disk:	1.98 MB (2,076,672 bytes)

Fig (a.4): Blue Component Image in Gray Scale



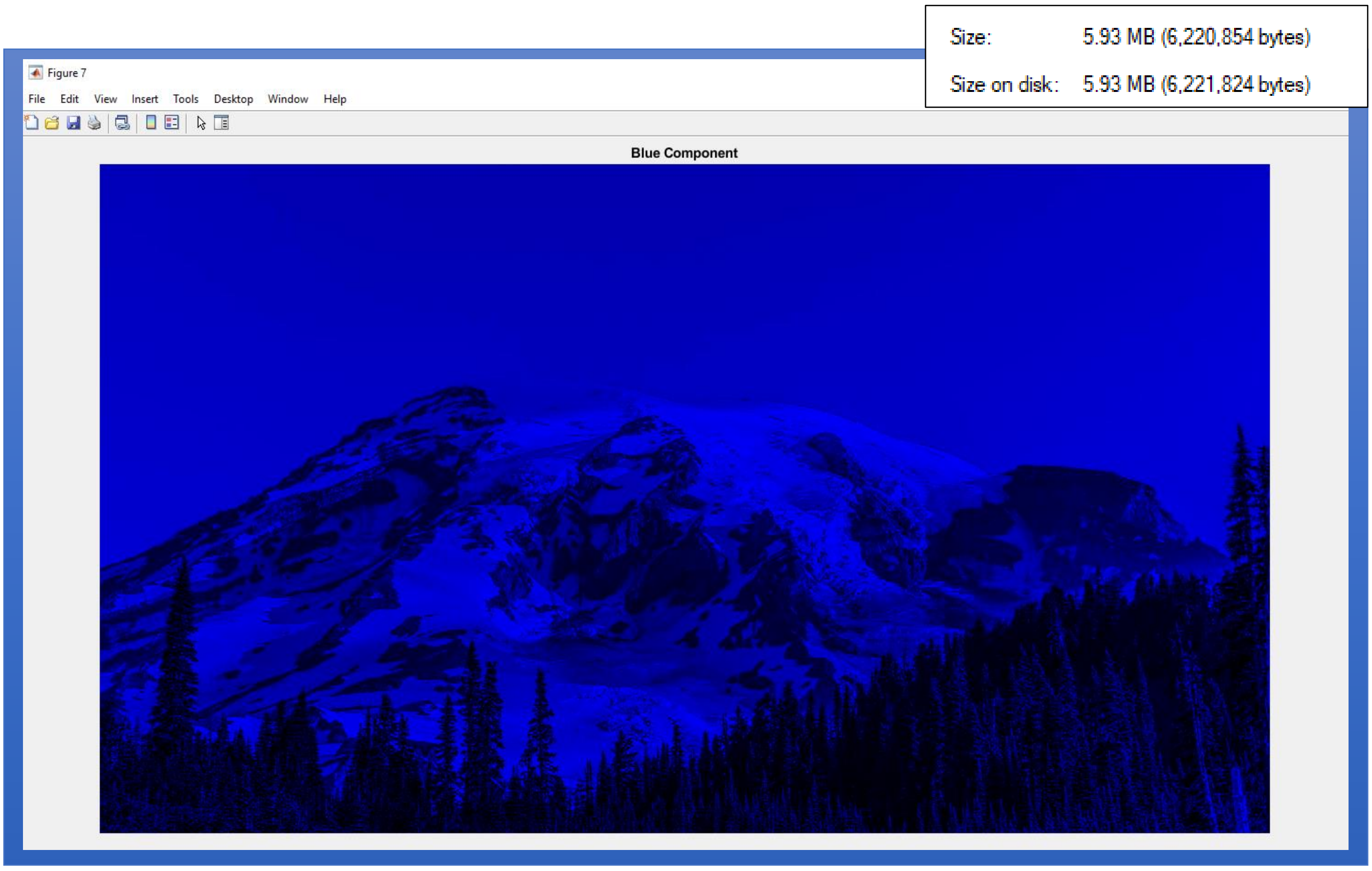
Size: 5.93 MB (6,220,854 bytes)
Size on disk: 5.93 MB (6,221,824 bytes)

Fig (a.5): Red Component Image in RGB Scale



Size: 5.93 MB (6,220,854 bytes)
Size on disk: 5.93 MB (6,221,824 bytes)

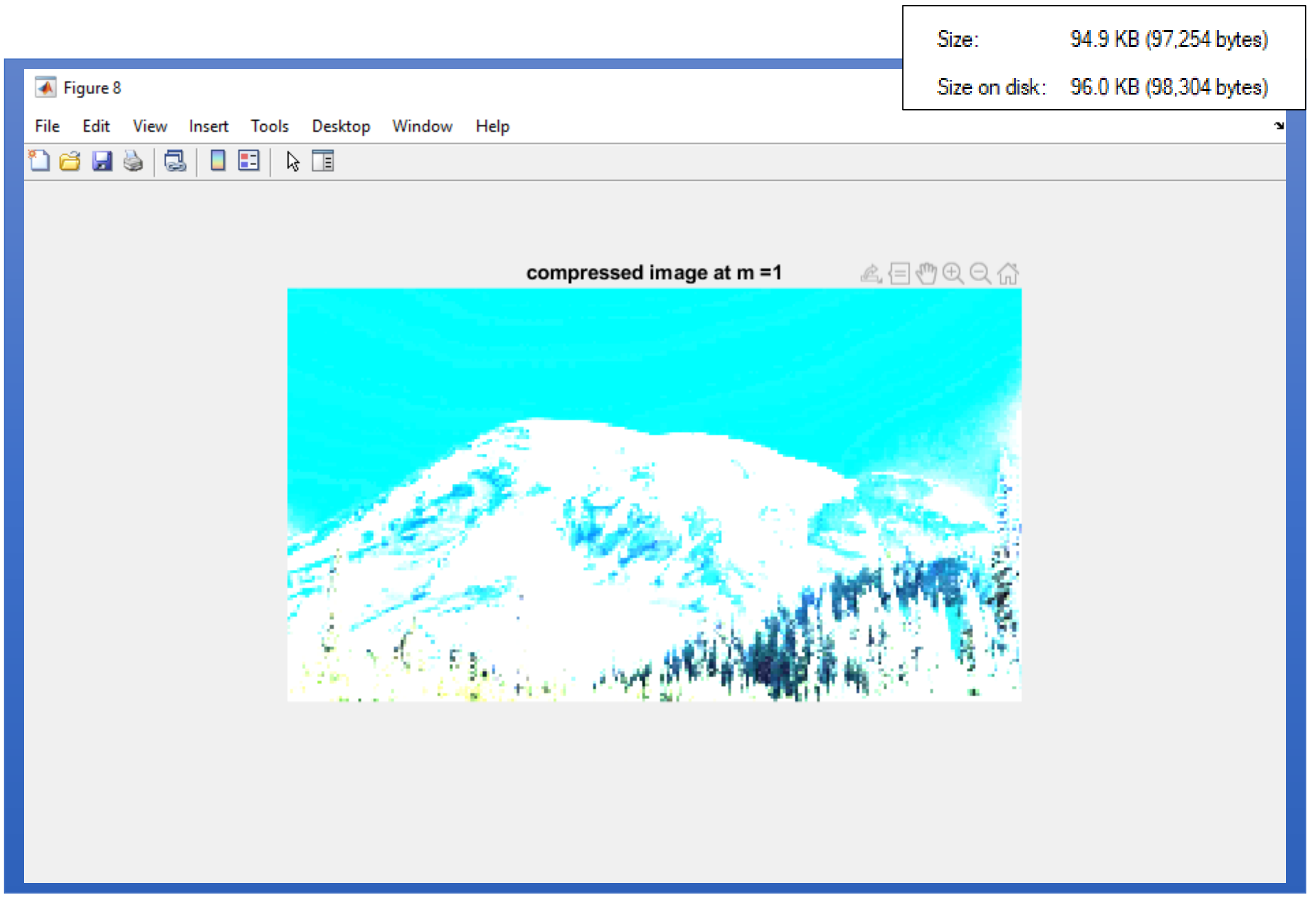
Fig (a.6): Green Component Image in RGB Scale



Size: 5.93 MB (6,220,854 bytes)

Size on disk: 5.93 MB (6,221,824 bytes)

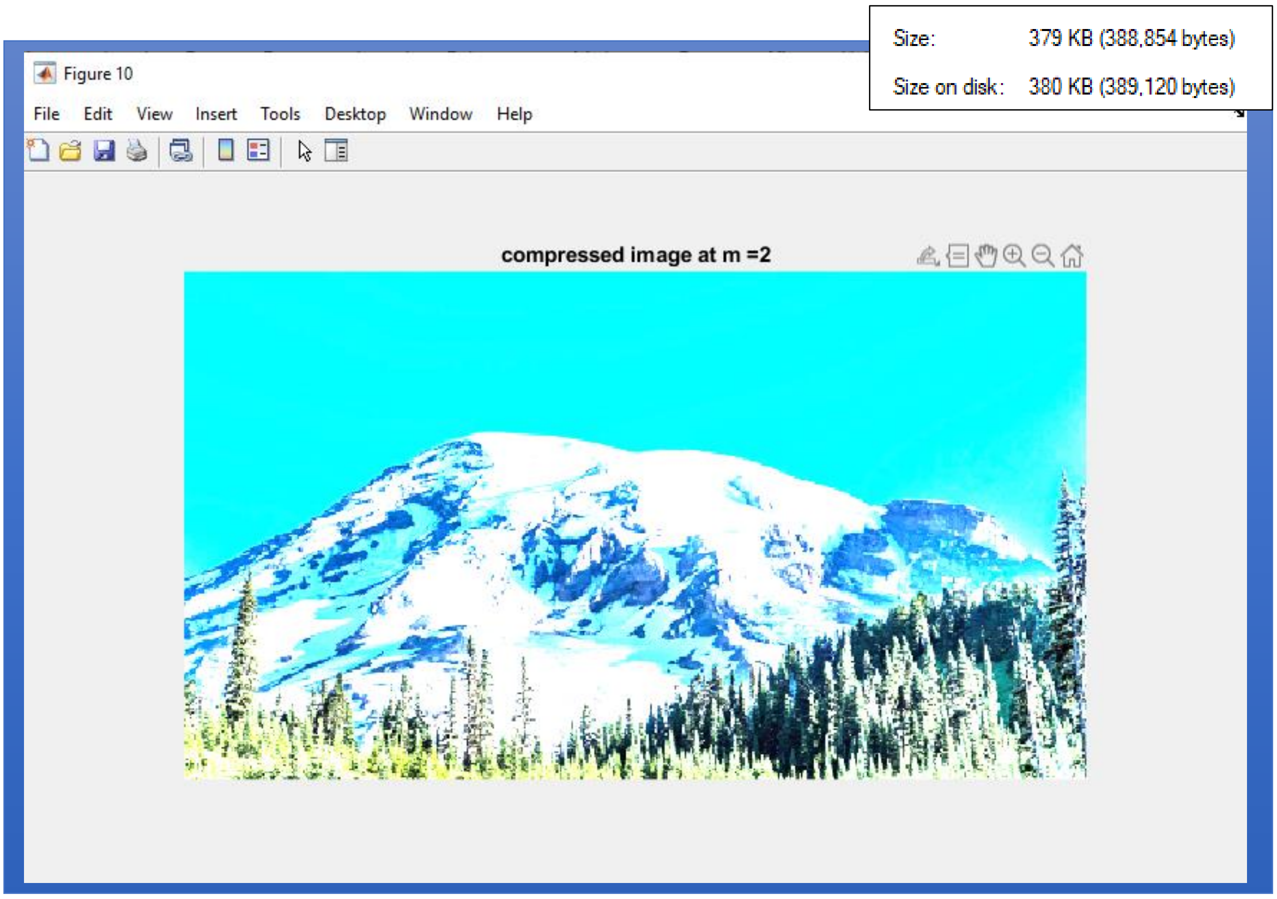
Fig (a.7): Blue Component Image in RGB Scale



Size: 94.9 KB (97,254 bytes)

Size on disk: 96.0 KB (98,304 bytes)

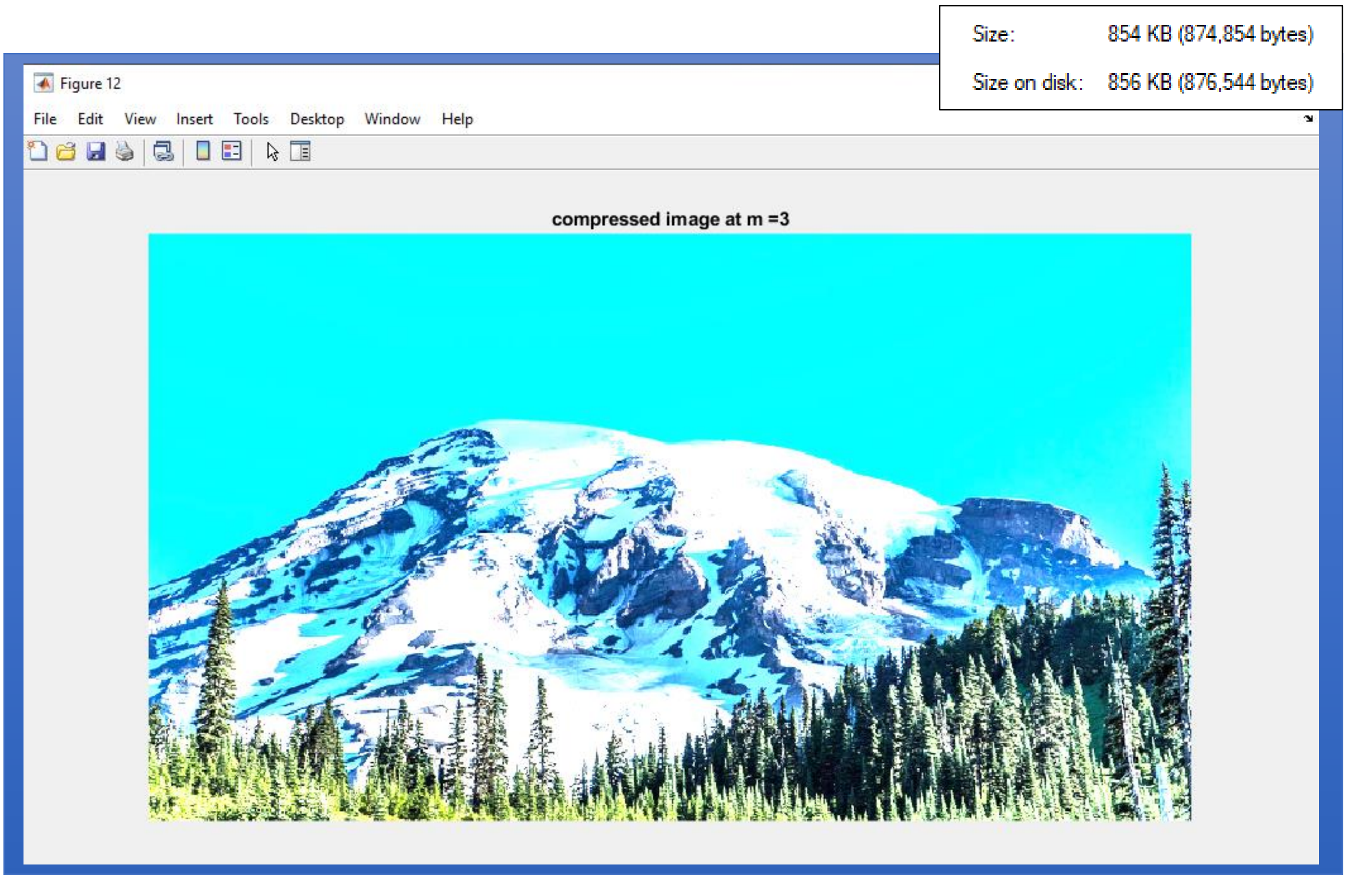
Fig (b & c.8): Compressed Image at m = 1 & its Size



Size: 379 KB (388,854 bytes)

Size on disk: 380 KB (389,120 bytes)

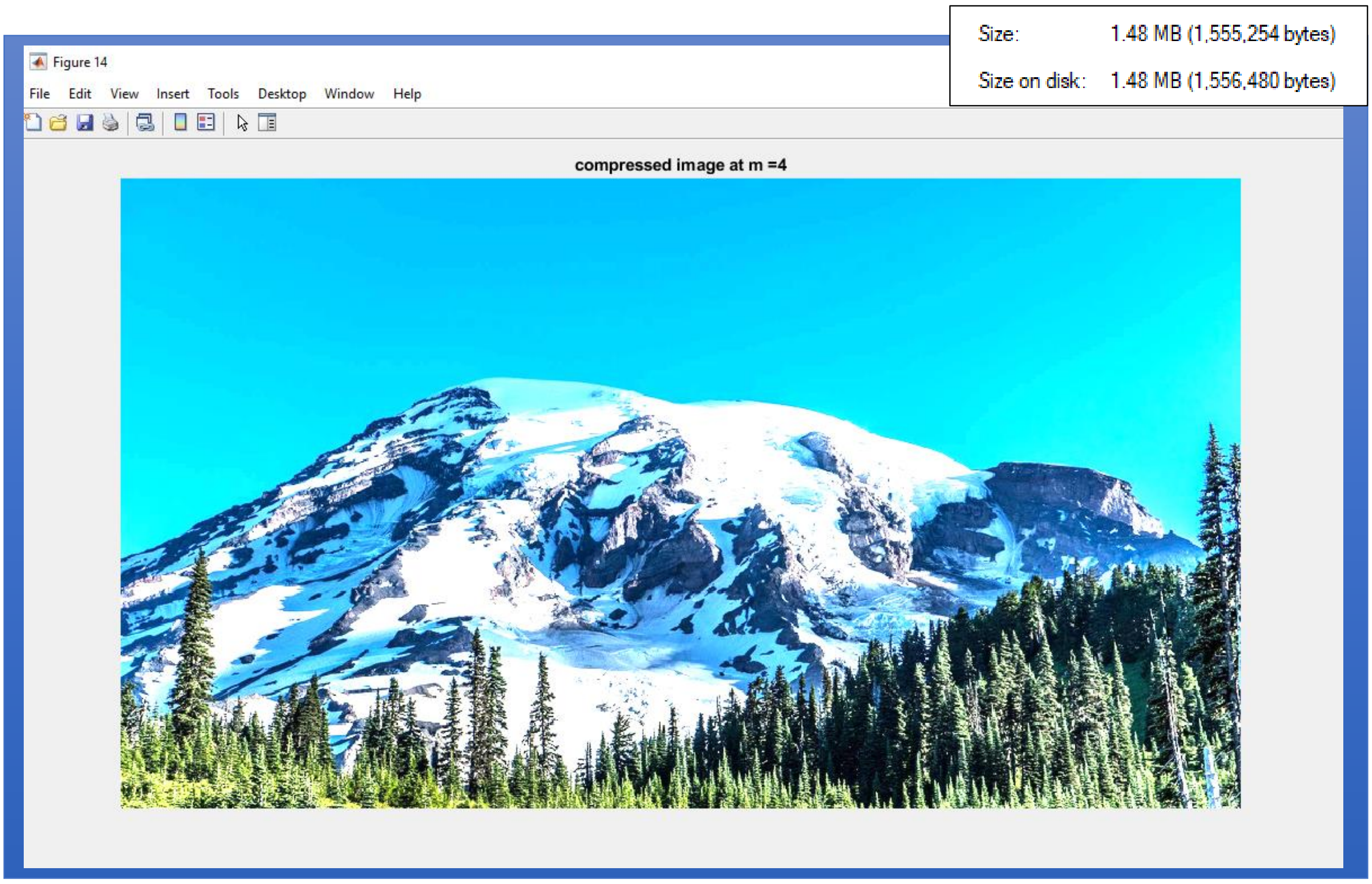
Fig (b & c.9): Compressed Image at m = 2 & its Size



Size: 854 KB (874,854 bytes)

Size on disk: 856 KB (876,544 bytes)

Fig (b & c.10): Compressed Image at m = 3 & its Size



Size: 1.48 MB (1,555,254 bytes)
Size on disk: 1.48 MB (1,556,480 bytes)

Fig (b & c.11): Compressed Image at m = 4 & its Size

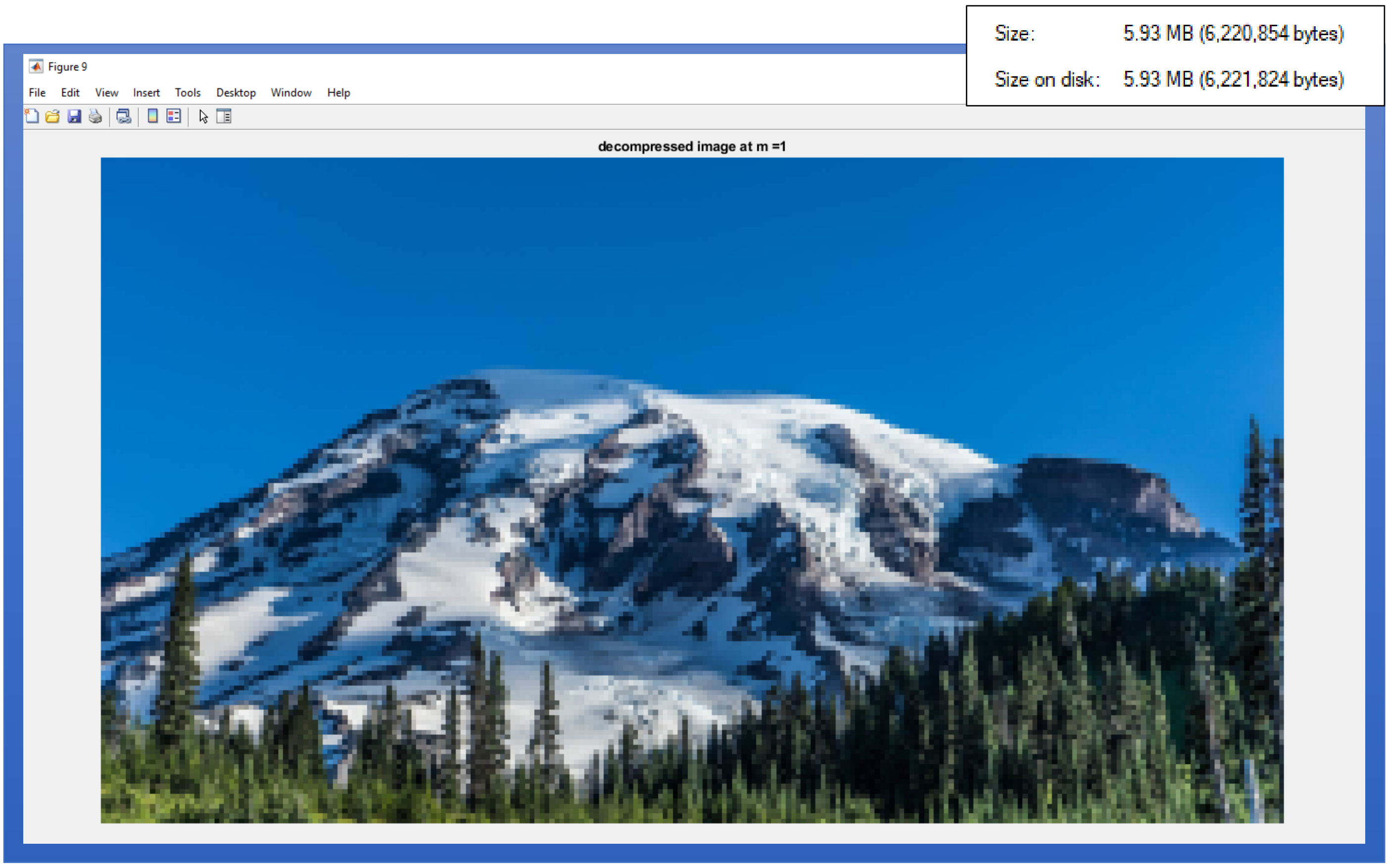


Figure 9
File Edit View Insert Tools Desktop Window Help
decompressed image at m =1

Size: 5.93 MB (6,220,854 bytes)
Size on disk: 5.93 MB (6,221,824 bytes)

Fig (d & c.12): Decompressed Image at m = 1 & its Size

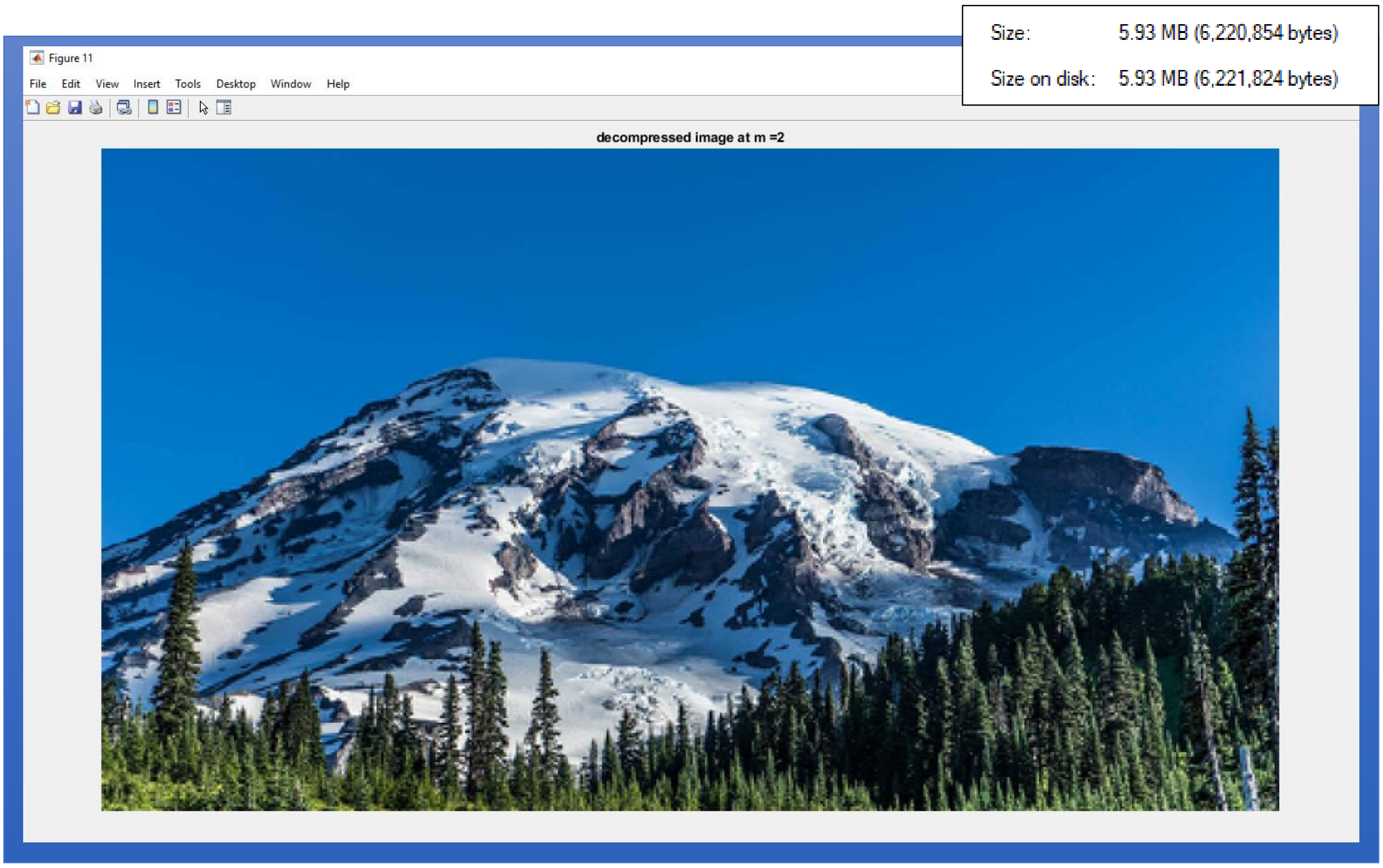


Figure 11
File Edit View Insert Tools Desktop Window Help
decompressed image at m =2

Size:	5.93 MB (6,220,854 bytes)
Size on disk:	5.93 MB (6,221,824 bytes)

Fig (d & c.13): Decompressed Image at m = 2 & its Size

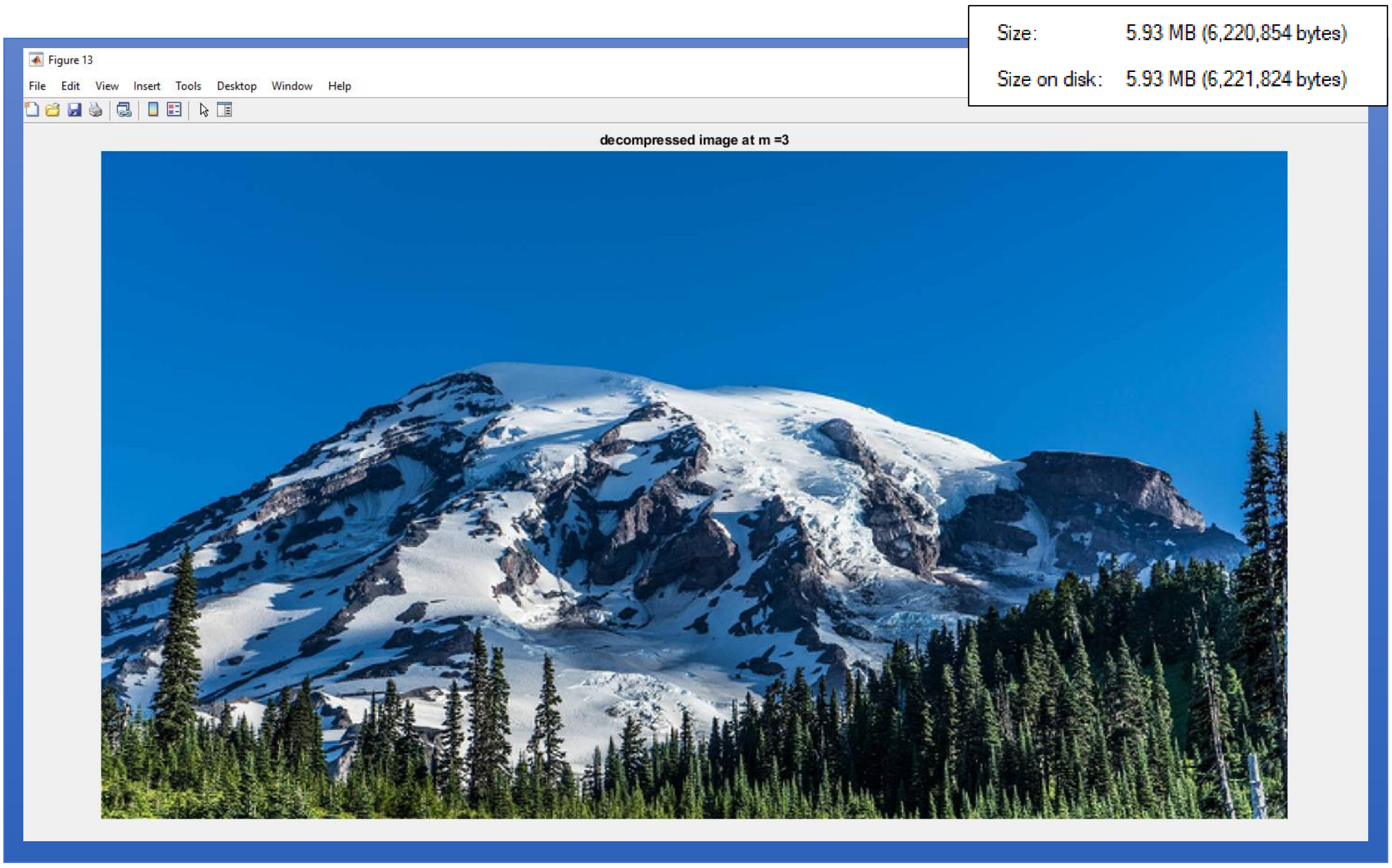


Figure 13
File Edit View Insert Tools Desktop Window Help
decompressed image at m =3

Size:	5.93 MB (6,220,854 bytes)
Size on disk:	5.93 MB (6,221,824 bytes)

Fig (d & c.14): Decompressed Image at m = 3 & its Size

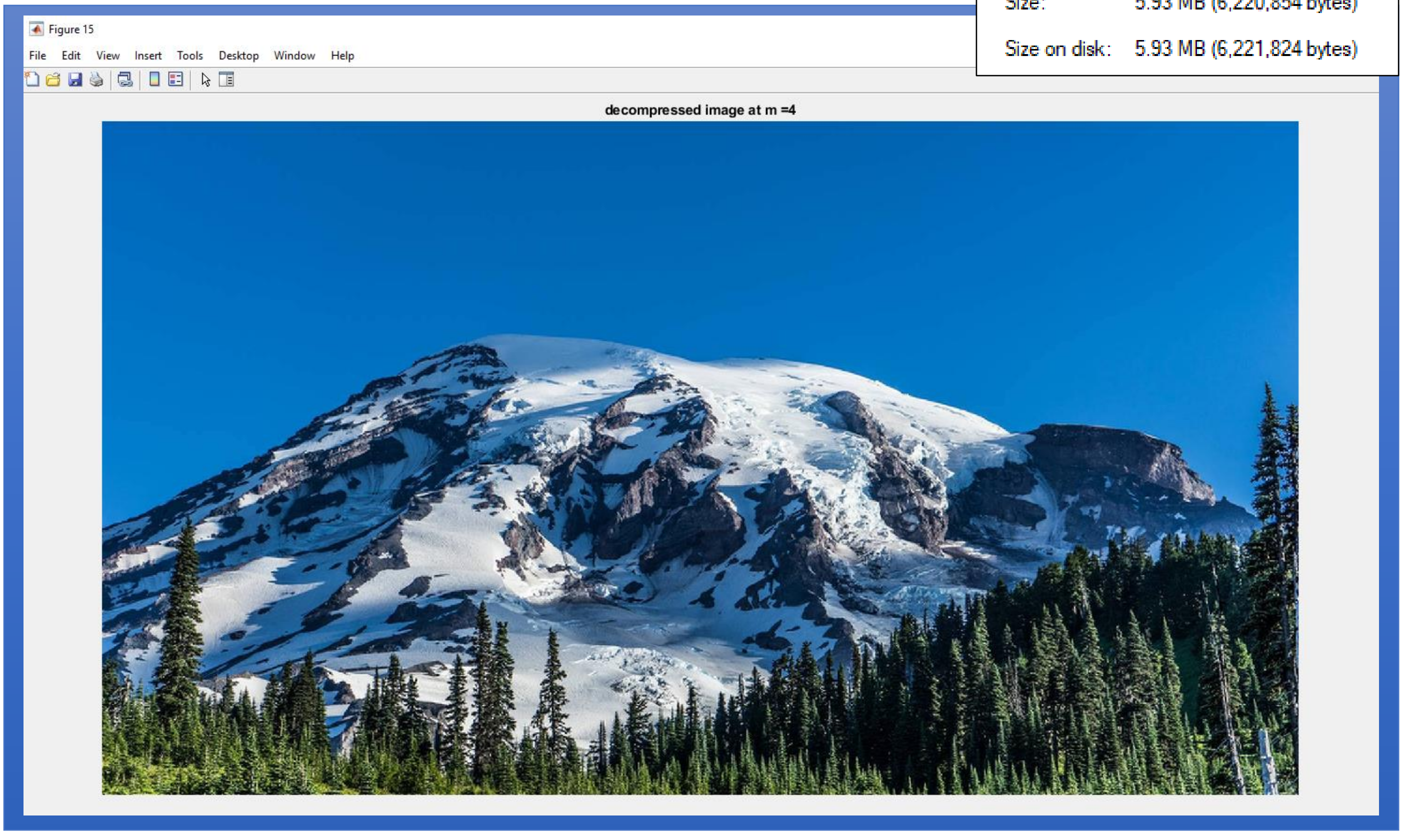
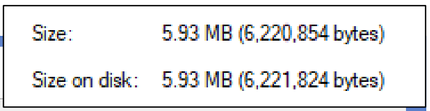


Figure 15
File Edit View Insert Tools Desktop Window Help
decompressed image at m =4



Size:	5.93 MB (6,220,854 bytes)
Size on disk:	5.93 MB (6,221,824 bytes)

Fig (d & c.15): Decompressed Image at $m = 4$ & its Size

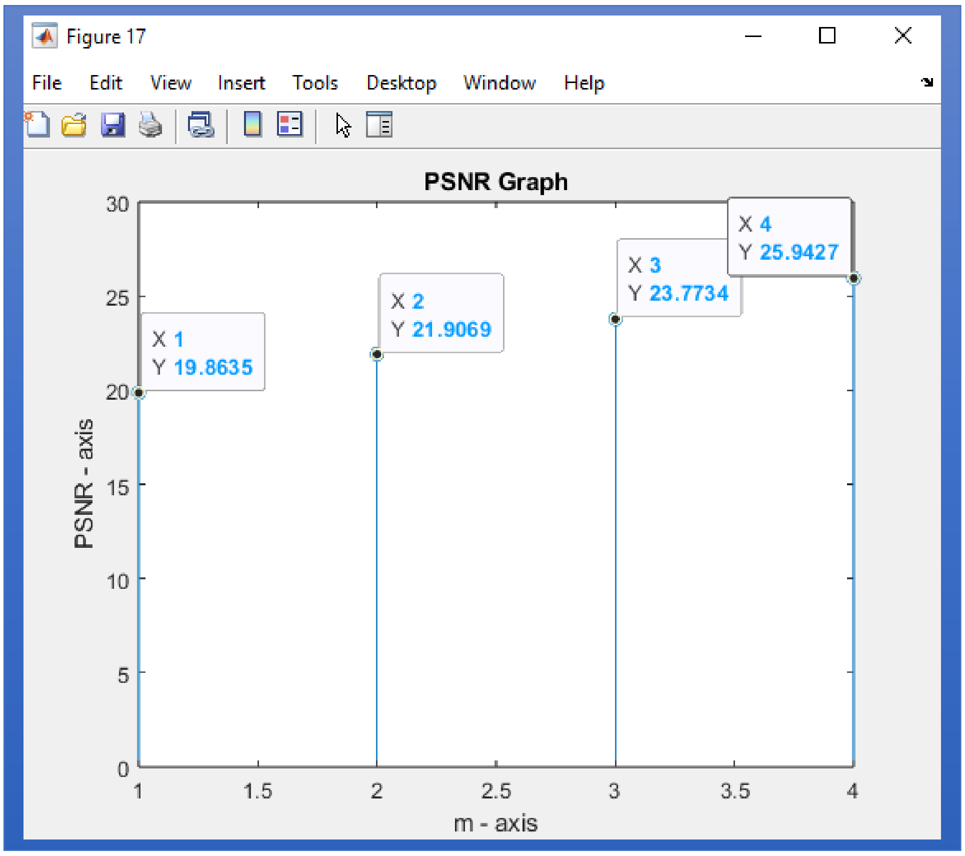


Fig (e & f.16): PSNR Graph in Discrete Values

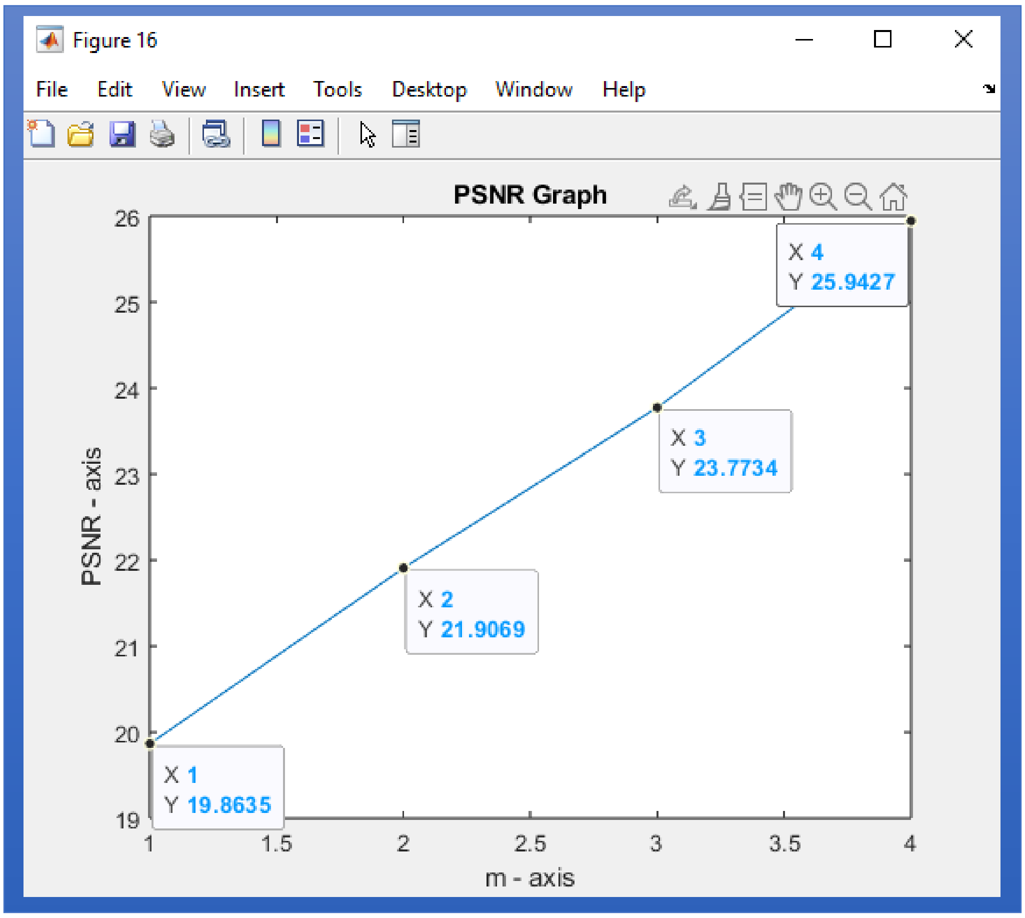
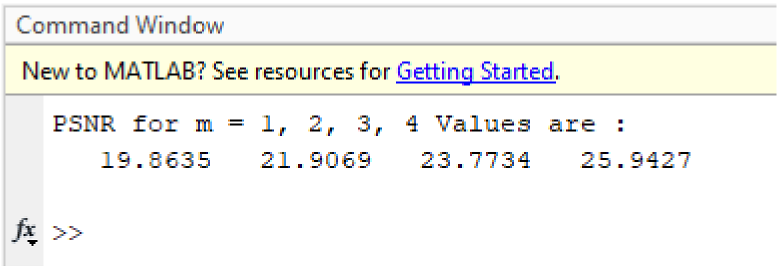


Fig (e & f.17): PSNR Graph at $m = 1, 2, 3, 4$



Command Window

New to MATLAB? See resources for Getting Started.

```
PSNR for m = 1, 2, 3, 4 Values are :
   19.8635   21.9069   23.7734   25.9427

>>
```

MATLAB Codes

```
%% Task (a)
clc
close all;
% Read and display The image input file using Relative Path.
Input_image = imread('image1.bmp');
[l,w] = size(Input_image(:,:,1));
figure,imshow(Input_image);
title("Original Input Image")
```

```
% Extract and display each one of its three color components.
[R_component, G_component, B_component] = imsplit(Input_image);
black = zeros(l, w);
% In Gray Scale
figure
imshow(R_component);
title('Red Component');
imwrite(R_component, 'Red Component.bmp');
figure
imshow(G_component);
title('Green Component');
imwrite(G_component, 'Green Component.bmp');
figure
imshow(B_component);
title('Blue Component');
imwrite(B_component, 'Blue Component.bmp');
% In RGB Scale
red = cat(3, R_component, black, black);
green = cat(3, black, G_component, black);
blue = cat(3, black, black, B_component);
figure
imshow(red)
imwrite(red, 'Red Component RGB.bmp');
title('Red Component');
figure
imshow(green);
imwrite(green, 'Green Component RGB.bmp');
title('Green Component');
figure
imshow(blue);
imwrite(blue, 'Blue Component RGB.bmp');
title('Blue Component');
%% Applying DCT & Inverse DCT to 8 * 8 Blocks
% Task (b) & (c) & (d) Repeat for m = 1,2,3,4:
% compress the image, process each color component in blocks of 8 × 8 pixels.
% Each block will be converted into frequency domain using 2D DCT.
% then only few coefficients are retained, while the rest will be ignored.
% Apply 2D DCT of each block having the same dimensions as the input block
decompression = cell(1,4);
compression = cell(1,4);
get_dct2 = @(block_struct) dct2(block_struct.data);
get_invdct2 = @(block_struct) idct2(block_struct.data);
mask = zeros(8);
R_DCT = blockproc(R_component, [8 8], get_dct2);
G_DCT = blockproc(G_component, [8 8], get_dct2);
B_DCT = blockproc(B_component, [8 8], get_dct2);
```

```
for m = 1:4
  mask(1:m,1:m) = 1;
  temporary = @(block_struct) block_struct.data.* mask;
  % Control the DCT Coefficient Outputs
  R_NEW = blockproc(R_DCT,[8 8],temporary);
  G_NEW = blockproc(G_DCT,[8 8],temporary);
  B_NEW = blockproc(B_DCT,[8 8],temporary);
  temp = @(block_struct) block_struct.data(1:m,1:m);
  R_compressed = blockproc(R_DCT,[8 8],temp);
  G_compressed = blockproc(G_DCT,[8 8],temp);
  B_compressed = blockproc(B_DCT,[8 8],temp);
  R_compressed_INVDCT = blockproc(R_compressed,[m m],get_invdct2);
  G_compressed_INVDCT = blockproc(G_compressed,[m m],get_invdct2);
  B_compressed_INVDCT = blockproc(B_compressed,[m m],get_invdct2);
  compressed_image
= uint8(cat(3,R_compressed_INVDCT,G_compressed_INVDCT,B_compressed_INVDCT));
  compression{1,m} = compressed_image;
  R_INVDCT = blockproc(R_NEW,[8 8],get_invdct2);
  G_INVDCT = blockproc(G_NEW,[8 8],get_invdct2);
  B_INVDCT = blockproc(B_NEW,[8 8],get_invdct2);
  decompression{1,m} = uint8(cat(3,R_INVDCT,G_INVDCT,B_INVDCT));
end
%% Task (e) & Task (f) Calculation and Plotting PSNR (Peak signal − to Noise Ratio)
peaksnr = ones(1,4);
Remaining = 0;
for m = 1:4
  imwrite(compression{1,m},strcat('compressed image at m = ',num2str(m),'.bmp'));
  imwrite(decompression{1,m},strcat('decompressed image at m = ',num2str(m),'.bmp'));
  figure
  imshow(compression{1,m})
  title(strcat('compressed image at m = ',num2str(m)));
  figure
  imshow(decompression{1,m})
  title(strcat('decompressed image at m = ',num2str(m)));
  peaksnr(m) = psnr(uint8(decompression{1,m}),uint8(Input_image));
end
fprintf("PSNR for m = 1,2,3,4 Values are : \n");
disp(peaksnr);
figure
plot(1:4,peaksnr)
title('PSNR Graph')
xlabel('m − axis')
ylabel('PSNR − axis')
figure
stem(1:4,peaksnr)
title('PSNR Graph')
xlabel('m − axis')
ylabel('PSNR − axis')
```

Audio Spectral Analysis And Filtering

Task 2

Explanation

- Q (a): By Using *test_noisy_audio.m* MATLAB File To get input information as Shown in Fig (a.1) & Fig (a.2) we get that $F_{stop} = 3000\ HZ$ we Approximate it to $3000\ HZ$ in filter Designer tool and $F_{pass} = 2500\ HZ$ Approximately or any number between $2000\ HZ\ to\ 2700\ HZ$ in filter Designer tool & Then we Know that the Interfering signal: High Frequency Sinusoidal (Cosine) Signal (In Time Domain). Impulse Signal in (Freq. Domain).
- Q (b): As Shown in Fig (b.3) & Fig (b.4), In Response Type We use Lowpass Filter, In Design Method We use IIR (Butterworth), and In Frequency Specifications in Hz units we take $\mathrm{F_s = 48000\ Hz, F_{pass} = 2500\ Hz, F_{stop} = 3000\ HZ}$, And then We Click Design Filter and then Export the Variables to the Workspace as an object as shown in Fig 0. In ZIP file.
- Q (c): By Using audio.m MATLAB File after Run it will Create A filtered Sound file of the Output audio Without noise (Interfering signal).
- Q (d): As Shown in Fig (d.5) & Fig (d.6), we find that the last sound signals around 3000 HZ then it is Filtered Successfully in filtered.wav.
- Q (e): As Shown in Fig (e.7) it's the Magnitude (dB) of the Frequency Response & In Fig (e.8) it's the Phase of the Frequency Response & Fig (e.9) it's the Combination of the Magnitude (dB) & Phase of the Frequency Response all figures from freqz(Hd).
- Q (f): As Shown in Fig (f.10) We get Impulse Response by typing impz(Hd) in the terminal Shown in MATLAB.
- Q (h): By Using audiox2.m MATLAB File after Run it will Create filteredx2 Sound file of the Output audio Without noise (Interfering signal) but with Twice Speed and As Shown in Fig (h.11) & Fig (h.12) we find that the last sound signals around 4000 HZ then it is Filtered with Twice Speed Successfully in filteredx2.wav.

- Content In ZIP File in Task 2 (Audio Spectral Analysis & Filtering): -
  - ✓ Figures: In this Folder you Will Find Figures From (Fig 0) to (Fig 12).
  - ✓ Filtered Sounds: In this Folder you Will Find Audios (Original & Filtered & Filteredx2).
  - ✓ MATLAB.m Codes: In this Folder you Will Find .m files (audio.m & audiox2.m & test_noisy_audio. m).
- Content of This File Respectively:
  - ✓ One uncompressed pdf project report containing:
    a. Explanation of your work.
    b. All the required results and answers to questions.
    c. All the required figures. Label your figures properly.
    d. All the codes, included at the end.

QUESTIONS & ANSWERS

Q (a): Use FFT to obtain the magnitude spectrum of the audio file “audio.wav”. Plot it against the frequency in Hz. Use the function ‘fftshift’ to make the zero frequency in the center of the plot? What is the type of the Interfering signal?

Ans: -

- Plotting: In Figures & Labels Section (Fig (a.1) & Fig (a.2)).
- The type of the Interfering signal:
  High Freq. Sinusoidal (Cosine) Signal. (In Time Domain).
  Impulse Signal. (In Frequency Domain).

Q (b): Use a suitable filter of your choice to remove the interfering signal. The filter choice should be based on the spectrum in part (a).
(Moving average filter should not be used). Give all the filter specifications.

Ans: -

- Plotting: In Figures & Labels Section (Fig (b.3) & Fig (b.4)).
- Filter specifications: From Part (a) Put $F_{stop} \cong 3000$ & $F_{pass} \cong 2500$
  Design Method (IIR Butterworth) & Response Type (Low pass).

Q (c): Play the filtered audio. (Record it as “filtered.wav”): filtered

---

Q (d): Plot the spectrum of the filtered audio.

Ans: -

- Plotting: In Figures & Labels Section (Fig (d.5) & Fig (d.6)).

Q (e): Plot the frequency response of the filter.

Ans: -

- Plotting: In Figures & Labels Section (Fig (e.7) & Fig (e.8) & Fig (e.9)).

Q (f): Plot the impulse response of the filter. What is the relationship between the impulse response and the frequency response in part (e)?

Ans: -

- Plotting: In Figures & Labels Section (Fig (f.10)).
- the relationship between the impulse response and the frequency response: A system's frequency response $H(j\omega)$ is the Fourier Transform of its impulse response $h(t)$.

Q (g): Is the filter causal or not? Explain your answer.

Ans: -

- Yes, it is causal. Because $h(t) = 0$ for $t < 0$
  It had to be causal to be practical for a real-time application.

Q (h): Play the filtered audio at twice the speed. Plot the new spectrum. What do you notice on the spectrum compared to the spectrum of the normal speed file?

Ans: -

- Plotting: In Figures & Labels Section (Fig (h.11) & Fig (h.12)).
- Notice on the spectrum:
  Expansion of the Frequency Range (F_s) of the New Spectrum Approximately be Double the Normal Spectrum Frequency Range for the audio input file. the magnitude of the spectrum at Twice speed is the same as the magnitude of the normal speed from.

Figures & Labels

test_noisy_audio

Q (a): Plot the magnitude spectrum of the original audio file against the frequency in Hz.

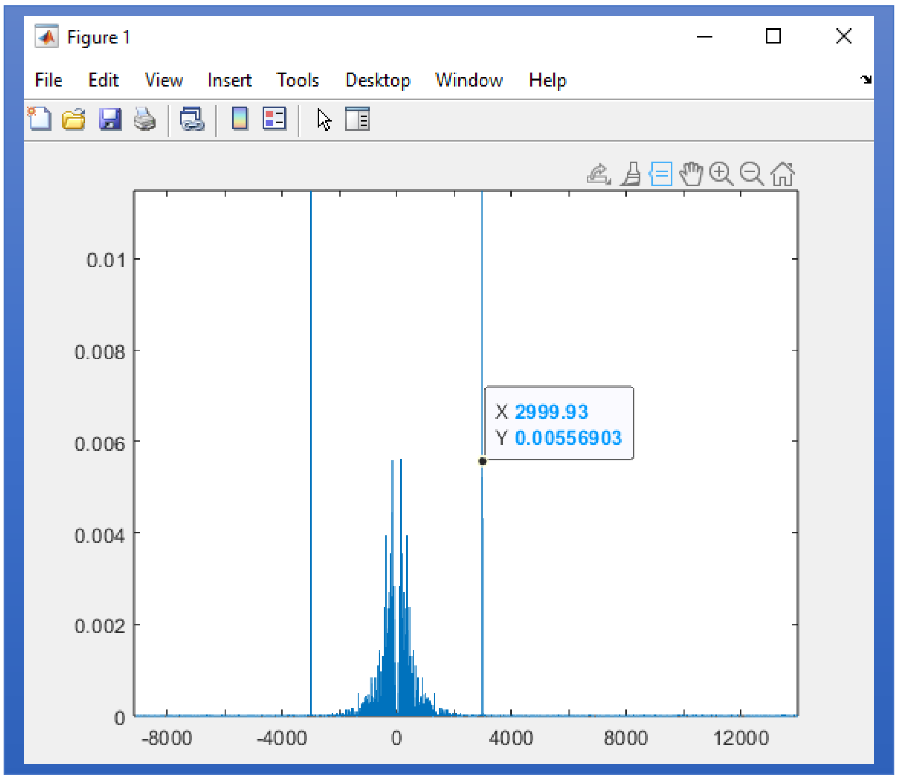


Fig (a.1): Input Audio Signal $F_{stop} \cong 3000$ HZ

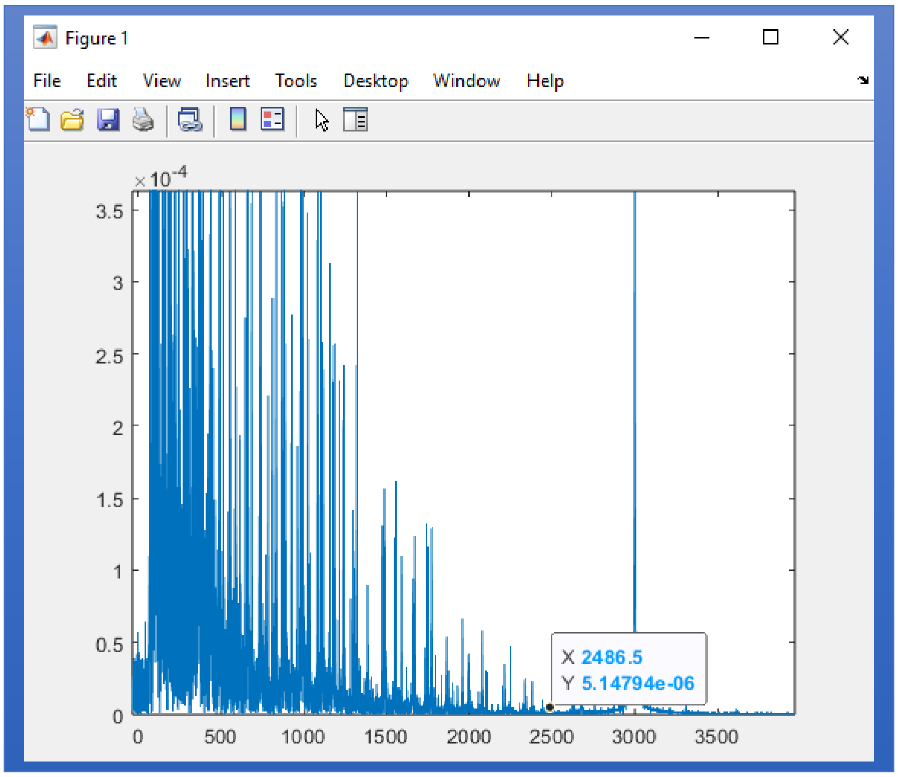
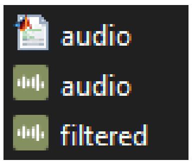
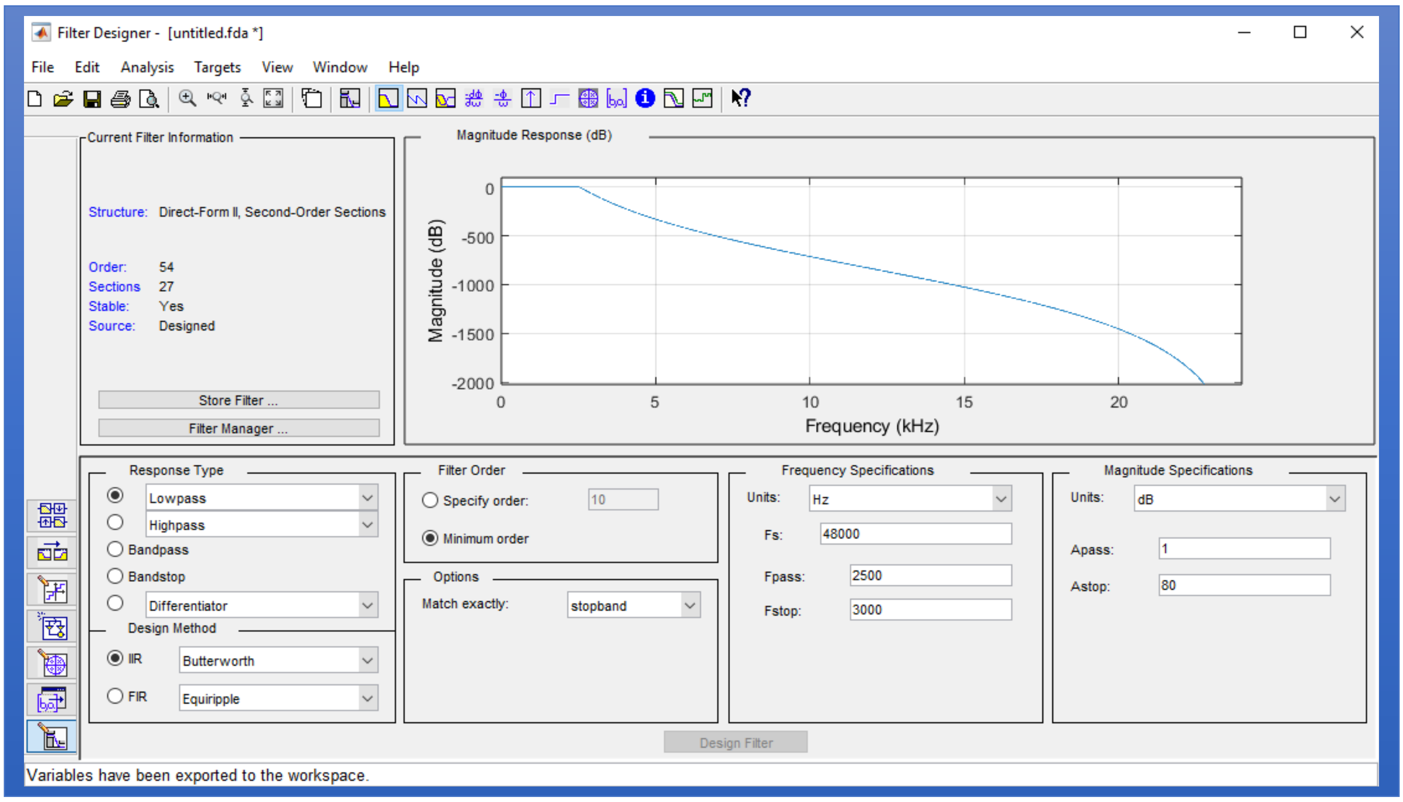


Fig (a.2): Input Audio Signal $F_{pass} \cong 2500$ HZ


audio
audio
filtered

Q (b): Filter Design Magnitude & Magnitude (dB) against the frequency in Hz.


Filter Designer - [untitled.fda *]

Current Filter Information

Structure: Direct-Form II, Second-Order Sections

Order: 54
Sections 27
Stable: Yes
Source: Designed

Response Type: [✓ Lowpass]
Design Method: [✓ IIR] Butterworth

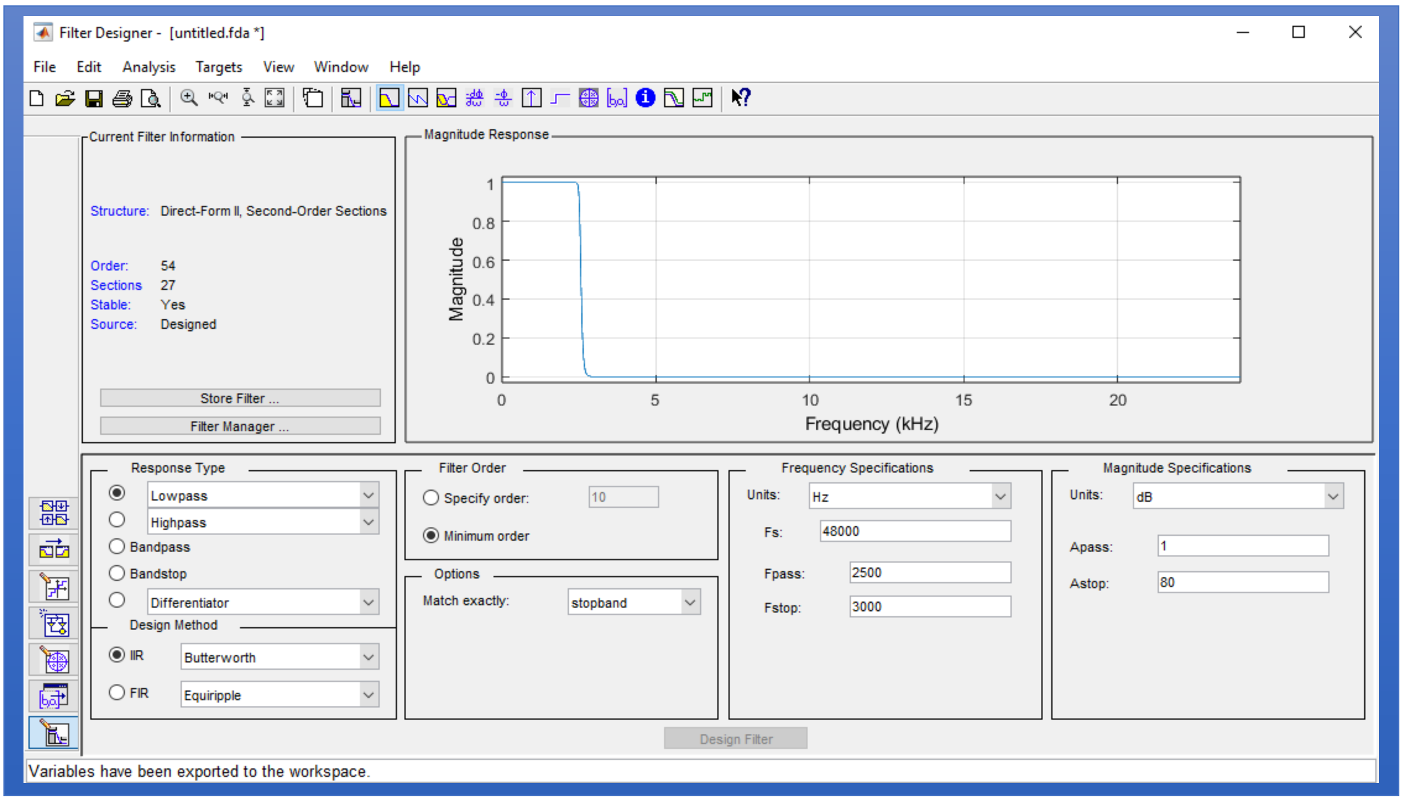
Filter Order: [✓ Minimum order]
Options: Match exactly: stopband

Frequency Specifications
Units: Hz
Fs: 48000
Fpass: 2500
Fstop: 3000

Magnitude Specifications
Units: dB
Apass: 1
Astop: 80

Variables have been exported to the workspace.

Fig (b.3): Filter Design (Magnitude (dB))


Filter Designer - [untitled.fda *]

Current Filter Information

Structure: Direct-Form II, Second-Order Sections

Order: 54
Sections 27
Stable: Yes
Source: Designed

Response Type: [✓ Lowpass]
Design Method: [✓ IIR] Butterworth

Filter Order: [✓ Minimum order]
Options: Match exactly: stopband

Frequency Specifications
Units: Hz
Fs: 48000
Fpass: 2500
Fstop: 3000

Magnitude Specifications
Units: dB
Apass: 1
Astop: 80

Variables have been exported to the workspace.

Fig (b.4): Filter Design (Magnitude)

Q (d): Plot the spectrum of the filtered audio.

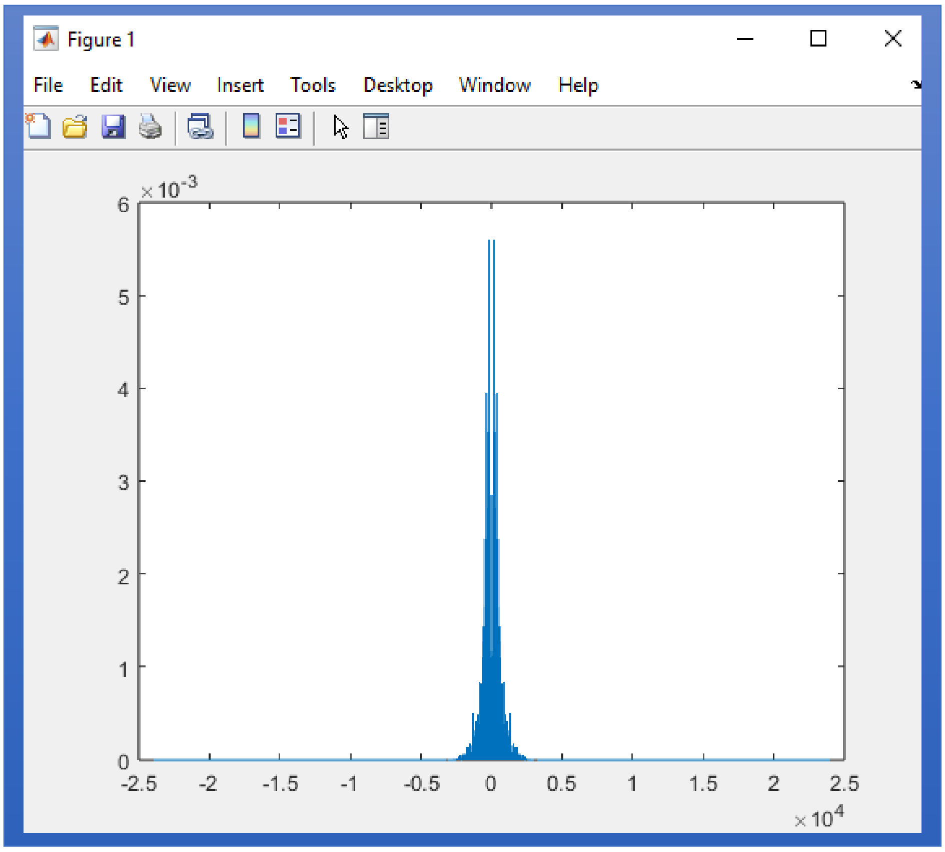


Fig (d.5): the spectrum of the filtered audio.

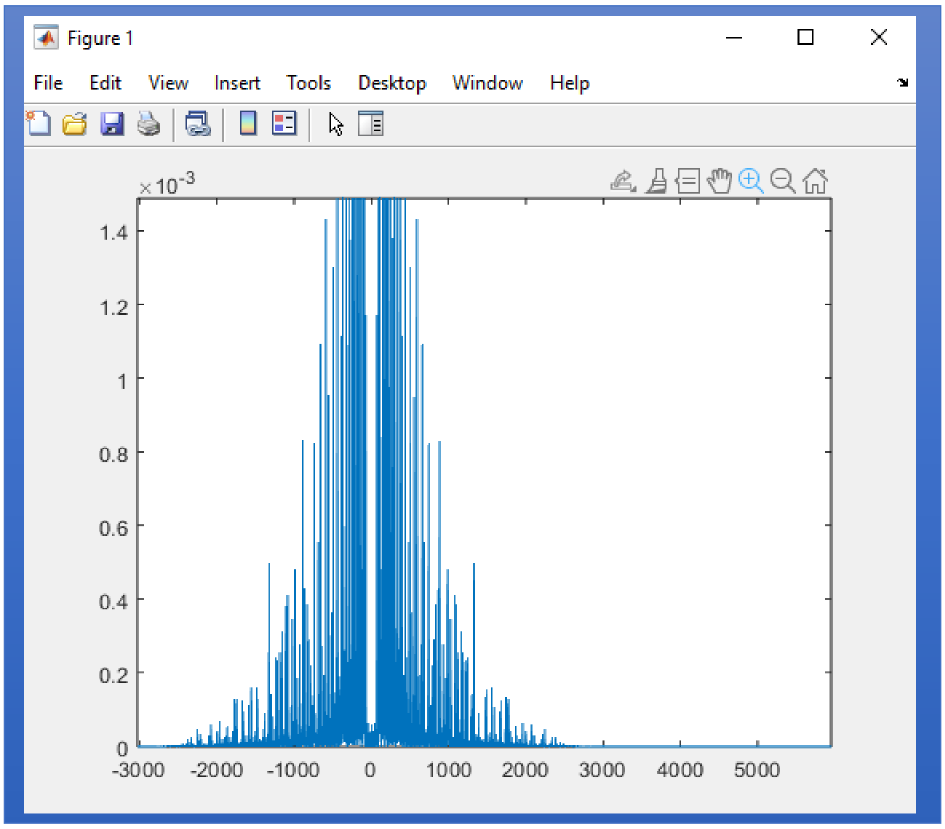


Fig (d.6): the spectrum of the filtered audio.

Q (e): Plot the spectrum of the filtered audio. (By typing freqz(Hd) in MATLAB)

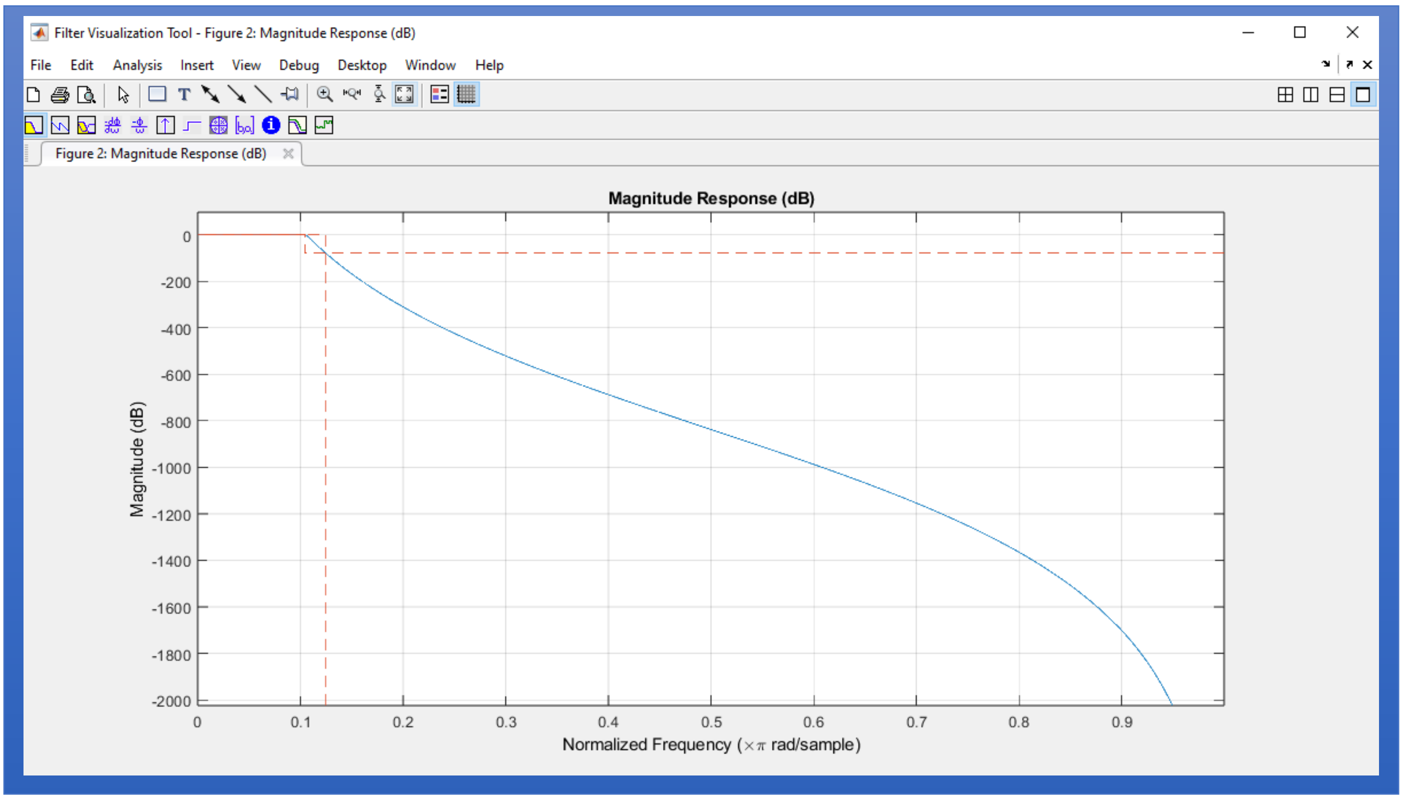


Fig (e.7): Frequency Response (Magnitude (dB))

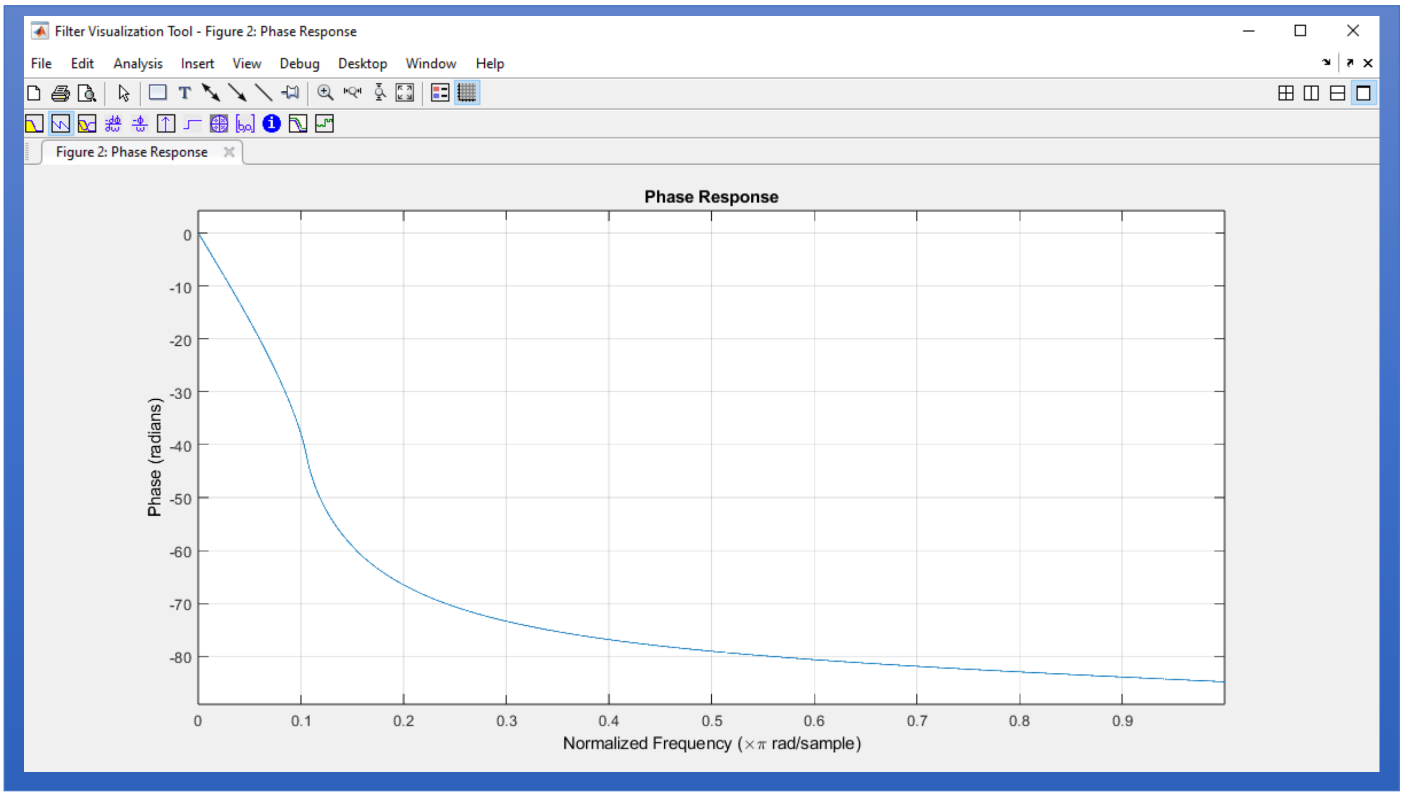


Fig (e.8): Frequency Response (Phase)

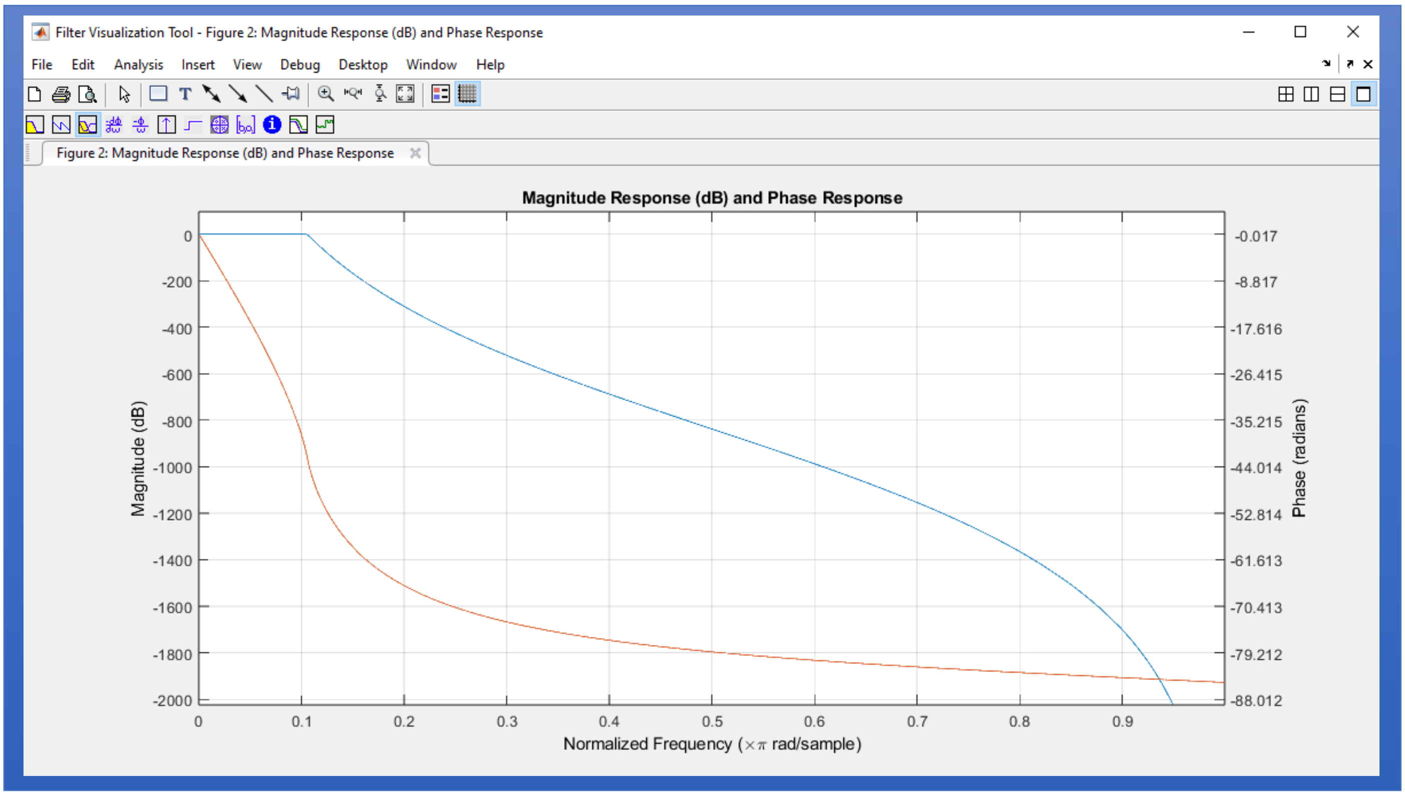

Filter Visualization Tool - Figure 2: Magnitude Response (dB) and Phase Response
File Edit Analysis Insert View Debug Desktop Window Help
Figure 2: Magnitude Response (dB) and Phase Response
Magnitude Response (dB) and Phase Response
Magnitude (dB)
Phase (radians)
Normalized Frequency (×π rad/sample)
0
-200
-400
-600
-800
-1000
-1200
-1400
-1600
-1800
-2000
-0.017
-8.817
-17.616
-26.415
-35.215
-44.014
-52.814
-61.613
-70.413
-79.212
-88.012
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9

Fig (e.9): Frequency Response (Magnitude (dB)) & (Phase)

Q (f): Plot the impulse response of the filter. (By typing impz(Hd) in MATLAB)

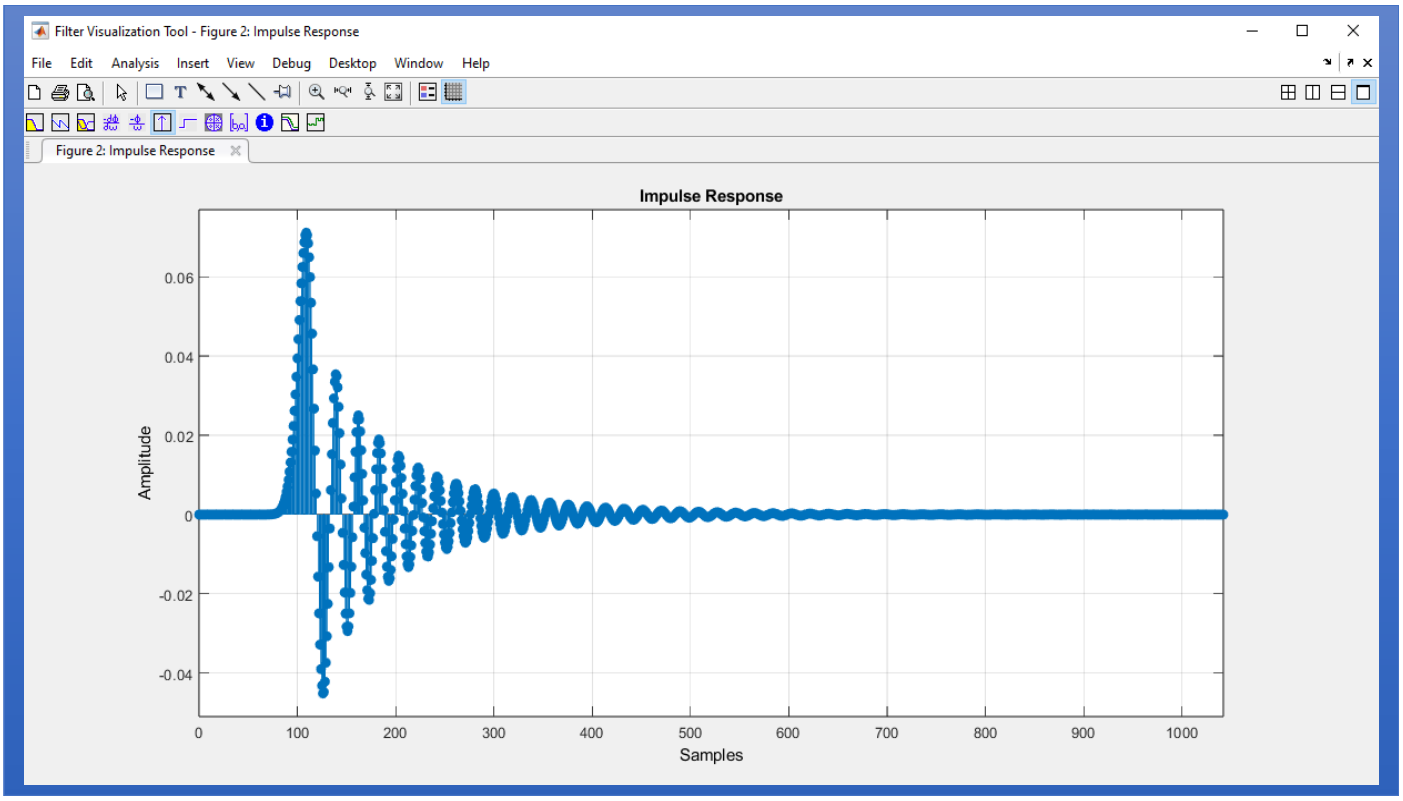
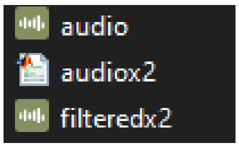

Filter Visualization Tool - Figure 2: Impulse Response
File Edit Analysis Insert View Debug Desktop Window Help
Figure 2: Impulse Response
Impulse Response
Amplitude
Samples
0.06
0.04
0.02
0
-0.02
-0.04
0 100 200 300 400 500 600 700 800 900 1000

Fig (f.10): Impulse Response (Amplitude)

Q (h): Play the filtered audio at twice the speed. Plot the new spectrum.



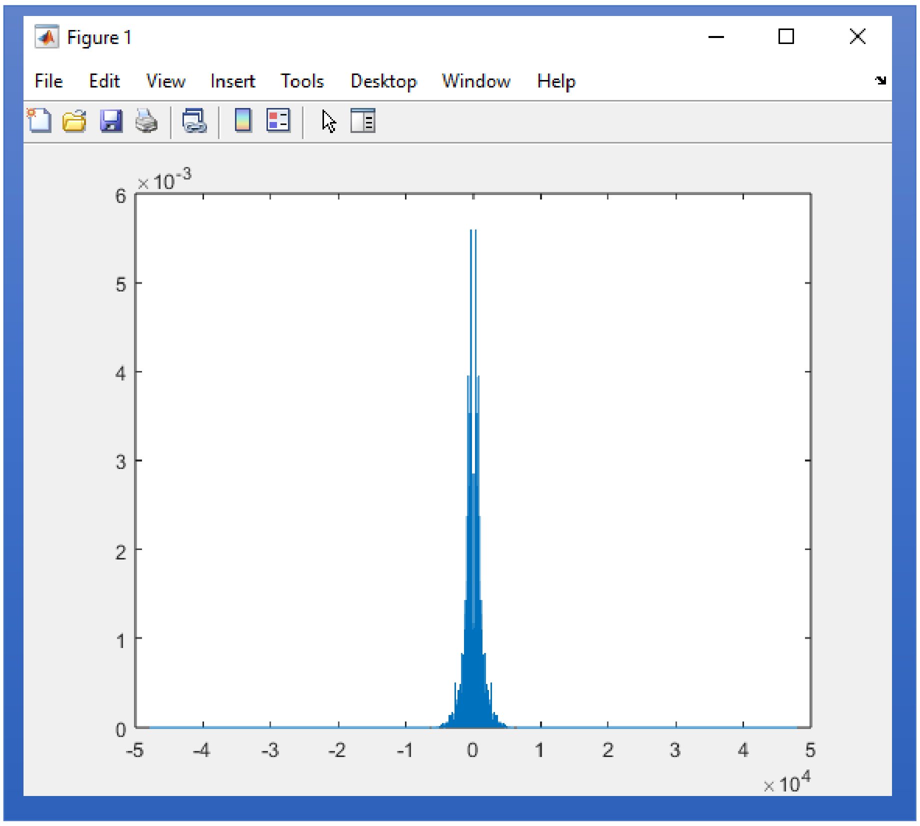


Fig (h.11): filtered audio at twice the speed

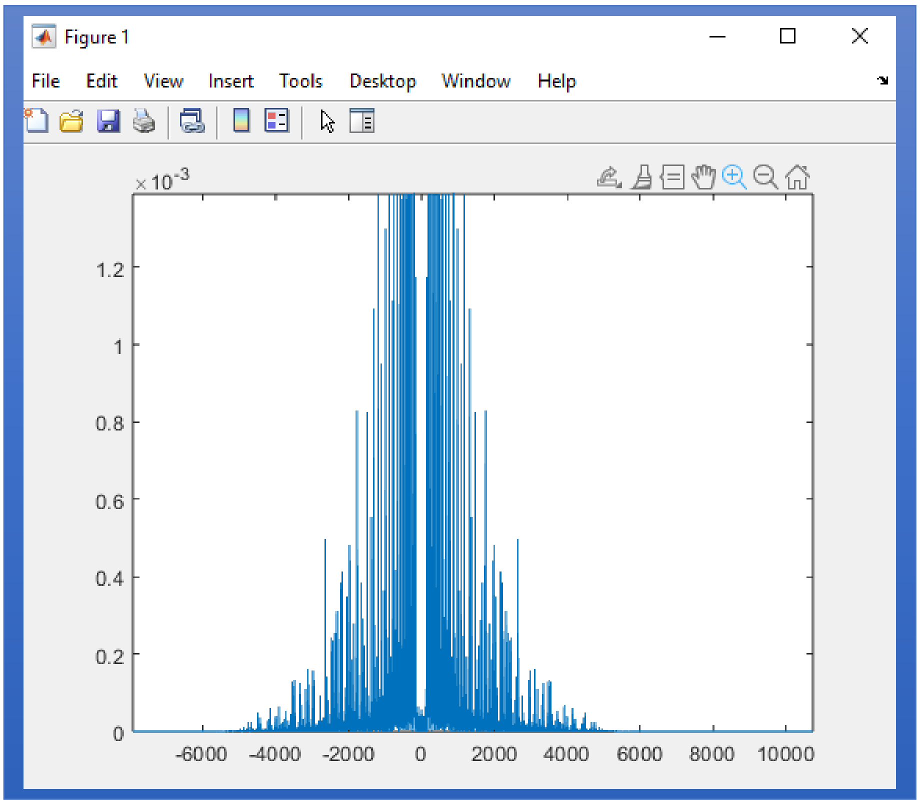
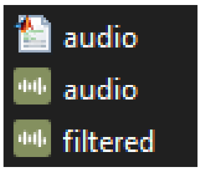


Fig (h.12): filtered audio at twice the speed (Double the Normal Frequency Range)

MATLAB Codes

test_noisy_audio

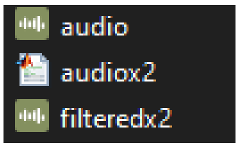
```
%% Test Input Audio File To get Fstop & Fpass
clc;
close all;
[x, fs] = audioread('audio.wav');
N = length(x);

%% Plotting against k
k = 0 : N − 1;
y = fft(x, N);
plot(k, abs(y));

%% Plotting against Frequency in HZ
f = (0 : N − 1) ∗ fs / N;
plot (f, abs(y) / N);

%% Shifting Zero to the center of Spectrum then Plotting against Frequency in HZ
f = (−N/2 : N/2 − 1) ∗ fs / N;
plot(f, abs(fftshift(y)) / N);
```



audio
audio
filtered

With Normal Speed.

```
%% Read The Input Audio
[x, fs] = audioread('audio.wav');
N = length(x);
%% Plotting against k
k = 0 : N − 1;
y = fft(x, N);
plot(k, abs(y));
%% Plotting against Frequency in HZ
f = (0 : N − 1) * fs / N;
plot(f, abs(y) / N);
%% Shifting Zero to the center of Spectrum
% then Plotting against Frequency in HZ
f = (−N / 2 : N / 2 − 1) * fs / N;
plot(f, abs(fftshift(y)) / N);
%% Filtering With Fstop ≅ 3000 & Fpass ≅ 2500
filtered = filter(Hd, x);
m = length(filtered);
Filtered_NEW = fft(filtered, m);
%% Plotting against k
plot(k, abs(Filtered_NEW));
%% Plotting against Frequency in HZ
f = (0 : N − 1) * fs / m;
plot(f, abs(Filtered_NEW) / m);
%% Shifting Zero to the center of Spectrum
f = (−N / 2 : N / 2 − 1) * fs / m;
%% Plotting against Frequency in HZ
plot(f, abs(fftshift(Filtered_NEW)) / m);
%% Write The Output Audio
audiowrite("filtered.wav", filtered, fs);
[z, fss] = audioread("filtered.wav");
sound(z, fss);
%% Plotting the Frequency Responce & Impulse Response
impz(Hd);
freqz(Hd);
```


audio
audiox2
filteredx2

With at Twice the Speed.

```
%% Read The Input Audio
[x, fs] = audioread('audio. wav');
N = length(x);
%% Plotting against k
k = 0 : N − 1;
y = fft(x, N);
plot(k, abs(y));
%% Plotting against Frequency in HZ
f = (0 : N − 1) ∗ fs / N;
plot(f, abs(y) / N);
%% Shifting Zero to the center of Spectrum
% then Plotting against Frequency in HZ
f = (−N / 2 : N / 2 − 1) ∗ fs / N;
plot(f, abs(fftshift(y)) / N);
%% Filtering With Fstop ≅ 3000 & Fpass ≅ 2500
filtered = filter(Hd, x);
m = length(filtered);
Filtered_NEW = fft(filtered, m);
%% Plotting against k
plot(k, abs(Filtered_NEW));
%% Plotting against Frequency in HZ x2 Speed
f = (0 : N − 1) ∗ (2 ∗ fs) / m;
plot(f, abs(Filtered_NEW) / m);
%% Shifting Zero to the center of Spectrum
f = (−N / 2 : N / 2 − 1) ∗ (2 ∗ fs) / m;
%% Plotting against Frequency in HZ
plot(f, abs(fftshift(Filtered_NEW)) / m);
%% Write The Output Audio x2 Speed
audiowrite("filteredx2. wav", filtered, 2 ∗ fs);
[z, fss] = audioread("filteredx2. wav");
sound(z, 2 ∗ fss);
%% Plotting the Frequency Responce & Impulse Response
impz(Hd);
freqz(Hd);
```