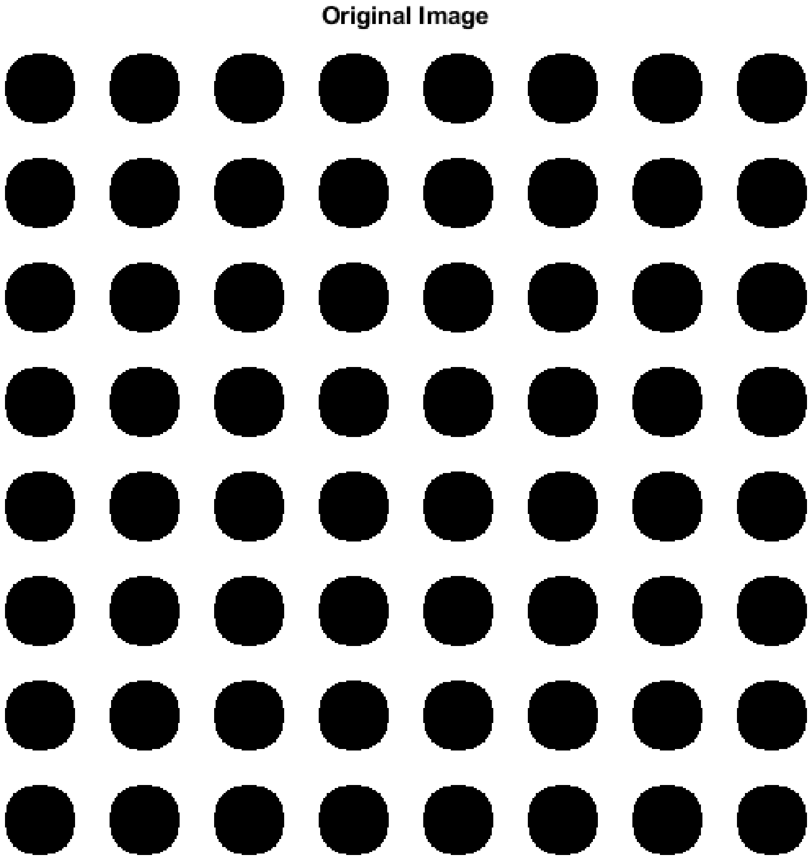
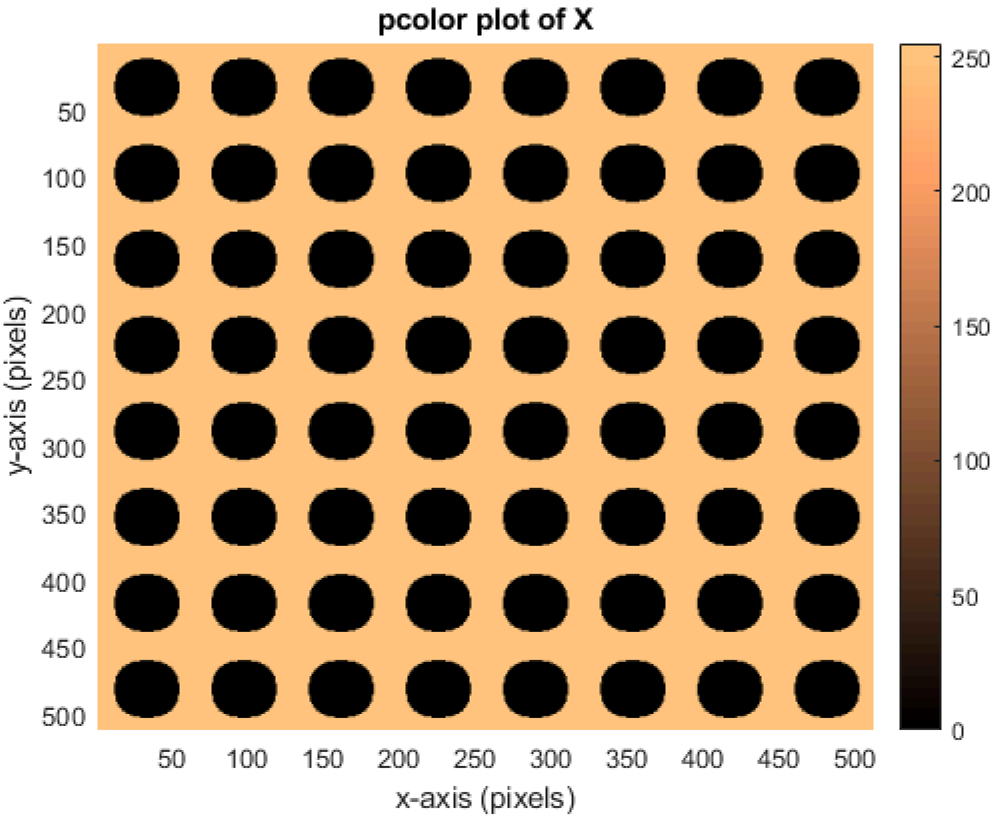
BE 601 HW3 Part 1a

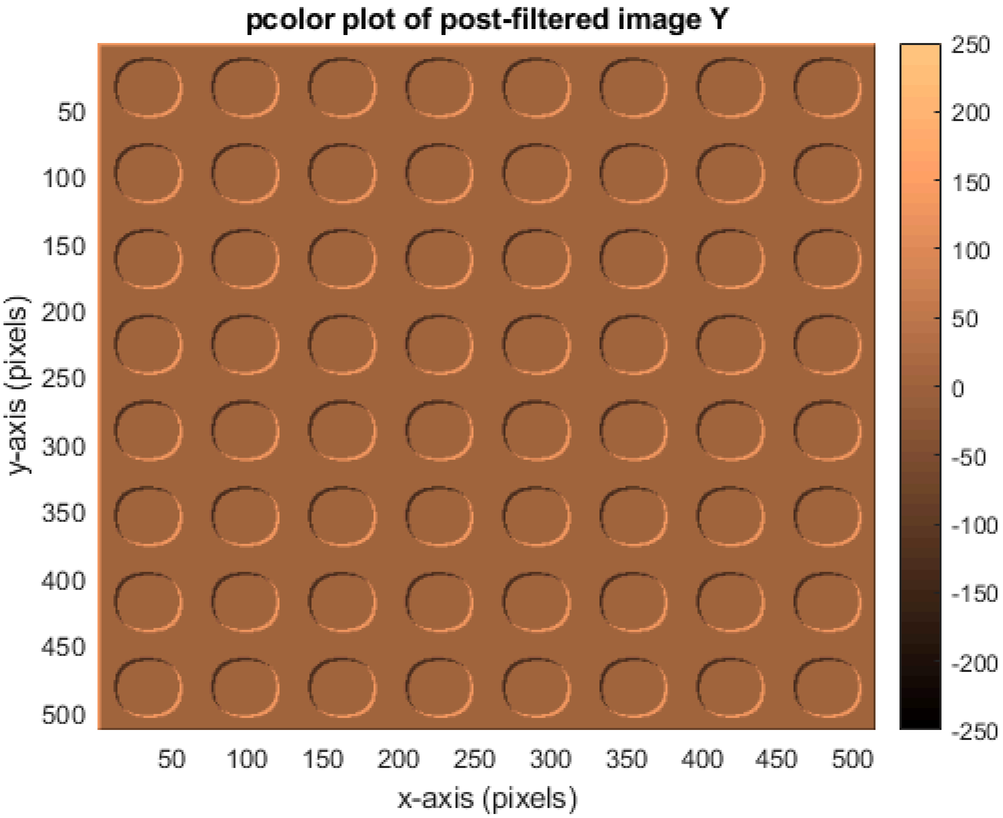
```
clear all
close all
clc

H = 0.5 * [1 0 0; 0 0 0; 0 0 -1];
X = imread('lattice_bigCircles_big_grid.tif');
imshow(X)
title('Original Image')

X = double(X);

f2 = figure('name', 'pcolor plot of X');
h = pcolor(X);
%Because all I see is a black image I removed edge colors to see the
%features
h.EdgeColor = 'none';

hold on
colormap(copper)
colorbar
xlabel('x-axis (pixels)');
ylabel('y-axis (pixels)');
title('pcolor plot of X');
set(gca, 'YDir', 'reverse');


Y = conv2(X, H);

f3 = figure('name', 'pcolor plot of post-filtered image Y');
h1 = pcolor(Y);
%Because all I see is a black image I removed edge colors to see the
%features
h1.EdgeColor = 'none';
title('pcolor plot of post-filtered image Y');
hold on
colormap(copper)
colorbar
caxis([-250 250]);
set(gca, 'YDir', 'reverse');
xlabel('x-axis (pixels)');
ylabel('y-axis (pixels)');

Y = uint8(Y);
f4 = figure('name', 'Post Filtered Image Y');
imshow(Y)
title('Post-Filtered Image')

%disp answer to give me a plausible, mathematical (calculus-based)
 explanation
```

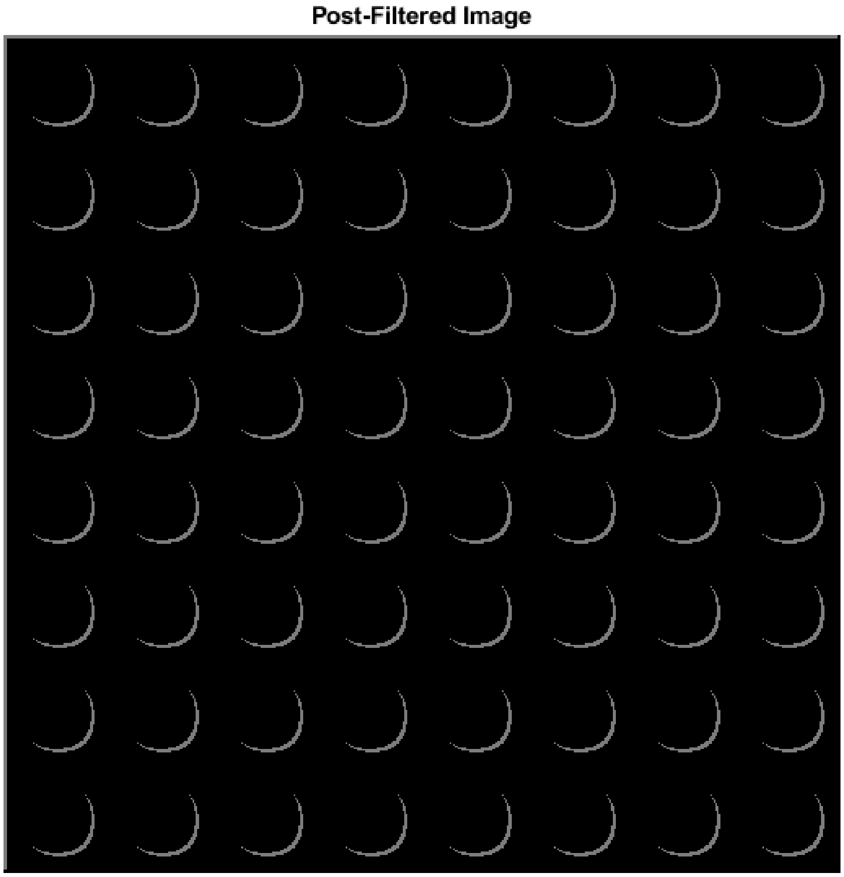
```
% on why the action of ?[?,?]is to highlight uni-directional edges on
 our input data.

disp('H[m,n] highlights uni-directional edges as du/dx(center
 diagonal)')
```

```
H[m,n] highlights uni-directional edges as du/dx(center diagonal)
```




pcolor plot of X
y-axis (pixels)
50
100
150
200
250
300
350
400
450
500
x-axis (pixels)
50
100
150
200
250
300
350
400
450
500
250
200
150
100
50
0


pcolor plot of post-filtered image Y
y-axis (pixels)
50
100
150
200
250
300
350
400
450
500
x-axis (pixels)
50
100
150
200
250
300
350
400
450
500
250
200
150
100
50
0
-50
-100
-150
-200
-250



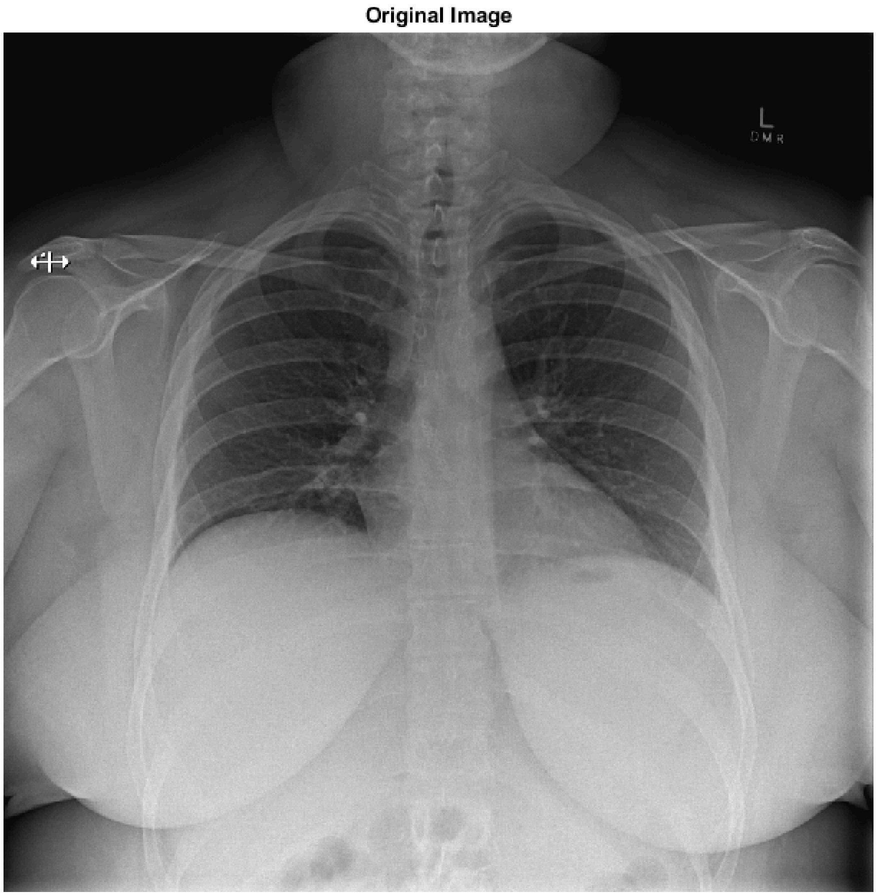
Post-Filtered Image

Published with MATLAB® R2018b

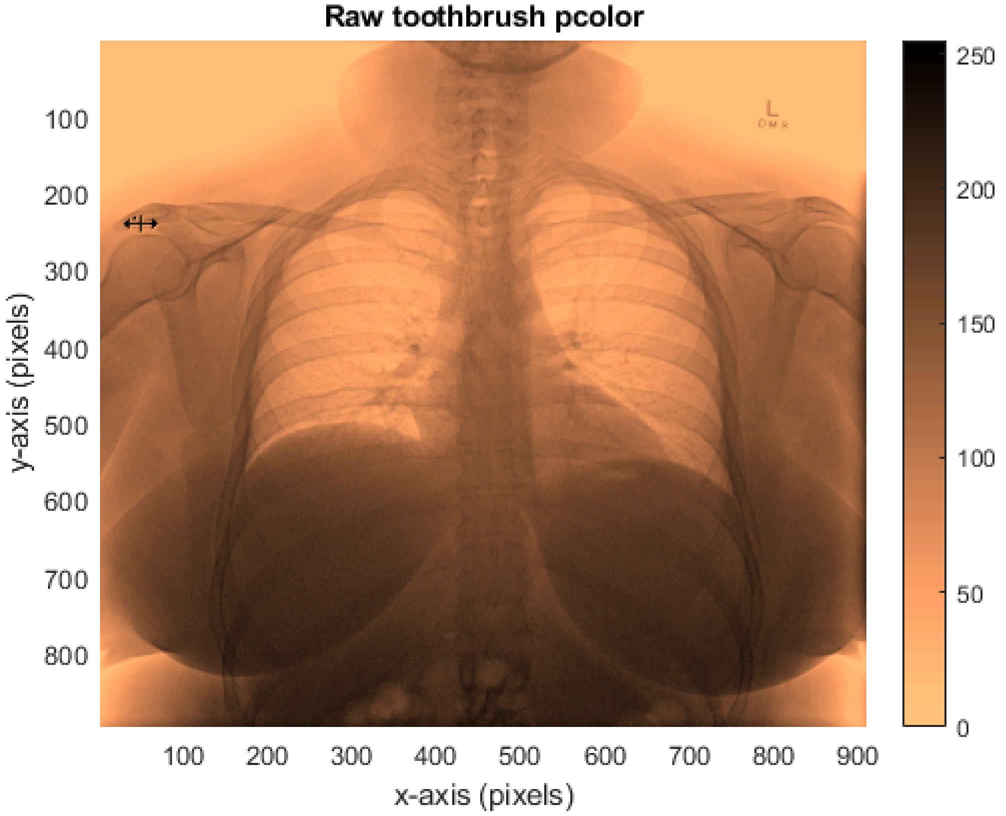
```
%%BE601HW3 Prob1b
%part 1dfeine the 6 filters using disp
close all
clear all
clc

H1 = [0 0 0; -1 2 -1; 0 0 0];
H2 = [0 -1 0; -1 4 -1; 0 -1 0];
H3 = 0.25 * [0 0 0; 1 2 1; 0 0 0];
H4 = [-1 0 1; -2 0 2; -1 0 1];
H5 = 0.0625 * [1 2 1; 2 4 2; 1 2 1];
H6 = [-1 -1 -1; -1 9 -1; -1 -1 -1];
disp('H4 is a edge detection filter with a first order x derivative
 times a gaussian filter')
disp('H3 is a smoothing filter using integrals  i.e. 1/(b-a) *
 integral(u)dx')
disp('H5 is a smoothing filter using an integral of the local area.')
disp('H1 is an edge detection filter taking the derivative along the
 center')
disp('H2 is an edge detection filter; 2nd order partial differential
 of the function')
disp('H6 is combination of both edge detection and smoothing, with
 part smoothing using similar integral gaussian as well as centerpoint
 detection using a first order derivative')
% explain here all the filters math and jargon

X = imread('swallowed_toothbrush_verb_frontal.tif');
imshow(X)
title('Original Image')
X = double(X);

f2 = figure('name', 'Toothbrush pcolor');
h = pcolor(X);
title('Raw toothbrush pcolor')
set(gca, 'YDir', 'reverse');
colorbar
h.EdgeColor = 'none';
xlabel('x-axis (pixels)');
ylabel('y-axis (pixels)');
colormap(flipud(copper));

H1 = [0 -1 0; -1 4 -1; 0 -1 0];
H2 = [0 -1 0; -1 5 -1; 0 -1 0];
H3 = [-1 -1 -1; -1 9 -1; -1 -1 -1];
H4 = 0.04 * ones(5,5);
H5 = 0.00390625 * [1 4 6 4 1; 4 16 24 16 4; 6 24 36 24 6; 4 16 24 16
 4; 1 4 6 4 1];
H6 = [-1 0 1; -2 0 2; -1 0 1];

xx = {H1 H2 H3 H4 H5 H6};
f3 = figure('name', 'Toothbrush pcolor_ID');
%
```

```
% for i = 1:length(xx)
%     X1 = conv2(X,xx{i});
% %     h1 = pcolor(X1);
% %     colormap(flipud(copper));
% %     h1.EdgeColor = 'none';
%     Y = uint8(X1);
%     imshow(Y);
%     disp(xx{i})
%     title(xx{i});
%     pause(5)
% end


X1 = conv2(X,xx{5});
h1 = pcolor(X1);
title('Filtered pcolor image');
colormap(flipud(copper)); %i prefer using colormap(bone)
h1.EdgeColor = 'none';
caxis([-250 250]);
colorbar
xlabel('x-axis (pixels)');
ylabel('y-axis (pixels)');
set(gca, 'YDir', 'reverse');

f4 = figure('name', 'Post Filtered Image');

Y = uint8(X1);
imshow(Y)
title('Post Filtered Image using H5');
```

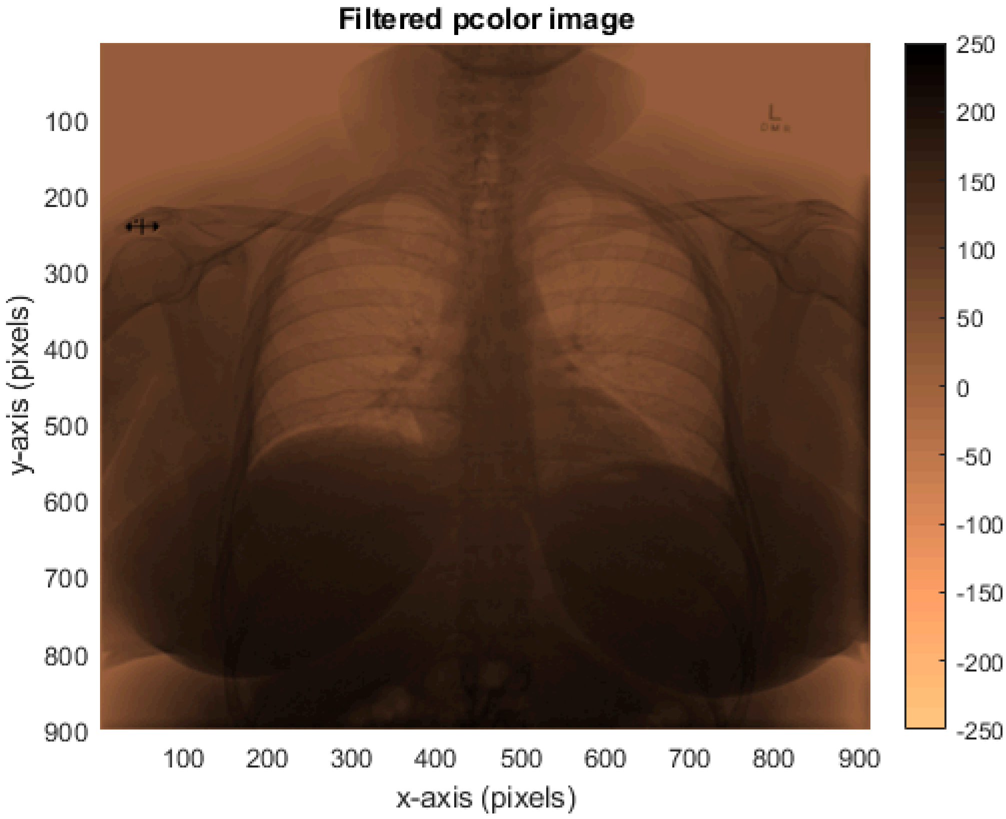
H4 is a edge detection filter with a first order x derivative times a gaussian filter
*H3 is a smoothing filter using integrals i.e. 1/(b-a) * integral(u)dx*
H5 is a smoothing filter using an integral of the local area.
H1 is an edge detection filter taking the derivative along the center
H2 is an edge detection filter; 2nd order partial differential of the function
H6 is combination of both edge detection and smoothing, with part smoothing using similar integral gaussian as well as centerpoint detection using a first order derivative



Original Image

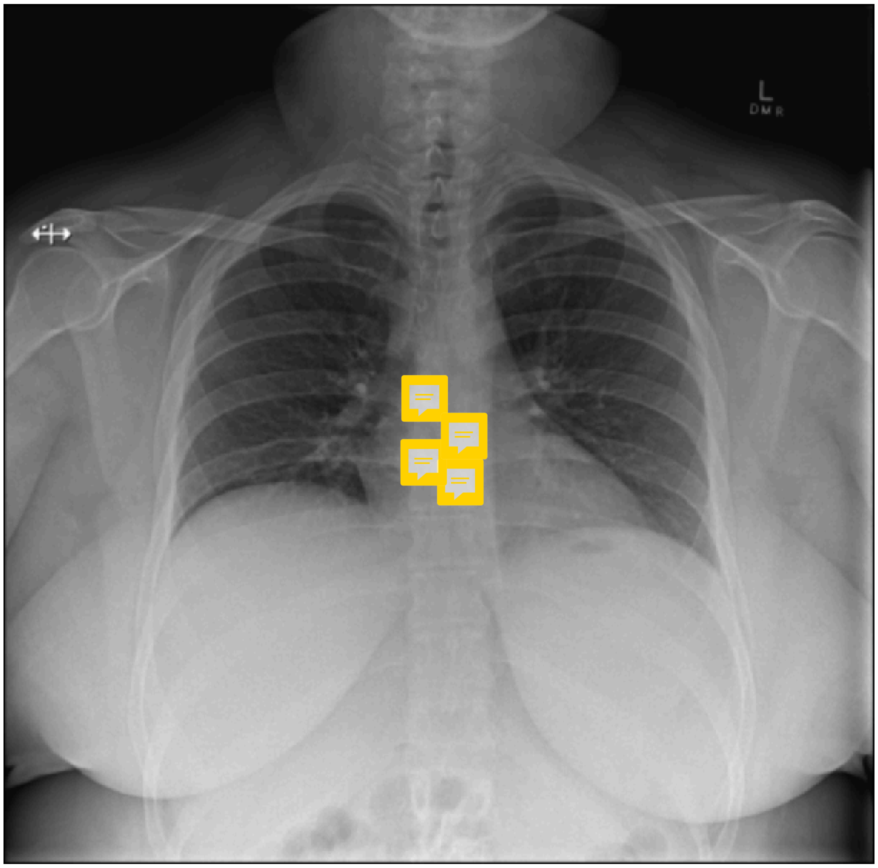


Raw toothbrush pcolor



Filtered pcolor image

Post Filtered Image using H5



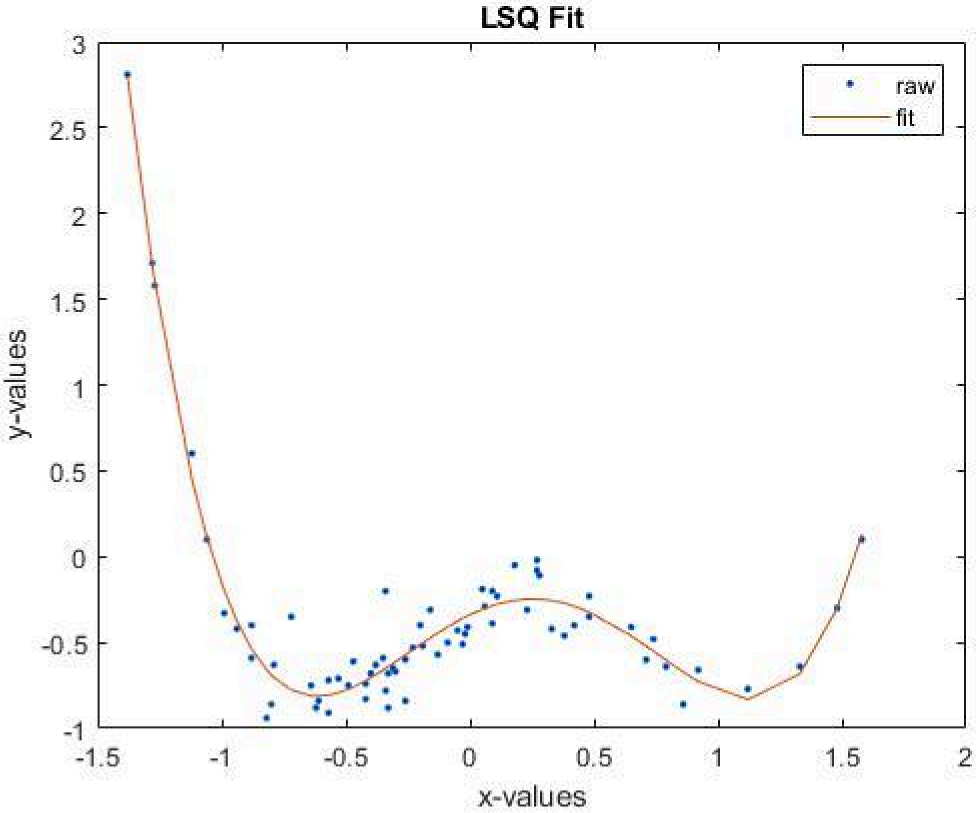
Published with MATLAB® R2018b

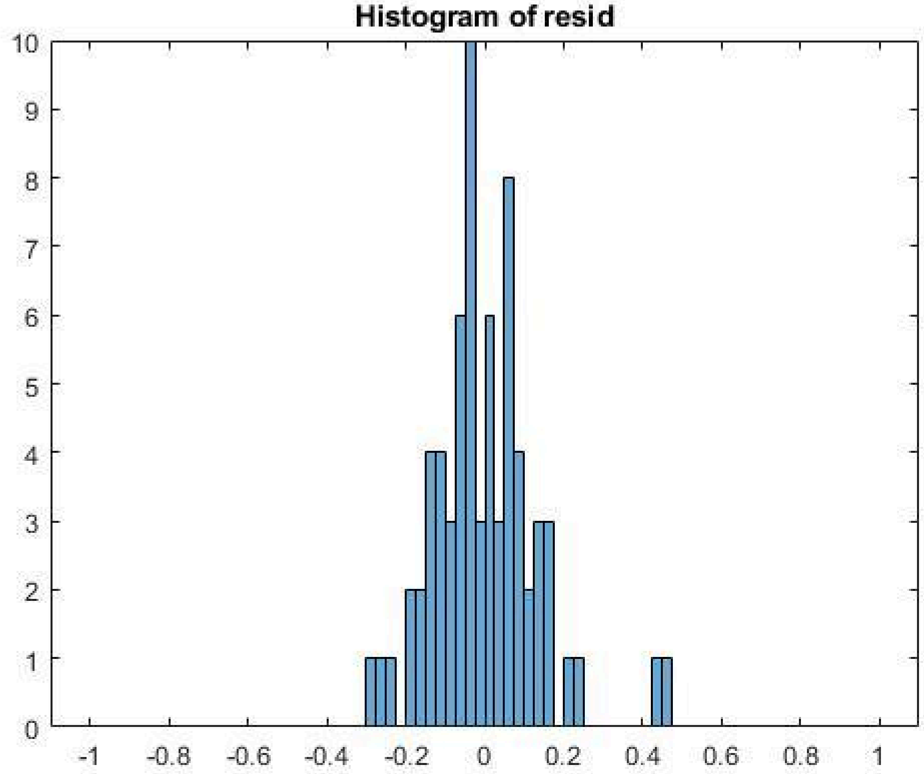
BE601HW3 Prob 2

```
[x, y] = textread('Problem2_polynomial_data.txt', '%f
%f', 'headerlines',1);


A = [ones(70,1) x x.^2 x.^3 x.^4];

c = (A'*A)\(A'*y);
yfit = ones(1,70);
cnew = fliplr(c');
x2 = x';
for i = 1:length(x')
    yfit(i) = polyval(cnew, x2(i));
end

f0 = figure('name', 'LSQ_Fit');

plot(x,y, 'o', 'MarkerFaceColor', 'b', 'MarkerSize',2)
hold on
plot(x, yfit)
title('LSQ Fit')
legend('raw', 'fit')
xlabel('x-values')
ylabel('y-values')


r = y - yfit';

r2_sum = sum(r.^2);

f1 = figure('name', 'Histogram of resid');
edges = [-1:0.025:1];
xlim([-1 1]);
ylim([0 10]);
h = histogram(r, edges);
title('Histogram of resid')

r_avg = mean(r);
samp_var = var(r); %built in matlab function uses N-1 formula


disp('Quartic fit seems to be the best.')
r2_sum
var_N1 = samp_var * (69)
```

```
Quartic fit seems to be the best.

r2_sum =

    1.2240
```

```
var_N1 =

    1.2240
```

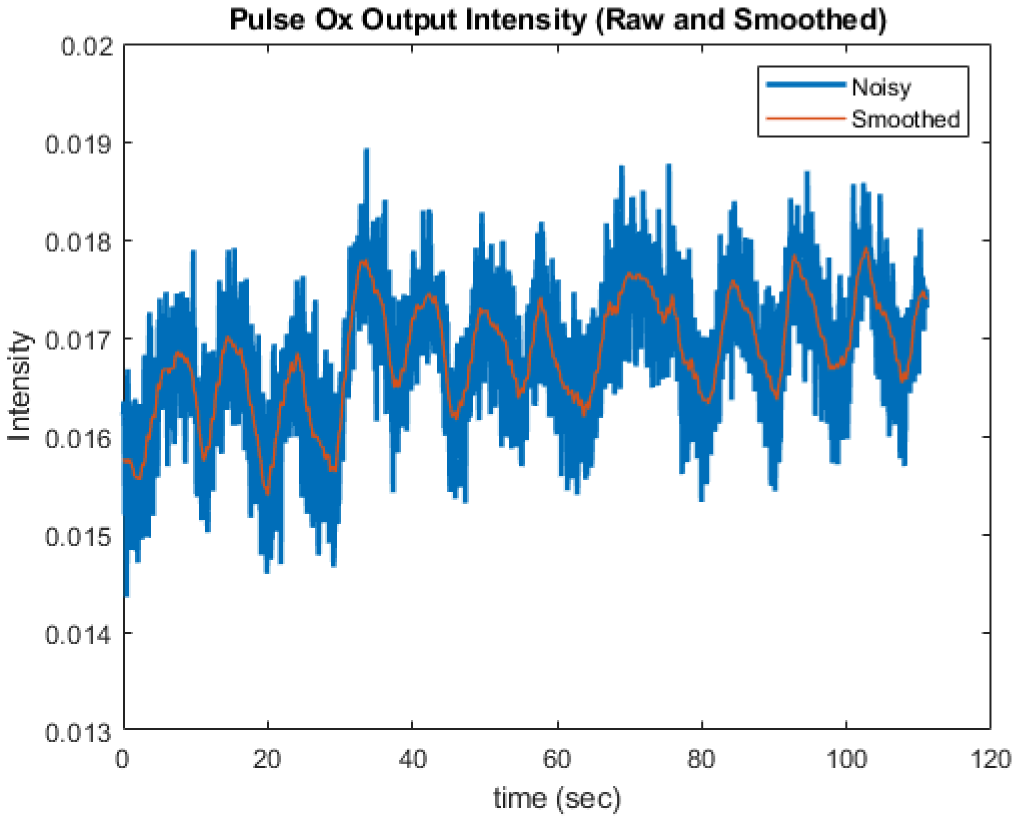

LSQ Fit
raw
fit
y-values
x-values
3
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-1
-0.5
0
0.5
1
1.5
2



Published with MATLAB® R2018b

```
%%BE601 HW3 Prob 3
clear all
close all
clc

[x, y] = textread('my_breathing_data_singleChannel.txt', '%f
%f', 'headerlines',1);
m = 51;
plot(x,y, 'LineWidth',2)
xold = x;

ylim([0.013 0.02])
hold on

y1 = y';
middleValue = [];
%add ghosts to signal
y2 = [fliplr(y1(1:26)) y1 fliplr(y1(1974:2000))];

for i = 1:2000
  x = i : (i + 52); %sliding window
  cnew = LSQ3(x, y2(x)); %cubic least squares
  middleX = mean(x); %finding midpoint of window to place new value
 into
  mv2(i) = polyval(cnew, middleX);
  warning('off', 'all')
end


plot(xold, mv2, 'LineWidth', 1)
legend('Noisy', 'Smoothed')
title('Pulse Ox Output Intensity (Raw and Smoothed)')
xlabel('time (sec)')
ylabel('Intensity')

sampleout = mv2(162:180);

echo on

sampleout

echo off
```

```
sampleout

sampleout =

  Columns 1 through 7

    0.0167    0.0167    0.0167    0.0167    0.0167    0.0167    0.0167
  Columns 8 through 14

    0.0167    0.0166    0.0166    0.0166    0.0166    0.0166    0.0165

  Columns 15 through 19

    0.0165    0.0165    0.0164    0.0164    0.0164

echo off
```



Published with MATLAB® R2018b

BE 601 HW 3 Problem 4

$$f(x) = \begin{cases} L - x & 0 < x < L \\ 0 & L < x < 2L \end{cases}$$

$$a_0 = 2\frac{1}{L}\int_0^{2L} f(x)\,dx = \frac{1}{2L}\int_0^{L} (L-x)\,dx + \frac{1}{L}\int_{L}^{2L} 0\,dx$$

$$= \frac{1}{2L}\left[Lx - \tfrac{1}{2}x^2\right]_0^L$$

$$\frac{1}{2L}\left[L^2 - L^2/2\right]$$

$$a_0 = \frac{L}{2} - L/\cancel{3}4 = \cancel{L/2}\ L/4$$

$$\left[\int_a^b u\frac{dv}{dx}dx = [uv]_a^b - \int_a^b \frac{du}{dx} v\right.$$

$$a_n = \frac{1}{L}\int_0^{2L} f(x)\cos nx\,dx$$

$$= \frac{1}{L}\int_0^{L} (L-x)\cos nx\,dx + \cancel{\frac{1}{L}\int_L^{2L} 0dx} \to 0$$

$$\frac{1}{L}\left[(L-x)\frac{\sin nx}{n}\Big|_0^L - (-1)\int_0^L \frac{\sin nx}{n}dx \right.$$

$$0 + 0 \qquad -\frac{\cos nx}{n}\Big|_0^L \Bigg]$$

$$1/Ln \cdot \frac{-\cos nx}{n}\Big|_0^L \longrightarrow \frac{1}{n^2L}\left(\cos Ln - 1\right)$$

$$a_m = \frac{1}{m^2L}\left(\cos Lm - 1\right)$$

$$b_n = \frac{1}{L}\int_0^{L} (L-x)\sin nx\,dx + \cancel{\int_L^{2L} 0dx} \to 0$$

$$(L-x)\frac{-\cos nx}{n}\Big|_0^L + \int_0^L -\frac{\cos nx}{n}dx$$

$$-L\frac{\cos 0}{n} \qquad \frac{\sin Ln}{n}$$

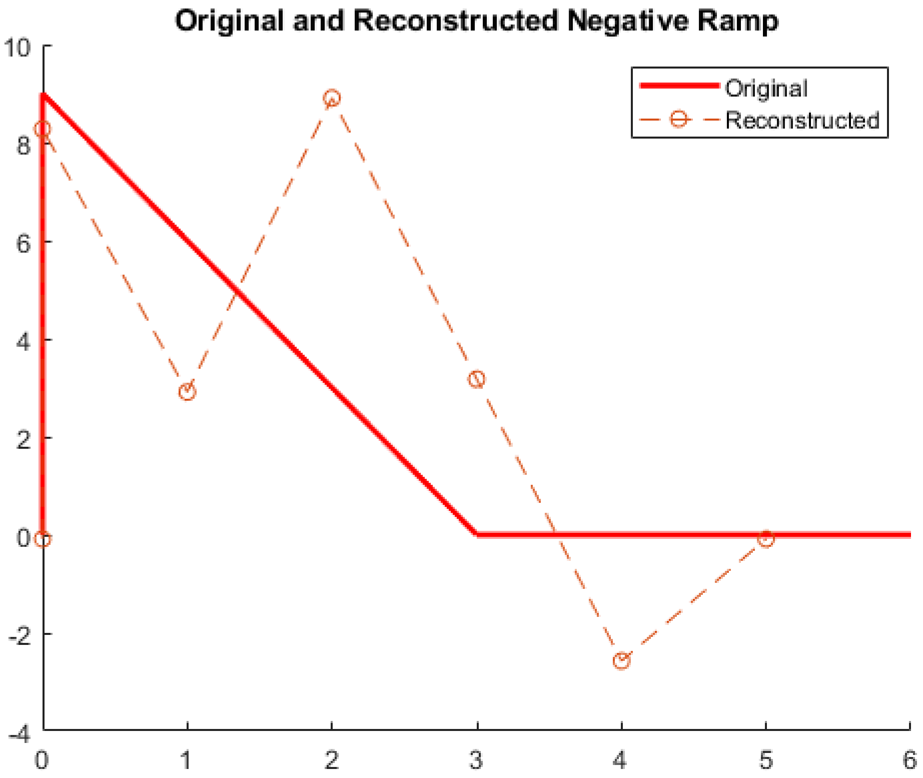
$$+L/n$$

$$1/L\left[L/n + \frac{\sin Ln}{n}\right]$$

$$b_n = \frac{1}{n} + \frac{\sin Ln}{n}$$

```
lh = line([0 0  3 ],  [0 9  0]);
set(lh, 'Color', 'red', 'Linewidth', 2);
x = 3:1:6;
y = zeros(1,4);
hold on
plot(x,y, 'Color', 'red', 'Linewidth', 2, 'HandleVisibility','off')


L = 3;
hold on

% an = 1/((n^2)*L)*(cos(L*n) - 1);
% bn = 1/n + sin(L*n)/n;

a0 = L/4;

for n = 1:5
    an(n) = 1/((n^2)*L)*(cos(L*n) - 1);
    bn(n) = 1/n + sin(L*n)/n;
end

% an(2) = 0;
% an(4) = 0;

x = 0:2*L;

cospart = a0 * cos(0*pi/L * x) + an(1) * cos(1*pi/L * x) + an(2) *
 cos(2*pi/L * x) + an(3) * cos(3*pi/L * x) +...
    an(4) * cos(4*pi/L * x) + an(5) * cos(5*pi/L * x);

sinpart = bn(1) * sin(pi/L * x) + bn(2) * sin(2*pi/L * x) + bn(3) *
 sin(3*pi/L * x) + bn(4) * sin(4*pi/L * x) +...
    bn(5) * sin(5*pi/L * x);


cospart(3) = 0; %n=2 should be 0

sum1 = cospart+sinpart;

plot([0 0 1 2 3 4 5 ],9/1.5166*sum1, 'o--')
%I messed up my integrals somewhere which is why my answers are so far
 off,
%I was unable to reconcile these answers

title('Original and Reconstructed Negative Ramp')
legend('Original', 'Reconstructed')

echo on
a0
an
bn
echo off
```

```
a0

a0 =

    0.7500

an

an =

   -0.6633   -0.0033   -0.0708   -0.0033   -0.0235

bn

bn =

    1.1411    0.3603    0.4707    0.1159    0.3301

echo off
```



Published with MATLAB® R2018b