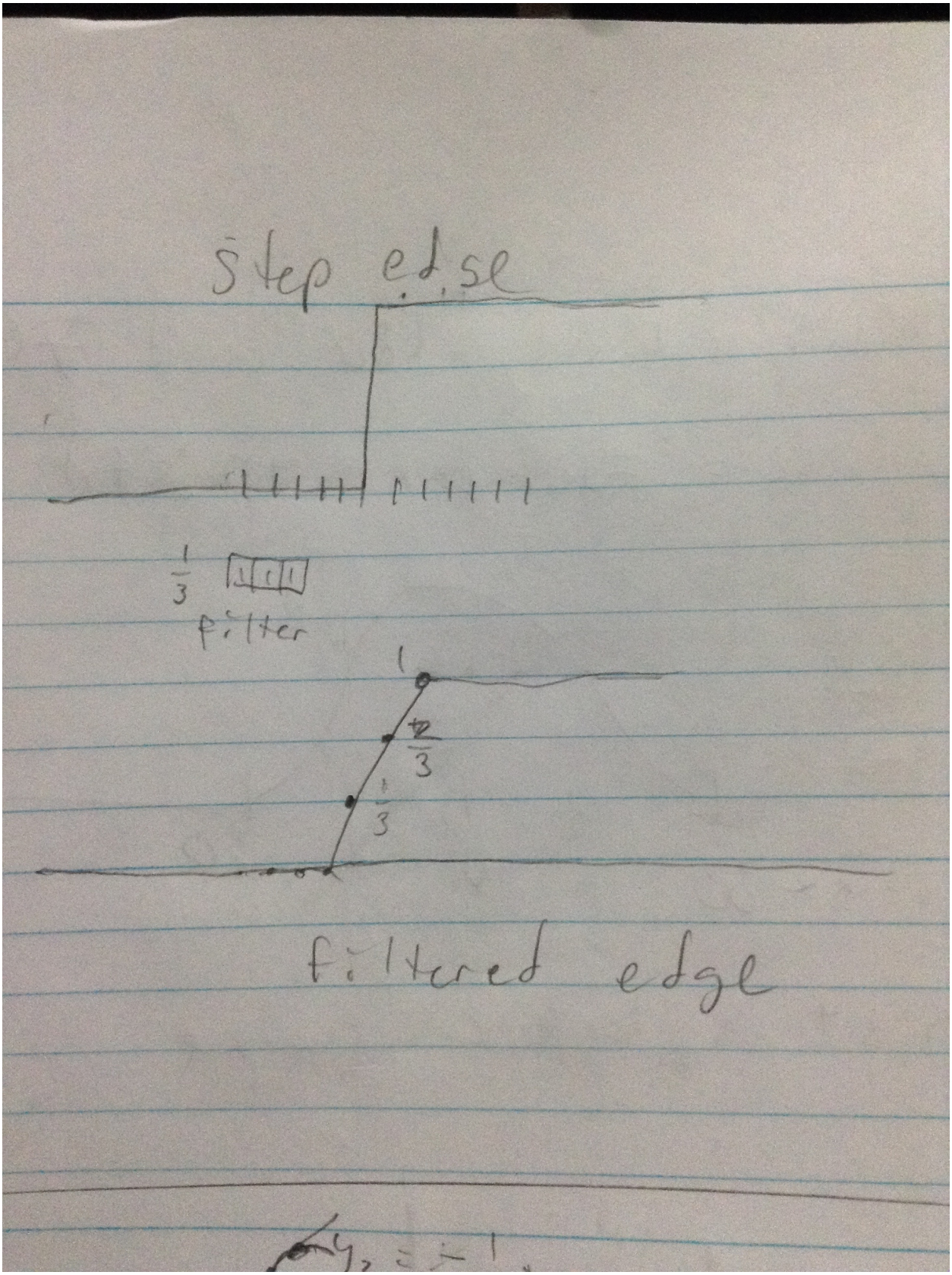
ECG 782 HW 4

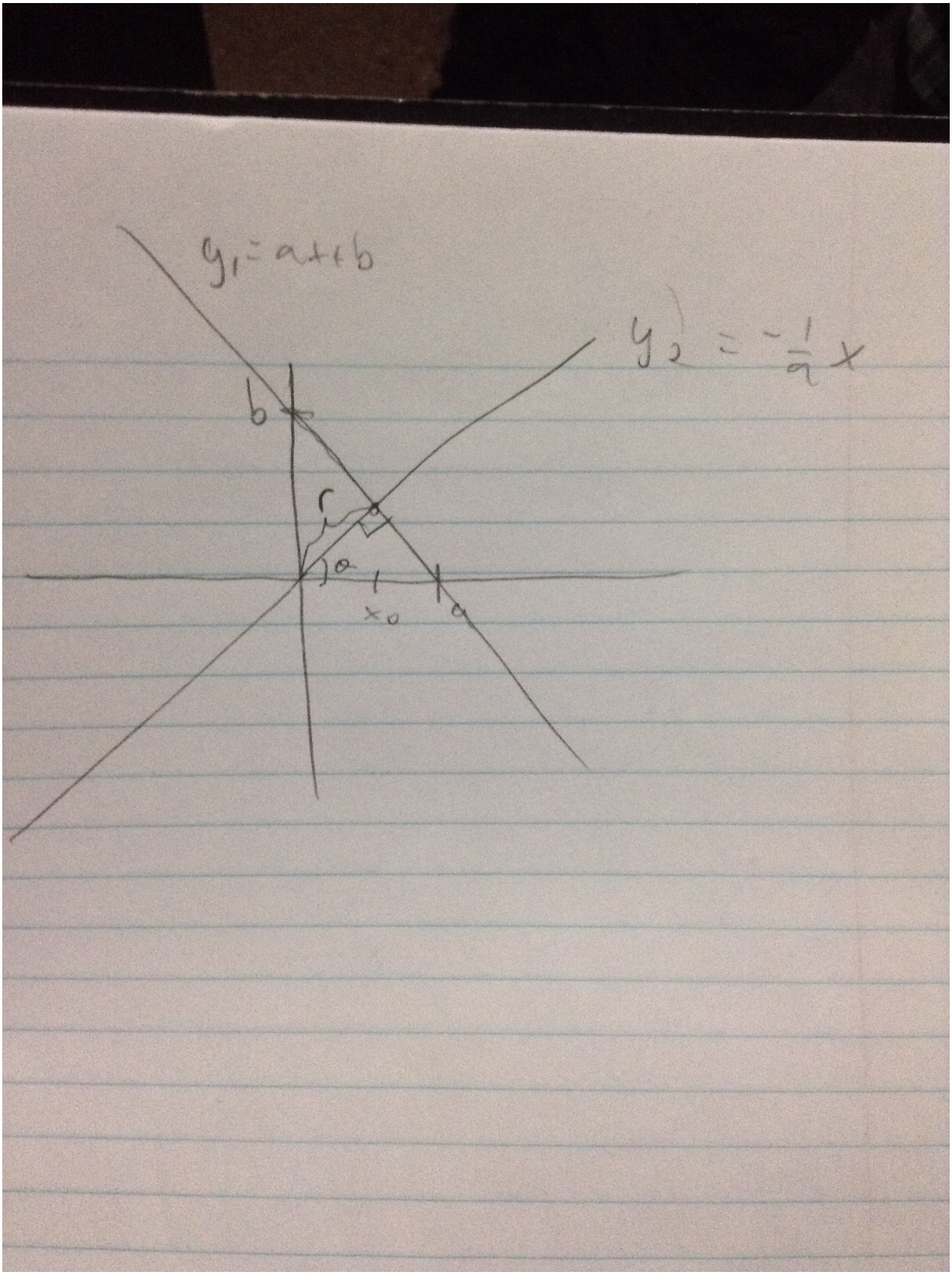
Carlo Lopez-Tello

2015-10-15

1 Problem 10.6



2 Problem 10.22



The slope of $y_2 = -\frac{1}{a}x$. This implies that $\theta = atan(-\frac{1}{a})$. We can find x_0 by solving the following equation.

$$y_1 = y_2$$

$$ax + b = -\frac{1}{a}x$$

$$b = -\frac{1}{a}x - ax$$

$$b = (\frac{1}{a} + a) - x$$

$$-b = (\frac{1}{a} + a)x$$

$$-\frac{b}{(\frac{1}{a} + a)} = x$$

$$x_0 = x$$

$$x_0 = -\frac{b}{(\frac{1}{a} + a)}$$

We can find r by solving the following equation.

$$rcos(\theta) = x_0$$

$$r = \frac{x_0}{cos(\theta)}$$

$$r = \frac{-\frac{b}{(\frac{1}{a}+a)}}{cos(atan(-\frac{1}{a}))}$$

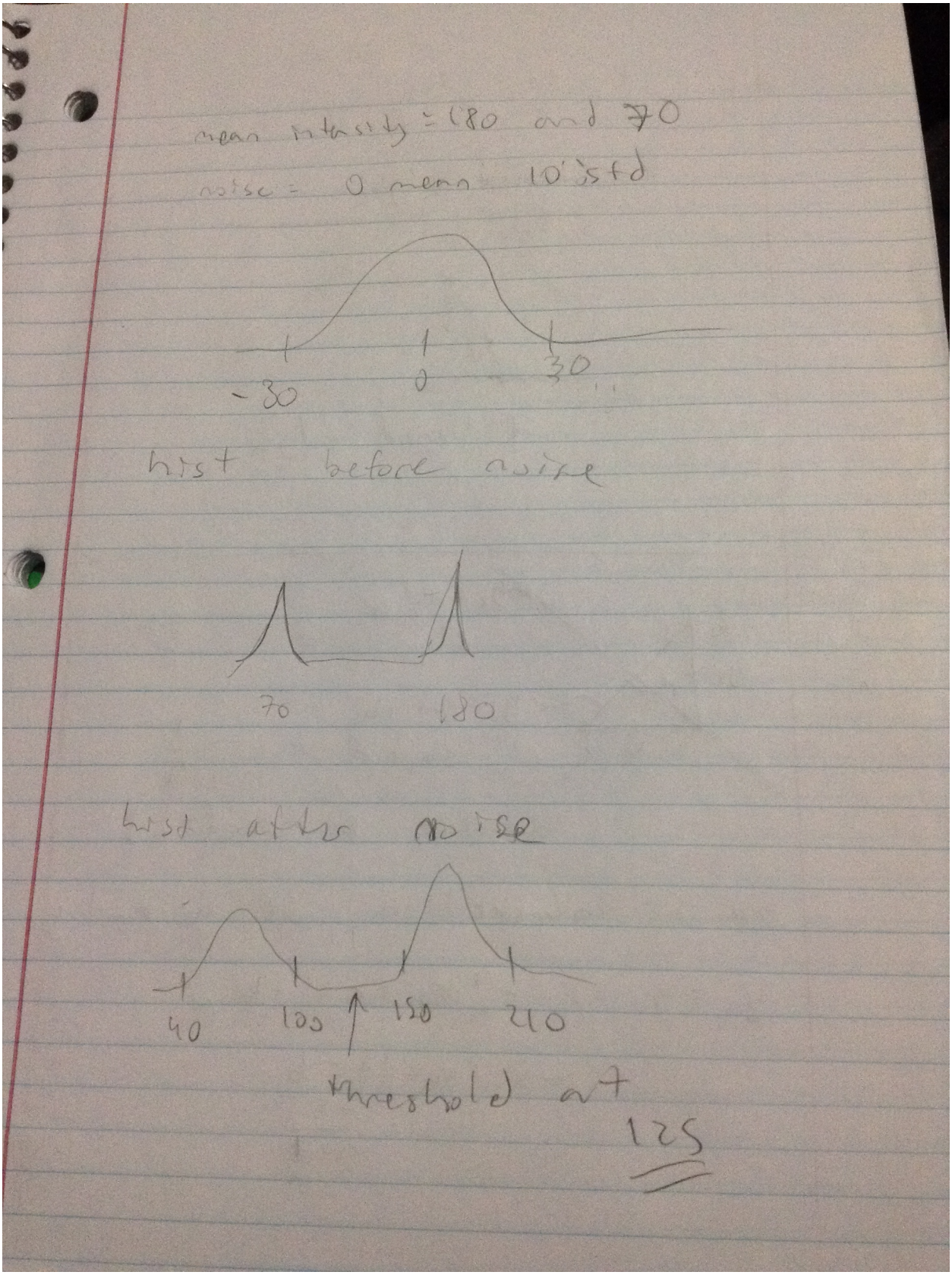
If $y = -3x + 2$ then $a = -3$ and $b = 2$

$$\theta = atan(-\frac{1}{a}) = atan(-\frac{1}{-3}) = 18.4°)$$

$$x_0 = -\frac{b}{(\frac{1}{a} + a)} = -\frac{2}{(\frac{1}{-3} + -3)} = .6$$

$$r = \frac{x_0}{cos(\theta)} = \frac{.6}{cos(18.4°)} = \frac{.6}{.95} = .63$$

3 Problem 10.36



4 Problem 4

The error for a given T threshold value can be obtained from the following:

$$E(T) = P_1 \int_T^{\infty} p_1(z) + P_2 \int_{-\infty}^{T} p_2(z)$$

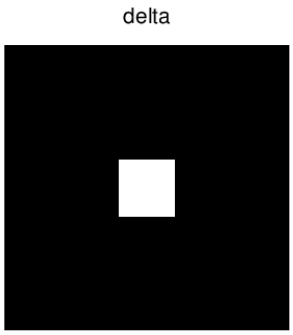
$$\frac{d}{dt}E(T) = -P_1 p_1(T) + P_2 p_2(T)$$
$$0 = -P_1 p_1(T) + P_2 p_2(T)$$
$$P_1 p_1(T) = P_2 p_2(T)$$

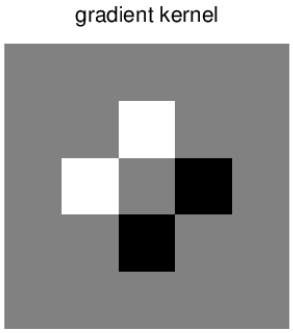
Since $P_2 = P_1$

$$p_1(T) = p_2(T)$$
$$1 - .5T = -.5 + .5T$$
$$1.5 = T$$

5 Problem 5

5.1 Gradient Kernel





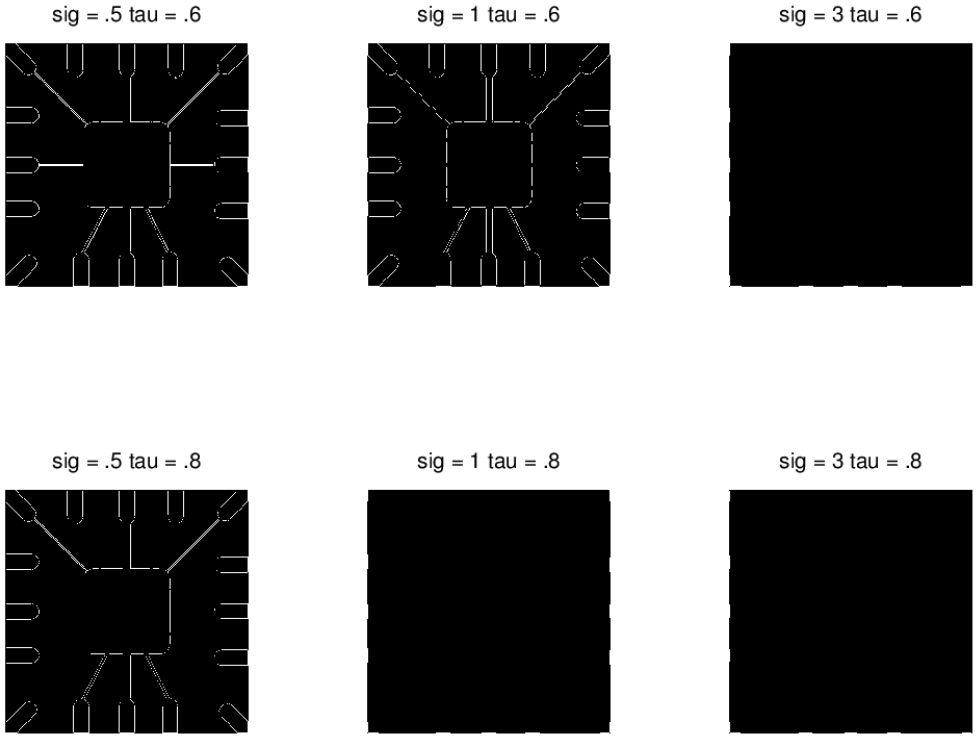
The kernel used is [.5 0 -.5] in each direction.

5.2 Implement a simple version of the Canny edge detector

```
function [E,M,A] = canny(I,sig,tau)

    smoothedImage = imsmooth(I,"Gaussian",sig);

    horizontalFilter = [-1 -1 -1;0 0 0;1 1 1;];
    horizontalEdges = imfilter(smoothedImage,horizontalFilter);

    verticalFilter = horizontalFilter';
    verticalEdges = imfilter(smoothedImage,verticalFilter);

    edgeGradients = sqrt(verticalEdges.^2 + horizontalEdges.^2);
    edgeGradients = normalizeImage(edgeGradients);
    M = edgeGradients;



    edgeDirections = abs(atan2(verticalEdges,horizontalEdges));
    edgeDirections = roundAngleValues(edgeDirections);
    A = edgeDirections;
    edgeGradients = nonMaxSuppression(edgeGradients, edgeDirections);
    E = (edgeGradients > tau);


end

function [roundedImage] = roundAngleValues(image)
    image(isnan(image)) = 0;
    image(image > 0 & image < 22.5*pi/180) = 0;
    image(image > 157.5*pi/180 & image < 180*pi/180) = 0;
    image(image > 22.5*pi/180 & image < 67.5*pi/180) = 45;
    image(image > 67.5*pi/180 & image < 112.5*pi/180) = 90;
    image(image > 112.5*pi/180 & image < 157.5*pi/180) = 135;
    roundedImage = image;
end

function [suppressedImage] = nonMaxSuppression(gradImage, angleImage)
    for i=2:rows(gradImage)-1
        for j=2:columns(gradImage)-1
            switch(angleImage(i,j))
                case 0
                    gradImage(i,j) = nonMaxPixel(gradImage(i,j),gradImage(i,j
                        -1),gradImage(i,j+1));
                case 45
                    gradImage(i,j) = nonMaxPixel(gradImage(i,j),gradImage(i-1,
                        j+1),gradImage(i+1,j-1));
                case 90
                    gradImage(i,j) = nonMaxPixel(gradImage(i,j),gradImage(i+1,
                        j),gradImage(i-1,j));
                case 135
                    gradImage(i,j) = nonMaxPixel(gradImage(i,j),gradImage(i-1,
                        j-1),gradImage(i+1,j+1));
                otherwise
```

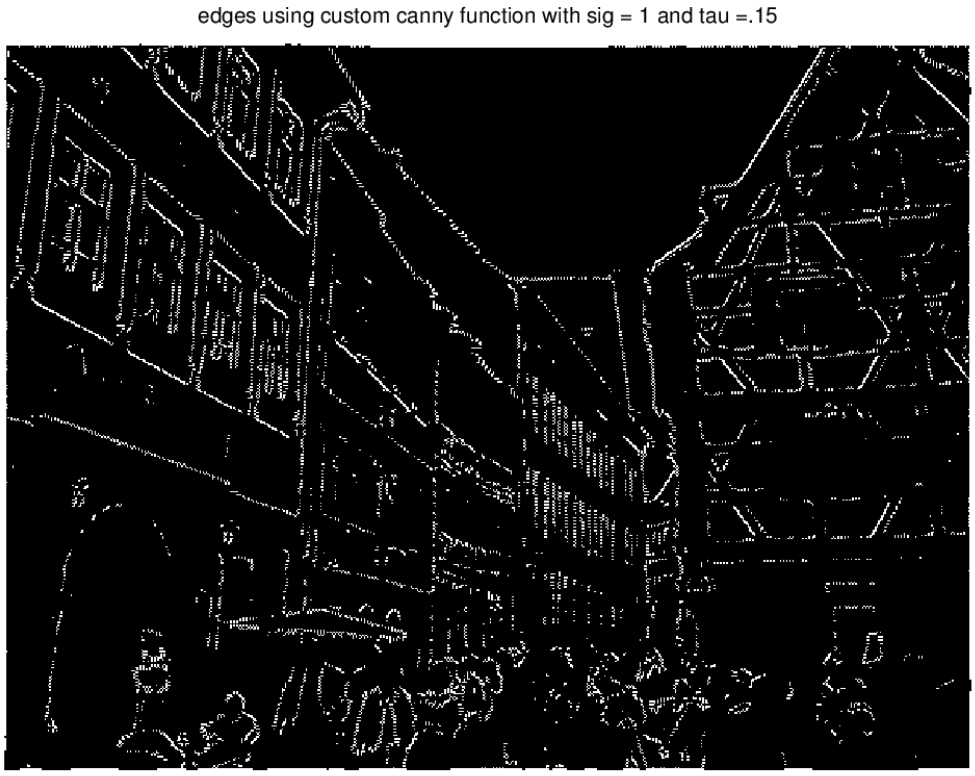
```
                    gradImage(i,j) = nonMaxPixel(gradImage(i,j),gradImage(i,j
                        -1),gradImage(i,j+1));
            end
        end
    end
    suppressedImage = gradImage;
end

function [pixelValue] = nonMaxPixel(currentPixel,adjPixel1,adjPixel2)
        if(currentPixel < adjPixel1 || currentPixel < adjPixel2)
                pixelValue = 0;
        else
                pixelValue = currentPixel;
        end
end
```

5.3 Apply the canny edge detector to wirebondmask.tif



Applying bigger filters to smooth the image before edge detection appears to decrease the intensity of the edges. This can be seen by the smaller amount of edges that are visible when using the same threshold value. This problem appears to be caused by border artifacts in the images.

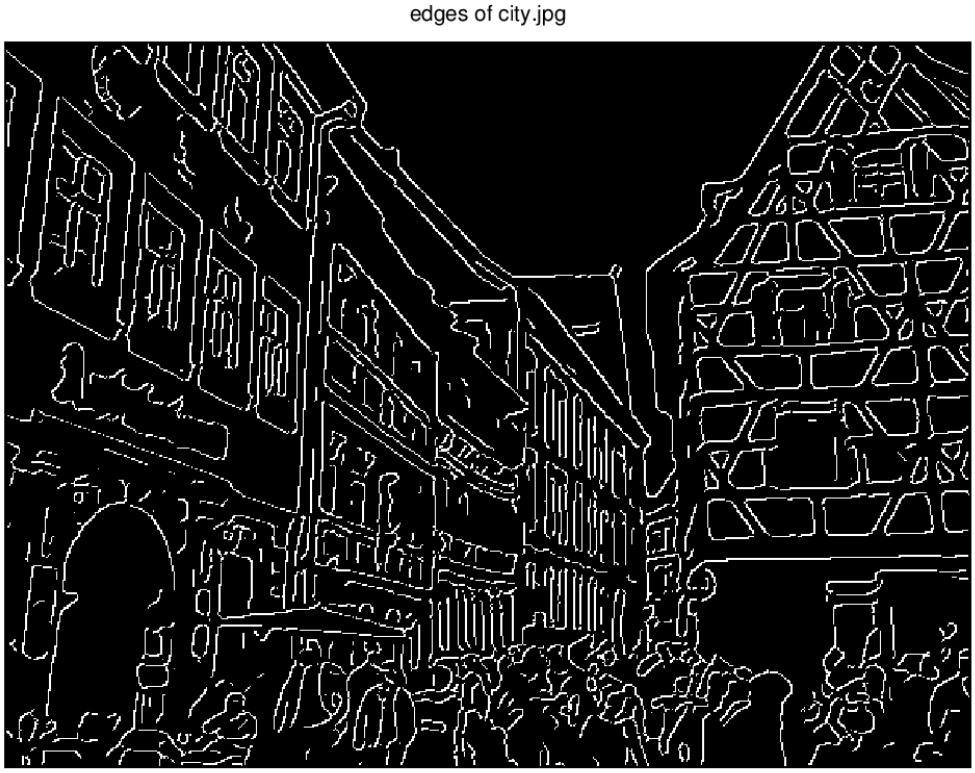
5.4 Apply the canny edge detector to city.jpg

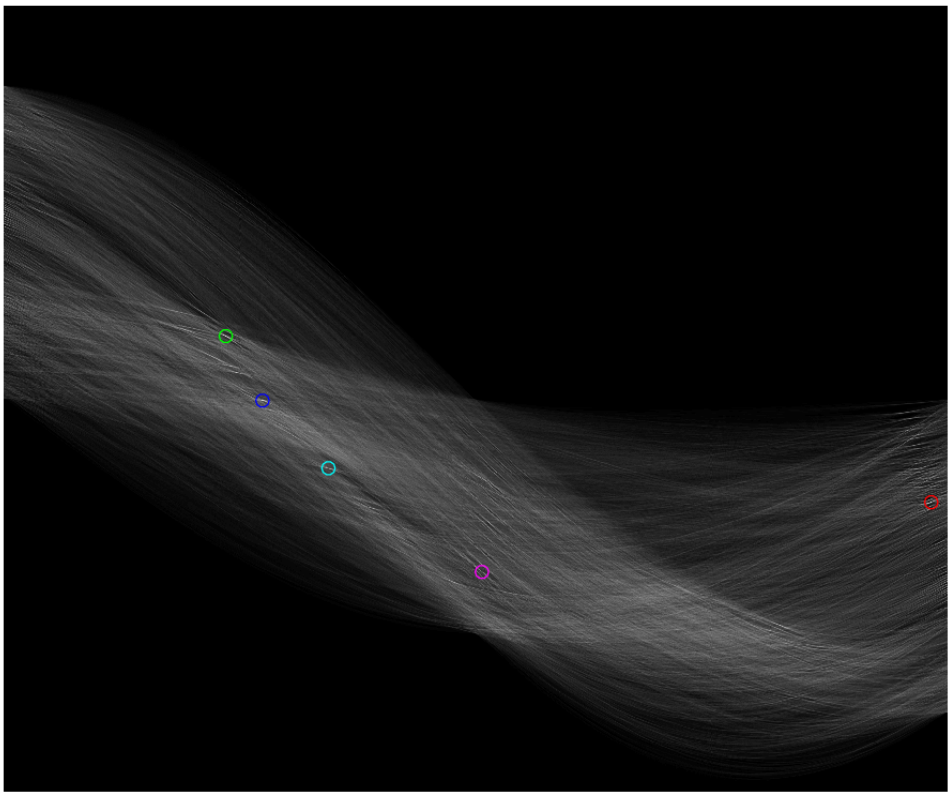


The results of the matlab function are much better than the custom made one. The difference in performance can be attributed to the lack of hysteresis thresholding and border artifacts. The border artifacts mess up the thresholding, since they have a much higher intensity, while the lack of hysteresis thresholding makes the edges look choppy.

6 Problem 6

6.1 Apply the line hough transform to city.jpg

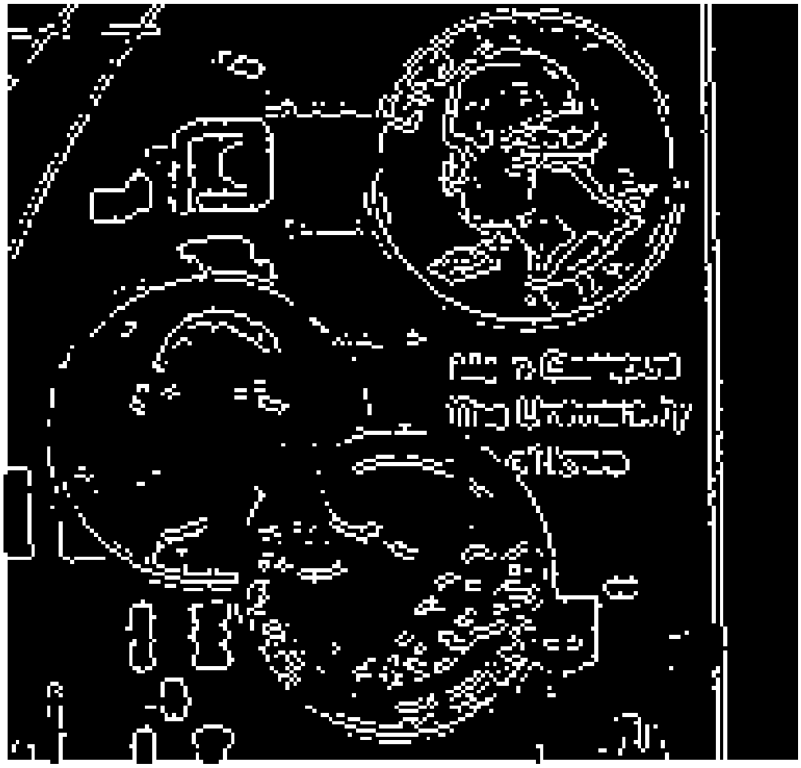



hough transform of city.jpg

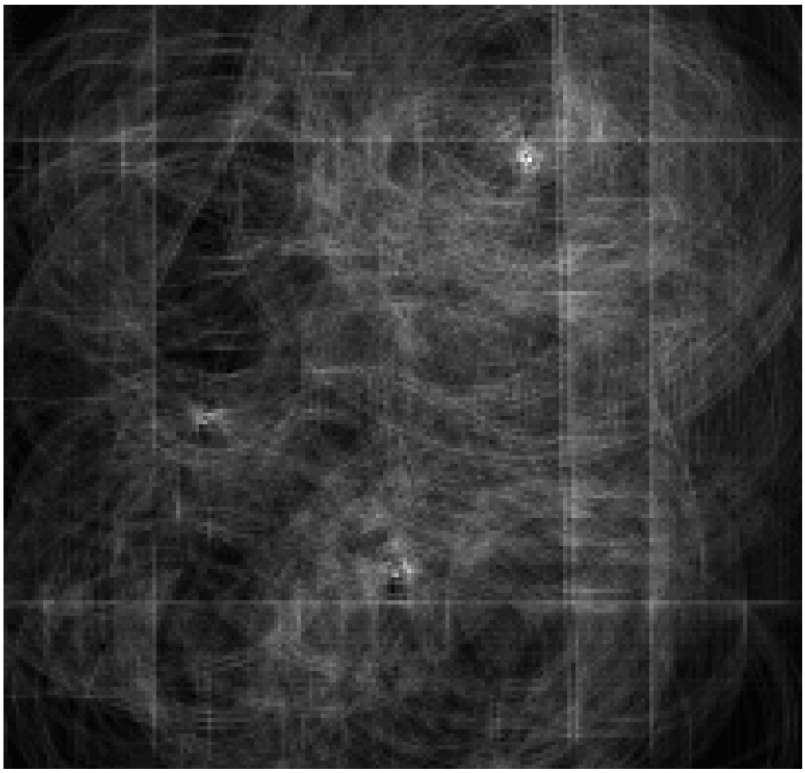
city.jpg + lines from hough transform

6.2 Apply the circle hough transform to quarters.bmp

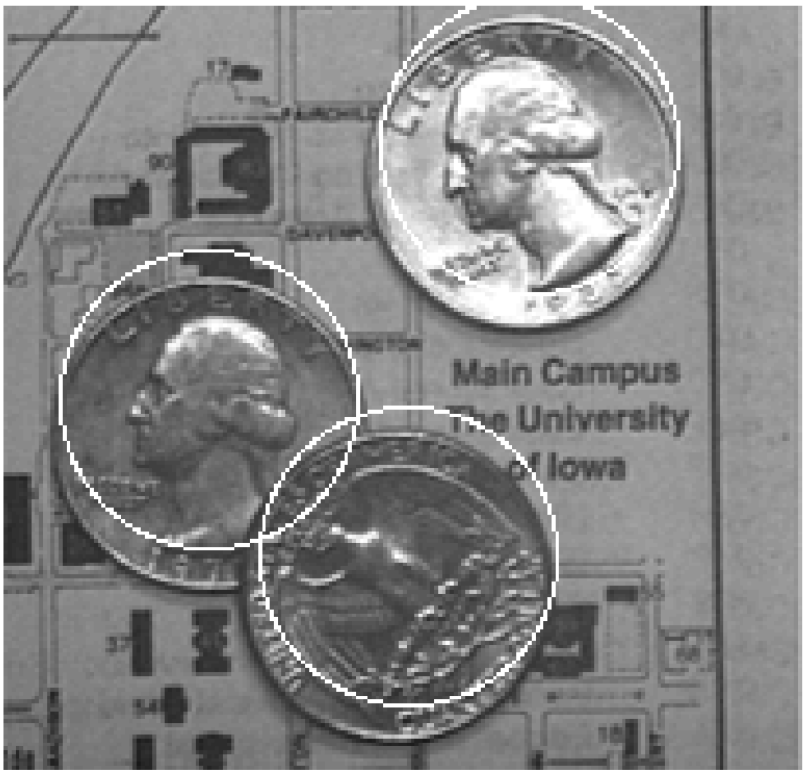
edges of quarters.bmp



hough transform of quarters.bmp



circles from hough transform + quarters.bmp



7 Code

Part 5.A

```
x = [0  0  0  0  0;
     0  0  0  0  0;
     0  0  1  0  0;
     0  0  0  0  0;
     0  0  0  0  0;]
[dx,dy] = gradient(x)
figure(1)
subplot(2,1,1)
imshow(x)
title('delta');
subplot(2,1,2)
imshow(dx+dy+.5);
title('gradient kernel');
print(figure(1),'partA.png');
```

Part 5.B

```
function [E,M,A] = canny(I,sig,tau)

    smoothedImage = imsmooth(I,"Gaussian",sig);

    horizontalFilter = [-1 -1 -1;0 0 0;1 1 1;];
    horizontalEdges = imfilter(smoothedImage,horizontalFilter);

    verticalFilter = horizontalFilter';
    verticalEdges = imfilter(smoothedImage,verticalFilter);

    edgeGradients = sqrt(verticalEdges.^2 + horizontalEdges.^2);
    edgeGradients = normalizeImage(edgeGradients);
    M = edgeGradients;



    edgeDirections = abs(atan2(verticalEdges,horizontalEdges));
    edgeDirections = roundAngleValues(edgeDirections);
    A = edgeDirections;
    edgeGradients = nonMaxSuppression(edgeGradients, edgeDirections);
    E = (edgeGradients > tau);


end

function [roundedImage] = roundAngleValues(image)
    image(isnan(image)) = 0;
    image(image > 0 & image < 22.5*pi/180) = 0;
    image(image > 157.5*pi/180 & image < 180*pi/180) = 0;
    image(image > 22.5*pi/180 & image < 67.5*pi/180) = 45;
    image(image > 67.5*pi/180 & image < 112.5*pi/180) = 90;
    image(image > 112.5*pi/180 & image < 157.5*pi/180) = 135;
    roundedImage = image;
end

function [suppressedImage] = nonMaxSuppression(gradImage, angleImage)
    for i=2:rows(gradImage)-1
        for j=2:columns(gradImage)-1
            switch(angleImage(i,j))
                case 0
                    gradImage(i,j) = nonMaxPixel(gradImage(i,j),gradImage(i,j
                        -1),gradImage(i,j+1));
                case 45
                    gradImage(i,j) = nonMaxPixel(gradImage(i,j),gradImage(i-1,
                        j+1),gradImage(i+1,j-1));
                case 90
                    gradImage(i,j) = nonMaxPixel(gradImage(i,j),gradImage(i+1,
                        j),gradImage(i-1,j));
                case 135
                    gradImage(i,j) = nonMaxPixel(gradImage(i,j),gradImage(i-1,
                        j-1),gradImage(i+1,j+1));
                otherwise
```

```
                    gradImage(i,j) = nonMaxPixel(gradImage(i,j),gradImage(i,j
                        -1),gradImage(i,j+1));
            end
        end
    end
    suppressedImage = gradImage;
end

function [pixelValue] = nonMaxPixel(currentPixel,adjPixel1,adjPixel2)
        if(currentPixel < adjPixel1 || currentPixel < adjPixel2)
                pixelValue = 0;
        else
                pixelValue = currentPixel;
        end
end
```

Part 5.C

```
image = imread('wirebond_mask.tif');
image = double(image)/255;
sig = [.5,1,3];
t = [.6,.8];
k = 1;
figure(1);

for i=1:length(t)
        for j=1:length(sig)
                edges = canny(image,sig(j),t(i));
                subplot(2,3,k);
                imshow(edges);
                k = k+1;
        end
end
```

Part 5.D

```
image = imread('city.jpg');
image = rgb2gray(image);
edges = edge(image,'canny');
figure(1);
title('edges using edge function with default parameters');
imshow(edges);
image = double(image);
[edges,gradMag,gradAngle] = canny(image,1,.15);
figure(2);
title('edges using custom canny function with sig = .5 and tau =.1');
imshow(edges);
```

Utility Code

```
function [normalizedImage] = normalizeImage(image)
        MI = min(min(image));
        MA = max(max(image));
        normalizedImage = (image-MI)./(MA-MI);
end
```

Part 5.A

```
image = imread('city.jpg');
image = rgb2gray(image);


edges = edge(image,'Canny');
theta =pi*(-90:.1:90)/180;
[houghTransform,R] = hough_line(edges,theta);
figure(1);
houghTransform = normalizeImage(houghTransform);
title('hough transform of city.jpg');
imshow(houghTransform);
[dIx,thetaIx] = find(houghTransform == max(max(houghTransform)));

figure(2)
imshow(image)
title('city.jpg + lines from hough transform');
hold on

figure(1)
title('hough transform of city.jpg');
k = 1;


hold on
colors = ['r' 'g' 'b' 'm' 'c'];
for i=1:5
        [dIx,thetaIx] = find(houghTransform == max(max(houghTransform)));
        x = -rows(image)*2:1:rows(image)*2;
        d = R(dIx(1));
        th = theta(thetaIx(1));
        figure(1)
        plot(thetaIx(1),dIx(1),[colors(i) 'o']);
        y = (x*cos(th) - d)/sin(th);
        figure(2)
        plot(x,-y,colors(i));
        for j= dIx(1)-10:dIx(1)+10
            for k = thetaIx(1)-10:thetaIx(1)+10
                if j > 0 & k > 0
                    houghTransform(j,k) = 0;
                end
            end
        end
end
hold off
figure(3)
title('edges of city.jpg');
imshow(edges);

print(figure(1),'partA1.png');
print(figure(2),'partA2.png');
print(figure(3),'partA3.png');
```

Part 5.B

```
function [accumulator] = houghCircle(image,radius)
        accScale = 1;
        [X,Y] = meshgrid(1:1/accScale:columns(image),1:1/accScale:rows(image))
            ;
        X = X-ceil(columns(image)/2);
        Y = Y-ceil(rows(image)/2);
        accumulator = zeros(rows(X),columns(X));
        for i=1:rows(image)
                for j=1:columns(image);
                        if(image(i,j) == 1);
                                x0 = j-ceil(columns(image)/2);
                                y0 = i-ceil(columns(image)/2);
                                Z = (X-x0).^2 + (Y-y0).^2 > radius.^2 -radius
                                    & (X-x0).^2 + (Y-y0).^2 < radius.^2 +
                                     radius;
                                accumulator = accumulator + Z;
                        end
                end
        end
        accumulator = normalizeImage(accumulator);
        %{
        figure(2)
        imshow(accumulator);

        circles = zeros(rows(image),columns(image));
        temp = accumulator;
        [X,Y] = meshgrid(1:1:columns(image),1:1:rows(image));

        for i=1:2
                [y,x] = find(temp == max(max(temp)));
                for j=1:length(x)
                        x0 = round(x(j,1)/accScale);
                        y0 = round(y(j,1)/accScale);
                        Z = ((X-x0).^2 + (Y-y0).^2 == radius.^2);
                        circles = circles + Z;
                        temp(temp == max(max(temp))) = 0;
                end
        end
        figure(3)
        imshow(circles);
        %}
end
```

```
%[X,Y] = meshgrid(-100:1:100,-100:1:100);
%Z = (X-20).^2 + (Y-20).^2 > 990 & (X-20).^2 + (Y-20).^2 < 1100;
%imshow(Z);
radius = 38;
image = imread('quarters.bmp');
edgeImage = edge(image,'Sobel');
figure(1)
title('edges of quarters.bmp');
imshow(edgeImage);

accumulator = houghCircle(edgeImage,radius);


figure(2)
title('hough transform of quarters.bmp');
imshow(accumulator)



circles = zeros(rows(image),columns(image));
[X,Y] = meshgrid(1:1:columns(image),1:1:rows(image));
X = X-ceil(columns(image)/2);
Y = Y-ceil(rows(image)/2);
k = 1;

for i=1:3
        [colIx,rIx] = find(accumulator == max(max(accumulator)));
        x0 = rIx(1)-ceil(columns(image)/2);
        y0 = colIx(1)-ceil(columns(image)/2);
        Z = (X-x0).^2 + (Y-y0).^2 > radius.^2 -radius & (X-x0).^2 + (Y-y0).^2
            < radius.^2 + radius;
        circles = circles + Z;
        for j= colIx(1)-10:colIx(1)+10
            for k = rIx(1)-10:rIx(1)+10
                if j > 0 & k > 0
                    accumulator(j,k) = 0;
                end
            end
        end
end
figure(3);
title('circles from hough transform + quarters.bmp');
imshow(circles*255+image);

print(figure(1),'partB1.png');
print(figure(2),'partB2.png');
print(figure(3),'partB3.png');
%}
```