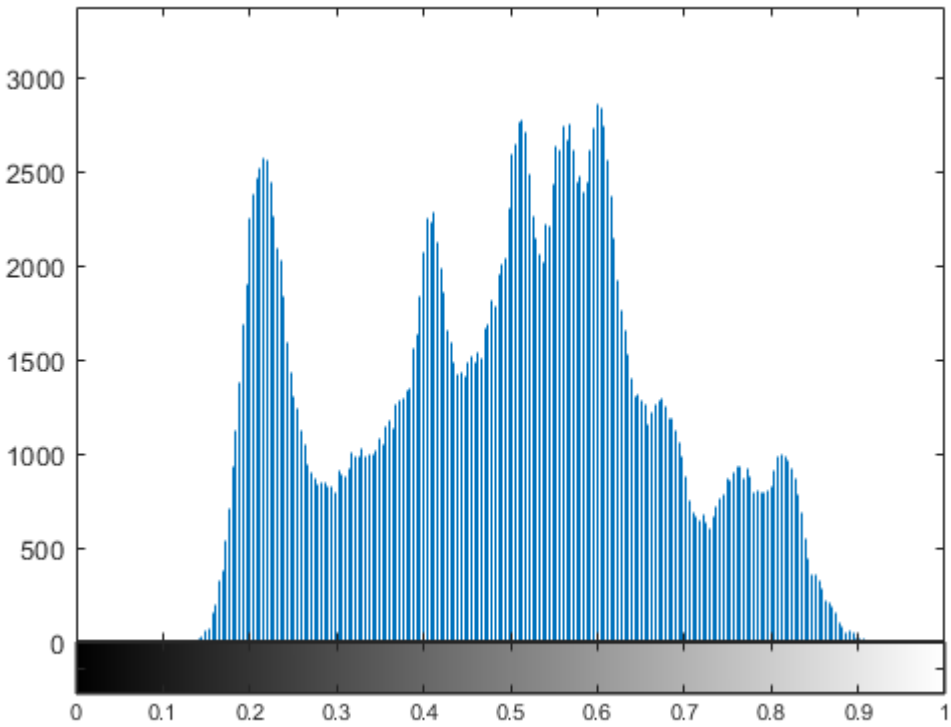
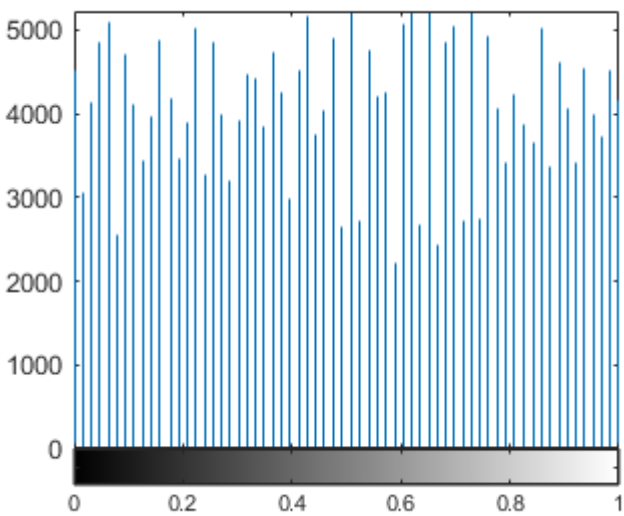
```
%Experiment 4-1
%section a
clc;
img = imread("lena.bmp");%loading image
imshow(img)%showing image
title('Original Image')

%section b
img = im2double(img);
%section c
figure;
imhist(img)%showing histogram
%section d
figure;
histeq(img)
%section e
imhist(histeq(img))
```


Original Image


3000
2500
2000
1500
1000
500
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


5000
4000
3000
2000
1000
0
0
0.2
0.4
0.6
0.8
1

```
%Experimnent 4-2
%section a
clc, close all;
img1 = imread("Image02.jpg");%loading image
img1 = im2double(img1);
imshow(img1)%showing image
title('Original Image')
```


```
%section b
J = imnoise(img1,'gaussian',0,0.2);%creating gaussian noise
figure;
imshow(J)
title('Gaussian noisy picture')
```


```
%section c
f = ones(3);
f = f/numel(f);
B = imfilter(J,f);%filtering the image
figure;
imshow(B)
title('Gaussian noisy filtered by 3x3 kernel')
```

Gaussian noisy filtered by 3x3 kernel

```
%section d
f1 = ones(5);
f1 = f1/numel(f1);
B1 = imfilter(J,f1);
figure;
imshow(B1)
```

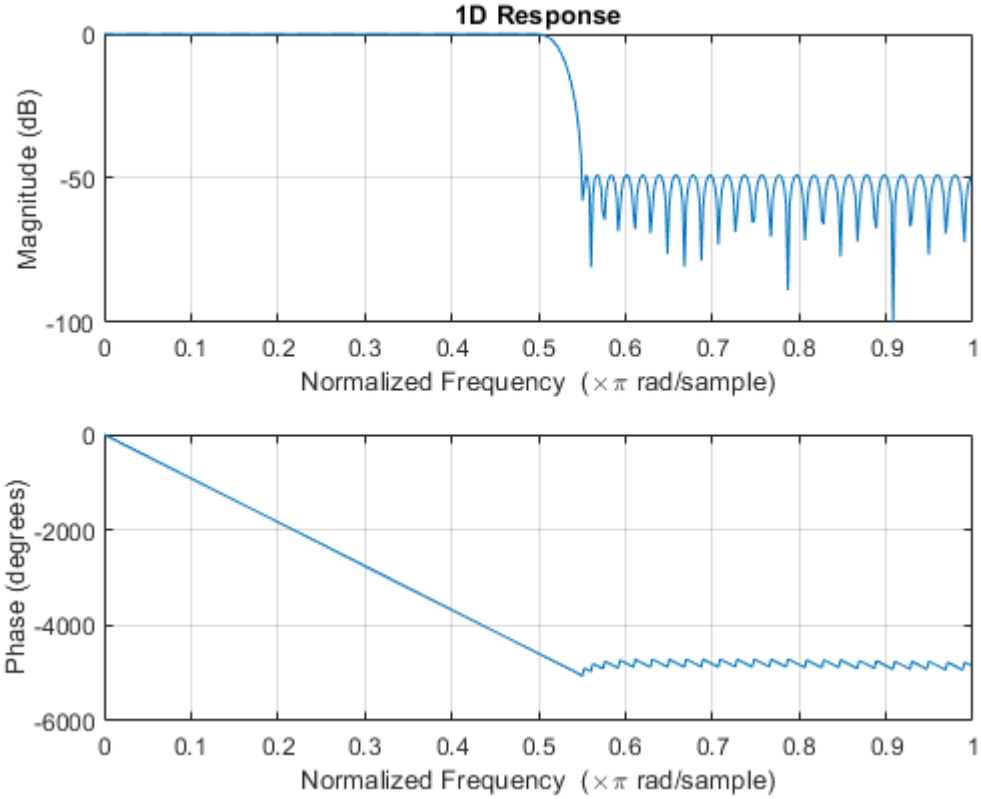
```
%section e
figure;
J1 = imnoise(img1,'salt & pepper',0.1);%creating slat & pepper noise
imshow(J1)
title('Salt & Pepper noisy picture')
```

Salt & Pepper noisy picture

```
%section f
f2 = ones(3);
f2 = f2/numel(f2);
B2 = imfilter(J1,f2);
```

```
figure;
imshow(B2)
title('Salt & Pepper noisy filtered by 3x3 kernel')
```


```
%section g
load("FIR1.mat")%loading filter
h = ftrans2(Num3);%converting to 2D
figure;
freqz(Num3)%1D resoponse
title('1D Response')
figure;
freqz2(h)%2D response
title('2D Response')
%testing the filter on gaussian noise
img_recovered = imfilter(J,h);
imshow(img_recovered)
title('filtering gaussian noise by filter designing')
%testing the filter on pepper & salt noise
img_recovered = imfilter(J1,h);
imshow(img_recovered)
title('filtering salt & pepper noise by filter disigning')
```



filtering salt & pepper noise by filter disigning

```
%section i
window = ones(3);
figure;
y1 = median_image(J,window);%filtering gaussian noise
imshow(y1)
title('filtered image implementing median method for gaussian noise')
figure;
y2 = median_image(J1,window);%filtering pepper & salt noise
imshow(y2)
title('filtered image implementing median method for pepper & salt noise')
```

image implementing median method for gaussia

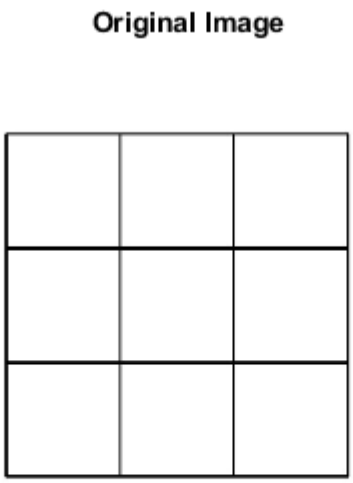
age implementing median method for pepper &

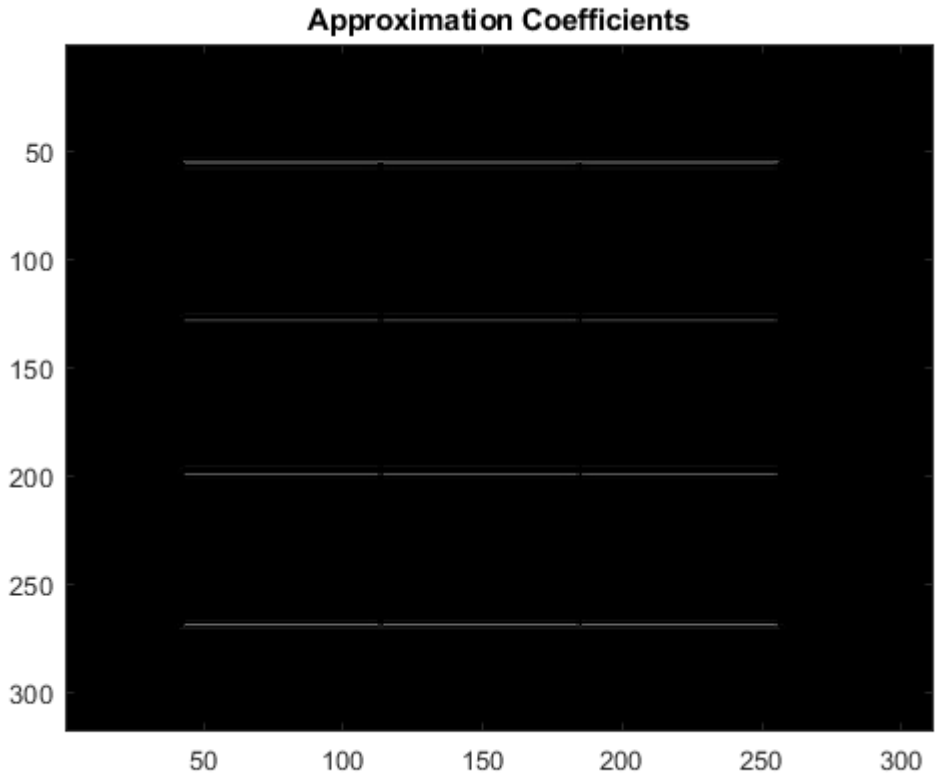
```
%section j
%comparing average and median methods
figure;
imshow(y2)%median method
title('median method for salt & pepper')
figure;
imshow(B2)%average method for pepper & salt
title('average method for salt & pepper')
```

median method for salt & pepper

average method for salt & pepper

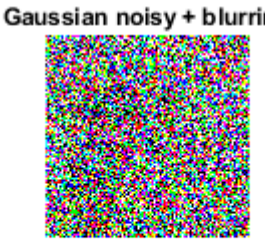
```
%Experiment 4-3
%section a
clc,close all;
img3 = imread("Image03.jpg");%loading image
imshow(img3)%showing image
title('Original Image')
img3 = im2double(img3);
[cA,cH,cV,cD] = dwt2(img3,'sym4','mode','per');
%section b
figure;
imagesc(cH)
title('Approximation Coefficients')
```






```
%Experiment 4-4
%section a
clc,close all;
img4 = imread("Image04.png");%loading image
imshow(img4)%showing image
title('Original Image')
img4 = im2double(img4);
PSF = fspecial('motion',15,20);
blurred = imfilter(img4,PSF,'conv','circular');
figure;
imshow(blurred)
title('Blurred Image')
%section b
nsr = 0;
u = deconvwnr(blurred,PSF,nsr);
figure;
```

```
imshow(u)
title('deblurring the image')
%section c
J3 = imnoise(blurred,'gaussian',0,10);%creating gaussian noise
figure;
imshow(J3)
title('Gaussian noisy + blurring')
%section d
nsr1 = 0.3;
u1 = deconvwnr(J3,PSF,nsr1);
figure;
imshow(u1)
title('Recovered Image')
```

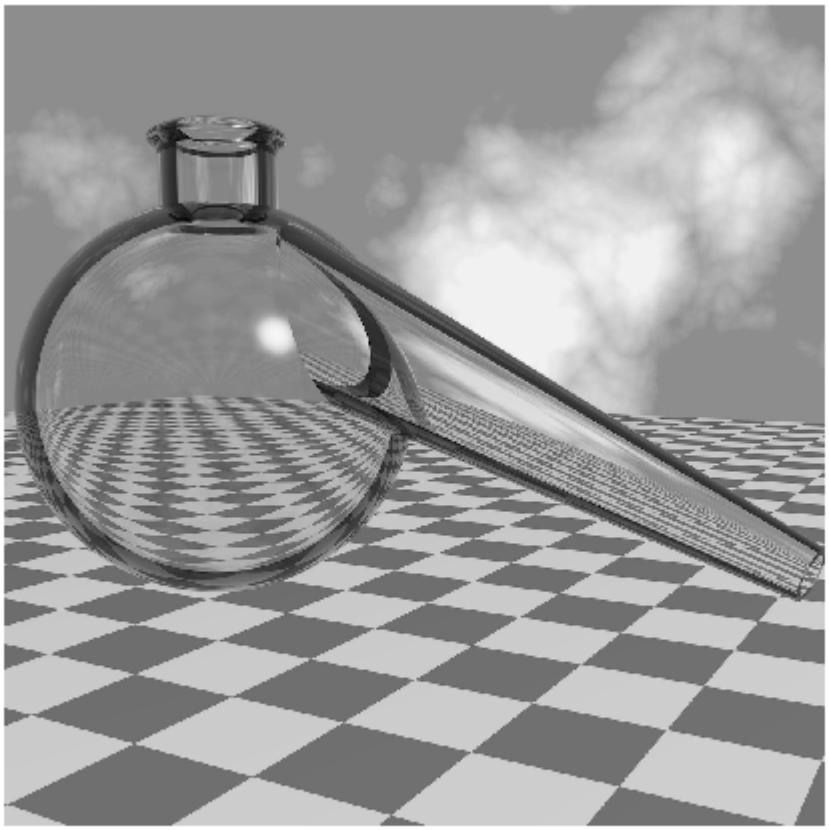



```
%Experiment 4-5
%section a
clc,close all;
img5 = imread("glass.tif");%loading image
imshow(img5)%showing image
title('Original Image')
img5 = im2double(img5);
%section b
Y5 = fftshift(fft2(img5));
figure;
imagesc(log10(abs(Y5)))
title('Amplitude')
```

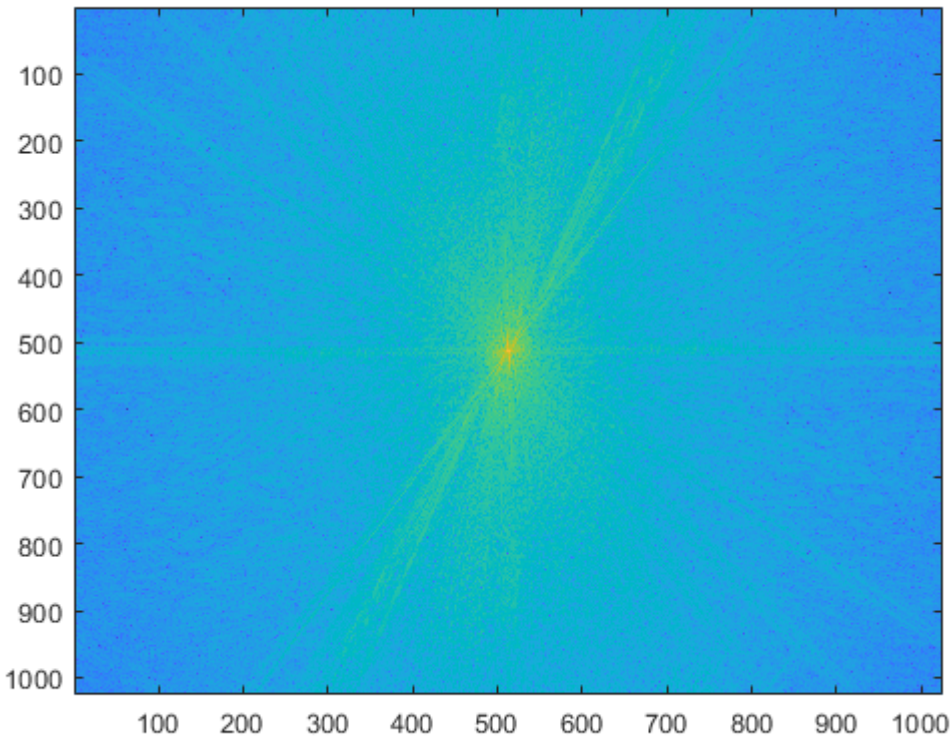
```
figure;
imagesc(angle(Y5))
title('Phase')
%section d
cutoff_frequency = 0.1 * pi;
Output_image = FFT_LP_2D(img5, cutoff_frequency);
figure;
imshow(Output_image)
title('filtering the original image')
%section e
img5_downsample = downsample(img5,2);
figure;
imshow(img5_downsample)
title('downsampling the original image')
Output_image1 = FFT_LP_2D(img5_downsample, cutoff_frequency);
figure;
imshow(Output_image1)
title('filtering the downsampled image')
```

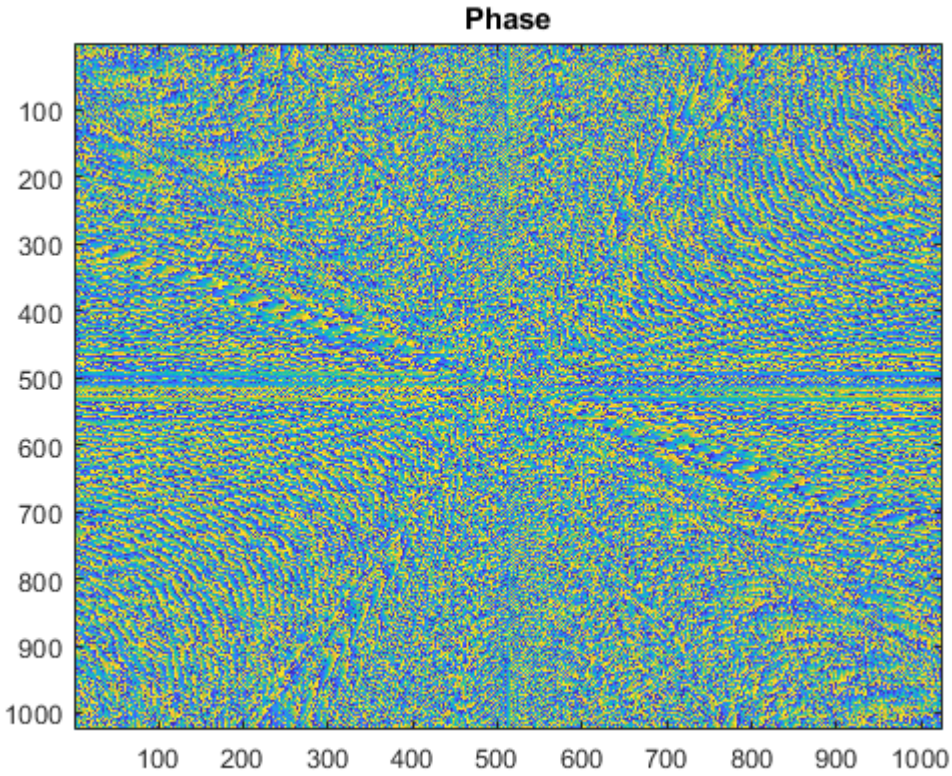
```
%section h (Experiment 4-2)
function y = median_image(img,window)
sz1 = size(img);%obtaining the size of image matrix
sz2 = size(window);%obtaining the size of window matrix
if rem(sz2,2)==[0 0]
    warning('the size of the window must be odd')
else
    tmp = zeros(sz2);%creating a temporary matrix with the size of window
    y_dim1 = sz1(1)-sz2(1)+1;%the number of rows in the output
    y_dim2 = sz1(2)-sz2(2)+1;%the number of columns in the output
    y_dim3 = sz1(3);%the number of heights in the output
    y = zeros(y_dim1,y_dim2,y_dim3);%initializing
  for z = 1:y_dim3
    for i=1:y_dim1
        for j=1:y_dim2
            tmp = img(i:sz2(1)+i-1,j:sz2(2)+j-1);
            y(i,j,z) = median(tmp,'all');
        end
    end
  end
end
end
```

Original Image



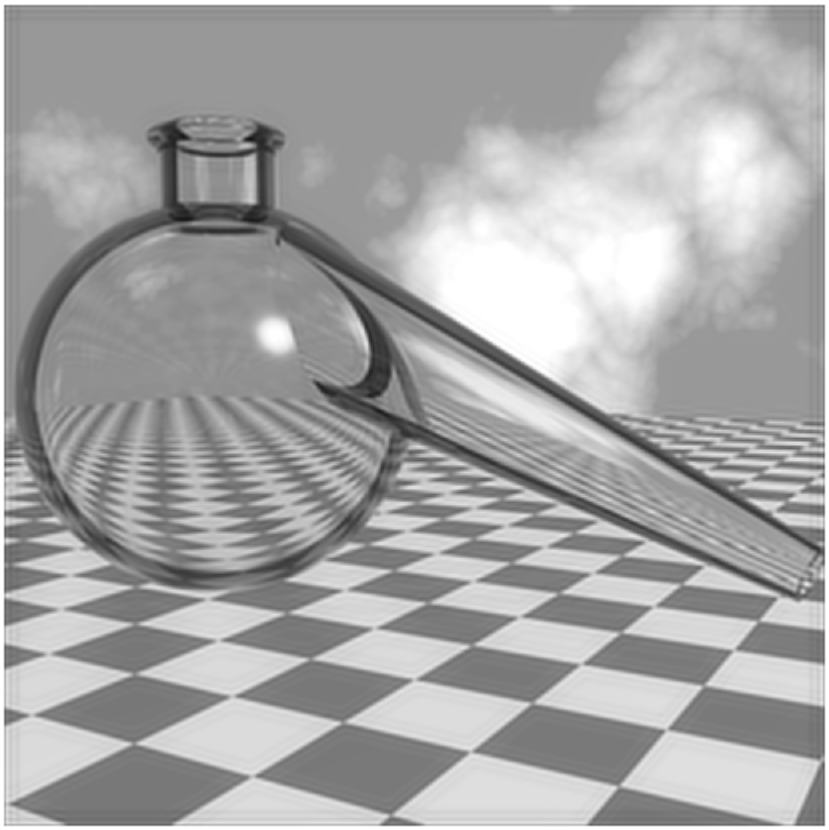
Amplitude



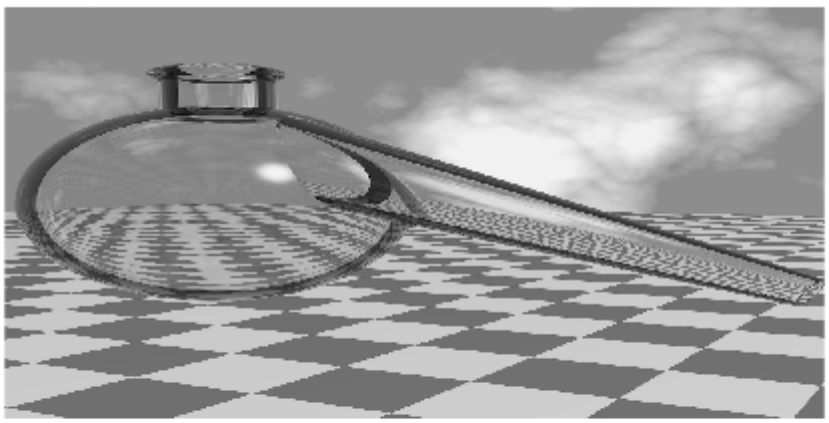


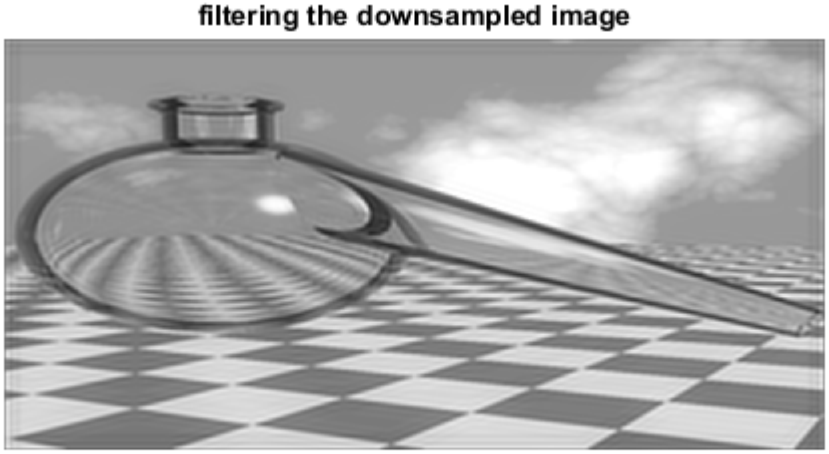

```
%section c (Experiment 4-5)
function Output_image = FFT_LP_2D(input_image, cutoff_frequency)
% Fs = 48000;  % Sampling Frequency

Fstop = cutoff_frequency;           % Stopband Frequency
Fpass = Fstop-0.1;             % Passband Frequency
Dpass = 0.057501127785;  % Passband Ripple
Dstop = 0.0001;          % Stopband Attenuation
dens  = 16;              % Density Factor

% Calculate the order from the parameters using FIRPMORD.
[N, Fo, Ao, W] = firpmord([Fpass, Fstop], [1 0], [Dpass, Dstop]);

% Calculate the coefficients using the FIRPM function.
b  = firpm(N, Fo, Ao, W, {dens});
Hd = dfilt.dffir(b);
Hd2 = ftrans2(Hd.Numerator);%converting to 2D
Output_image = imfilter(input_image,Hd2);
end
```

filtering the original image



downsampling the original image





Published with MATLAB® R2020b