Digital Signal Processing HW13 MATLAB Part

Name: Jingyang Zhang
Net ID: jz2807

1) Repeat the deblurring demo for images, using second order derivative as the regularization term. Try different blurring filters, different noise levels, and different λ values. Include your code, results (original, blurred and noise added images, deblurred images), and your comments about the effect of different parameters. (Note that the blurring filter does not have to be simple averaging. It could be weighted average, e.g. h = (1, 2, 1)/4. Generally, you should set all coefficients to be positive but sum to. Also, feel free to use any image with sharp edges).

Solution:

.m file(s): jyz_HW13_1.m

Code:

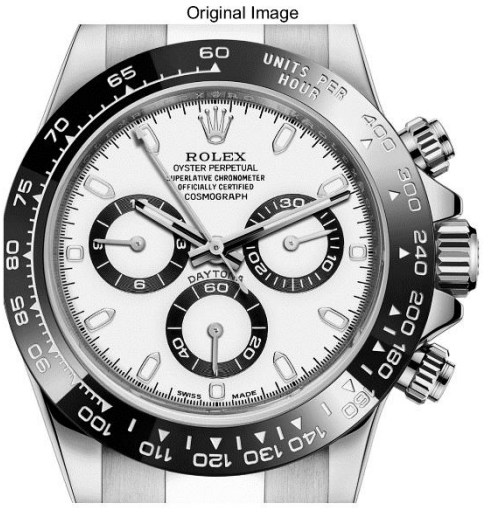
```
close all
clear

% load image and basic process ↓
img = imread('daytona_1024.png');
gimg = double(img);
[height, width] = size(gimg);
imin = min(min(gimg));
imax = max(max(gimg));

h1 = ones(1, 9)/9;
h2 = [1 8 28 56 70 56 28 8 1] / 256;
h3 = ones(1, 5)/5;
h4 = [1 4 6 4 1] / 16;
H1 = convmtx(h1', width);
H2 = convmtx(h2', width);
d_second_order = [1 -2 1]';
D = convmtx(d_second_order, width);
lam1 = 0.1;
lam2 = 1;
lam3 = 5;

% create blurred image
b_img1 = zeros(height, width + 9 - 1);
b_img2 = zeros(height, width + 9 - 1);
for i=1:height
   b_img1(i, :)=conv(gimg(i, 1 : width), h1);
   b_img2(i, :)=conv(gimg(i, 1 : width), h2);
   b_img3(i, :)=conv(gimg(i, 1 : width), h3);
   b_img4(i, :)=conv(gimg(i, 1 : width), h4);
end
[bheight1, bwidth1] = size(b_img1);
[bheight2, bwidth2] = size(b_img2);
```

```
%Add noise
b_img1_L_noi = b_img1 + 5 * randn(bheight1, bwidth1);
b_img1_M_noi = b_img1 + 15 * randn(bheight1, bwidth1);
b_img1_H_noi = b_img1 + 25 * randn(bheight1, bwidth1);
b_img2_L_noi = b_img2 + 5 * randn(bheight2, bwidth2);
b_img2_M_noi = b_img2 + 15 * randn(bheight2, bwidth2);
b_img2_H_noi = b_img2 + 25 * randn(bheight2, bwidth2);

%deconvolution
d_b_img1_L_noi_lam1 = ((H1'*H1 + lam1 * D'*D) \ (H1' * b_img1_L_noi'))';
d_b_img1_M_noi_lam1 = ((H1'*H1 + lam1 * D'*D) \ (H1' * b_img1_M_noi'))';
d_b_img1_H_noi_lam1 = ((H1'*H1 + lam1 * D'*D) \ (H1' * b_img1_H_noi'))';
d_b_img2_L_noi_lam1 = ((H1'*H1 + lam1 * D'*D) \ (H2' * b_img2_L_noi'))';
d_b_img2_M_noi_lam1 = ((H1'*H1 + lam1 * D'*D) \ (H2' * b_img2_M_noi'))';
d_b_img2_H_noi_lam1 = ((H1'*H1 + lam1 * D'*D) \ (H2' * b_img2_H_noi'))';

d_b_img1_L_noi_lam2 = ((H2'*H2 + lam2 * D'*D) \ (H1' * b_img1_L_noi'))';
d_b_img1_M_noi_lam2 = ((H2'*H2 + lam2 * D'*D) \ (H1' * b_img1_M_noi'))';
d_b_img1_H_noi_lam2 = ((H2'*H2 + lam2 * D'*D) \ (H1' * b_img1_H_noi'))';
d_b_img2_L_noi_lam2 = ((H2'*H2 + lam2 * D'*D) \ (H2' * b_img2_L_noi'))';
d_b_img2_M_noi_lam2 = ((H2'*H2 + lam2 * D'*D) \ (H2' * b_img2_M_noi'))';
d_b_img2_H_noi_lam2 = ((H2'*H2 + lam2 * D'*D) \ (H2' * b_img2_H_noi'))';

d_b_img1_L_noi_lam3 = ((H1'*H1 + lam3 * D'*D) \ (H1' * b_img1_L_noi'))';
d_b_img1_M_noi_lam3 = ((H1'*H1 + lam3 * D'*D) \ (H1' * b_img1_M_noi'))';
d_b_img1_H_noi_lam3 = ((H1'*H1 + lam3 * D'*D) \ (H1' * b_img1_H_noi'))';
d_b_img2_L_noi_lam3 = ((H1'*H1 + lam3 * D'*D) \ (H2' * b_img2_L_noi'))';
d_b_img2_M_noi_lam3 = ((H1'*H1 + lam3 * D'*D) \ (H2' * b_img2_M_noi'))';
d_b_img2_H_noi_lam3 = ((H1'*H1 + lam3 * D'*D) \ (H2' * b_img2_H_noi'))';

figure (1)
imshow(gimg,[imin,imax]);
title('Original Image');

figure (2)
subplot(2,2,1)
imshow(b_img1, [imin, imax])
title('Blurred Image 1 by Averaging Filter, L = 9');
subplot(2,2,2)
imshow(b_img2, [imin, imax])
title('Blurred Image 2 by Weighted Filter, L = 9');
subplot(2,2,3)
imshow(b_img3, [imin, imax])
title('Blurred Image 3 by Averaging Filter, L = 5');
subplot(2,2,4)
imshow(b_img4, [imin, imax])
title('Blurred Image 4 by Weighted Filter, L = 5');
```

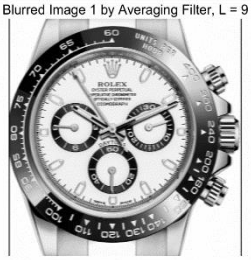
```
figure (3)
subplot(2,3,1)
imshow(b_img1_L_noi, [imin, imax]);
title('Blurred Image 1, Low Noise');
subplot(2,3,2)
imshow(b_img1_M_noi, [imin, imax]);
title('Blurred Image 1, Mid Noise');
subplot(2,3,3)
imshow(b_img1_H_noi, [imin, imax]);
title('Blurred Image 1, High Noise');
subplot(2,3,4)
imshow(b_img2_L_noi, [imin, imax]);
title('Blurred Image 2, Low Noise');
subplot(2,3,5)
imshow(b_img2_M_noi, [imin, imax]);
title('Blurred Image 2, Mid Noise');
subplot(2,3,6)
imshow(b_img2_H_noi, [imin, imax]);
title('Blurred Image 2, High Noise');

figure(4)
subplot(2,3,1)
imshow(d_b_img1_L_noi_lam1,[imin,imax])
title('Img1, Low Noise Deconvolution (2nd, \lambda = 0.10)');
subplot(2,3,2)
imshow(d_b_img1_M_noi_lam1,[imin,imax])
title('Img1, Mid Noise Deconvolution (2nd, \lambda = 0.10)');
subplot(2,3,3)
imshow(d_b_img1_H_noi_lam1,[imin,imax])
title('Img1, High Noise Deconvolution (2nd, \lambda = 0.10)');
subplot(2,3,4)
imshow(d_b_img2_L_noi_lam1,[imin,imax])
title('Img2, Low Noise Deconvolution (2nd, \lambda = 0.10)');
subplot(2,3,5)
imshow(d_b_img2_M_noi_lam1,[imin,imax])
title('Img2, Mid Noise Deconvolution (2nd, \lambda = 0.10)');
subplot(2,3,6)
imshow(d_b_img2_H_noi_lam1,[imin,imax])
title('Img2, High Noise Deconvolution (2nd, \lambda = 0.10)');

figure (5)
subplot(2,3,1)
imshow(d_b_img1_L_noi_lam2,[imin,imax])
title('Img1, Low Noise Deconvolution (2nd, \lambda = 1.00)');
subplot(2,3,2)
imshow(d_b_img1_M_noi_lam2,[imin,imax])
title('Img1, Mid Noise Deconvolution (2nd, \lambda = 1.00)');
subplot(2,3,3)
imshow(d_b_img1_H_noi_lam2,[imin,imax])
```

```
title('Img1, High Noise Deconvolution (2nd, \lambda = 1.00)');
subplot(2,3,4)
imshow(d_b_img2_L_noi_lam2,[imin,imax])
title('Img2, Low Noise Deconvolution (2nd, \lambda = 1.00)');
subplot(2,3,5)
imshow(d_b_img2_M_noi_lam2,[imin,imax])
title('Img2, Mid Noise Deconvolution (2nd, \lambda = 1.00)');
subplot(2,3,6)
imshow(d_b_img2_H_noi_lam2,[imin,imax])
title('Img2, High Noise Deconvolution (2nd, \lambda = 1.00)');

figure (6)
subplot(2,3,1)
imshow(d_b_img1_L_noi_lam3,[imin,imax])
title('Img1, Low Noise Deconvolution (2nd, \lambda = 5.00)');
subplot(2,3,2)
imshow(d_b_img1_M_noi_lam3,[imin,imax])
title('Img1, Mid Noise Deconvolution (2nd, \lambda = 5.00)');
subplot(2,3,3)
imshow(d_b_img1_H_noi_lam3,[imin,imax])
title('Img1, High Noise Deconvolution (2nd, \lambda = 5.00)');
subplot(2,3,4)
imshow(d_b_img2_L_noi_lam3,[imin,imax])
title('Img2, Low Noise Deconvolution (2nd, \lambda = 5.00)');
subplot(2,3,5)
imshow(d_b_img2_M_noi_lam3,[imin,imax])
title('Img2, Mid Noise Deconvolution (2nd, \lambda = 5.00)');
subplot(2,3,6)
imshow(d_b_img2_H_noi_lam3,[imin,imax])
title('Img2, High Noise Deconvolution (2nd, \lambda = 5.00)');
```

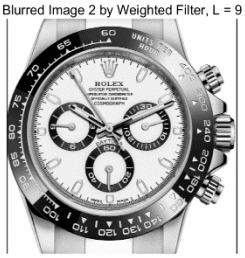
Result(Plots):



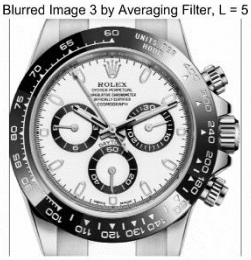
Blurred Image 1 by Averaging Filter, L = 9



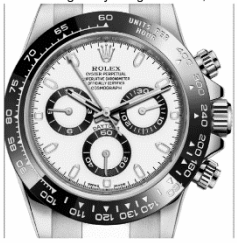
Blurred Image 2 by Weighted Filter, L = 9



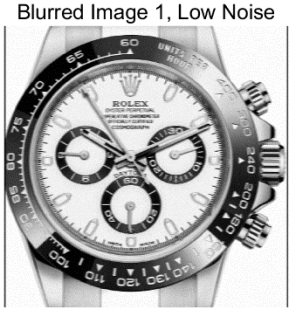
Blurred Image 3 by Averaging Filter, L = 5



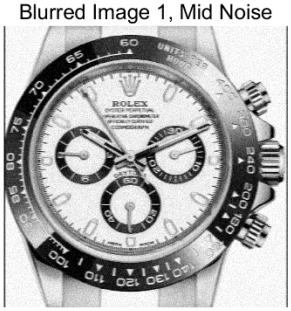
Blurred Image 4 by Weighted Filter, L = 5



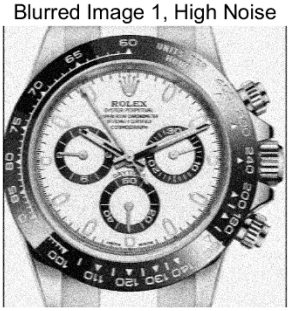
Blurred Image 1, Low Noise



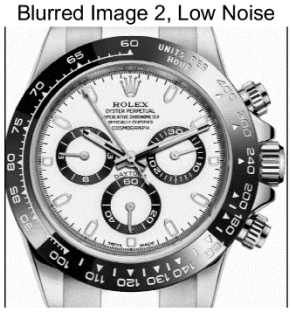
Blurred Image 1, Mid Noise



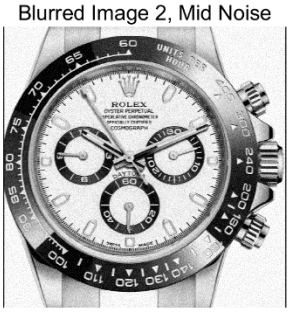
Blurred Image 1, High Noise



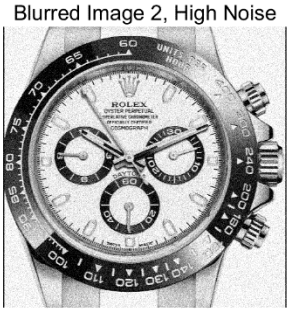
Blurred Image 2, Low Noise



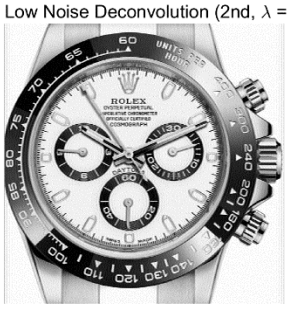
Blurred Image 2, Mid Noise



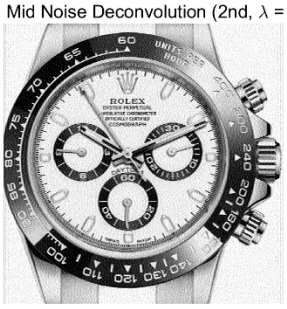
Blurred Image 2, High Noise



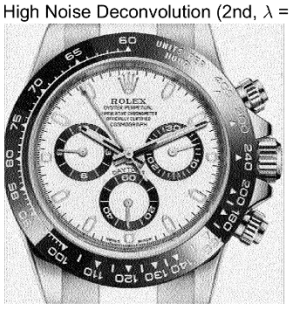
Img1, Low Noise Deconvolution (2nd, λ = 0.10)



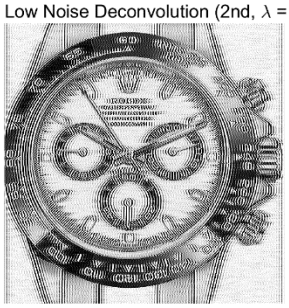
Img1, Mid Noise Deconvolution (2nd, λ = 0.10)



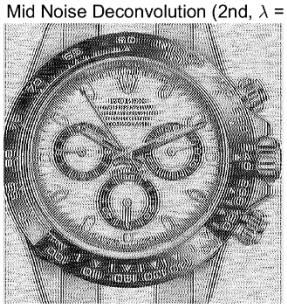
Img1, High Noise Deconvolution (2nd, λ = 0.10)



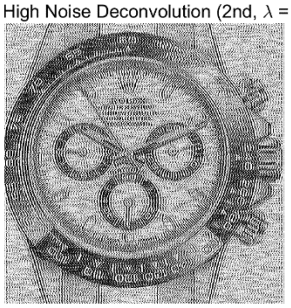
Img2, Low Noise Deconvolution (2nd, λ = 0.10)



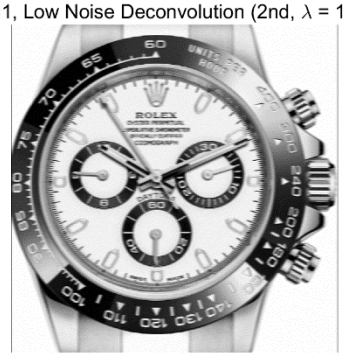
Img2, Mid Noise Deconvolution (2nd, λ = 0.10)



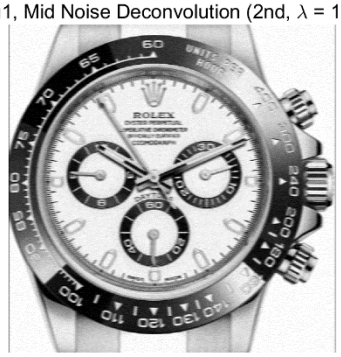
Img2, High Noise Deconvolution (2nd, λ = 0.10)



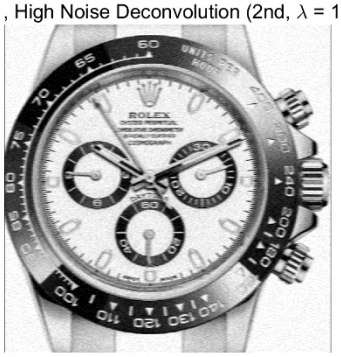
Img1, Low Noise Deconvolution (2nd, λ = 1.00)



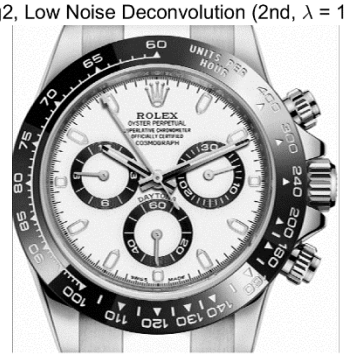
Img1, Mid Noise Deconvolution (2nd, λ = 1.00)



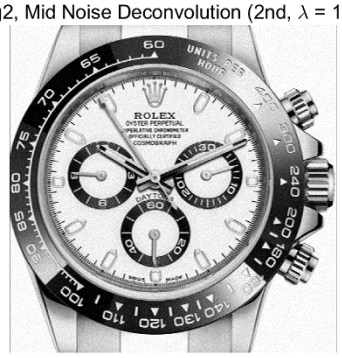
Img1, High Noise Deconvolution (2nd, λ = 1.00)



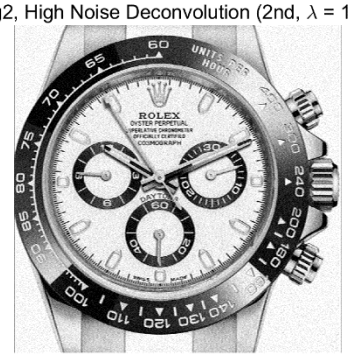
Img2, Low Noise Deconvolution (2nd, λ = 1.00)



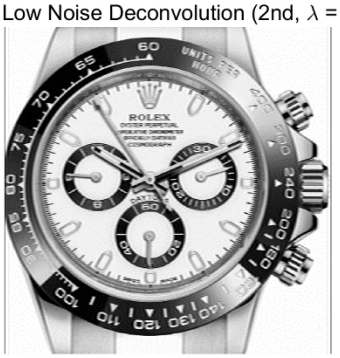
Img2, Mid Noise Deconvolution (2nd, λ = 1.00)



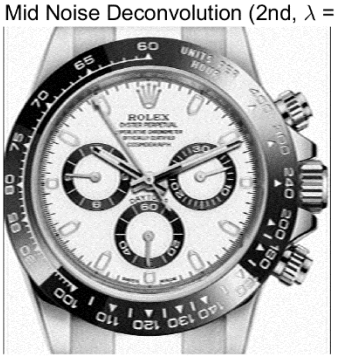
Img2, High Noise Deconvolution (2nd, λ = 1.00)



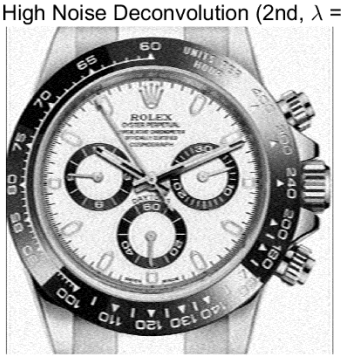
Img1, Low Noise Deconvolution (2nd, λ = 5.00)



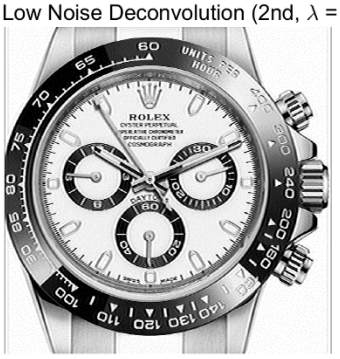
Img1, Mid Noise Deconvolution (2nd, λ = 5.00)



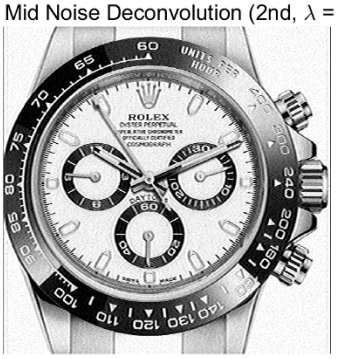
Img1, High Noise Deconvolution (2nd, λ = 5.00)



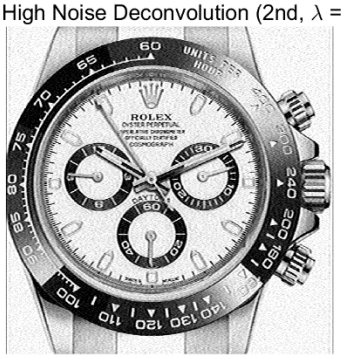
Img2, Low Noise Deconvolution (2nd, λ = 5.00)



Img2, Mid Noise Deconvolution (2nd, λ = 5.00)



Img2, High Noise Deconvolution (2nd, λ = 5.00)



Comments:

I applied simple and weighted averaging blurring filters of length 5 and 9 to the original image (4 filters in total). As we can see from the output images, the longer the filter is, the more is image blurred, simple average filter affects more than the weighted filter.

By setting different coefficients of the noise, low level, middle level and high level of noise were added to the blurred image, as is shown in the output images of image1 and image2. With the level of noise becomes higher, there are more noise points appearing on the images.

After using second order derivative as the regularization term to deblur the images with different lambda

values, we can see the result on the plots. When lambda = 0.1, it cannot deblur the image 2 with noise very well, we can see a very bad reconstruction. As lambda becomes larger, the deblurred images is smoother, as a trade-off, the images lose more details. Even when lambda = 5.0, it cannot remove the high-level noise in both cases.

2) Demonstrate least square smoothing of noisy data. Your program should generate a signal (use your own imagination or take a sample signal from Sample Data folder), add noise to it. Then generate the smoothed (denoised) signal with 1) using a regularization term that minimizes the energy of the denoised signal, 2) using a regularization term that minimizes the second order derivative of the denoised signal. For each method, try different values for the regularization parameter λ. You should plot the denoised signal and the corresponding denoising filter frequency response, for different λ values for each method.

Solution:

.m file(s): jyz_HW13_2.m

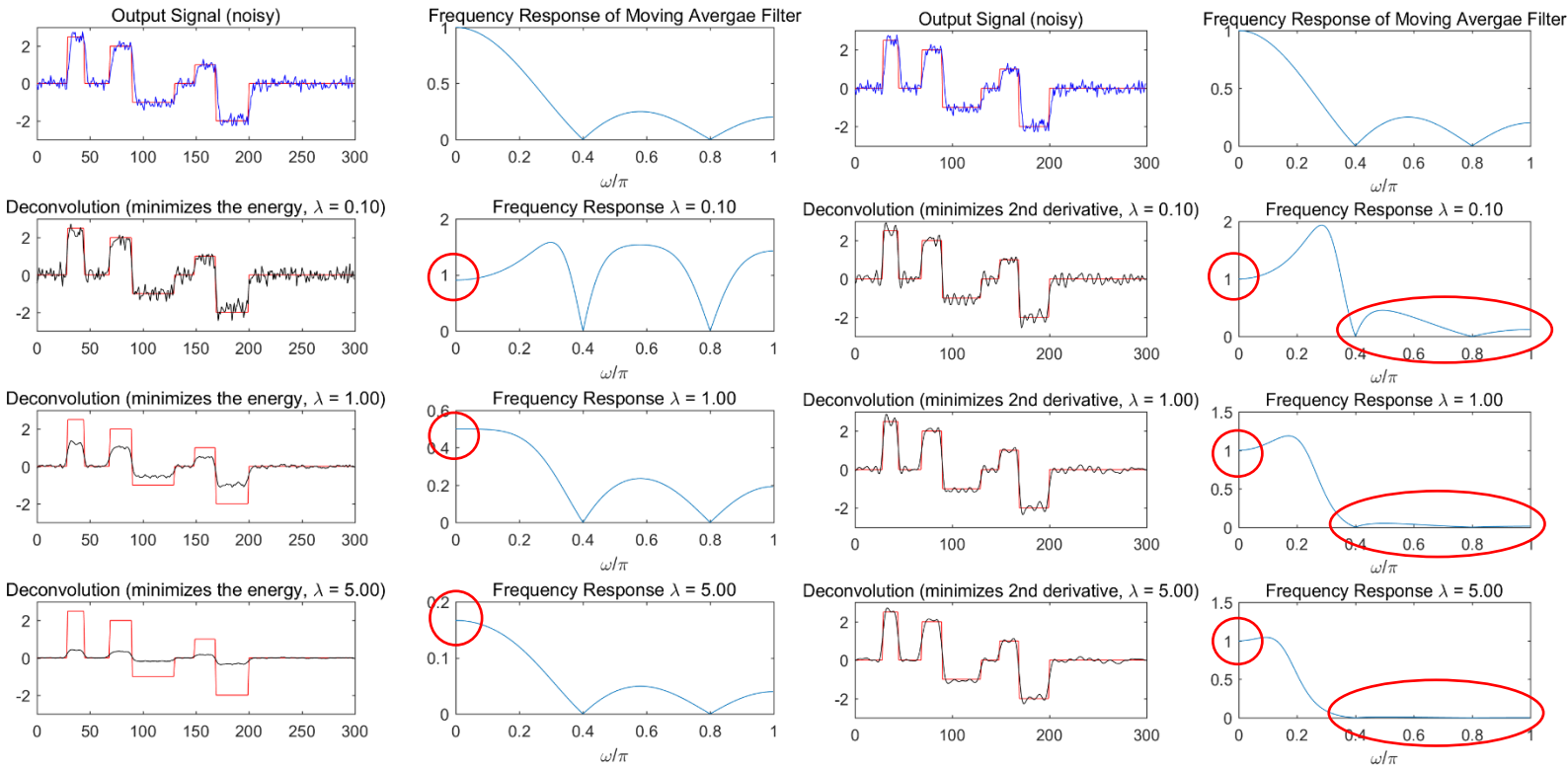
Code:

```
close all
clear

N = 300;
n = (0:N-1)';
n1 = 70;
n2 = 130;
x = zeros(N,1);
x(30:45) = 2.5;
x(70:90) = 2;
x(91:130) = -1;
x(150:170) = 1;
x(170:200) = -2;
h=[1 1 1 1 1]'/5;

% add noise
randn('state', 0);
y = conv(h, x);
Ny = length(y);
yn = y + 0.2 * randn(Ny, 1);
H = convmtx(h, N);

% minimizes the energy
lam1 = 0.1;
lam2 = 1;
lam3 = 5;
g_min_e1 = (H'*H + lam1 * eye(N)) \ (H' * yn);
g_min_e2 = (H'*H + lam2 * eye(N)) \ (H' * yn);
g_min_e3 = (H'*H + lam3 * eye(N)) \ (H' * yn);

% minimizes the second order derivative
```

```
d_second_order = [1 -2 1]';
D = convmtx(d_second_order, N);
g_min_d1 = (H'*H + lam1 * (D'*D)) \ (H' * yn);
g_min_d2 = (H'*H + lam2 * (D'*D)) \ (H' * yn);
g_min_d3 = (H'*H + lam3 * (D'*D)) \ (H' * yn);

hr = h(end : -1 : 1);
hff = conv(h, hr);
num = [hr; zeros(length(h) - 2, 1)];
[HF, ~] = freqz(h, 1);

d0 = [1 0 0 0 0]';
d0r = d0(end : -1 : 1);
d0ff = conv(d0, d0r);
den10 = hff + lam1 * d0ff;
den20 = hff + lam2 * d0ff;
den30 = hff + lam3 * d0ff;
[HF10, ~] = freqz(num, den10);
[HF20, ~] = freqz(num, den20);
[HF30, ~] = freqz(num, den30);

d2 = [1 -2 1 0 0]';
d2r = d2(end : -1 : 1);
d2ff = conv(d2, d2r);
den12 = hff + lam1 * d2ff;
den22 = hff + lam2 * d2ff;
den32 = hff + lam3 * d2ff;
[HF12, ~] = freqz(num, den12);
[HF22, ~] = freqz(num, den22);
[HF32, w] = freqz(num, den32);

figure(1)
subplot(4, 2, 1)
plot(n,x,'r',n,yn(1:N),'b');
ylim([-3 3])
title('Output Signal (noisy)');
subplot(4, 2, 2)
plot(w/pi, abs(HF))
xlabel('\omega/\pi')
title('Frequency Response of Moving Avergae Filter')
subplot(4, 2, 3)
plot(n,x,'r',n,g_min_e1,'k')
ylim([-3 3])
title('Deconvolution (minimizes the energy, \lambda = 0.10)');
subplot(4, 2, 4)
plot(w/pi,abs(HF10))
xlabel('\omega/\pi')
title('Frequency Response \lambda = 0.10')
subplot(4, 2, 5)
```

```
plot(n,x,'r',n,g_min_e2,'k')
ylim([-3 3])
title('Deconvolution (minimizes the energy, \lambda = 1.00)');
subplot(4, 2, 6)
plot(w/pi,abs(HF20))
xlabel('\omega/\pi')
title('Frequency Response \lambda = 1.00')
subplot(4, 2, 7)
plot(n,x,'r',n,g_min_e3,'k')
ylim([-3 3])
title('Deconvolution (minimizes the energy, \lambda = 5.00)');
subplot(4, 2, 8)
plot(w/pi,abs(HF30))
xlabel('\omega/\pi')
title('Frequency Response \lambda = 5.00')

figure(2)
subplot(4, 2, 1)
plot(n,x,'r',n,yn(1:N),'b');
ylim([-3 3])
title('Output Signal (noisy)');
subplot(4, 2, 2)
plot(w/pi, abs(HF))
xlabel('\omega/\pi')
title('Frequency Response of Moving Avergae Filter')
subplot(4, 2, 3)
plot(n,x,'r',n,g_min_d1,'k')
ylim([-3 3])
title('Deconvolution (minimizes 2nd derivative, \lambda = 0.10)');
subplot(4, 2, 4)
plot(w/pi,abs(HF12))
xlabel('\omega/\pi')
title('Frequency Response \lambda = 0.10')
subplot(4, 2, 5)
plot(n,x,'r',n,g_min_d2,'k')
ylim([-3 3])
title('Deconvolution (minimizes 2nd derivative, \lambda = 1.00)');
subplot(4, 2, 6)
plot(w/pi,abs(HF22))
xlabel('\omega/\pi')
title('Frequency Response \lambda = 1.00')
subplot(4, 2, 7)
plot(n,x,'r',n,g_min_d3,'k')
ylim([-3 3])
title('Deconvolution (minimizes 2nd derivative, \lambda = 5.00)');
subplot(4, 2, 8)
plot(w/pi,abs(HF32))
xlabel('\omega/\pi')
title('Frequency Response \lambda = 5.00')
```

Result(Plots):



Comments:

When using a regularization term that minimizes the energy of the denoised signal, it could be concluded that with lambda increases, the result is smoother, that is, noise is reduced more, but the signal amplitude also reduces more, because more emphasis is put on minimizing the signal energy. The corresponding frequency frequencies also show the effect of that. As is marked on the plot, when lambda becomes larger, the dc gain of the system goes smaller, the amplitude thus decreases.

When using a regularization term that minimizes the second order derivative of the denoised signal, we can see that the denoised signal becomes smother as the lambda increases. As is seen on the Frequency Responses Plots, the filter with larger lambda value put more emphasis on suppressing high frequency part of the signal. But the dc gain always remains 1, which ensures that the amplitude does not change rapidly with lambda.