**PROJECT-9**

**LOW PASS & HIGH PASS**

**FILTER**

EE5356 Digital Image Processing

Dr. K. R. Rao

Submitted By:

Ashutosh Desai

UTA ID:1001388602

Email: - ashutoshrajesh.desai@mavs.uta.edu

SPRING-2018

EE 5356 - DIGITAL IMAGE PROCESSING - PROJECT 14

Low-pass and High-pass filter

Steps:

1. Read any 512\*512 grayscale image and compute its Fourier transform.

2. Generate an ideal LPF for your 512\*512 image with cutoff frequency (Do) 40, 60 and compute its frequency response .

3. Generate a Gaussian LPF with σ = 16, 24 and compute its frequency response.

4. Generate a Butterworth LPF having order n = 3, 4 and Do = 50 and compute its frequency response.

5. Generate an ideal HPF with cutoff frequency 30, 50 and compute its frequency response.

6. Generate a Gaussian HPF with σ = 16, 24 and compute its frequency response.

7. Generate a Butterworth HPF with n = 3, 4 and Do = 50 and compute its frequency response.

8. Apply all the generated filters on your image in frequency domain and obtain the filtered images.

Submit the following:

1. Display the image and the log of its shifted Fourier transform image (magnitude spectrum).

2. Display the 3D plot of all the LPF and HPF filter responses.

3. Display all the filtered images in frequency domain.

4. Display all the filtered images in spatial domain.

5. Give the MATLAB code.

References:

1. Marques, Oge “Practical image and video processing using MATLAB” pp 243-251, Wiley, 2011

**THEORY**

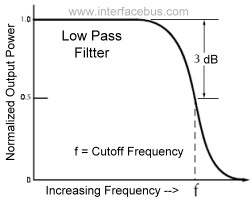
**LOW PASS FILTER**

A low-pass filter (LPF) is a [filter](https://en.wikipedia.org/wiki/Filter_(signal_processing)) that passes [signals](https://en.wikipedia.org/wiki/Signal_(electrical_engineering)) with a [frequency](https://en.wikipedia.org/wiki/Frequency) lower than a certain [cutoff frequency](https://en.wikipedia.org/wiki/Cutoff_frequency) and [attenuates](https://en.wikipedia.org/wiki/Attenuate) signals with frequencies higher than the cutoff frequency. The exact [frequency response](https://en.wikipedia.org/wiki/Frequency_response) of the filter depends on the [filter design](https://en.wikipedia.org/wiki/Filter_design). The filter is sometimes called a high-cut filter, or treble-cut filter in audio applications. A low-pass filter is the complement of a [high-pass filter](https://en.wikipedia.org/wiki/High-pass_filter).

Low-pass filters exist in many different forms, including electronic circuits such as a hiss filter used in [audio](https://en.wikipedia.org/wiki/Sound_recording), [anti-aliasing filters](https://en.wikipedia.org/wiki/Anti-aliasing_filter) for conditioning signals prior to [analog-to-digital conversion](https://en.wikipedia.org/wiki/Analog-to-digital_conversion), [digital filters](https://en.wikipedia.org/wiki/Digital_filter) for smoothing sets of data, acoustic barriers, [blurring](https://en.wikipedia.org/wiki/Gaussian_blur) of images, and so on. The [moving average](https://en.wikipedia.org/wiki/Moving_average_(finance)) operation used in fields such as finance is a particular kind of low-pass filter, and can be analyzed with the same [signal processing](https://en.wikipedia.org/wiki/Signal_processing) techniques as are used for other low-pass filters. Low-pass filters provide a smoother form of a signal, removing the short-term fluctuations and leaving the longer-term trend.

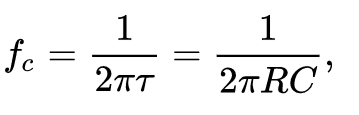
Filter designers will often use the low-pass form as a [prototype filter](https://en.wikipedia.org/wiki/Prototype_filter). That is, a filter with unity bandwidth and impedance. The desired filter is obtained from the prototype by scaling for the desired bandwidth and impedance and transforming into the desired bandform (that is low-pass, high-pass, [band-pass](https://en.wikipedia.org/wiki/Band-pass_filter) or [band-stop](https://en.wikipedia.org/wiki/Band-stop_filter)).

**Frequency Response Of a Low pass Filter**



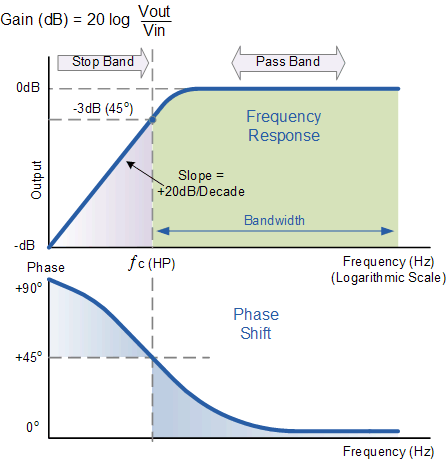
**HIGH PASS FILTER**

A high-pass filter (HPF) is an [electronic filter](https://en.wikipedia.org/wiki/Filter_(signal_processing)) that passes [signals](https://en.wikipedia.org/wiki/Signal_(electrical_engineering)) with a [frequency](https://en.wikipedia.org/wiki/Frequency) higher than a certain [cutoff frequency](https://en.wikipedia.org/wiki/Cutoff_frequency) and [attenuates](https://en.wikipedia.org/wiki/Attenuate) signals with frequencies lower than the cutoff frequency. The amount of [attenuation](https://en.wikipedia.org/wiki/Attenuation) for each frequency depends on the filter design. A high-pass [filter](https://en.wikipedia.org/wiki/Filter_(signal_processing)) is usually modeled as a [linear time-invariant system](https://en.wikipedia.org/wiki/Linear_time-invariant_system). It is sometimes called a low-cut filter or bass-cut filter.[[1]](https://en.wikipedia.org/wiki/High-pass_filter#cite_note-Watkinson1998-1) High-pass filters have many uses, such as blocking DC from circuitry sensitive to non-zero average voltages or [radio frequency](https://en.wikipedia.org/wiki/Radio_frequency) devices. They can also be used in conjunction with a [low-pass filter](https://en.wikipedia.org/wiki/Low-pass_filter) to produce a [bandpass filter](https://en.wikipedia.org/wiki/Bandpass_filter).



**Cut-off Frequency:-**

**Frequency Response Of a High pass Filter**



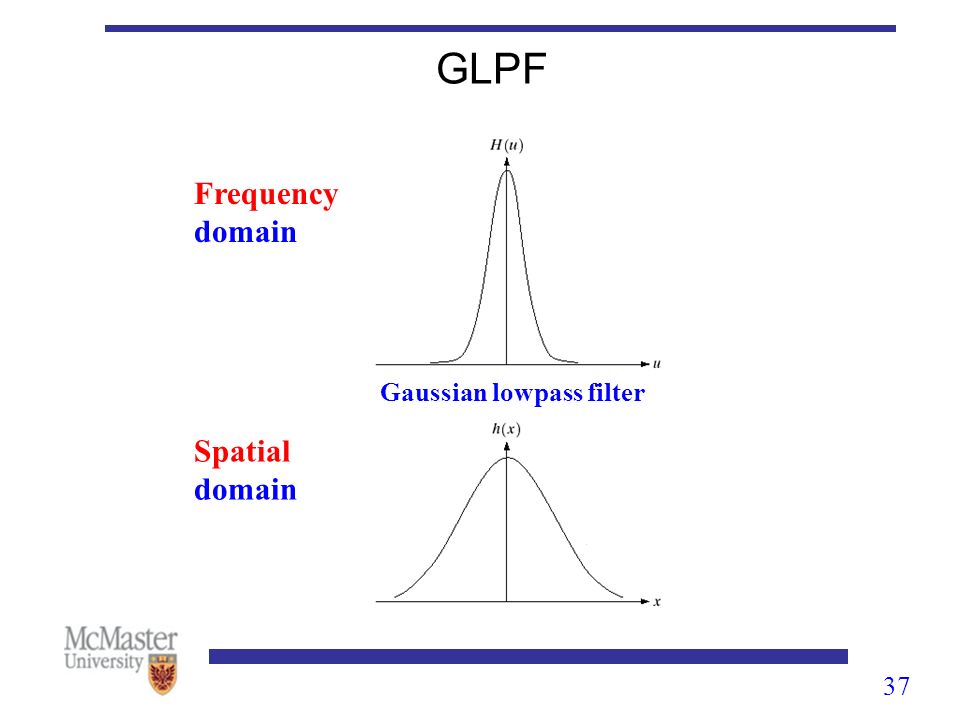
**GAUSSIAN FILTER:**

In [electronics](https://en.wikipedia.org/wiki/Electronics) and [signal processing](https://en.wikipedia.org/wiki/Signal_processing), a Gaussian filter is a [filter](https://en.wikipedia.org/wiki/Filter_(signal_processing)) whose [impulse response](https://en.wikipedia.org/wiki/Impulse_response) is a [Gaussian function](https://en.wikipedia.org/wiki/Gaussian_function) (or an approximation to it). Gaussian filters have the properties of having no overshoot to a step function input while minimizing the rise and fall time. This behavior is closely connected to the fact that the Gaussian filter has the minimum possible [group delay](https://en.wikipedia.org/wiki/Group_delay). It is considered the ideal [time domain](https://en.wikipedia.org/wiki/Time_domain) filter, just as the [sinc](https://en.wikipedia.org/wiki/Sinc_filter) is the ideal frequency domain filter.[[1]](https://en.wikipedia.org/wiki/Gaussian_filter#cite_note-1) These properties are important in areas such as [oscilloscopes](https://en.wikipedia.org/wiki/Oscilloscope#The_vertical_amplifier)[[2]](https://en.wikipedia.org/wiki/Gaussian_filter#cite_note-2) and digital telecommunication systems.[[3]](https://en.wikipedia.org/wiki/Gaussian_filter#cite_note-3)

Mathematically, a Gaussian filter modifies the input signal by [convolution](https://en.wikipedia.org/wiki/Convolution) with a Gaussian function; this transformation is also known as the [Eigenstress transform](https://en.wikipedia.org/wiki/Weierstrass_transform).

The Gaussian function is for {\displaystyle x\in (-\infty ,\infty )} and would theoretically require an infinite window length. However, since it decays rapidly, it is often reasonable to truncate the filter window and implement the filter directly for narrow windows, in effect by using a simple rectangular window function. In other cases, the truncation may introduce significant errors. Better results can be achieved by instead using a different [window function](https://en.wikipedia.org/wiki/Window_function); see [scale space implementation](https://en.wikipedia.org/wiki/Scale_space_implementation) for details.

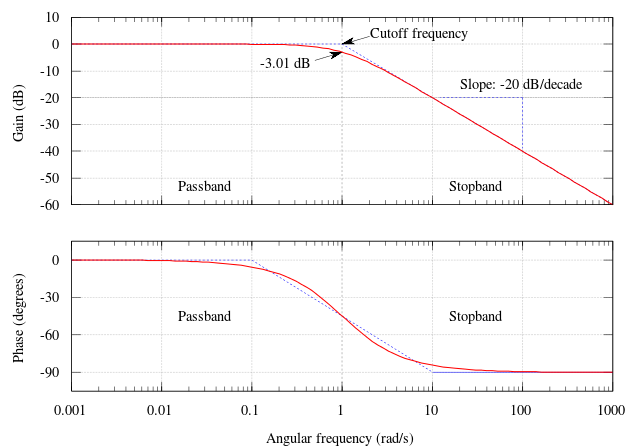
Filtering involves convolution. The filter function is said to be the kernel of an integral transform. The Gaussian kernel is continuous. Most commonly, the discrete equivalent is the [sampled Gaussian kernel](https://en.wikipedia.org/wiki/Sampled_Gaussian_kernel)that is produced by sampling points from the continuous Gaussian. An alternate method is to use the [discrete Gaussian kernel](https://en.wikipedia.org/wiki/Discrete_Gaussian_kernel) which has superior characteristics for some purposes. Unlike the sampled Gaussian kernel, the discrete Gaussian kernel is the solution to the discrete [diffusion equation](https://en.wikipedia.org/wiki/Diffusion_equation).



**BUTTERWORTH FILTER:**

The Butterworth filter is a type of [signal processing filter](https://en.wikipedia.org/wiki/Filter_(signal_processing)) designed to have a [frequency response](https://en.wikipedia.org/wiki/Frequency_response) as flat as possible in the [passband](https://en.wikipedia.org/wiki/Passband). It is also referred to as a maximally flat magnitude filter. It was first described in 1930 by the British [engineer](https://en.wikipedia.org/wiki/Engineer) and [physicist](https://en.wikipedia.org/wiki/Physicist) [Stephen Butterworth](https://en.wikipedia.org/wiki/Stephen_Butterworth) in his paper entitled "On the Theory of Filter Amplifiers"

The frequency response of the Butterworth filter is maximally flat (i.e. has no [ripples](https://en.wikipedia.org/wiki/Ripple_(filters))) in the passband and rolls off towards zero in the stopband.[[2]](https://en.wikipedia.org/wiki/Butterworth_filter#cite_note-bianchi-2) When viewed on a logarithmic [Bode plot](https://en.wikipedia.org/wiki/Bode_plot), the response slopes off linearly towards negative infinity. A first-order filter's response rolls off at −6 dB per [octave](https://en.wikipedia.org/wiki/Octave_(electronics)) (−20 dB per [decade](https://en.wikipedia.org/wiki/Decade_(log_scale))) (all first-order lowpass filters have the same normalized frequency response). A second-order filter decreases at −12 dB per octave, a third-order at −18 dB and so on. Butterworth filters have a monotonically changing magnitude function with ω, unlike other filter types that have non-monotonic ripple in the passband and/or the stopband.

Compared with a [Chebyshev](https://en.wikipedia.org/wiki/Chebyshev_filter) Type I/Type II filter or an [elliptic filter](https://en.wikipedia.org/wiki/Elliptic_filter), the Butterworth filter has a slower [roll-off](https://en.wikipedia.org/wiki/Roll-off), and thus will require a higher order to implement a particular [stopband](https://en.wikipedia.org/wiki/Stopband) specification, but Butterworth filters have a more linear phase response in the pass-band than Chebyshev Type I/Type II and elliptic filters can achieve.

**MATLAB SCRIPT:**

clc;

clear all;

close all;

Input\_Img = imread('D:\STUDY\DIP\Test img\girl512.bmp');

[m,n] = size(Input\_Img);

figure(1);

imshow(Input\_Img);

title('Original Image');

Image\_Double = im2double(Input\_Img);

Img\_DFT = fft2(Image\_Double);

figure(2);

subplot(1,2,1);

imshow(Img\_DFT);

title('Fourier Transform of Original Image');

Img\_DFT\_Shift = log(1 + abs(fftshift(Img\_DFT)));

subplot(1,2,2);

imshow(Img\_DFT\_Shift);

title('Shifted Fourier Transform of Original Image');

a = 0:(m-1);

b = 0:(n-1);

i = find(a > m/2);

a(i) = a(i)-m;

j = find(b > n/2);

b(j)=b(j)-n;

[k,l]=meshgrid(b,a);

D=sqrt((k.^2+l.^2));

figure(3);

mesh(real(fftshift(D)));

title('D');

Image1 = Ideal\_Low\_Pass\_Filter(40,D);

Image2 = Ideal\_Low\_Pass\_Filter(60,D);

Image3 = Gaussian\_Low\_Pass\_Filter(16,D);

Image4 = Gaussian\_Low\_Pass\_Filter(24,D);

Image5 = Butterworth\_Low\_Pass\_Filter(50,3,D);

Image6 = Butterworth\_Low\_Pass\_Filter(50,4,D);

Image7 = Ideal\_High\_Pass\_Filter(30,D);

Image8 = Ideal\_High\_Pass\_Filter(50,D);

Image9 = Gaussian\_High\_Pass\_Filter(16,D);

Image10 = Gaussian\_High\_Pass\_Filter(24,D);

Image11 = Butterworth\_High\_Pass\_Filter(50,3,D);

Image12 = Butterworth\_High\_Pass\_Filter(50,4,D);

Image1 = Image1.\*Img\_DFT;

Image1\_ifft = real(ifft2(Image1));

figure();

subplot(2,2,1);

imshow(log(1+abs(fftshift(Image1))));

title('filtered DFT image when D0 = 40');

subplot(2,2,2);

imshow(Image1\_ifft);

title('filtered Original image when D0 = 40');

Image2 = Image2.\*Img\_DFT;

Image2\_ifft = real(ifft2(Image2));

subplot(2,2,3);

imshow(log(1+abs(fftshift(Image2))));

title('filtered DFT image when D0 = 60');

subplot(2,2,4);

imshow(Image2\_ifft);

title('filtered Original image when D0 = 40');

Image3 = Image3.\*Img\_DFT;

Image3\_ifft = real(ifft2(Image3));

figure();

subplot(2,2,1);

imshow(log(1+abs(fftshift(Image3))));

title('filtered DFT image when sigma = 16');

subplot(3,2,2);

imshow(Image3\_ifft);

title('filtered Original image when sigma = 16');

Image4 = Image4.\*Img\_DFT;

Image4\_ifft = real(ifft2(Image4));

subplot(2,2,3);

imshow(log(1+abs(fftshift(Image4))));

title('filtered DFT image when sigma = 24');

subplot(2,2,4);

imshow(Image4\_ifft);

title('filtered Original image when sigma = 24');

Image5 = Image5.\*Img\_DFT;

Image5\_ifft = real(ifft2(Image5));

figure();

subplot(2,2,1);

imshow(log(1+abs(fftshift(Image5))));

title('filtered DFT image when n = 3 & D0 = 50');

subplot(2,2,2);

imshow(Image5\_ifft);

title('filtered Original image when n = 3 & D0 = 50');

Image6 = Image6.\*Img\_DFT;

Image6\_ifft = real(ifft2(Image6));

subplot(2,2,3);

imshow(log(1+abs(fftshift(Image6))));

title('filtered DFT image when n = 4 & D0 = 50');

subplot(2,2,4);

imshow(Image6\_ifft);

title('filtered Original image when n = 4 & D0 = 50');

Image7 = Image7.\*Img\_DFT;

Image7\_ifft = real(ifft2(Image7));

figure();

subplot(2,2,1);

imshow(log(1+abs(fftshift(Image7))));

title('filtered DFT image when D0 = 30');

subplot(2,2,2);

imshow(Image7\_ifft);

title('filtered Original image when D0 = 30');

Image8 = Image8.\*Img\_DFT;

Image8\_ifft = real(ifft2(Image8));

subplot(2,2,3);

imshow(log(1+abs(fftshift(Image8))));

title('filtered DFT image when D0 = 50');

subplot(2,2,4);

imshow(Image8\_ifft);

title('filtered Original image when D0 = 50');

Image9 = Image9.\*Img\_DFT;

Image9\_ifft = real(ifft2(Image9));

figure();

subplot(2,2,1);

imshow(log(1+abs(fftshift(Image9))));

title('filtered DFT image when sigma = 16');

subplot(2,2,2);

imshow(Image9\_ifft);

title('filtered Original image when sigma = 16');

Image10 = Image10.\*Img\_DFT;

Image10\_ifft = real(ifft2(Image10));

subplot(2,2,3);

imshow(log(1+abs(fftshift(Image10))));

title('filtered DFT image when sigma = 24');

subplot(2,2,4);

imshow(Image10\_ifft);

title('filtered Original image when sigma = 24');

Image11 = Image11.\*Img\_DFT;

Image11\_ifft = real(ifft2(Image11));

figure();

subplot(2,2,1);

imshow(log(1+abs(fftshift(Image11))));

title('filtered DFT image when n = 3 & D0 = 50');

subplot(2,2,2);

imshow(Image11\_ifft);

title('filtered Original image when n = 3 & D0 = 50');

Image12 = Image12.\*Img\_DFT;

Image12\_ifft = real(ifft2(Image12));

subplot(2,2,3);

imshow(log(1+abs(fftshift(Image12))));

title('filtered DFT image when n = 4 & D0 = 50');

subplot(2,2,4);

imshow(Image12\_ifft);

title('filtered Original image when n = 4 & D0 = 50');

Function for Ideal Low Pass Filter:

function H = Ideal\_Low\_Pass\_Filter(D0,D)

for u = 1:512

for v = 1:512

if(D(u,v) <= D0)

H(u,v) = 1;

else

H(u,v) = 0;

end

end

end

figure();

mesh(fftshift(H)),

str = sprintf('Low Pass Filter for D0 = %d',D0);

title(str);

axis tight;

figure();

imshow(fftshift(H));

str = sprintf('Low Pass Filter for D0 = %d',D0);

title(str);

end

%Function for Ideal High Pass Filter:

function H = Ideal\_High\_Pass\_Filter(D0,D)

for u = 1:512

for v = 1:512

if(D(u,v)<=D0)

H(u,v) = 0;

else

H(u,v) = 1;

end

end

end

figure();

mesh(fftshift(H)),

str = sprintf('High Pass Filter for D0 = %d',D0);

title(str);

axis tight;

figure();

imshow(fftshift(H));

str = sprintf('High Pass Filter for D0 = %d',D0);

title(str);

end

%Function for Gaussian Low Pass Filter:

function H = Gaussian\_Low\_Pass\_Filter(sigma,D)

for u = 1:512

for v = 1:512

H(u,v) = exp(-1\*(D(u,v)^2)/(2\*sigma^2));

end

end

figure();

mesh(fftshift(H)),

str = sprintf('Gaussian Low Pass Filter for sigma = %d',sigma);

axis tight;

title(str);

figure();

imshow(fftshift(H));

str = sprintf('Gaussian Low Pass Filter for sigma = %d',sigma);

title(str);

end

%Function for Gaussian High Pass Filter:

function H = Gaussian\_High\_Pass\_Filter(sigma,D)

for u = 1:512

for v = 1:512

H(u,v) = 1-exp(-1\*(D(u,v)^2)/(2\*sigma^2));

end

end

figure();

mesh(fftshift(H)),

str = sprintf('Gaussian High Pass Filter for sigma = %d',sigma);

axis tight;

title(str);

figure();

imshow(fftshift(H));

str = sprintf('Gaussian High Pass Filter for sigma = %d',sigma);

title(str);

end

%Function for Butterworth Low Pass Filter:

function H = Butterworth\_Low\_Pass\_Filter(D0,N,D)

for u=1:512

for v=1:512

H(u,v) = 1/(1+(D(u,v)/D0)^(2\*N));

end

end

figure();

mesh(fftshift(H)),

str = sprintf('Butterworth Low Pass Filter for D0 = %d & n = %d',D0,N);

title(str);

axis tight;

figure();

imshow(fftshift(H));

str = sprintf('Butterworth Low Pass Filter for D0 = %d & n = %d',D0,N);

title(str);

end

%Function for Butterworth High Pass Filter:

function H = Butterworth\_High\_Pass\_Filter(D0,N,D)

for u = 1:512

for v = 1:512

H(u,v) = 1/(1+(D0/D(u,v))^(2\*N));

end

end

figure();

mesh(fftshift(H)),

str = sprintf('Butter High Pass Filter for D0 = %d & n = %d',D0,N);

axis tight;

title(str);

figure();

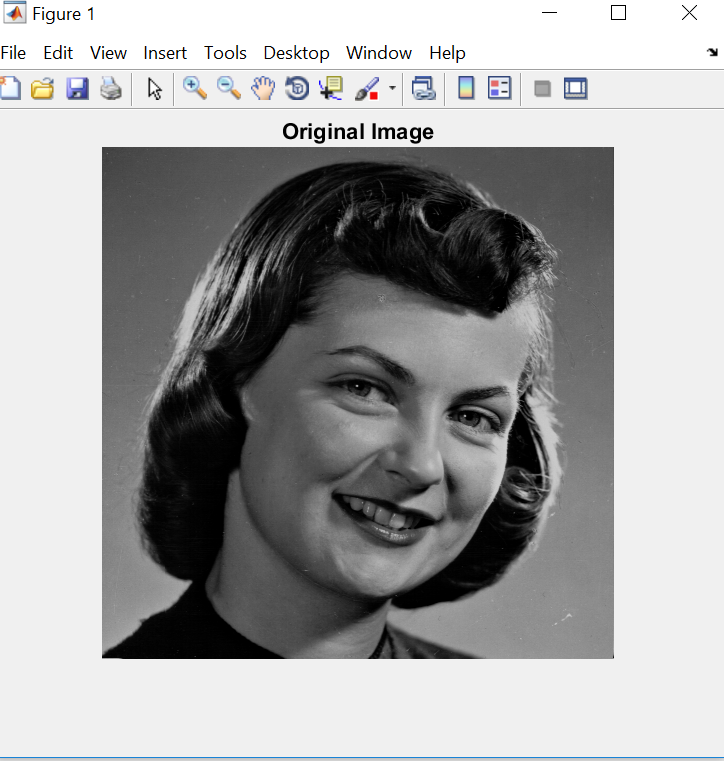
imshow(fftshift(H));

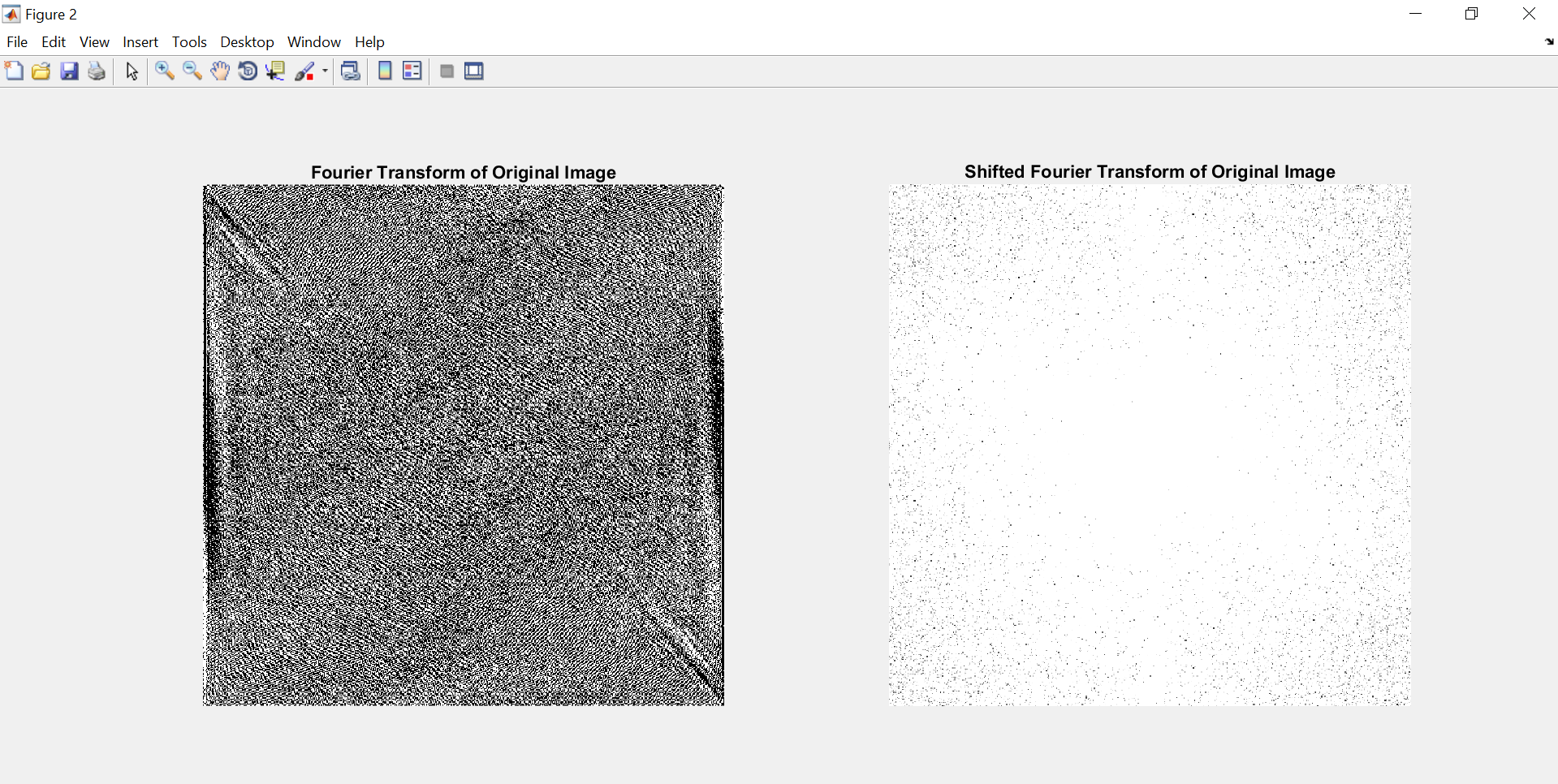
str = sprintf('Butterworth High Pass Filter for D0 = %d & n = %d',D0,N);

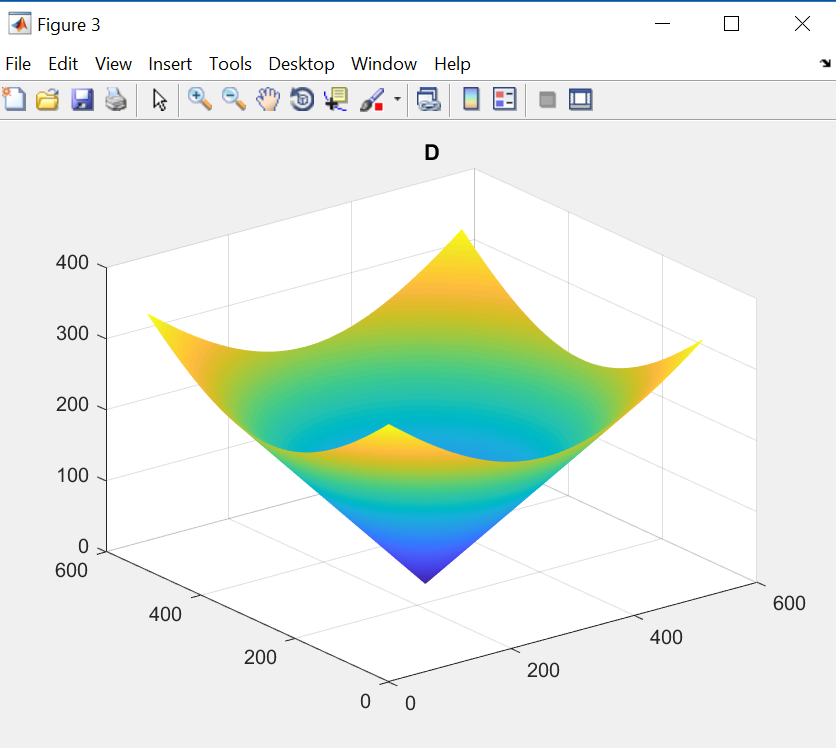
title(str);

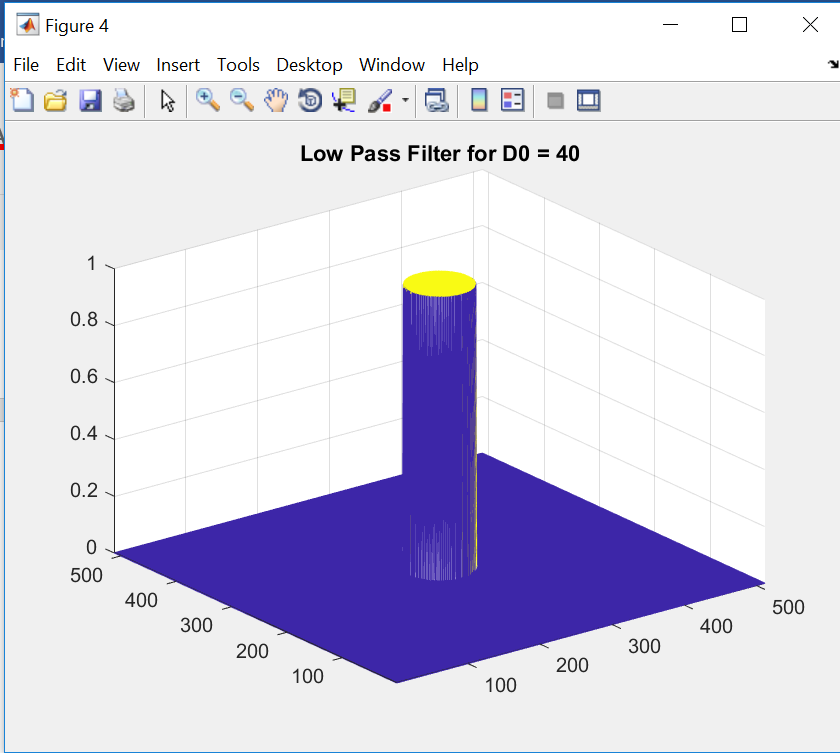
end

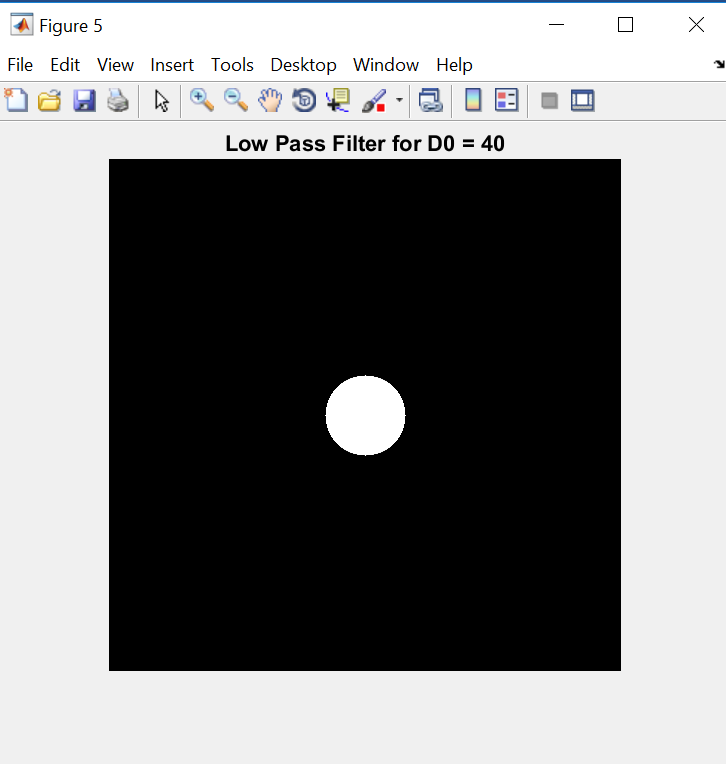
**OUTPUT:**

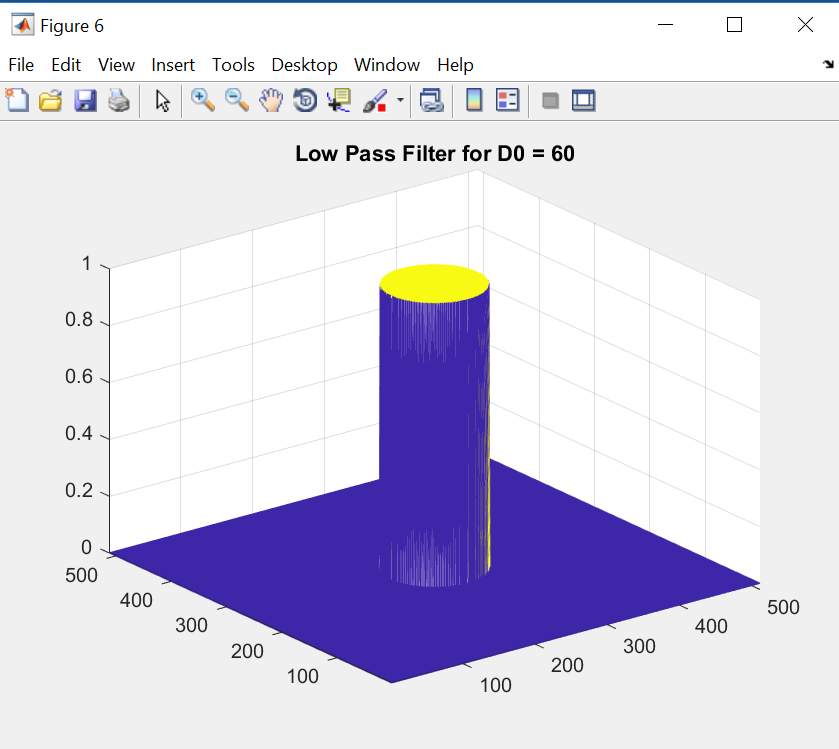
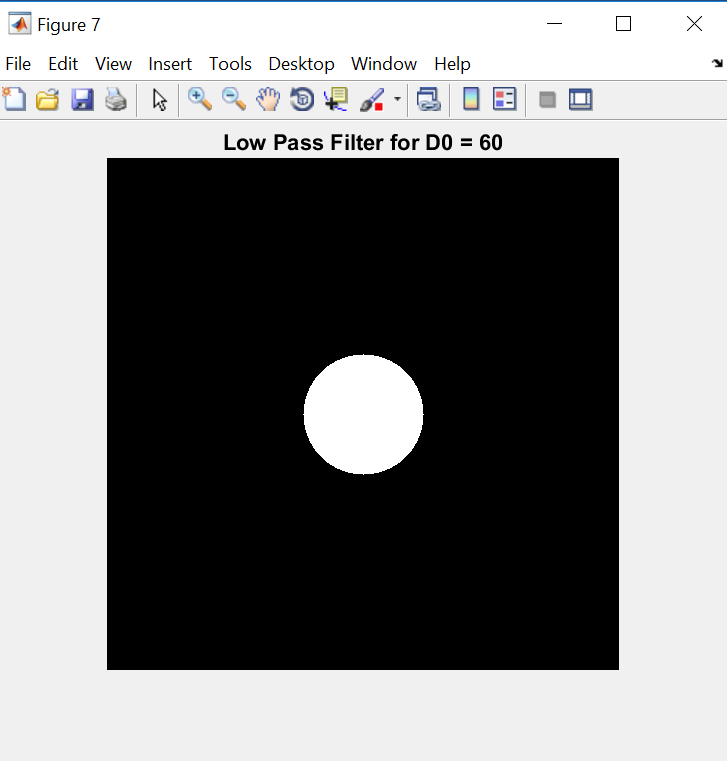


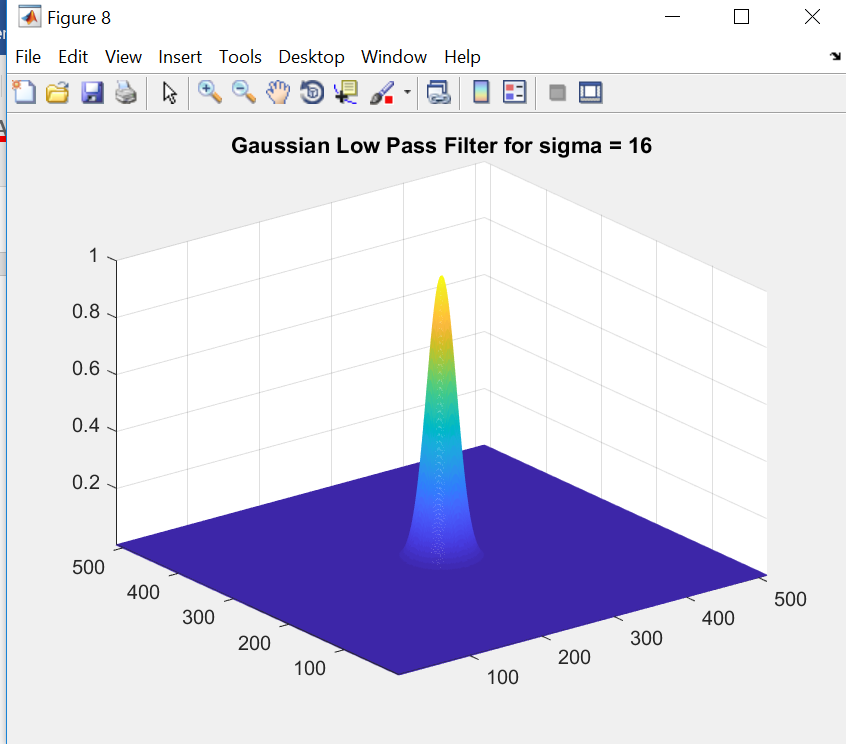
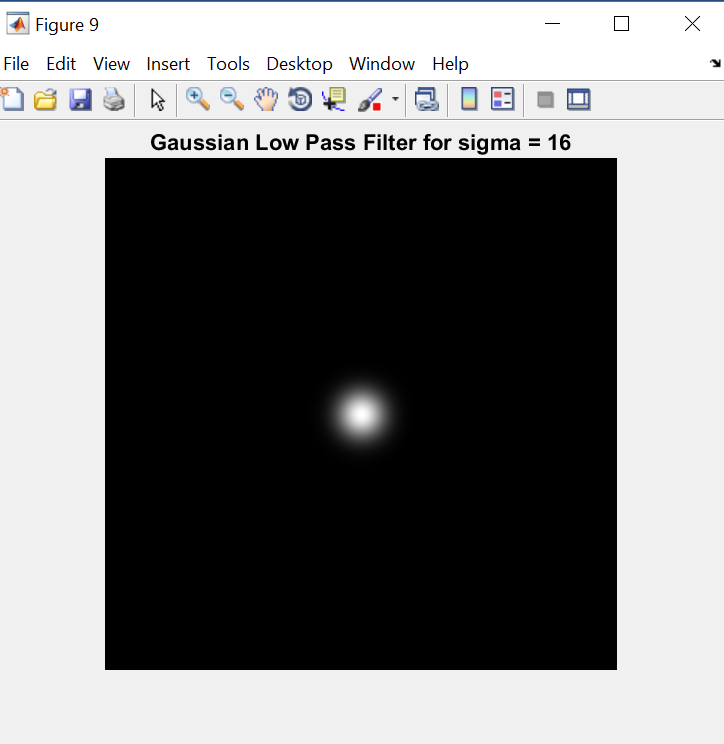


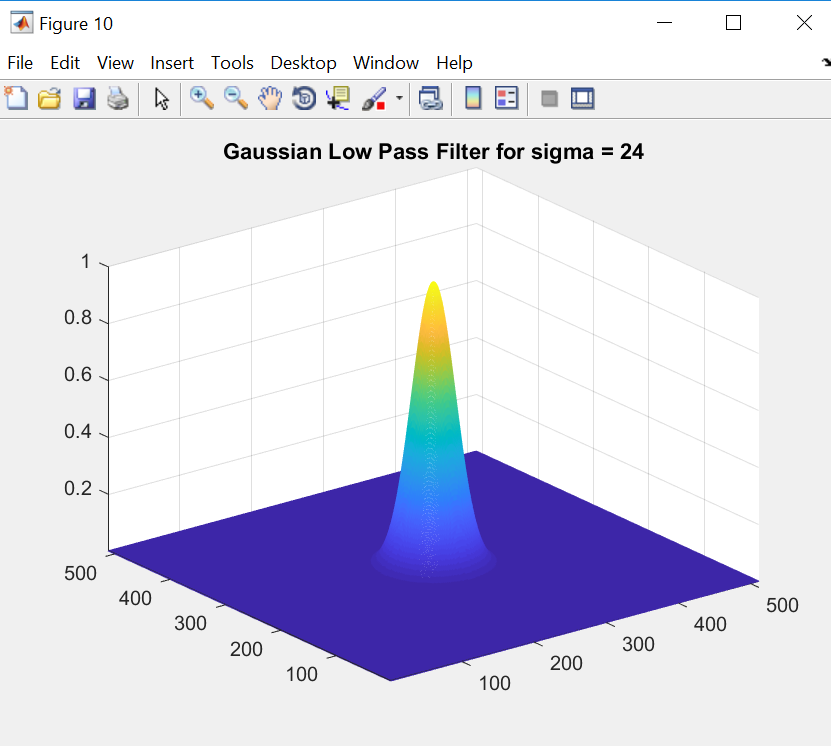


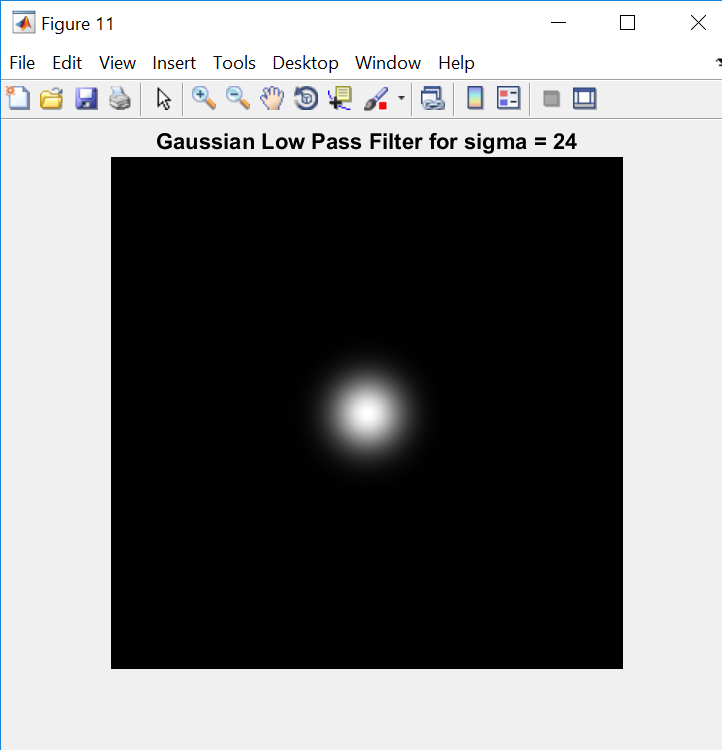


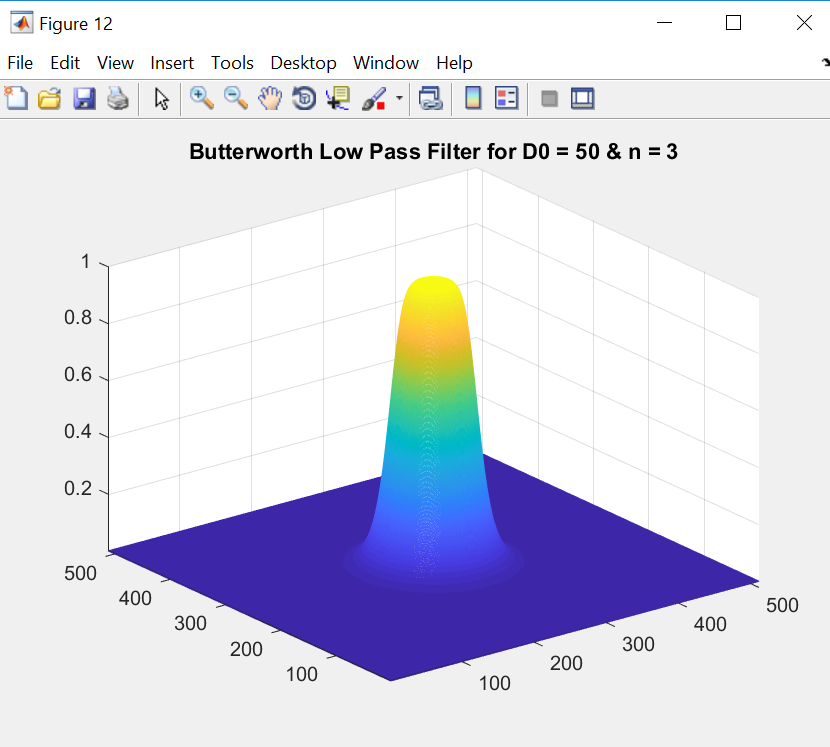
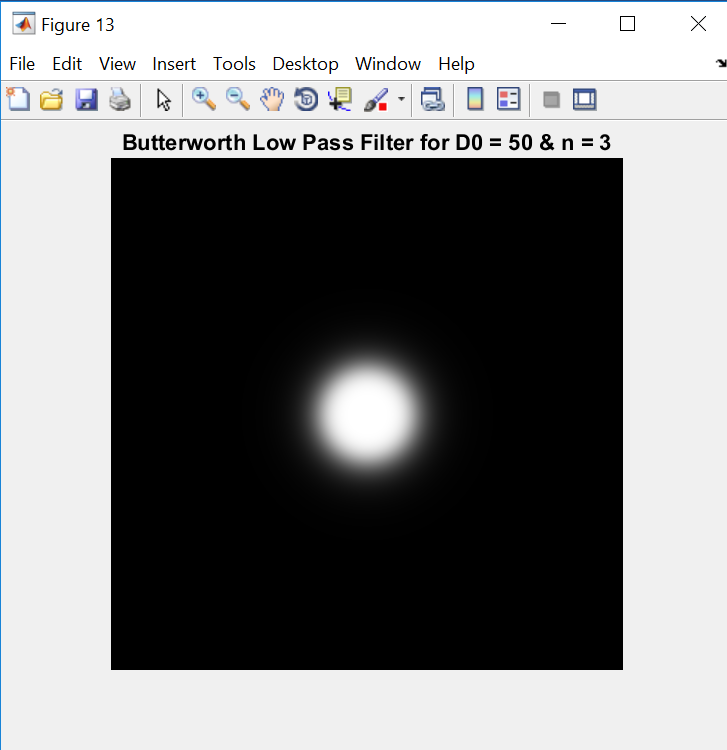


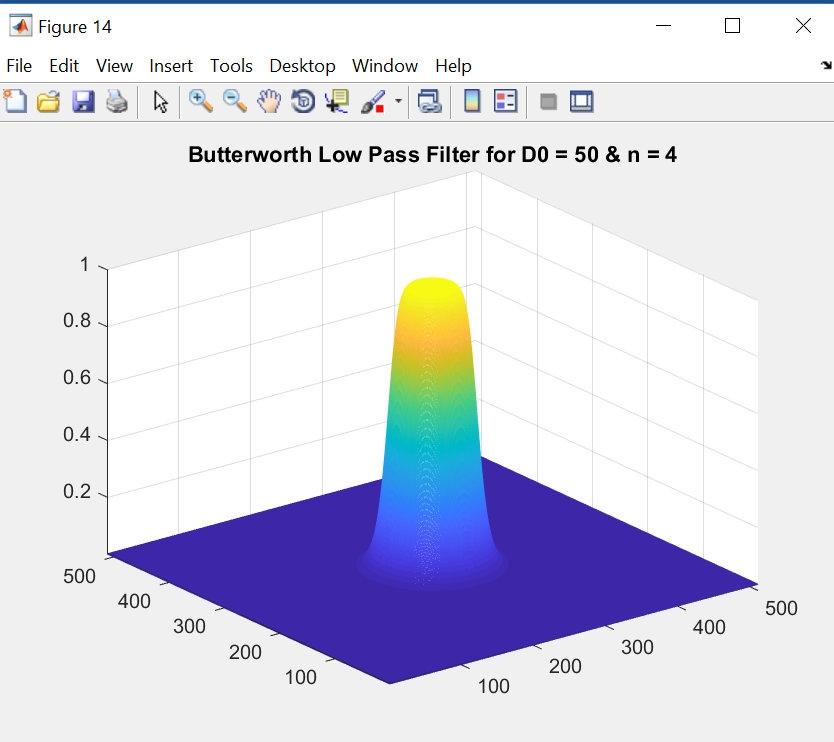
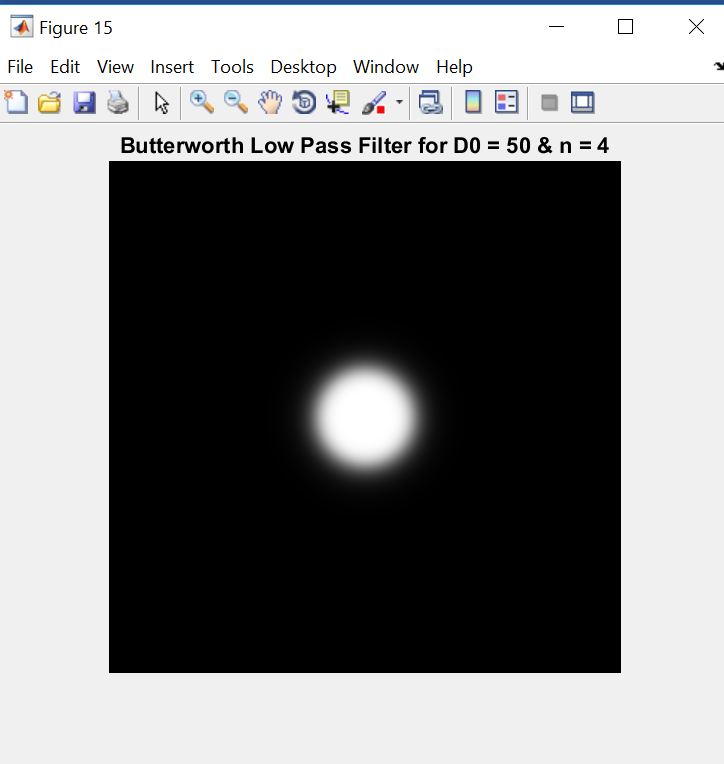


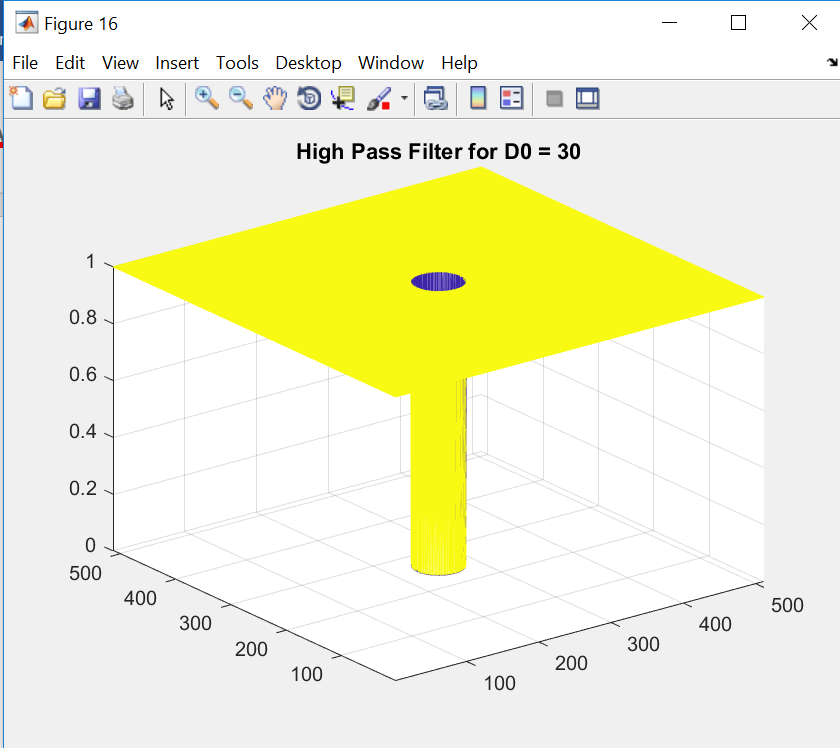
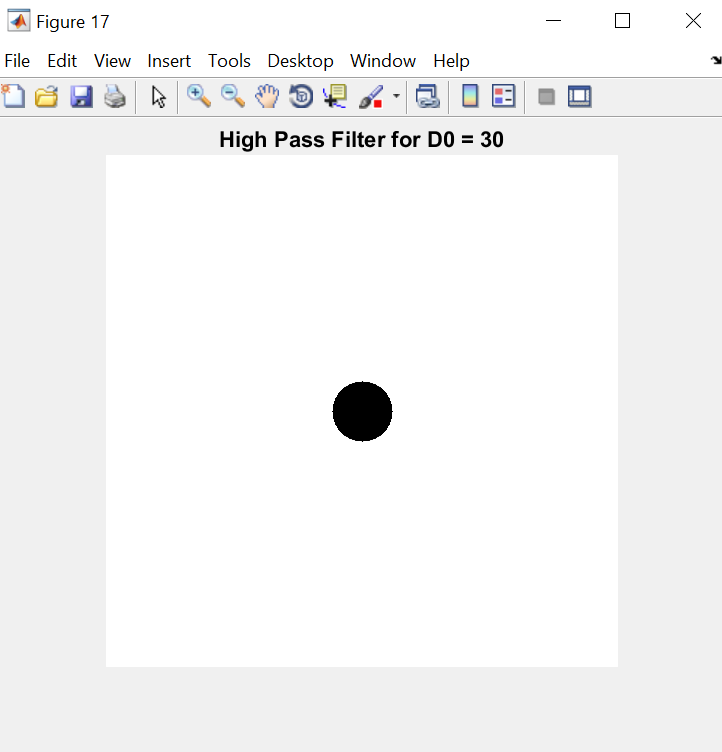


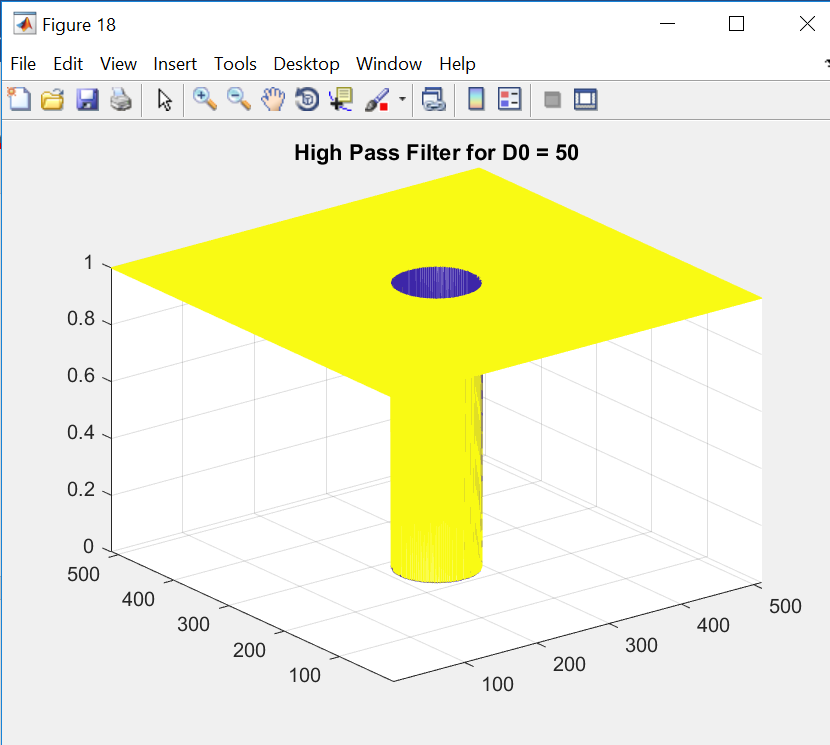


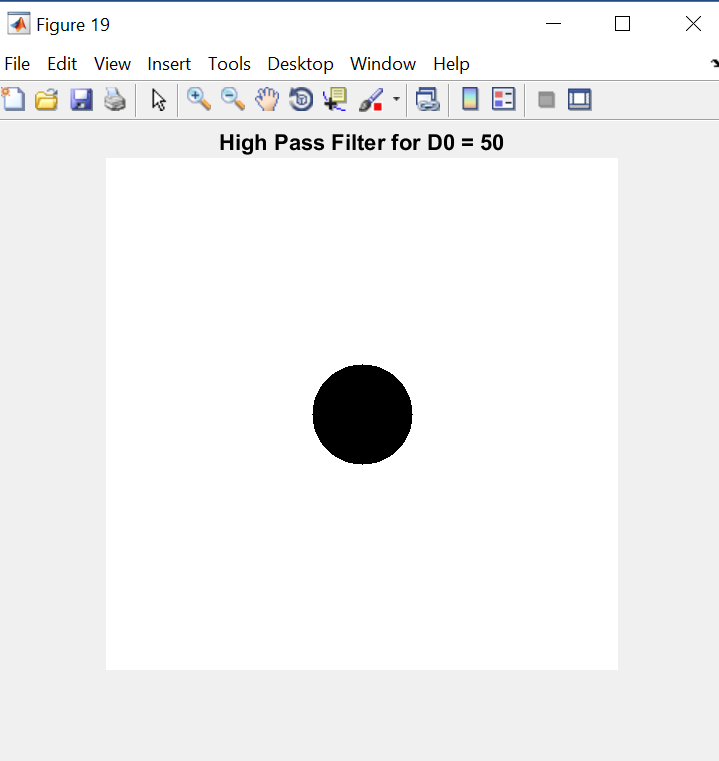


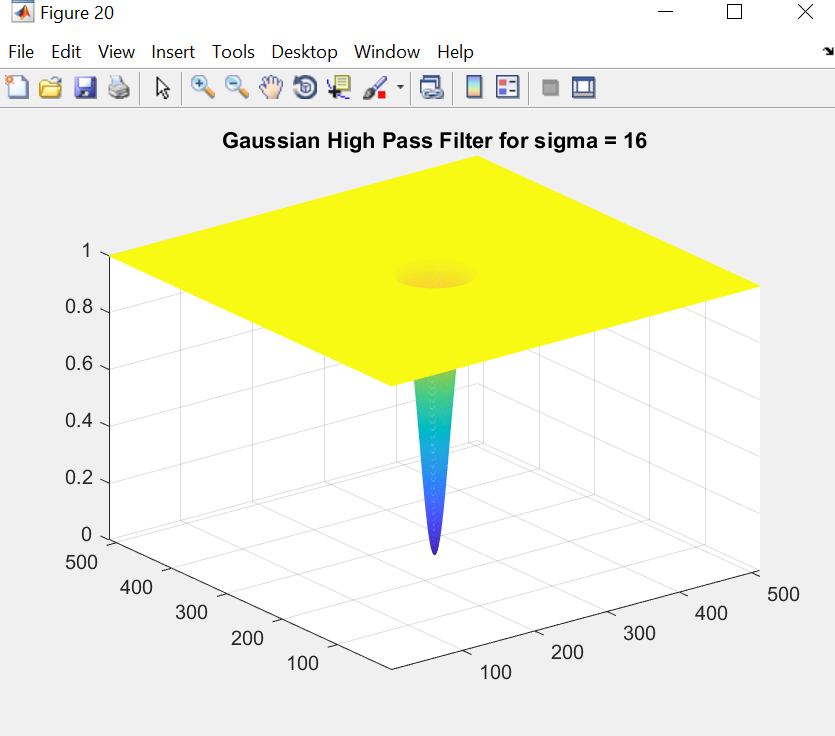
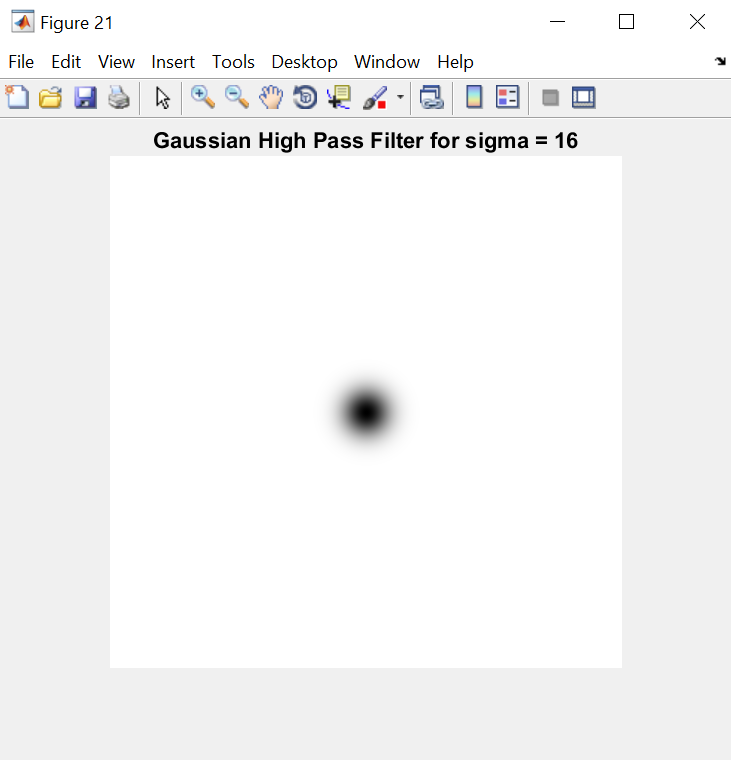


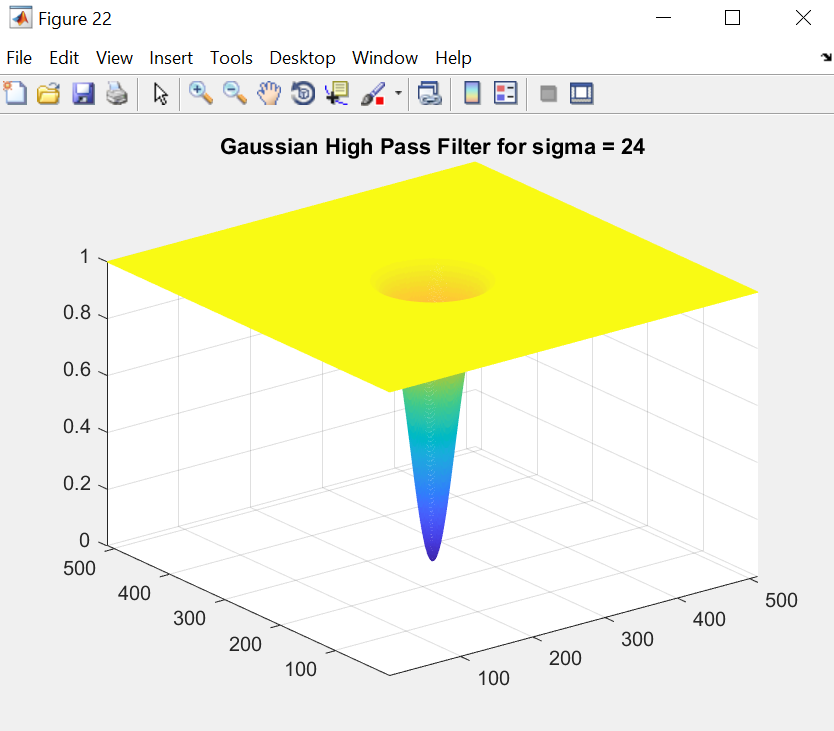


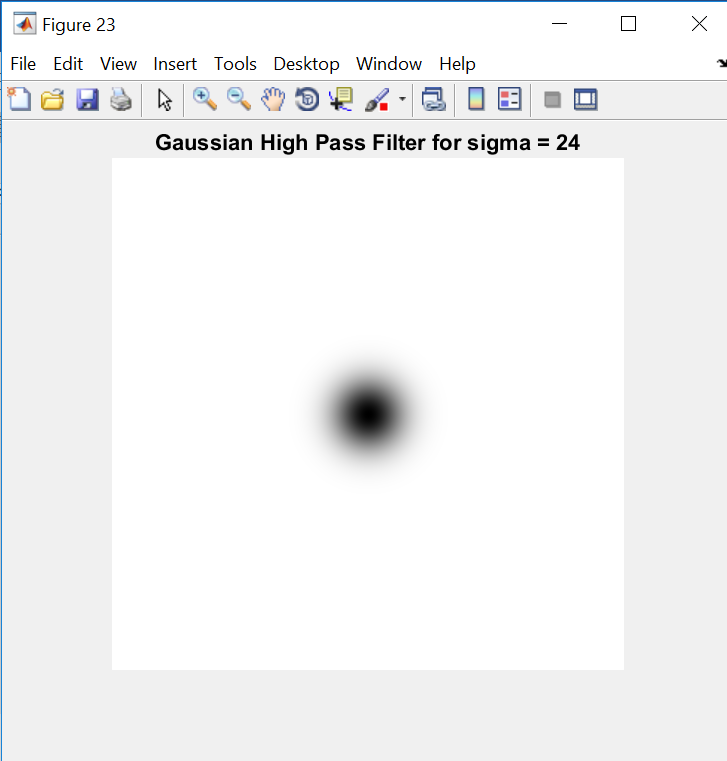


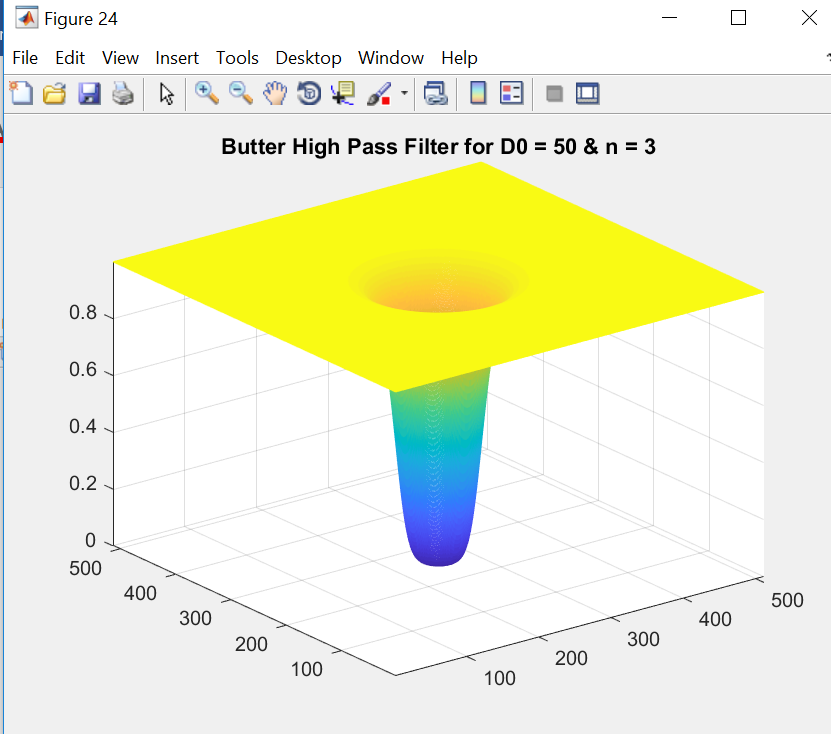


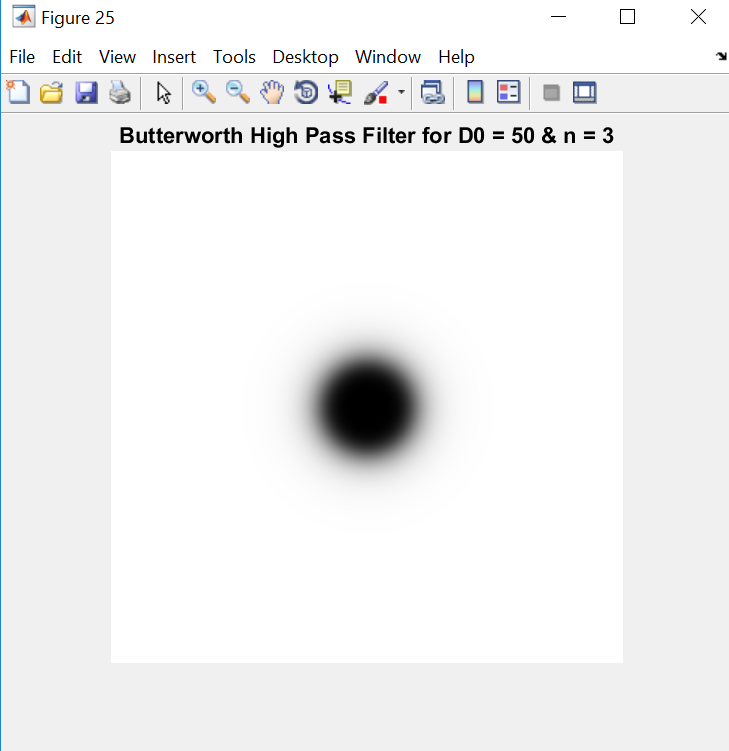


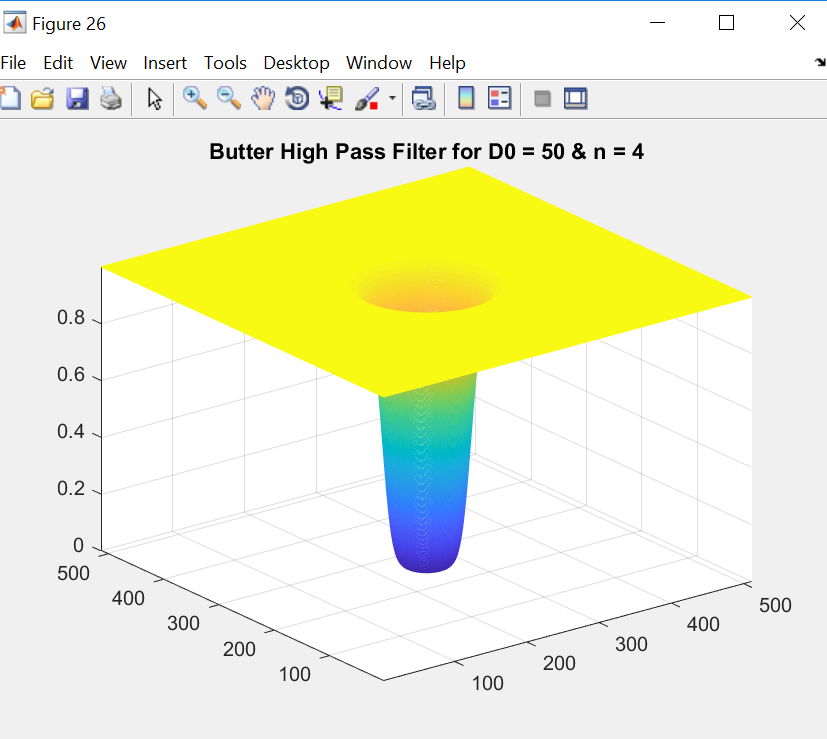


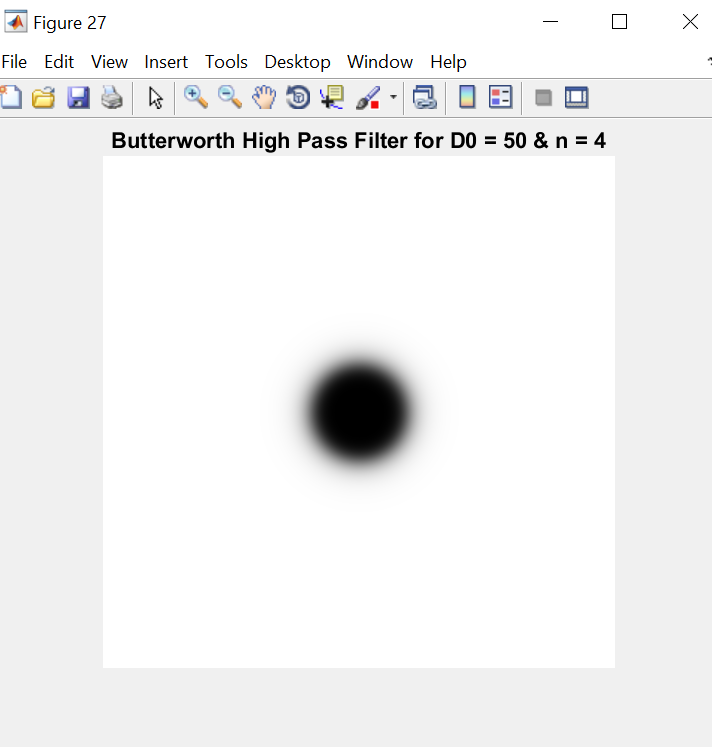


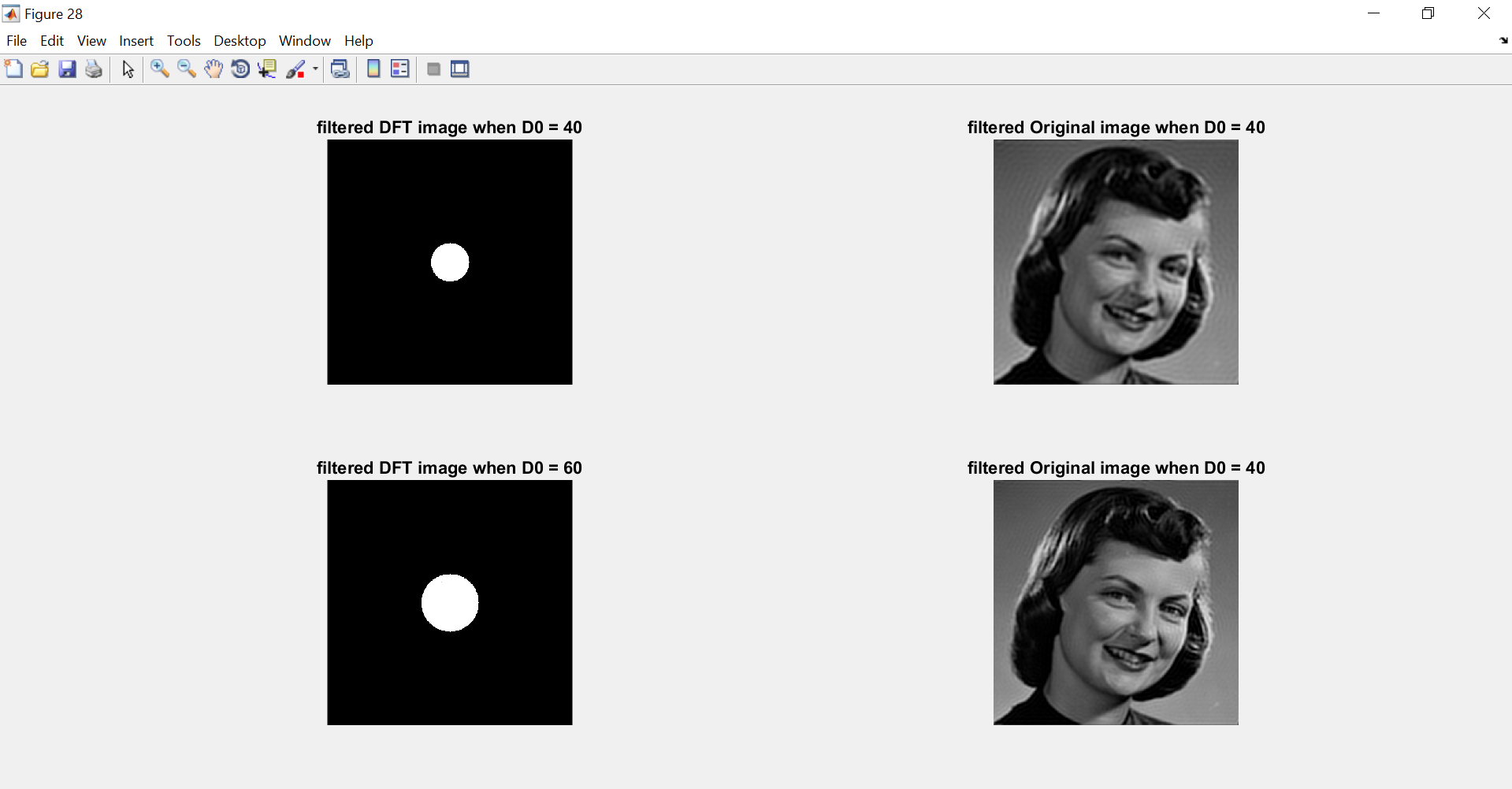


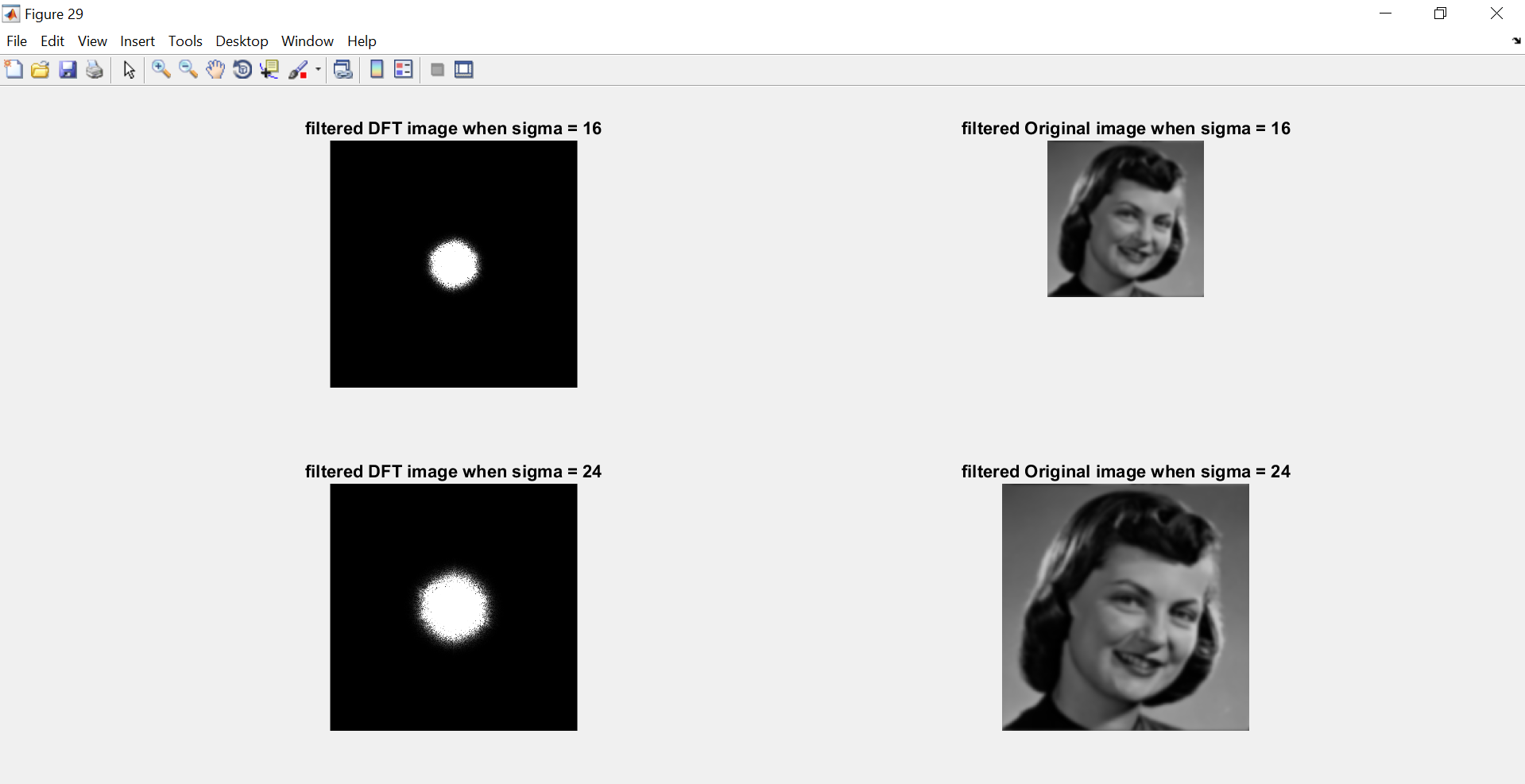








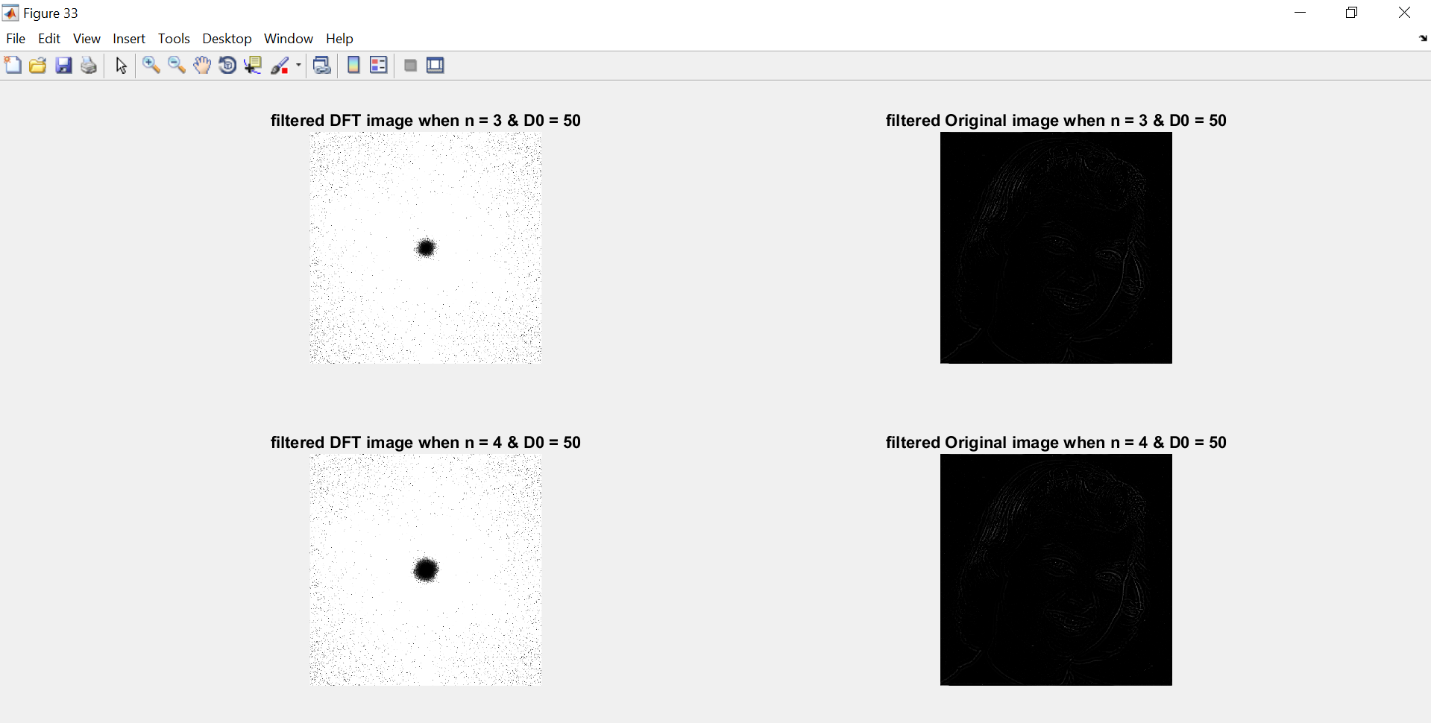












**EXPLAINATION:**

* A low-pass filter is a filter that passes low-frequency signals and attenuates (reduces the amplitude of) signals with frequencies higher than the cutoff frequency.
* A High-pass filter is a filter that passes High-frequency signals and attenuates signals with frequencies lower than the cutoff frequency.
* Gaussian low pass and Gaussian high pass filter minimize the problem that occur in ideal low pass and high pass filter.
* This problem is known as ringing effect. This is due to reason because at some points transition between one color to the other cannot be defined precisely, due to which the ringing effect appears at that point.

**CONCLUSION:**  
 This in this experiment, we studied about operation of various filters on the image and observed the variations at the output.