ECE280 - Lab 7: Image Processing 2

Sylvester Johannes Arizie (sna31)

November 20, 2024

I have adhered to the Duke Community Standard in completing this assignment.

ECE 280L – Fall 2024

Image Processing 2

Contents

1 Exercises 1 and 2: Calculating Fourier Transforms for 1D Finite-Duration Discrete Data Sets, and Synthesizing Data Sets from Fourier Transforms

2 Exercise 3: Exploring Round LPFs Applied to Coins

In Exercise 3, we explored round Low Pass Filters which operate by essentially removing high-frequency noise or details from a signal, in this case an image, and the size of the filter (radius) determines how much detail is retained. When the radius is larger (e.g., 0.5), the filter allows more of the high-frequency components of the image to pass through, leading to a less smoothed image but finer details may be preserved. As the radius is reduced, (e.g., 0.1), the filter blocks more high-frequency components, which results in a more blurred and smooth image. Fine details such as edges are thus begin to disappear as the radius is decreased even further

3 Exercise 4: Exploring Round LPFs Applied to Noise

in Exercise 4, we work with a noise image whose pixel values are randomly generated between 0 and 255, whilst applying a circular Low Pass Filter with varying radii. For larger radii, the filter allows a broader range of low frequencies to pass through. This results in less filtering of noise, thus the image retains a good amount of high-frequency content. Although some portions appear blurred out, noise is still predominant in the final image thus not that much of a change is seen. As the radius gets smaller, the filter blocks more high- frequency content/ components and leads to more pronounced smoothing of the image. Thus the image gets more significantly blurred and the image loses clarity.

4 Exercise 5: Further Exploring Round LPFs Applied to Noise

For Exercise 5, 2D images and a 3D image are obtained. The image is more simplified compared to past images generated. It shows a Grey scale with dark hues. It looks flat and has a shift in brightness in certain areas. The 3D Surface plot shows the pixel values and color to represent the intensity of color/ hue at each point of the filtered image. It has undulations and while they both provide the same information in general, the surface plot shows a 3-dimensional perspective whilst the filtered image is in 2D(like a top-down view).

5 Exercise 6: Exploring Rectangular LPFs Applied to Coins

In this exercise, we explore Rectangular Low Pass Filters hence a need for 2 bounds for each side of the rectangle. These bounds approximately show the same behavior with changes in values. When —u— and —v— are set to large cut-offs, high frequency content pass through and thus the image still has some fineness in detail. As these bounds are reduced, the image loses higher frequencies and thus gets blurrier.

6 Exercise 7: Exploring Rectangular LPFs Applied to Noise

This behaves in the same way as in exercise 6. The only difference is the image used. The image retains more texture and sharp features, and the random noise pattern is more apparent for higher limits and reduces as the bounds are decreased, leading to blurring.

7 Exercise 8: Further Exploring Rectangular LPFs Applied to Noise

The filtered image looks brighter in Exercise 8 relative to Exercise 5. It has more grey sections than black and seems somewhat inverted as compared to exercise 5 for areas of brightness. The surface plot has more undulations and is more complex compared to that in exercise 5. Overall, the two images perform the same functions as they did in exercise 5. The image is in 2D with more brightness whilst the Surface plot is a complex 3D plot/ figure with interwoven undulations.

8 Exercise 9: Exploring Rectangular HPFs Applied to Coins

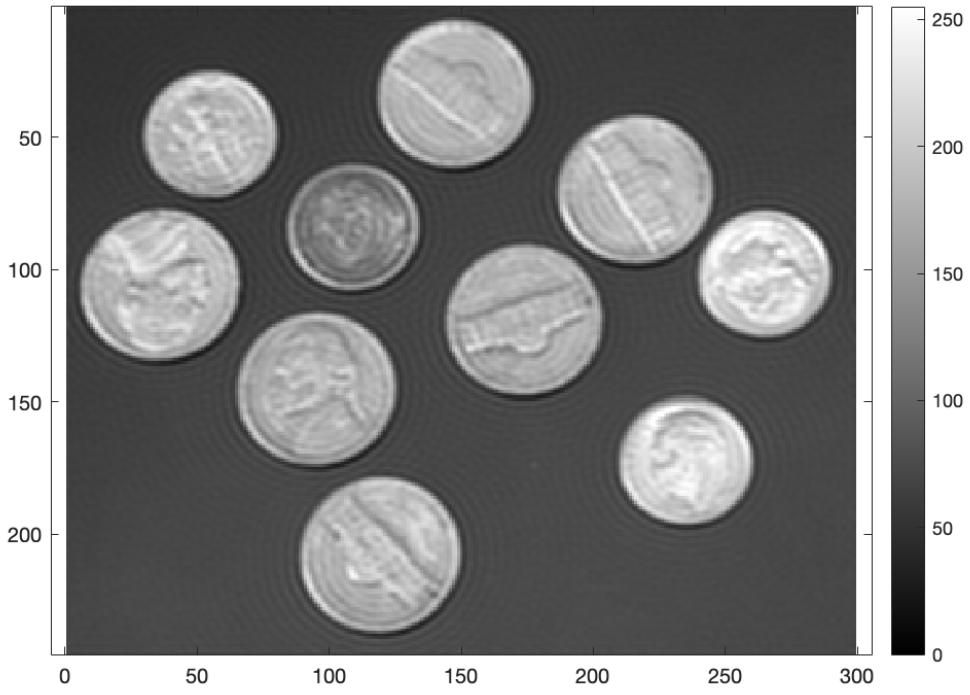
In this exercise, we explore the workings of a High Pass Filter, which operates by removing low frequency portions of a signal whilst maintaining the higher frequency content. For each —u— and —v—, the filter allows high frequency material outside the cutoffs to pass through and blocks otherwise. Increasing the limits removes more low frequency content and emphasizes fine edges of the image. Decreasing the limits allows more low frequency content to pass thus introducing blurriness. The directional nature of —u— and —v— enhances either horizontal or vertical edges.

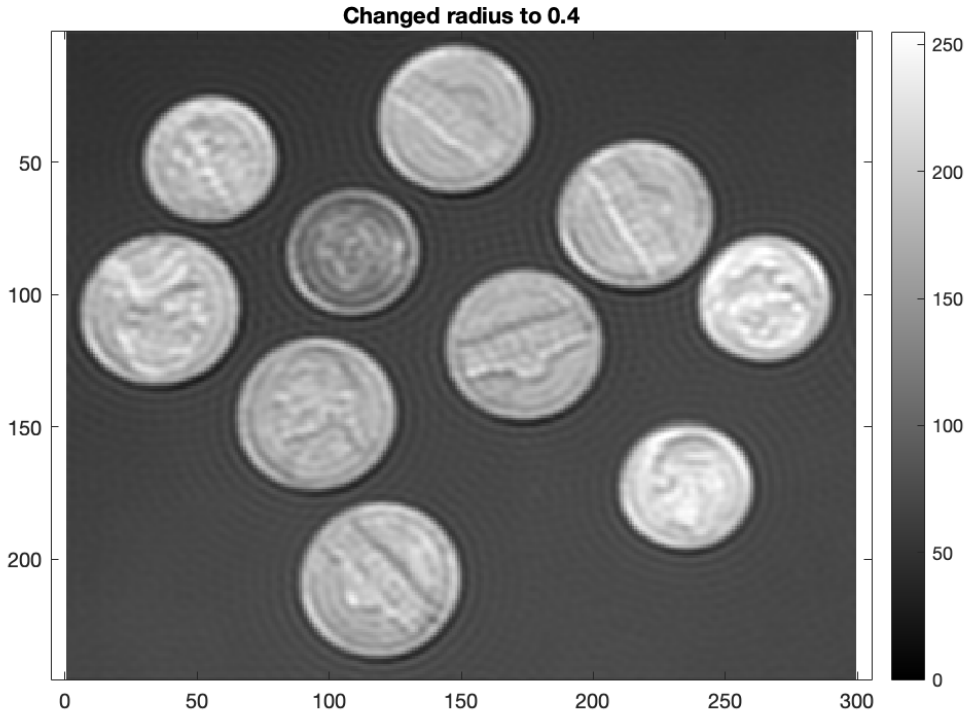
9 Exercise 10: Exploring Rectanguar BPFs Applied to Coins

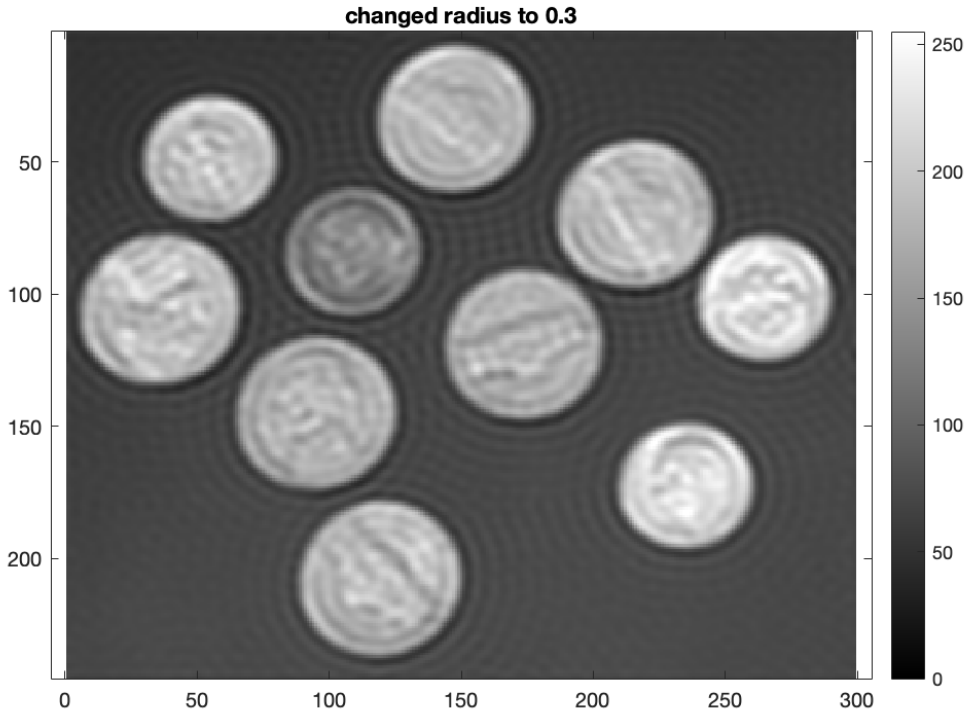
In this exercise, we explore Band Pass Filters which operate by allowing frequencies within specific ranges to pass through and blocking all frequencies outside that range. By applying the Band Pass Filter, we retain the middle frequency details which produces a "balanced" image. The image thus contains moderate texture and has moderate sharpness, without the noise from HPF or the smoothness from LPFs.

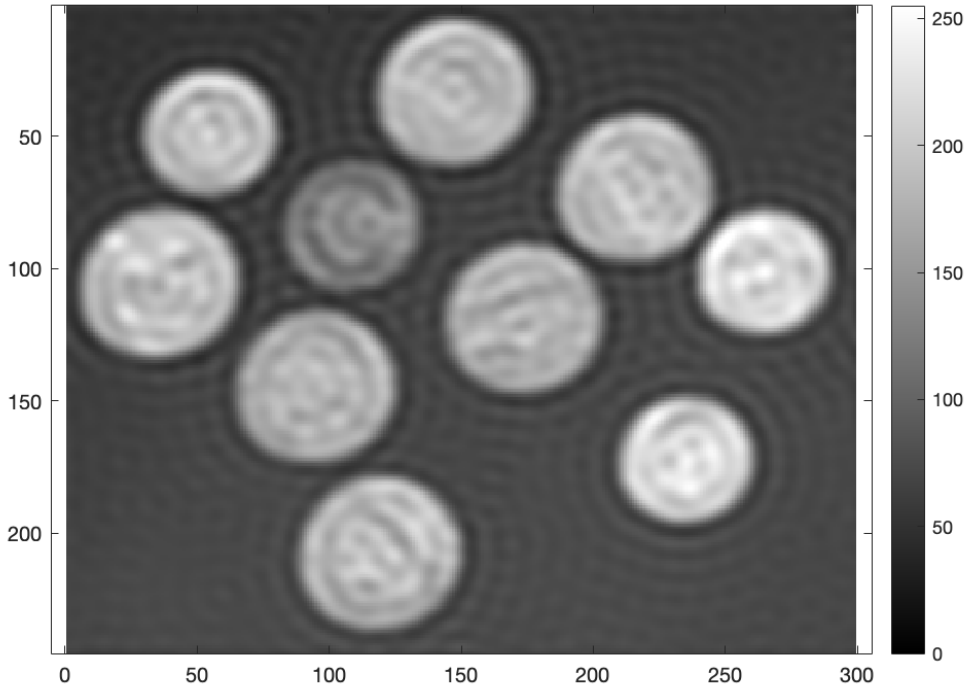
10 Graphs

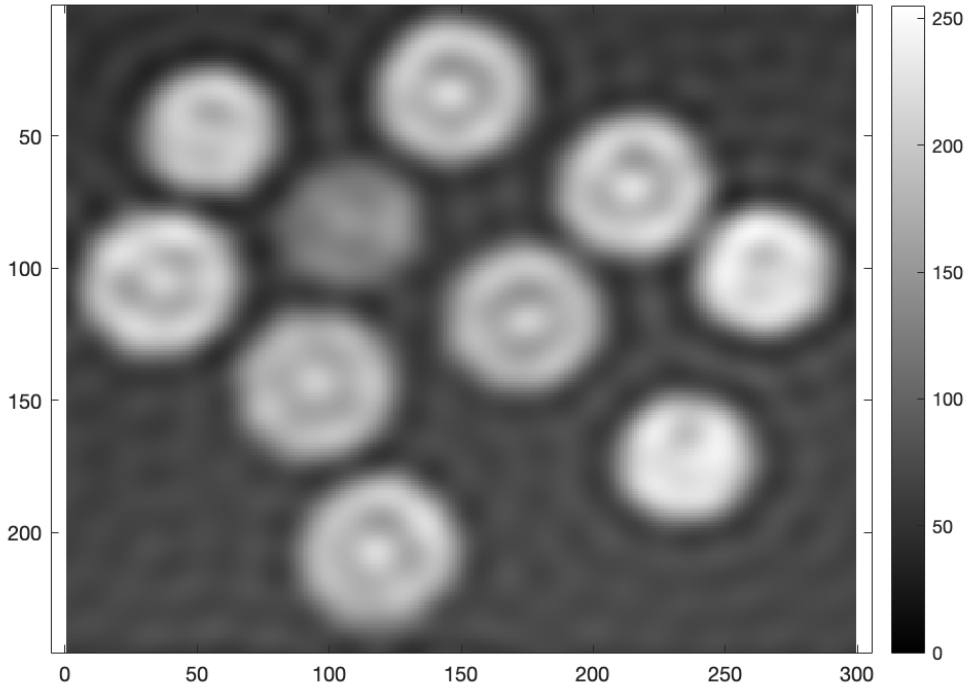
10.1 Exercise 3



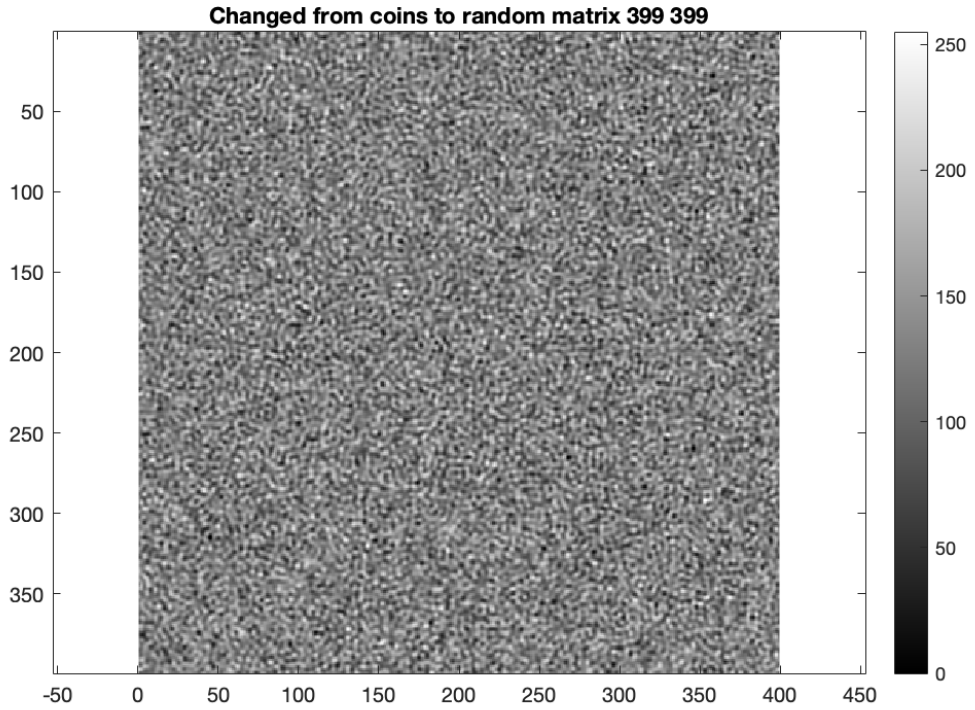


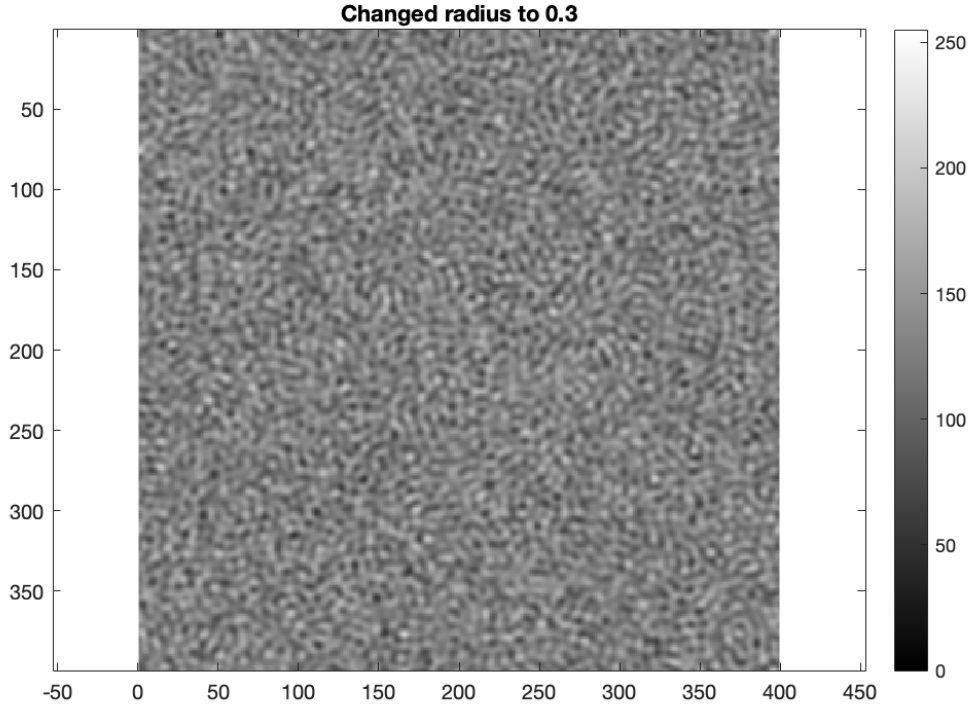

changed radius to 0.3
50
100
150
200
0
50
100
150
200
250
300
250
200
150
100
50
0

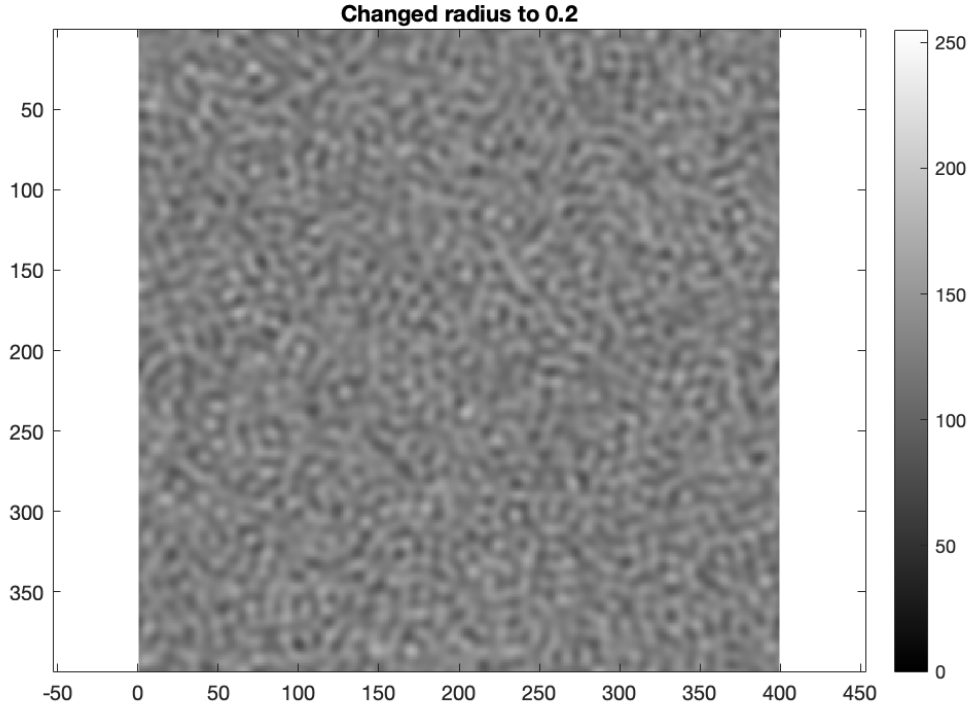

50
100
150
200
0
50
100
150
200
250
300
250
200
150
100
50
0

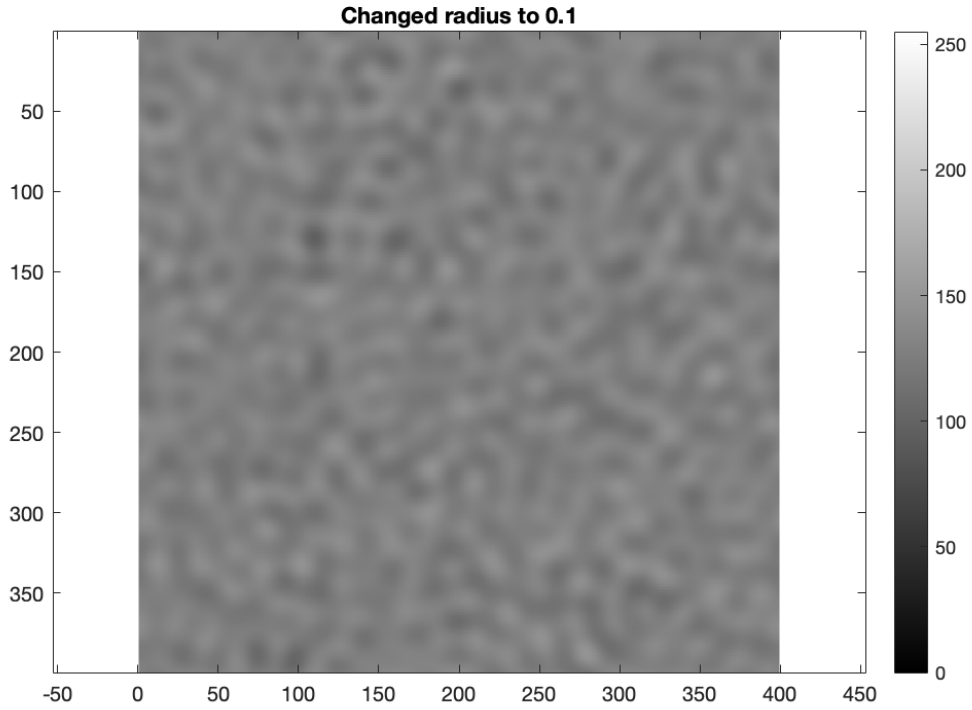

50
100
150
200
0
50
100
150
200
250
300
250
200
150
100
50
0

10.2 Exercise 4

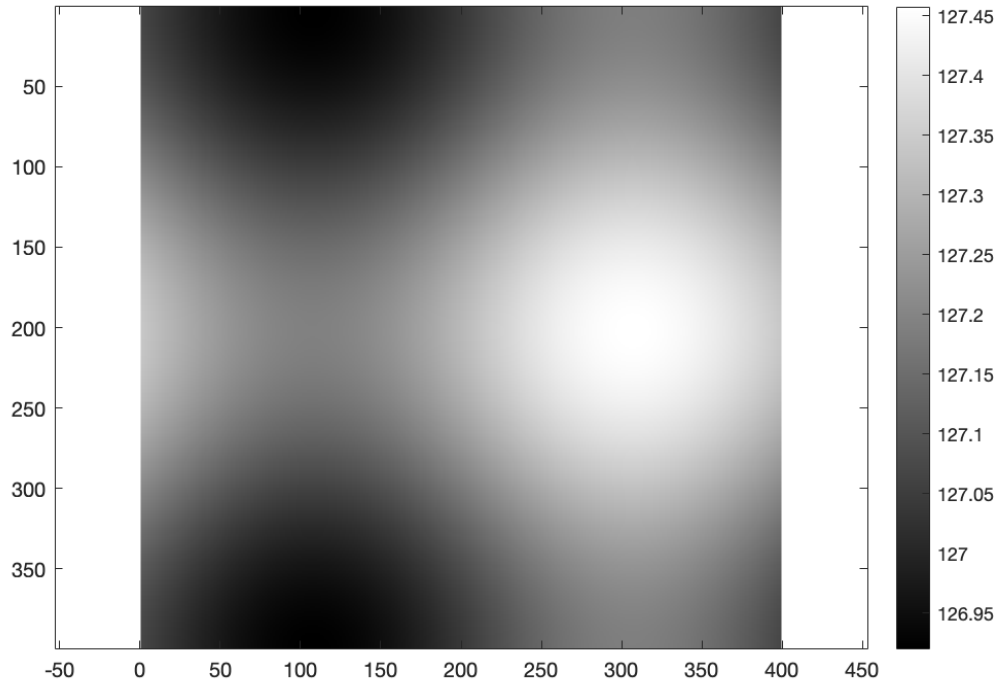


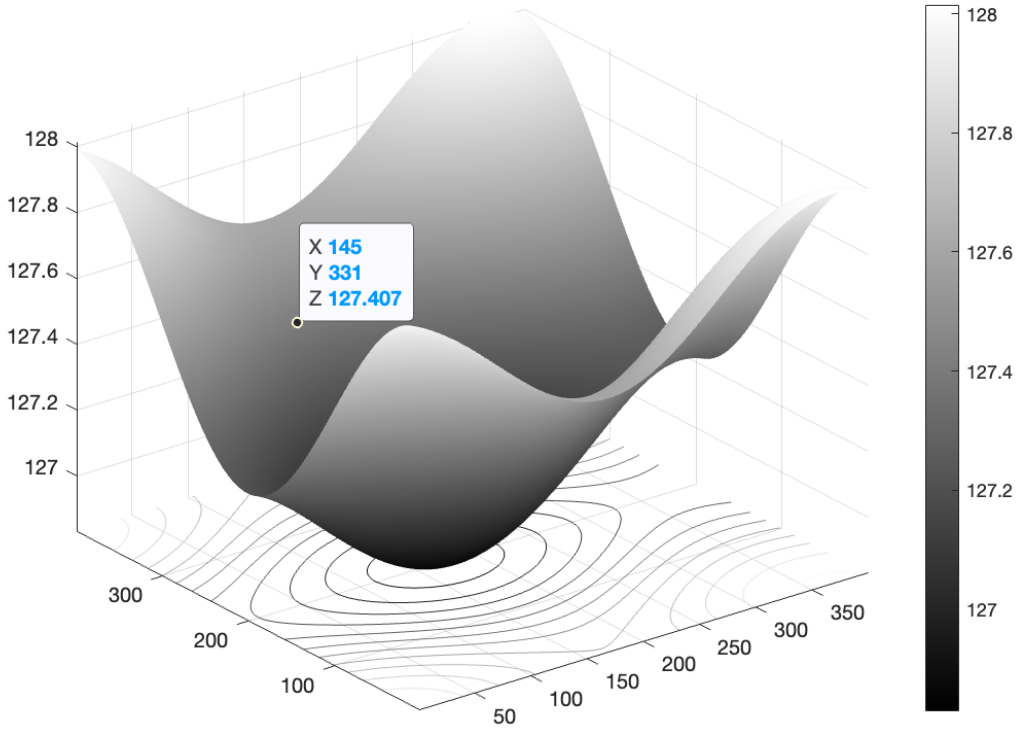

Changed radius to 0.3
50
100
150
200
250
300
350
-50
0
50
100
150
200
250
300
350
400
450
250
200
150
100
50
0


Changed radius to 0.2
50
100
150
200
250
300
350
-50
0
50
100
150
200
250
300
350
400
450
250
200
150
100
50
0

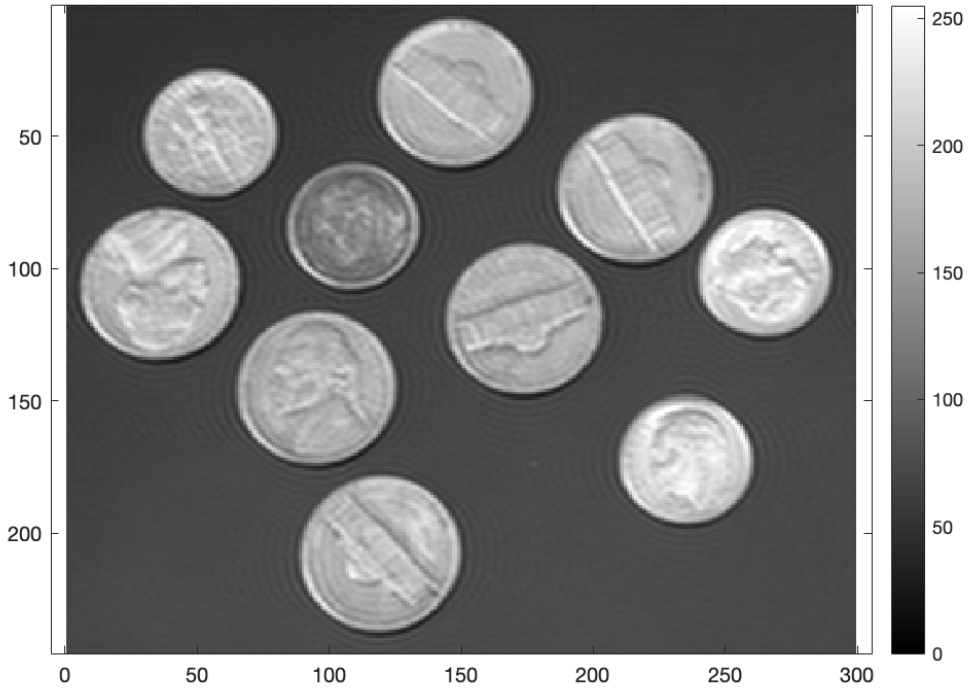

Changed radius to 0.1
50
100
150
200
250
300
350
-50
0
50
100
150
200
250
300
350
400
450
250
200
150
100
50
0

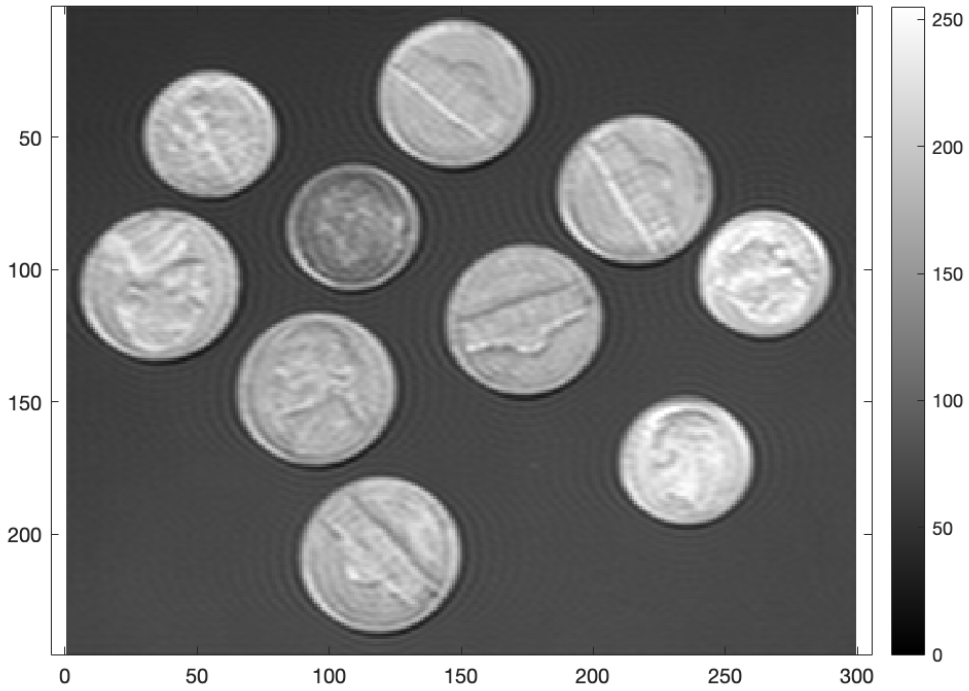
10.3 Exercise 5

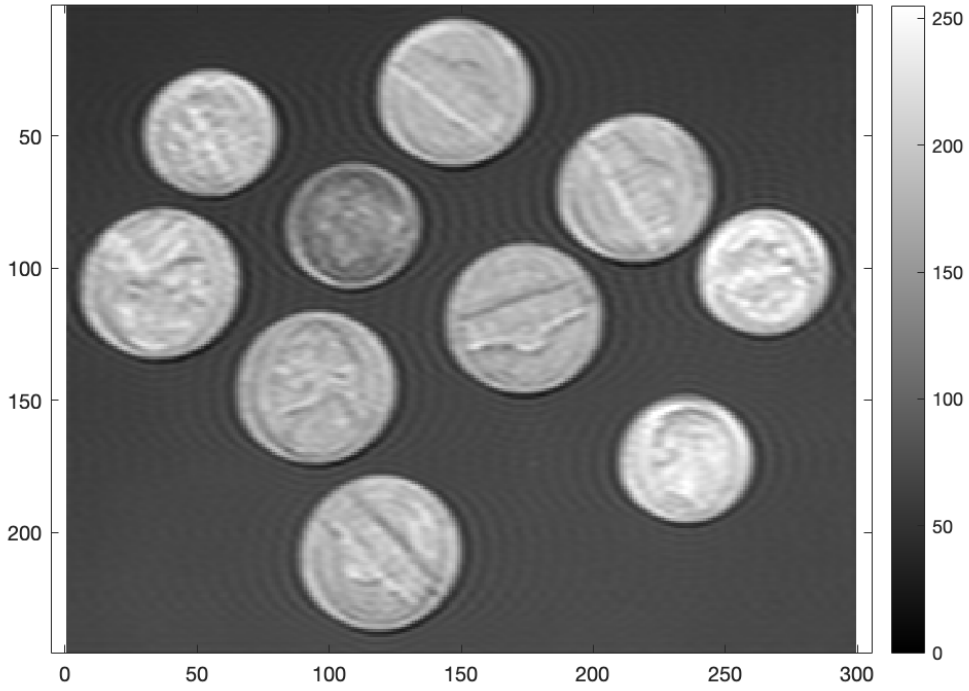


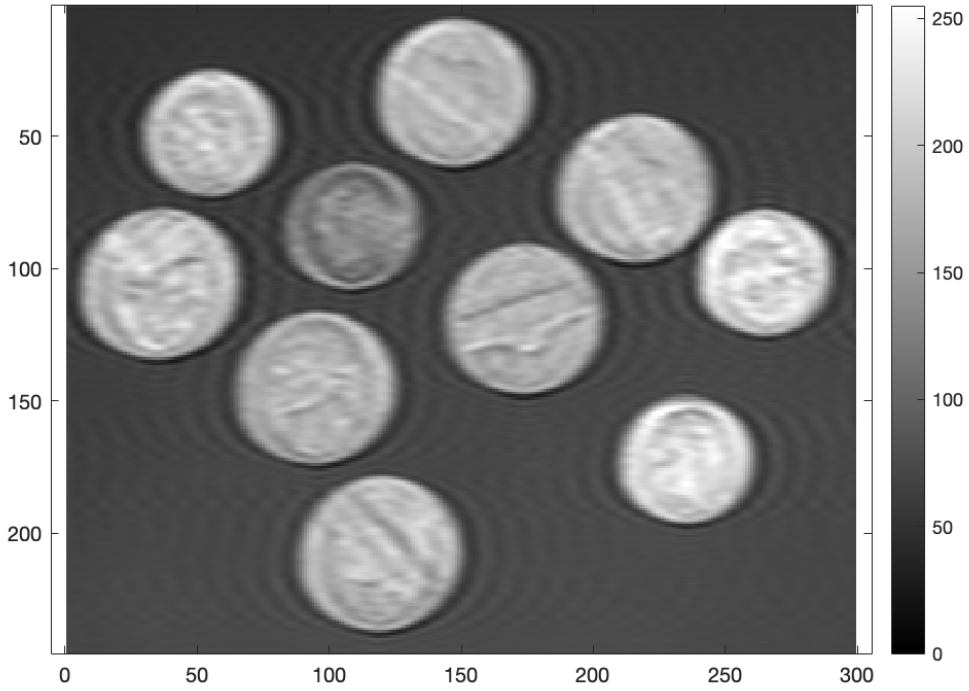


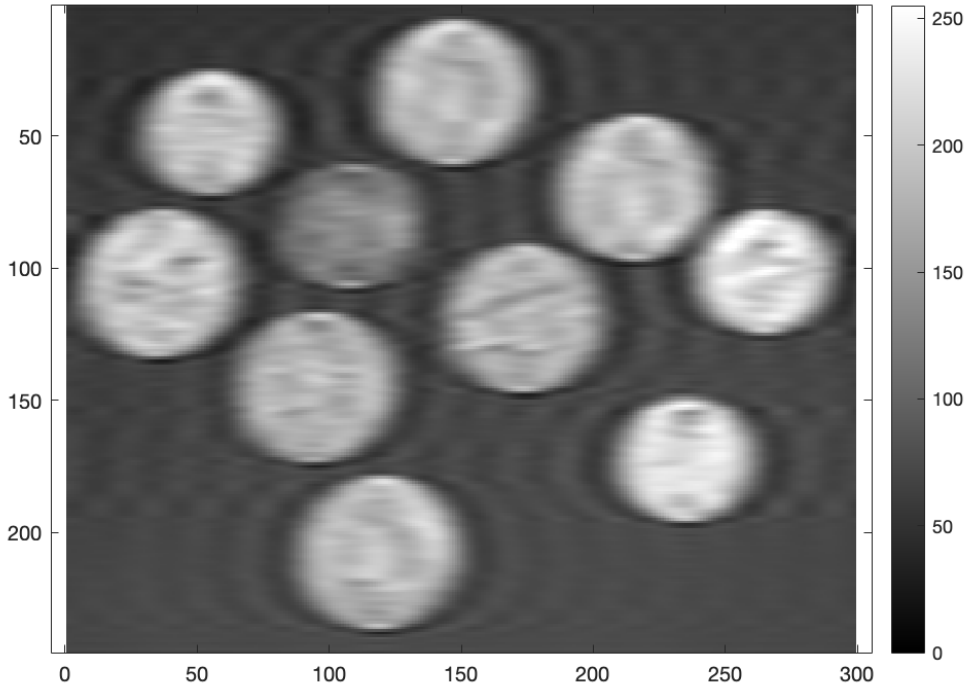
10.4 Exercise 6

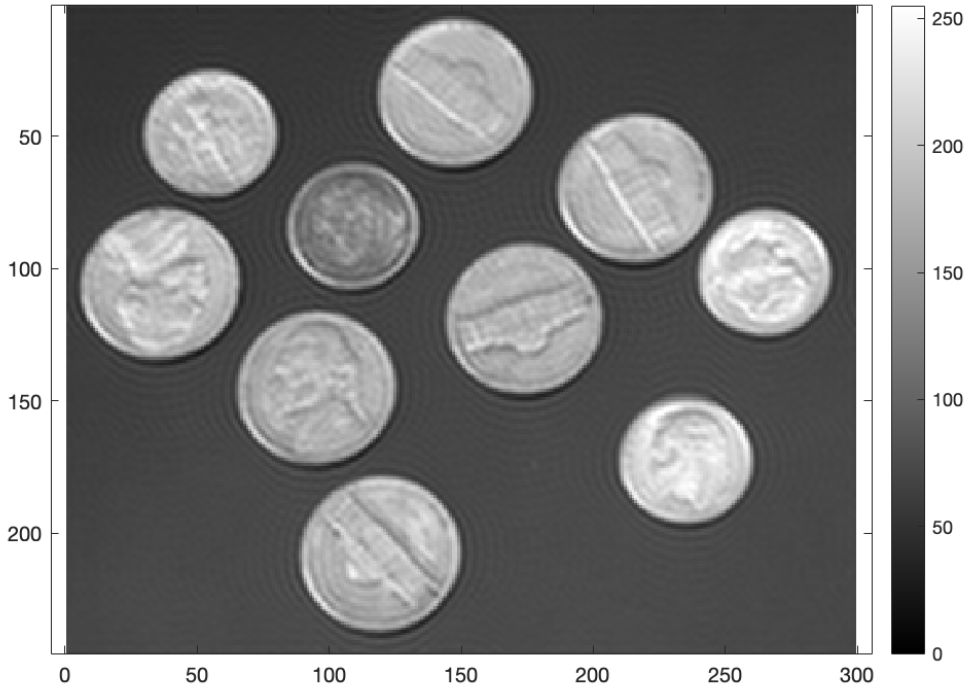


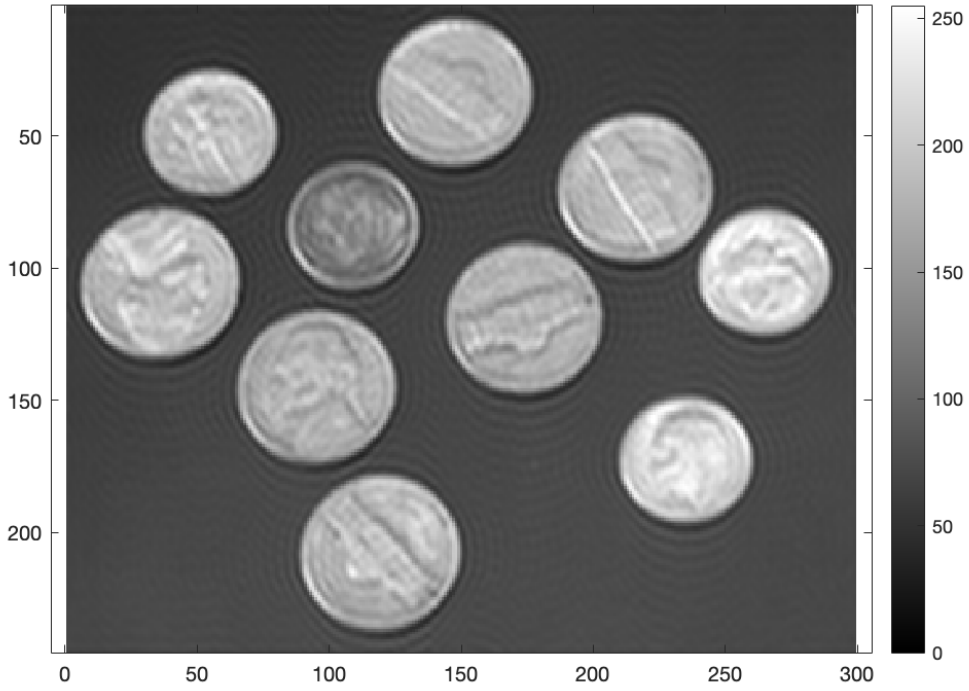


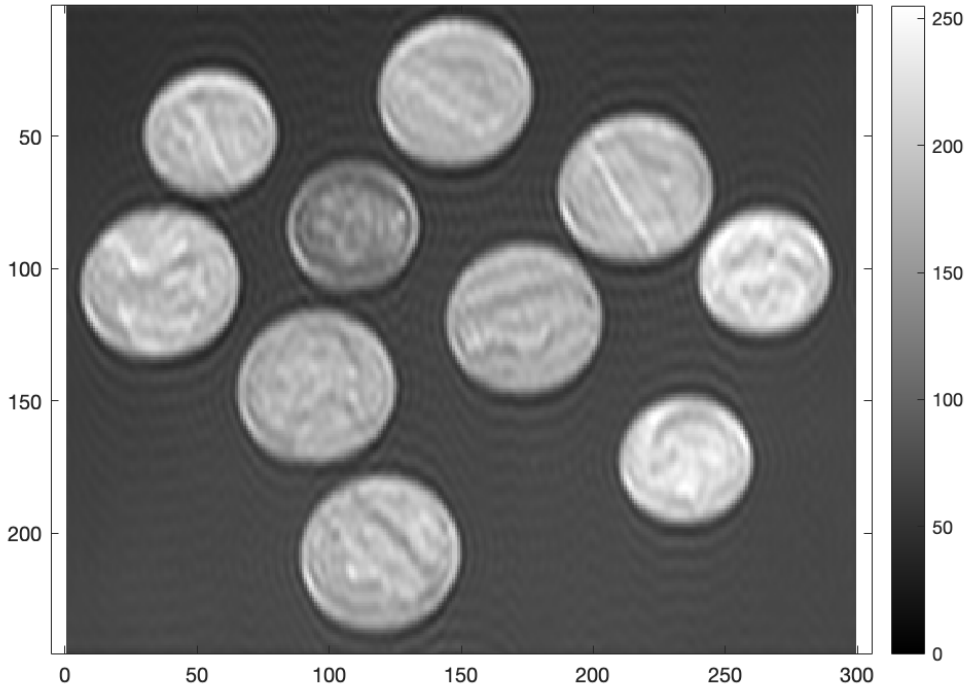

50
100
150
200
0
50
100
150
200
250
300
250
200
150
100
50
0

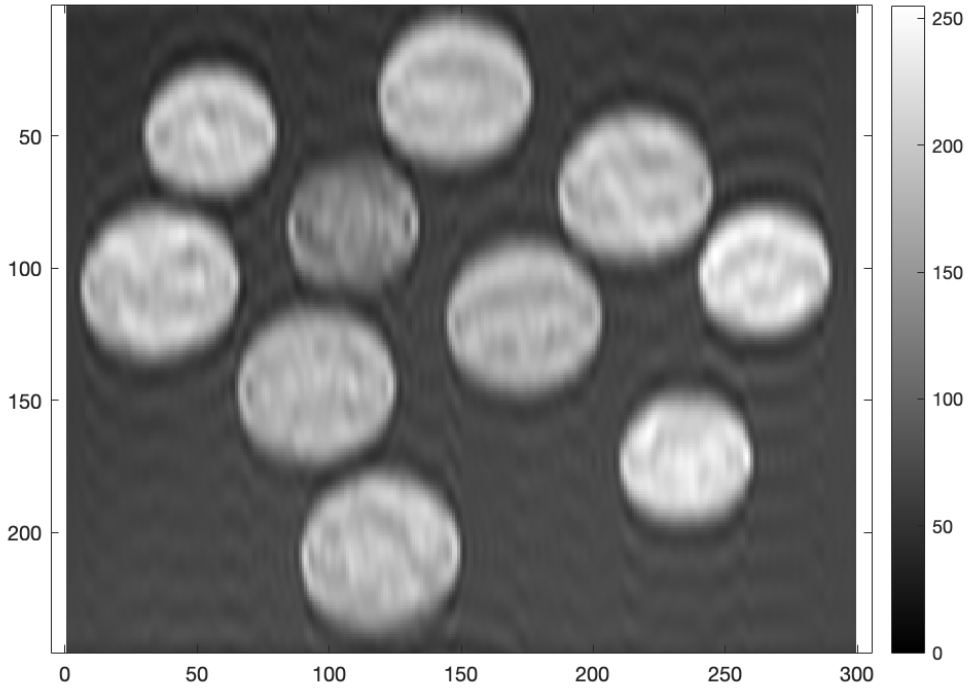

50
100
150
200
0
50
100
150
200
250
300
250
200
150
100
50
0


50
100
150
200
0
50
100
150
200
250
300
250
200
150
100
50
0

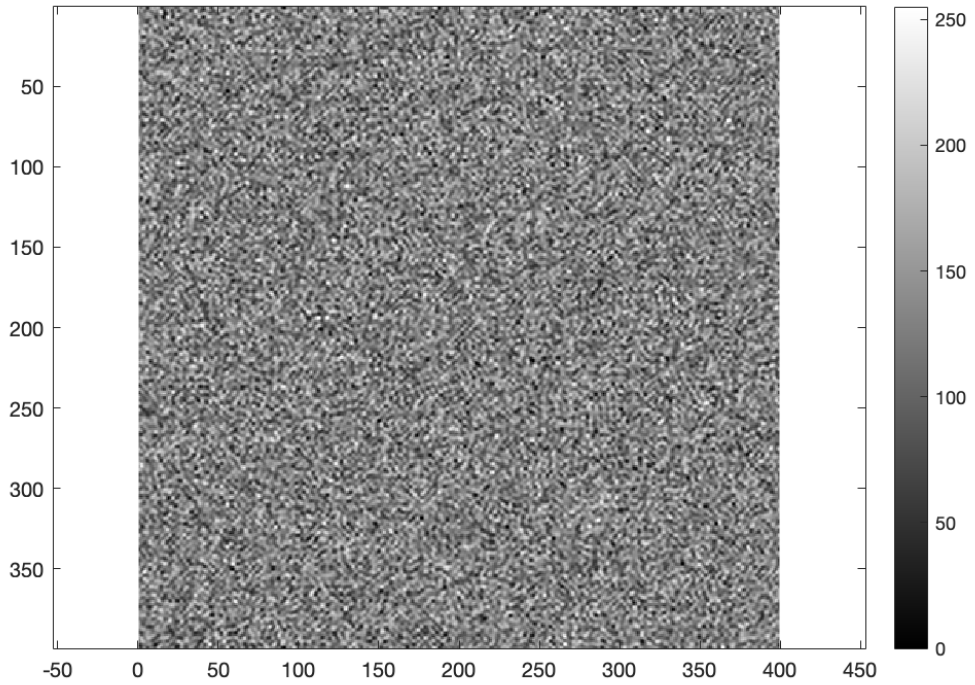

50
100
150
200
0
50
100
150
200
250
300
250
200
150
100
50
0

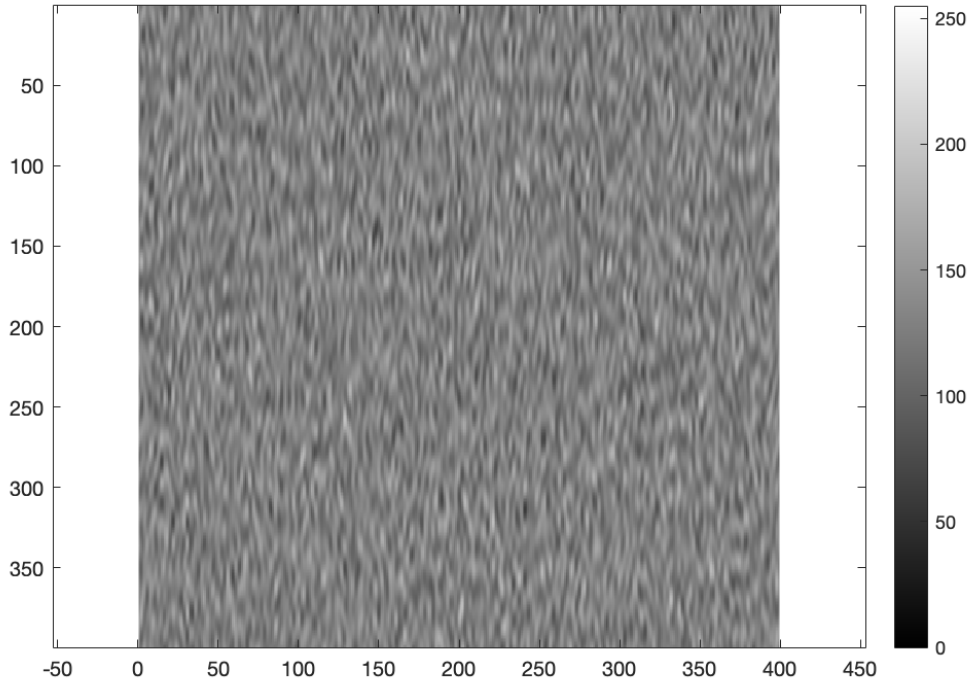

50
100
150
200
0
50
100
150
200
250
300
250
200
150
100
50
0

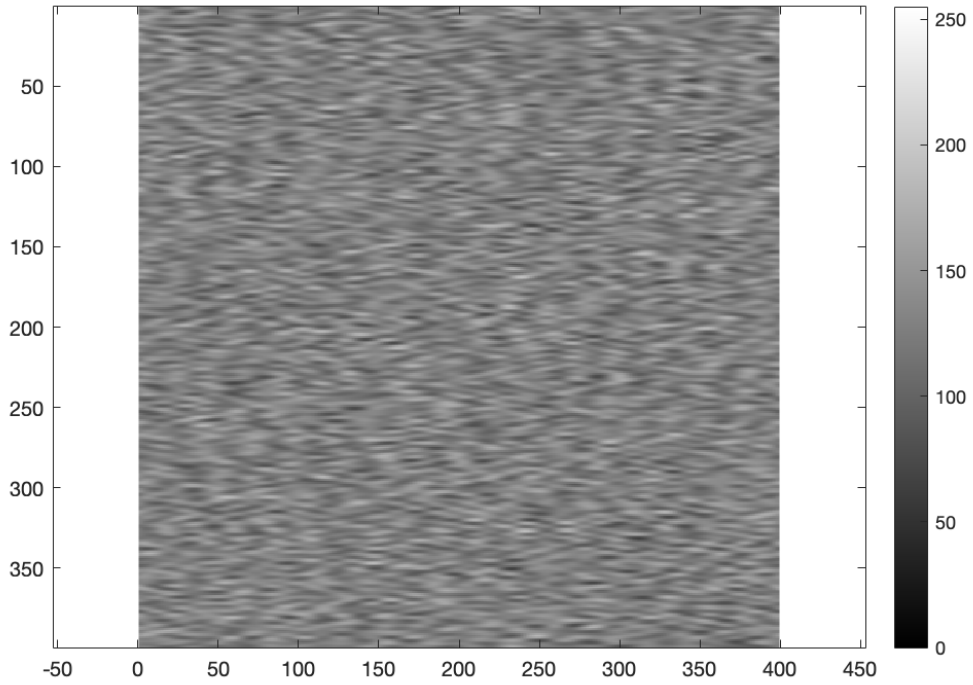

50
100
150
200
0
50
100
150
200
250
300
250
200
150
100
50
0

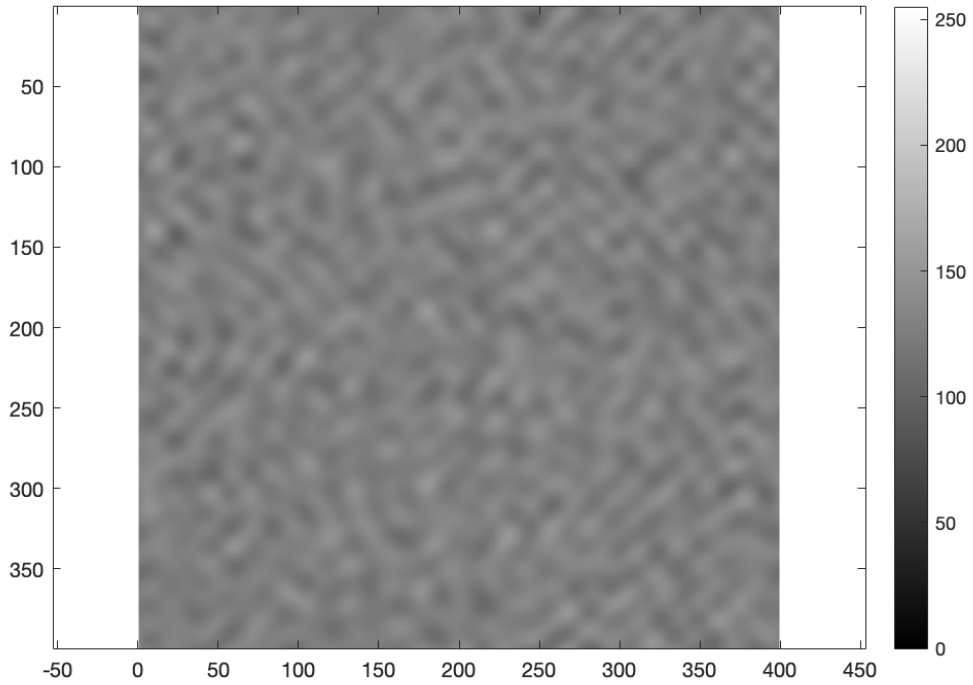

50
100
150
200
0
50
100
150
200
250
300
250
200
150
100
50
0

10.5 Exercise 7

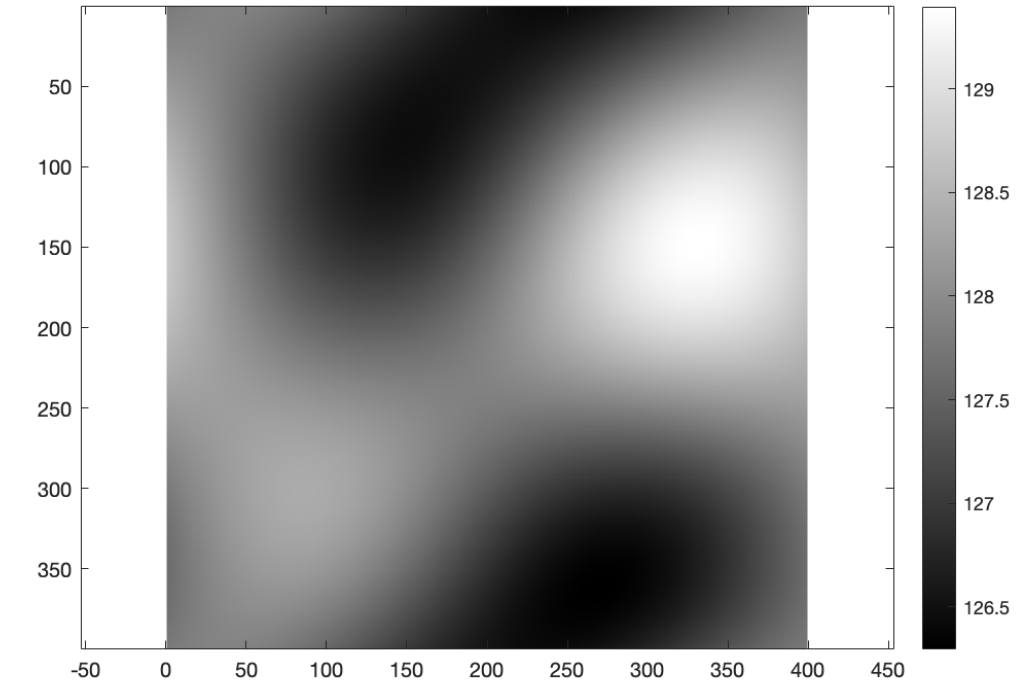


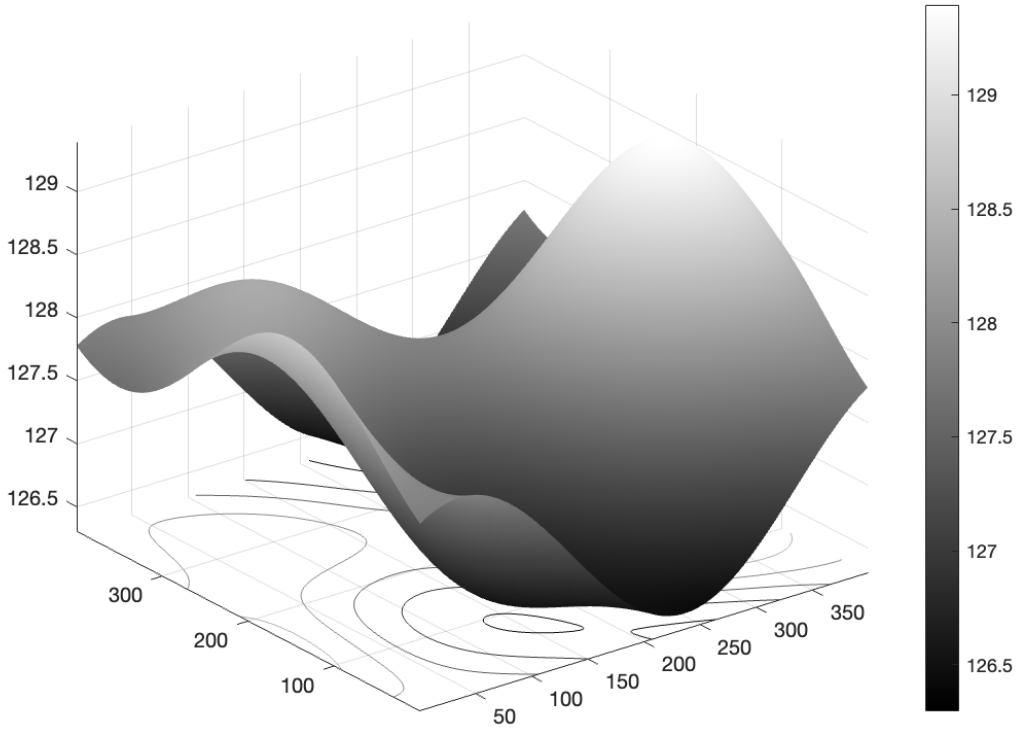



50
100
150
200
250
300
350
-50
0
50
100
150
200
250
300
350
400
450
250
200
150
100
50
0

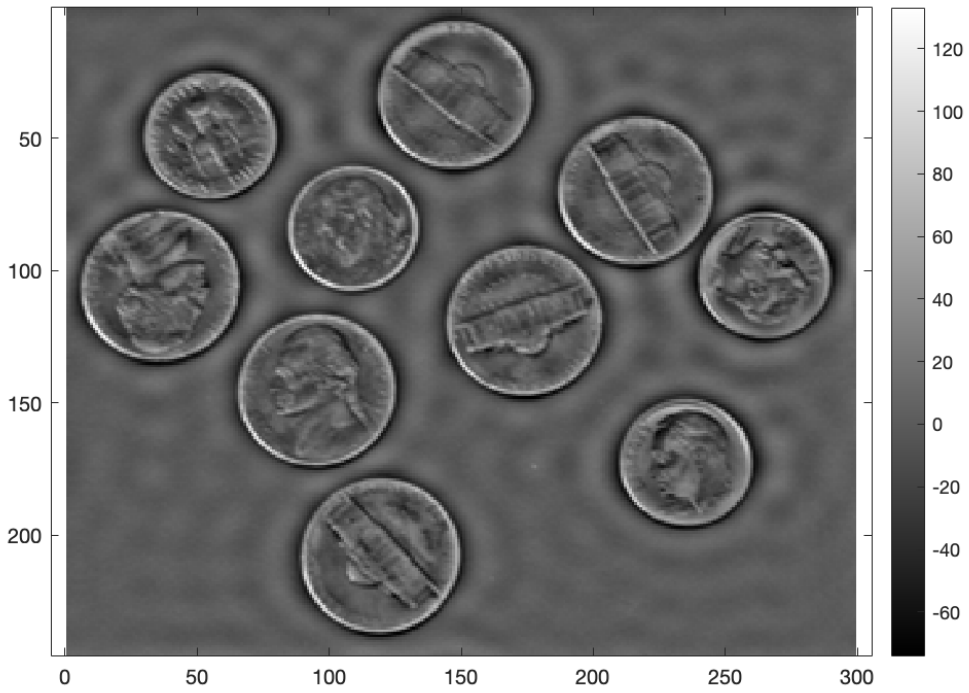

50
100
150
200
250
300
350
-50
0
50
100
150
200
250
300
350
400
450
250
200
150
100
50
0

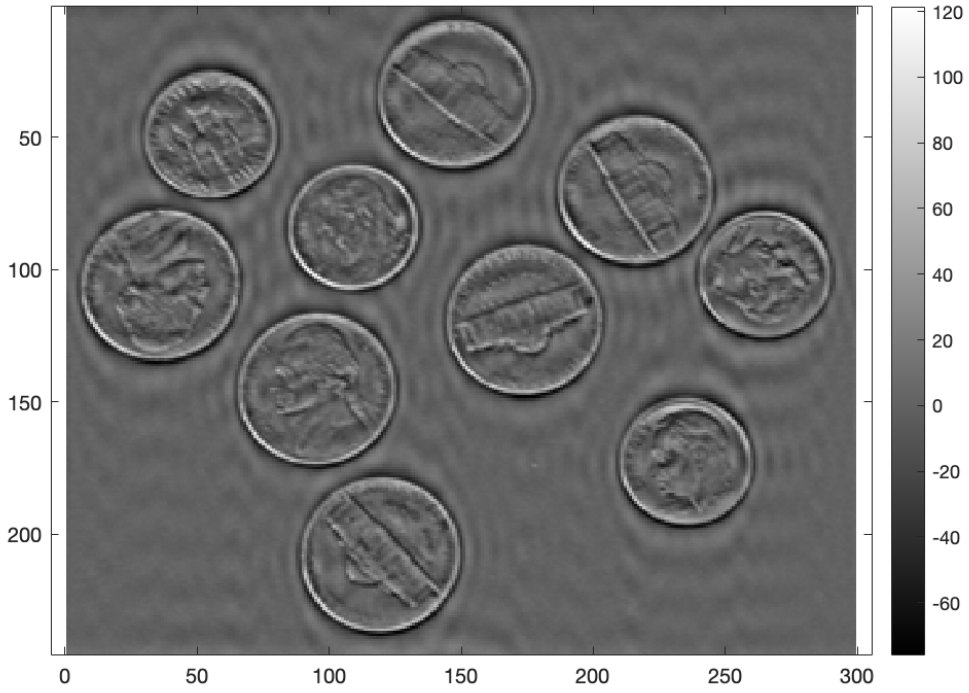
10.6 Exercise 8

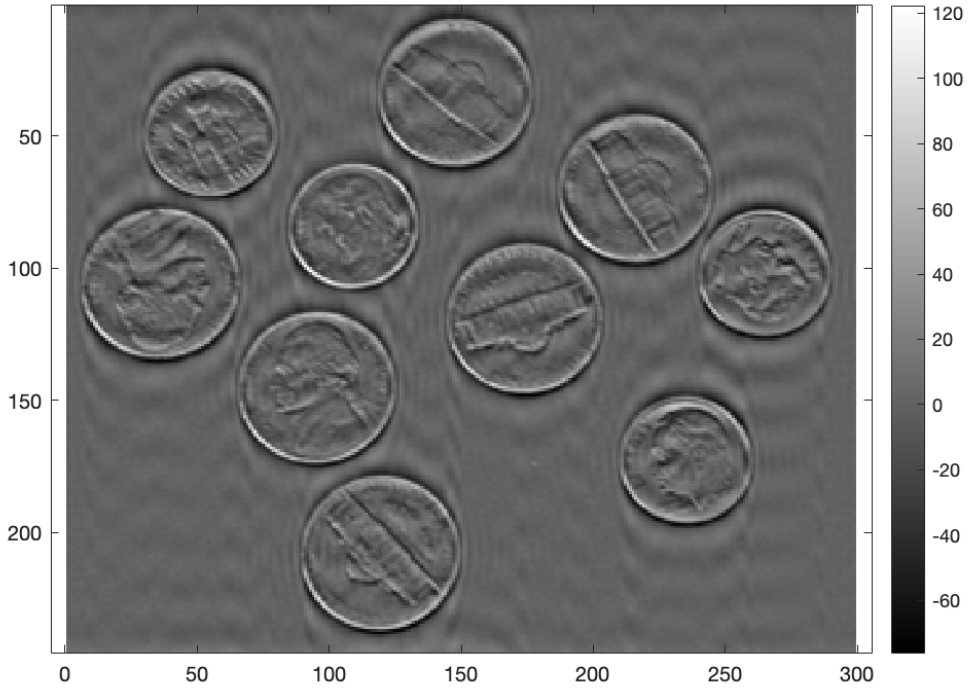


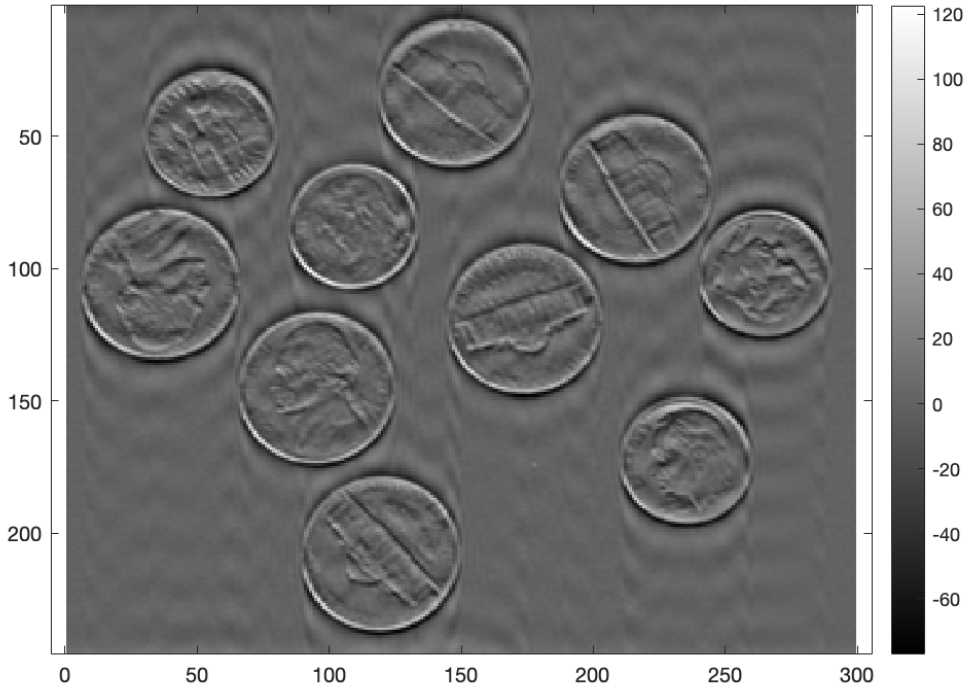


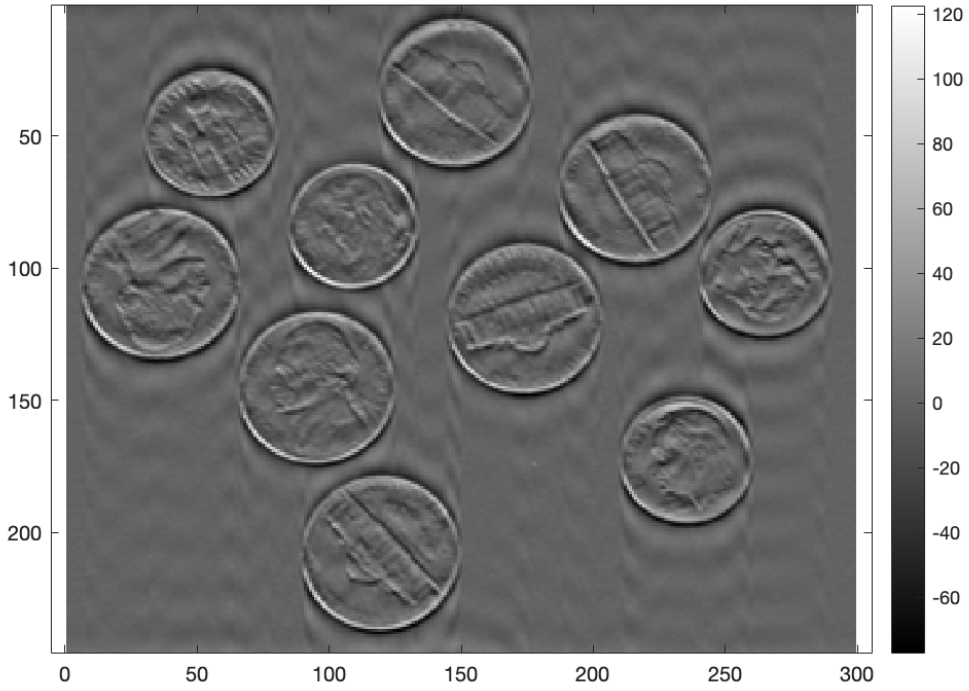
10.7 Exercise 9

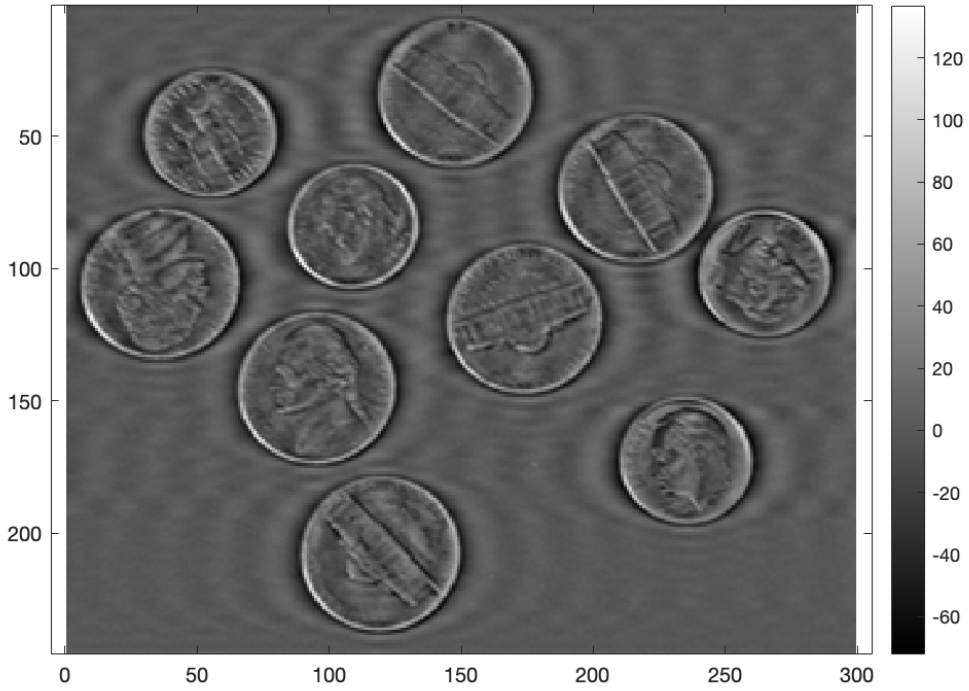


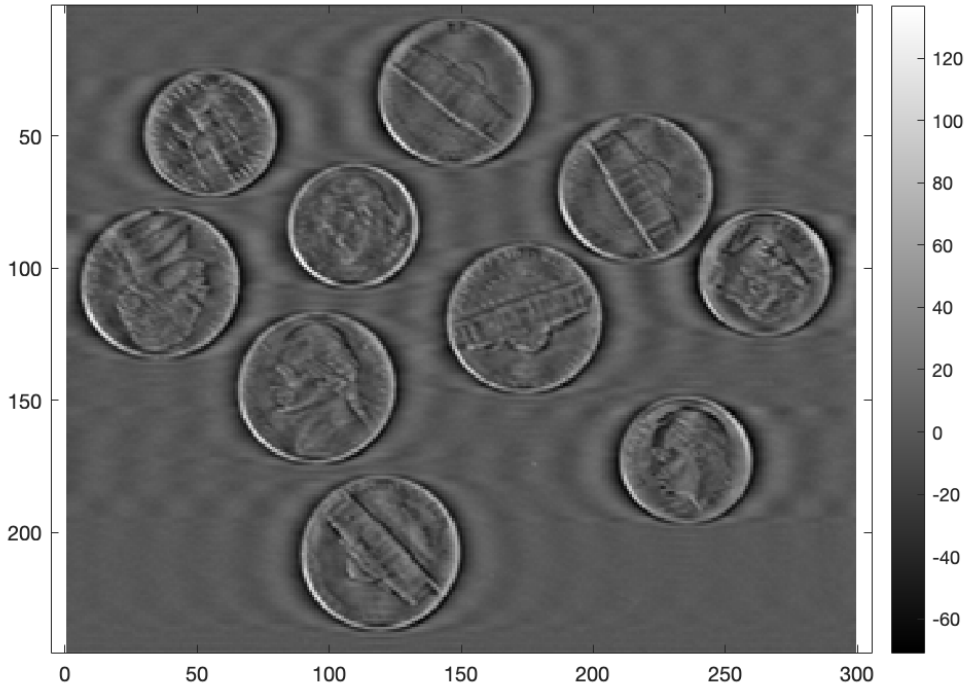


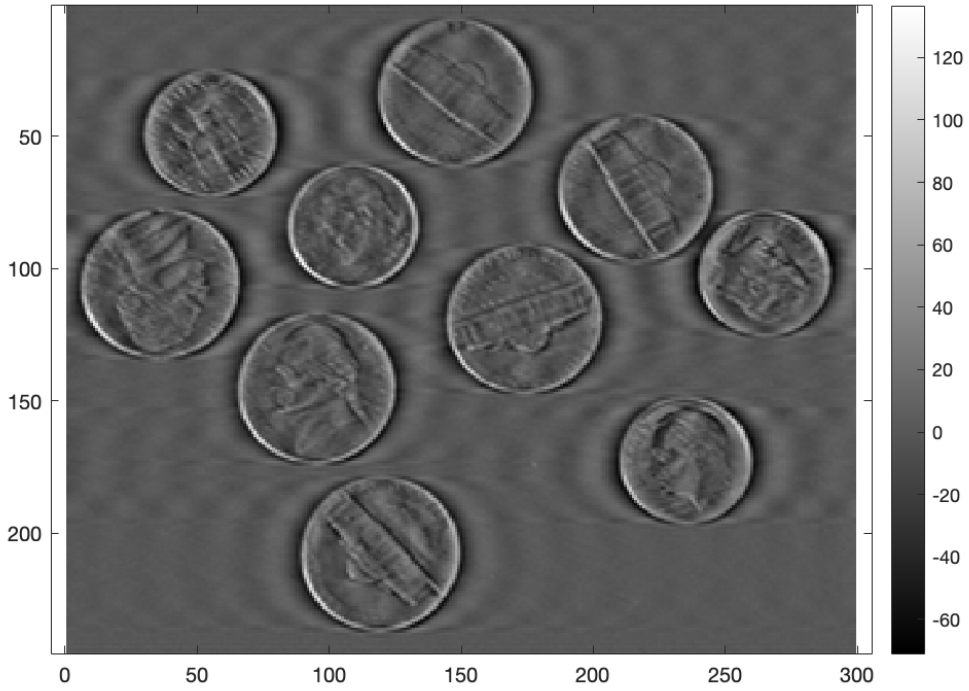

50
100
150
200
0
50
100
150
200
250
300
120
100
80
60
40
20
0
-20
-40
-60

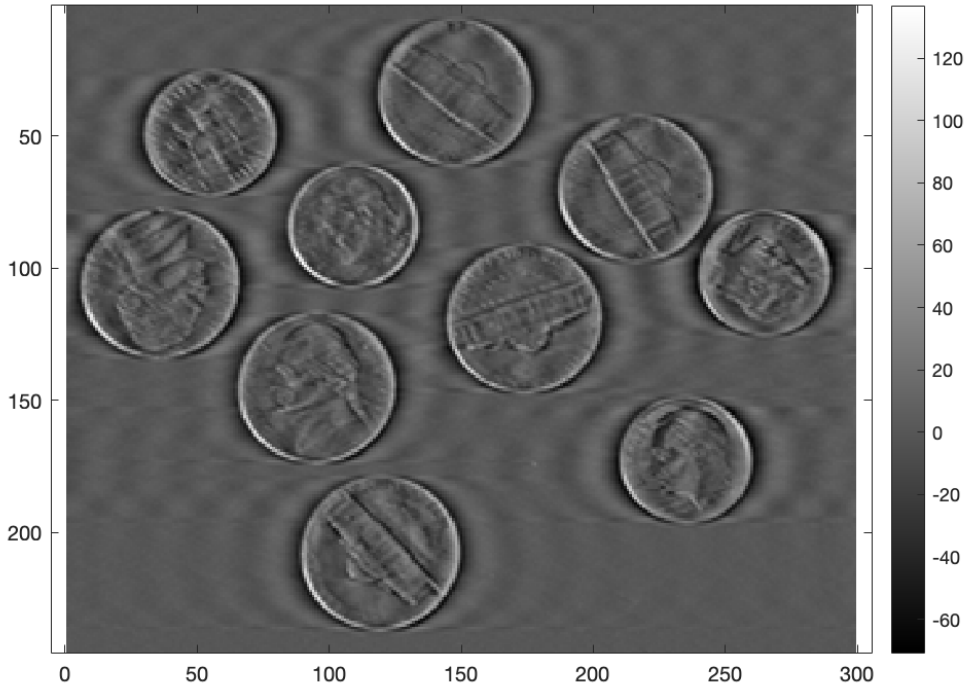

50
100
150
200
0
50
100
150
200
250
300
120
100
80
60
40
20
0
-20
-40
-60


50
100
150
200
0
50
100
150
200
250
300
120
100
80
60
40
20
0
-20
-40
-60

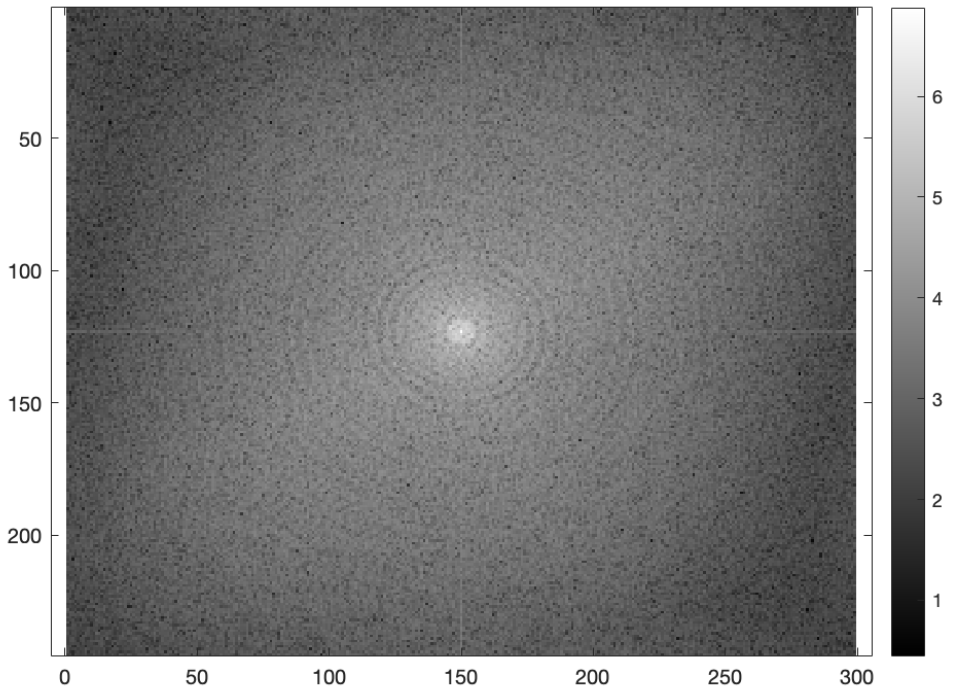

50
100
150
200
0
50
100
150
200
250
300
120
100
80
60
40
20
0
-20
-40
-60

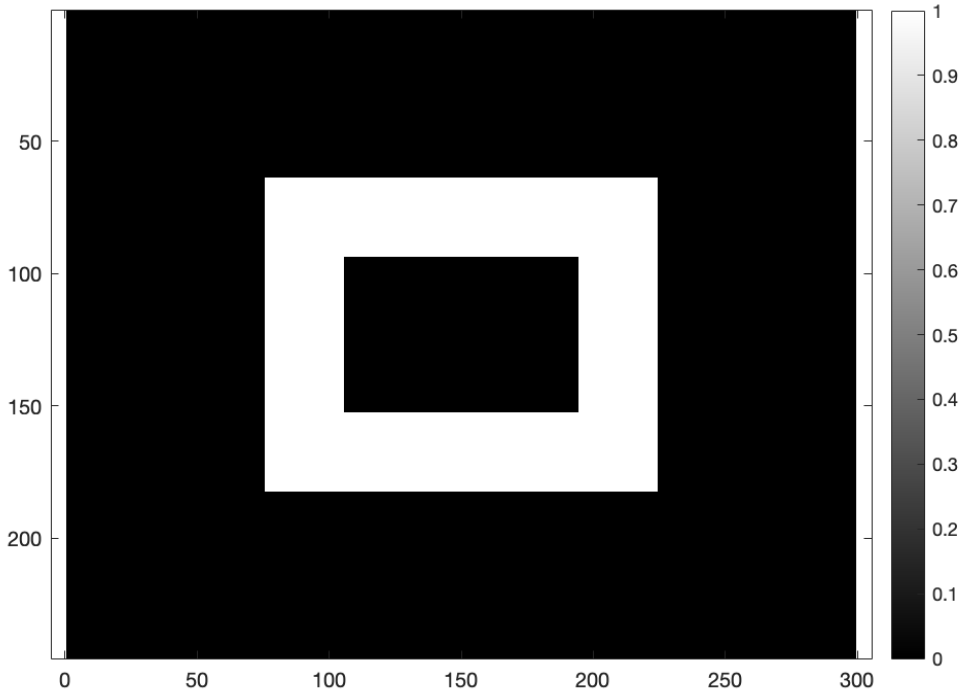

50
100
150
200
0
50
100
150
200
250
300
120
100
80
60
40
20
0
-20
-40
-60

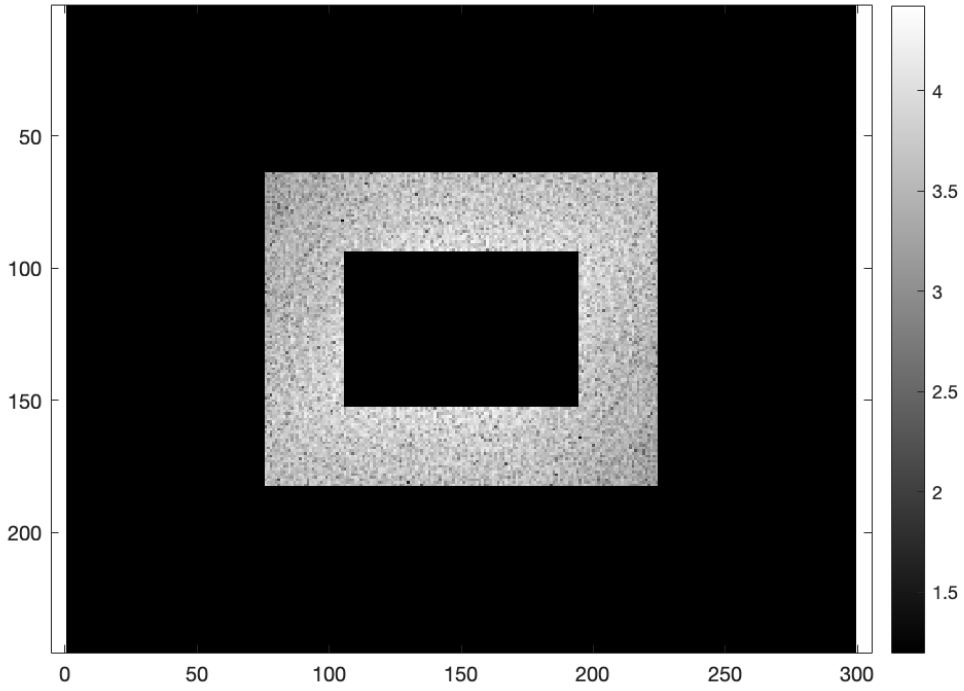

50
100
150
200
0
50
100
150
200
250
300
120
100
80
60
40
20
0
-20
-40
-60

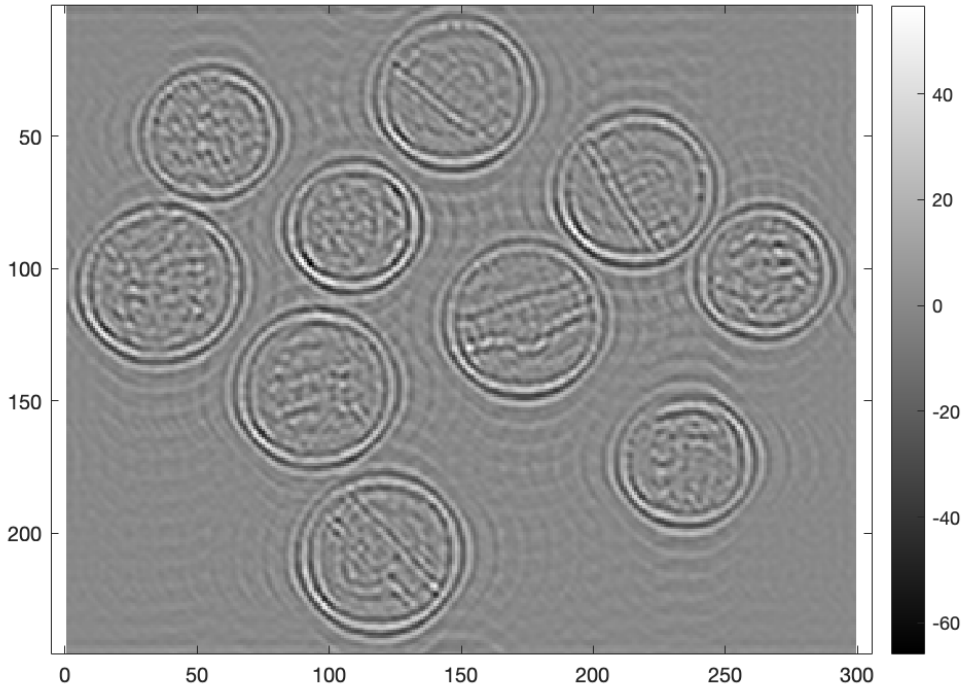

50
100
150
200
0
50
100
150
200
250
300
120
100
80
60
40
20
0
-20
-40
-60

10.8 Exercise 10






50
100
150
200
0
50
100
150
200
250
300
4
3.5
3
2.5
2
1.5


50
100
150
200
0
50
100
150
200
250
300
40
20
0
-20
-40
-60

11 Codes

11.1 Exercises 1 and 2

```
function[A, B, X] = analysis(x);
M = length(x);
A = zeros (1, M);
B = zeros (1, M);
X = zeros (1, M);

%calculating A[k], B[k], and X[k]
for k = 0:M-1
        for m = 0:M-1
            A(k+1) = A(k+1) + x(m+1) * cos(2 * pi * k * m / 5);
            B(k+1) = B(k+1) + x(m+1) * sin(2 * pi * k * m / 5);
            X(k+1) =  X(k+1) + x(m+1) * exp(-1j * 2 * pi * k * m / 5);
        end
    end

end

x_test= [1, 5, 2, 3, 1];
[test1, test2, test3] = analysis(x_test)




function[x]= trig_synth(A, B);
M = length(A);
x = [];

for k = 1:M
    x_sum=0;
        for m = 1:M
            x_sum = x_sum + ((A(m) * cos(2 * pi * (k-1) * (m-1) / M)) +(B(m) * sin(2 * pi
        end
        x = [x (1/M)*x_sum];
end
end

%second_test = trig_synth(test1, test2)

function[x] = exp_synth(X);
M = length(X);
x = [];

for k = 1:M
    x_sum=0;
        for m = 1:M
            x_sum = x_sum + X(m)*exp((1i*2*pi*(k-1)*(m-1)/M));
        end
        x = [x (1/M)*x_sum];
end
end

x_back= exp_synth(test3)
```

11.2 Exercise 3

```
x = imread('coins.png');
x = x(2:end, 2:end);
figure(1); clf
image(x); axis equal; colormap gray; colorbar

X = fft2(x);
Xs = fftshift(X);
figure(2); clf
%imagesc(log10(abs(Xs))); axis equal; colormap gray; colorbar

[rows, cols] = size(x);
max_size = max(rows, cols);
rnorm = rows/max_size; cnorm = cols/max_size;
[v, u] = meshgrid(linspace(-cnorm, cnorm, cols),...
    linspace(-rnorm, rnorm, rows)) ;
filter = sqrt(u.^2+v.^2)<0.1;
figure(3); clf
%imagesc(filter); axis equal; colormap gray; colorbar

Xsfiltered = Xs.*filter;
figure(4); clf
%imagesc(log10(abs(Xsfiltered))); axis equal; colormap gray; colorbar

Xfiltered = ifftshift(Xsfiltered);
xfiltered = ifft2(Xfiltered);
figure(5); clf
imagesc(xfiltered, [0, 255]); axis equal; colormap gray; colorbar
```

11.3 Exercise 4

```
x = randi([0, 255], 399, 399);
figure(1); clf
image(x); axis equal; colormap gray; colorbar

X = fft2(x);
Xs = fftshift(X);
figure(2); clf
%imagesc(log10(abs(Xs))); axis equal; colormap gray; colorbar

[rows, cols] = size(x);
max_size = max(rows, cols);
rnorm = rows/max_size; cnorm = cols/max_size;
[v, u] = meshgrid(linspace(-cnorm, cnorm, cols),...
    linspace(-rnorm, rnorm, rows)) ;
filter = sqrt(u.^2+v.^2)<0.1;
figure(3); clf
%imagesc(filter); axis equal; colormap gray; colorbar

Xsfiltered = Xs.*filter;
figure(4); clf
%imagesc(log10(abs(Xsfiltered))); axis equal; colormap gray; colorbar

Xfiltered = ifftshift(Xsfiltered);
xfiltered = ifft2(Xfiltered);
figure(5); clf
imagesc(xfiltered, [0, 255]); axis equal; colormap gray; colorbar
```

11.4 Exercise 5

```
x = randi([0, 255], 399, 399);
figure(1); clf
image(x); axis equal; colormap gray; colorbar

X = fft2(x);
Xs = fftshift(X);
figure(2); clf
imagesc(log10(abs(Xs))); axis equal; colormap gray; colorbar

[rows, cols] = size(x);
max_size = max(rows, cols);
rnorm = rows/max_size; cnorm = cols/max_size;
[v, u] = meshgrid(linspace(-cnorm, cnorm, cols),...
    linspace(-rnorm, rnorm, rows)) ;
filter = sqrt(u.^2+v.^2)<0.006;
figure(3); clf
imagesc(filter); axis equal; colormap gray; colorbar

Xsfiltered = Xs.*filter;
figure(4); clf
imagesc(log10(abs(Xsfiltered))); axis equal; colormap gray; colorbar

Xfiltered = ifftshift(Xsfiltered);
xfiltered = ifft2(Xfiltered);
figure(5); clf
imagesc(xfiltered, [0, 255]); axis equal; colormap gray; colorbar

figure(6); clf
imagesc(xfiltered); axis equal; colormap gray; colorbar

figure(7); clf
surfc(xfiltered); colormap gray; shading interp; colorbar
```

11.5 Exercise 6

```
x = imread('coins.png');
x = x(2:end, 2:end);
figure(1); clf
%image(x); axis equal; colormap gray; colorbar

X = fft2(x);
Xs = fftshift(X);
figure(2); clf
%imagesc(log10(abs(Xs))); axis equal; colormap gray; colorbar

[rows, cols] = size(x);
max_size = max(rows, cols);
rnorm = rows/max_size; cnorm = cols/max_size;
[v, u] = meshgrid(linspace(-cnorm, cnorm, cols),...
    linspace(-rnorm, rnorm, rows)) ;
filter = (abs(u)<0.1) & (abs(v)<0.5);
figure(3); clf
%imagesc(filter); axis equal; colormap gray; colorbar

Xsfiltered = Xs.*filter;
figure(4); clf
%imagesc(log10(abs(Xsfiltered))); axis equal; colormap gray; colorbar

Xfiltered = ifftshift(Xsfiltered);
xfiltered = ifft2(Xfiltered);
figure(5); clf
imagesc(xfiltered, [0, 255]); axis equal; colormap gray; colorbar
```

11.6 Exercise 7

```
x = randi([0, 255], 399, 399);
figure(1); clf
%image(x); axis equal; colormap gray; colorbar

X = fft2(x);
Xs = fftshift(X);
figure(2); clf
%imagesc(log10(abs(Xs))); axis equal; colormap gray; colorbar

[rows, cols] = size(x);
max_size = max(rows, cols);
rnorm = rows/max_size; cnorm = cols/max_size;
[v, u] = meshgrid(linspace(-cnorm, cnorm, cols),...
    linspace(-rnorm, rnorm, rows)) ;
filter = (abs(u)<0.1) & (abs(v)<0.1);
figure(3); clf
%imagesc(filter); axis equal; colormap gray; colorbar

Xsfiltered = Xs.*filter;
figure(4); clf
%imagesc(log10(abs(Xsfiltered))); axis equal; colormap gray; colorbar

Xfiltered = ifftshift(Xsfiltered);
xfiltered = ifft2(Xfiltered);
figure(5); clf
imagesc(xfiltered, [0, 255]); axis equal; colormap gray; colorbar
```

11.7 Exercise 8

```
x = randi([0, 255], 399, 399);
figure(1); clf
image(x); axis equal; colormap gray; colorbar

X = fft2(x);
Xs = fftshift(X);
figure(2); clf
imagesc(log10(abs(Xs))); axis equal; colormap gray; colorbar

[rows, cols] = size(x);
max_size = max(rows, cols);
rnorm = rows/max_size; cnorm = cols/max_size;
[v, u] = meshgrid(linspace(-cnorm, cnorm, cols),...
    linspace(-rnorm, rnorm, rows)) ;
filter = (abs(u)<0.006) & (abs(v)<0.006);
figure(3); clf
imagesc(filter); axis equal; colormap gray; colorbar

Xsfiltered = Xs.*filter;
figure(4); clf
imagesc(log10(abs(Xsfiltered))); axis equal; colormap gray; colorbar

Xfiltered = ifftshift(Xsfiltered);
xfiltered = ifft2(Xfiltered);
figure(5); clf
imagesc(xfiltered, [0, 255]); axis equal; colormap gray; colorbar

figure(6); clf
imagesc(xfiltered); axis equal; colormap gray; colorbar

figure(7); clf
surfc(xfiltered); colormap gray; shading interp; colorbar
```

11.8 Exercise 9

```
x = imread('coins.png');
x = x(2:end, 2:end);
figure(1); clf
%image(x); axis equal; colormap gray; colorbar

X = fft2(x);
Xs = fftshift(X);
figure(2); clf
%imagesc(log10(abs(Xs))); axis equal; colormap gray; colorbar

[rows, cols] = size(x);
max_size = max(rows, cols);
rnorm = rows/max_size; cnorm = cols/max_size;
[v, u] = meshgrid(linspace(-cnorm, cnorm, cols),...
    linspace(-rnorm, rnorm, rows)) ;
filter = (abs(u)>0.9) | (abs(v)>0.1);
figure(3); clf
%imagesc(filter); axis equal; colormap gray; colorbar

Xsfiltered = Xs.*filter;
figure(4); clf
%imagesc(log10(abs(Xsfiltered))); axis equal; colormap gray; colorbar

Xfiltered = ifftshift(Xsfiltered);
xfiltered = ifft2(Xfiltered);
figure(5); clf
imagesc(xfiltered); axis equal; colormap gray; colorbar
```

11.9 Exercise 10

```
x = imread('coins.png');
x = x(2:end, 2:end);
figure(1); clf
image(x); axis equal; colormap gray; colorbar

X = fft2(x);
Xs = fftshift(X);
figure(2); clf
imagesc(log10(abs(Xs))); axis equal; colormap gray; colorbar

[rows, cols] = size(x);
max_size = max(rows, cols);
rnorm = rows/max_size; cnorm = cols/max_size;
[v, u] = meshgrid(linspace(-cnorm, cnorm, cols),...
    linspace(-rnorm, rnorm, rows)) ;

filter = (((abs(u)>0.2) | (abs(v)>0.3)) &((abs(u)<0.4) & (abs(v)< 0.5)));
figure(3); clf
imagesc(filter); axis equal; colormap gray; colorbar

Xsfiltered = Xs.*filter;
figure(4); clf
imagesc(log10(abs(Xsfiltered))); axis equal; colormap gray; colorbar

Xfiltered = ifftshift(Xsfiltered);
xfiltered = ifft2(Xfiltered);
figure(5); clf
imagesc(xfiltered); axis equal; colormap gray; colorbar
```