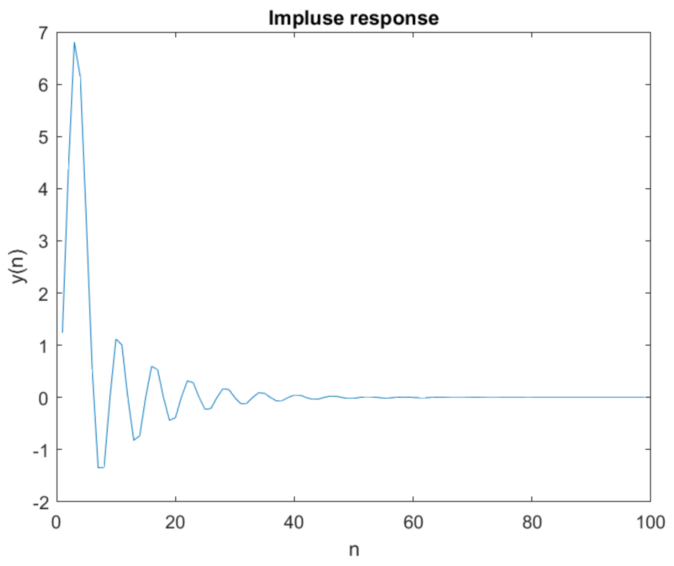
ELEC421 – HW2 - Or Bahari 51277200

1.

For t1, t2 I padded with zeros the rest of the values. For t3 I padded with zeros up to index 103 (this is the length of the result of convolving h1, h2). The transfer function of t4 is:

$$H(z) = \frac{Y(z)}{V(z)} = \frac{1 + z^{-1}}{1 - 0.9z^{-1} + 0.81z^{-2}}$$

The impulse response of y(n) with 0<n<=99 is:



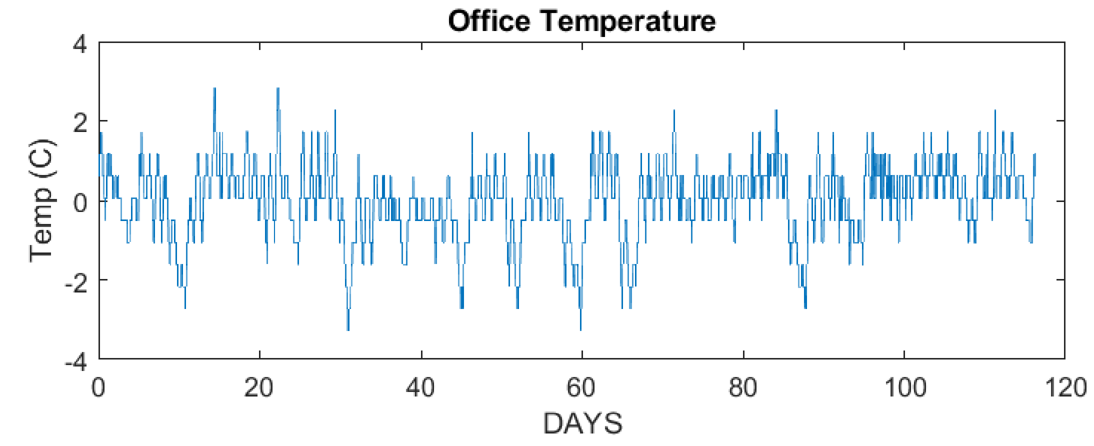
Code:

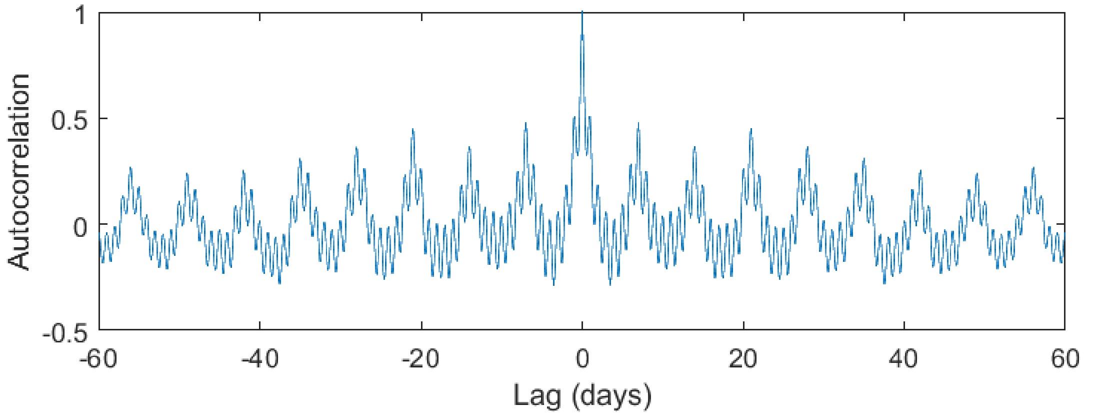
```
%1
v1=zeros([1 99])
v1(1:5)=[1,1/2,1/8,1/16,1/32];
v2 = conv([1,1,1,1,1],v1);
t3=zeros([1 103]);
t3(1:3)=[1/4,1/2,1/4];
vn=t3+v2;
yn=filter([1 1],[1 -0.9 0.81],vn);
yn=yn(1:99);
figure(1);
plot(yn);
xlabel("n");
ylabel("y(n)");
title("Impluse response");
```

2.a.

We measure a room temperature twice an hour for about 120 days. I expect that over time the temperatures will be the same in a period of one week or/and one day (depends if the specific day of the week matters for the office or not..)

2.b. +2.c.





Code:

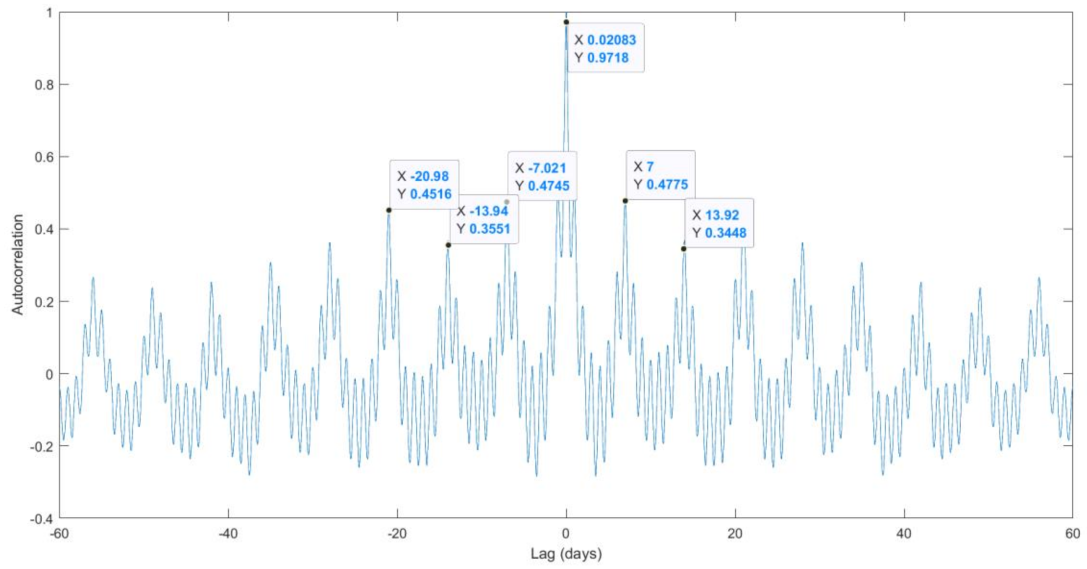
```
%2.b+2.c.
load officetemp;
 % convert from 'F' to 'C'
celsius=5*(temp-32)/9;
c=celsius-mean(celsius);
t=0:1/48:5583/48;
figure(2);
subplot(2,1,1);
plot (t,c);
xlabel('DAYS');
ylabel('Temp (C)');
title("Office Temperature");
subplot(2,1,2);
[autocor,lags] = xcorr(c,60*48,'coeff');
plot(lags/48,autocor)
xlabel('Lag (days)')
ylabel('Autocorrelation')
```

```
Note for TA: I converted to Celsius before subtracting the mean as I think it is better this way.
```

2.d. +2.e.

Periods: 1day,7days.

According to what we studied, the period of the auto-correlation and the signal are the same, so it's easy to find the period in the correlation instead of the temperature graph. You can see that the peak repeats itself about every 7 days I put data marks to show it:



Moreover, there's also a 1-day period derived from the fact that we have 6 small peaks between every big peak. So the big peaks are week cycles and the small peaks are day cycles.

Expectations: yeah the results match.

3.a.

Code:

```
%3.a
h1=[0 0 0;1 2 1;0 0 0];
h2=[0 1 0;0 0 0;0 -1 0];
h3 = conv2(h1,h2);
```

3.b.

Well we got:

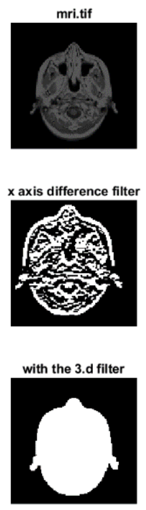
h3 =

```
0   0   0   0   0
0   1   2   1   0
0   0   0   0   0
0  -1  -2  -1   0
0   0   0   0   0
```

→ According to what we learned in class, h3 calculates neighbourhood differences along the ‘y’ axis.

3.c.

We can see that after assigning the filter the shape remains and some differences appear (color differences in the original picture appear on the filtered one). Big differences along the ‘y’ axis appears brighter.



Code:

```
%3.c.
imread('mri.tif');
figure(3);
subplot (3,1,1);
imshow(ans);
title("mri.tif");
subplot(3,1,2);
imshow(conv2(ans,h3));
title("x axis difference filter");
```

3.d.

(Picture is in 3.c.)

We got:

h13 =

```
    0    0    0    0    0

    0    1    2    1    0

    0    2    4    2    0

    0    1    2    1    0

    0    0    0    0    0
```

➔ This filter smooths the image (only the large differences appear).

And, indeed this is what we get in the example above.

Code:

```
%3.d
h12=[0 1 0;0 2 0;0 1 0];
h13 = conv2(h1,h12);
subplot(3,1,3);
imshow(conv2(ans,h13));
title("with the 3.d filter");
```

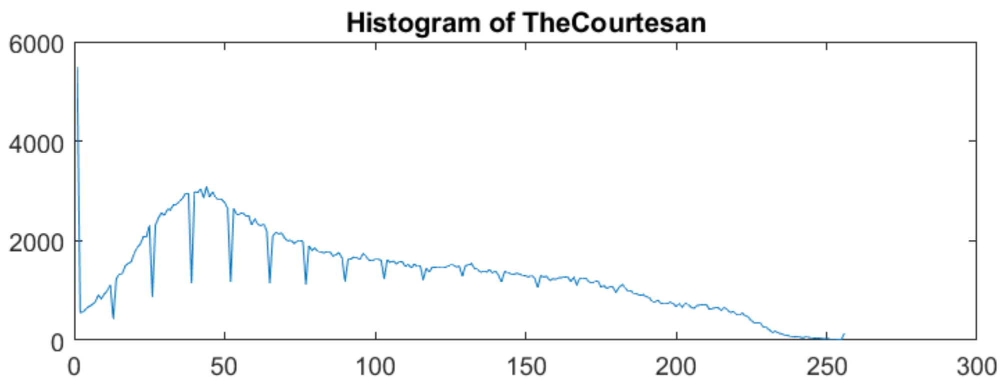
4.a.

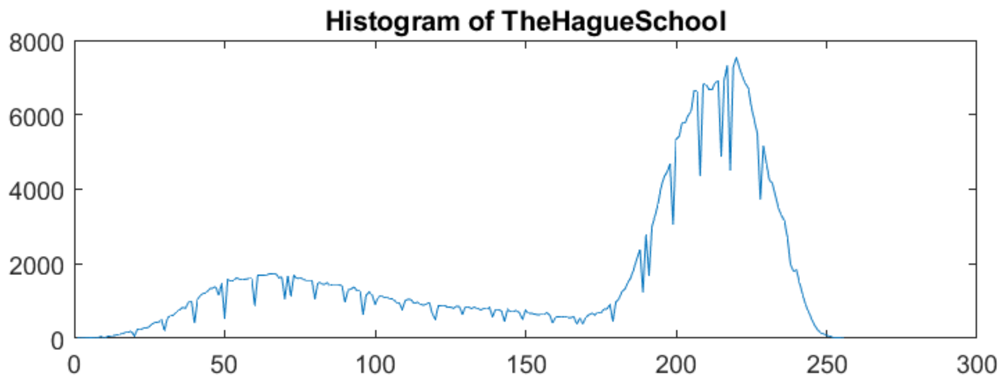
My function (colorhist.m):

```
function hist = colorhist(image,n1,n2,n3)
R=histcounts(image(:,:,1),n1);
G=histcounts(image(:,:,2),n2);
B=histcounts(image(:,:,3),n3);

hist=R+G+B;
end
```

Applying function on both images results in:





Discriminating: The top one (“The Courtesan”) has less intensity in the color (most value between 20-80) and the lower image (”TheHagueSchool”) is more intense (most colors between 200-255).

Code:

```
%4.a.
im1=imread('TheCourtesan.bmp');
im2=imread('TheHagueSchool.bmp');
hist1=colorhist(im1,256,256,256);
hist2=colorhist(im2,256,256,256);
figure(4);
subplot(2,1,1);
plot(hist1);
title("Histogram of TheCourtesan");
subplot(2,1,2);
plot(hist2);
title("Histogram of TheHagueSchool");
```

4.b.

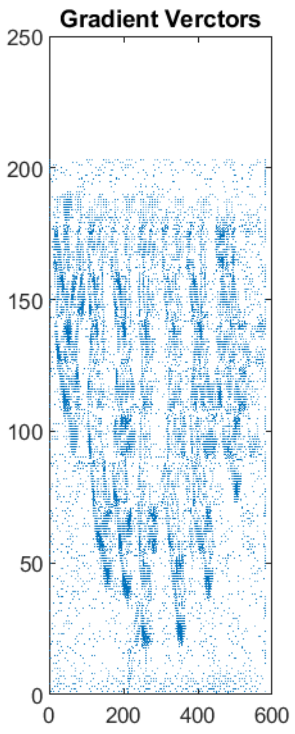
Gradients are on the left.

For the threshold I calculated the average of M and got:

mean(M,'all')

ans =

    5.1042

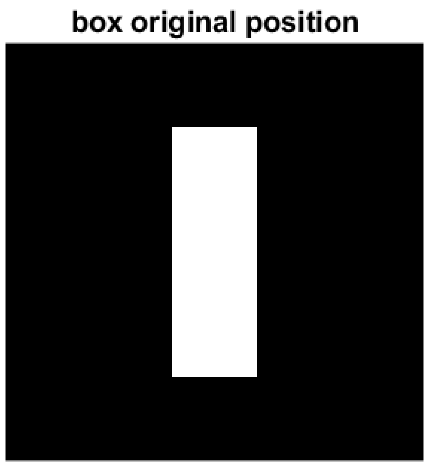


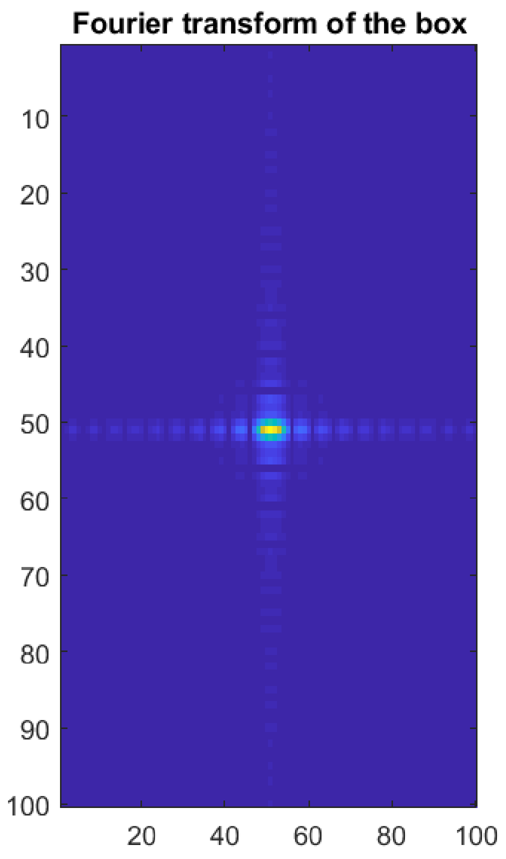
So I picked all the bright (larger than 5.1) values to be very bright (50) and all the dark elements to be completely black (0). The result makes the 'edges' of the image clearer.

Code:

```
%4.b
im1=imread('evolution.jpg');
im2=imgaussfilt(im1,[4 1]);
[dy,dx]=gradient(double(im2));
M=sqrt(dx.^2+dy.^2);
figure(45);
subplot(1,3,1);
imshow(M);
quiver(dy,dx);
title("Gradient Verctors");
subplot(1,3,2);
imshow(M);
title("Original image");
subplot(1,3,3);
M(M>=5.1)=50;
M(M<5.1)=0;
imshow(M);
title("Threshold");
```

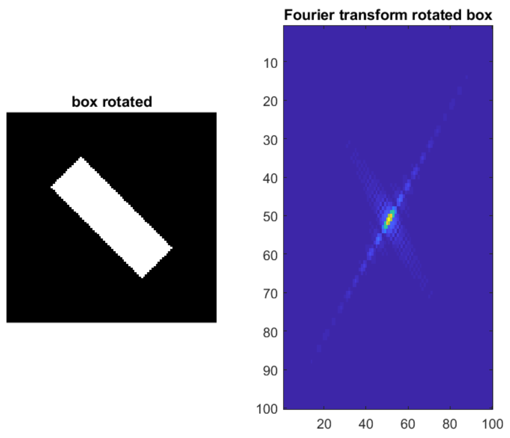
5.a



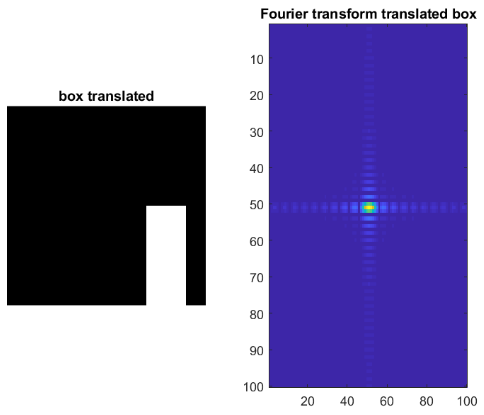


5.b.

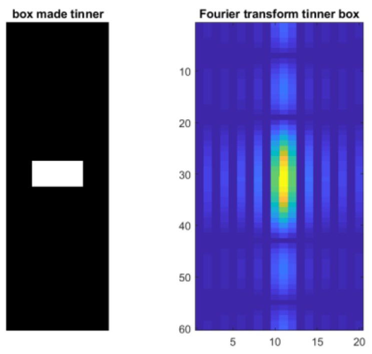
Rotated:



Moved/Translated:



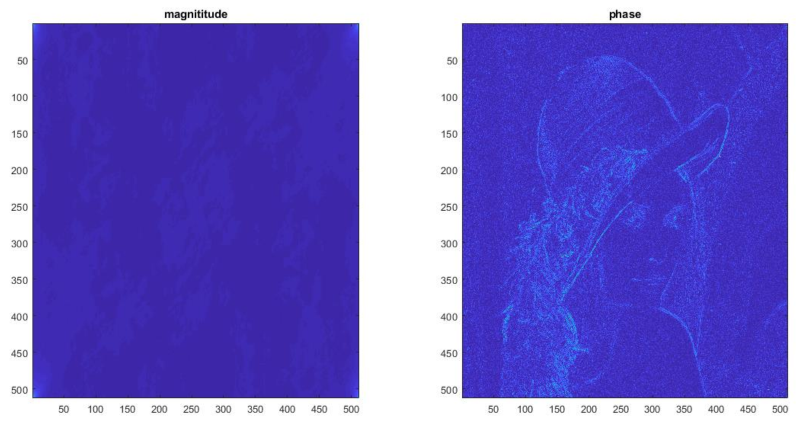
Box made thinner:



Code:

```
%5.a+5.b.
img=zeros([100 100]);
img(21:80,41:60)= (255*ones([60,20]));
figure(51);
subplot (1,2,1);
imshow(img);
title('box original position');
f_img=abs(fft2(img));
f_img=fftshift(f_img);
subplot(1,2,2);
imagesc(f_img);
title("Fourier transform of the box");
figure(52);
subplot(1,2,1);
img_rot=imrotate(img,45,'bilinear','crop');
imshow(img_rot);
title('box rotated');
subplot(1,2,2);
f_img_rot=fftshift(abs(fft2(img_rot)));
imagesc(f_img_rot);
title("Fourier transform rotated box");
figure(53);
subplot(1,2,1);
img_tran = imtranslate(img,[30 30]);
imshow(img_tran);
title('box translated');
subplot(1,2,2);
f_img_tran=fftshift(abs(fft2(img_tran)));
imagesc(f_img_tran);
title("Fourier transform translated box");
figure(54);
subplot(1,2,1);
img2=zeros([60 20]);
img2(28:32,6:15)= (255*ones([5,10]));
imshow(img2);
title('box made tinner');
subplot(1,2,2);
f_img2=fftshift(abs(fft2(img2)));
imagesc(f_img2);
title("Fourier transform tinner box");
```

5.c.



As we learned in class, the ‘phase’ has more details about the original image than the magnitude.

Code:

```
%5.c
lena=imread('lena512.bmp');
lena_fft = fft2(lena);
figure (55);
subplot(1,2,1);
imagesc(ifft2((abs(lena_fft))));
title('magnititude');
subplot(1,2,2);
imagesc(abs(ifft2(exp(1i*angle((lena_fft))))));
title('phase');
```

6.a.

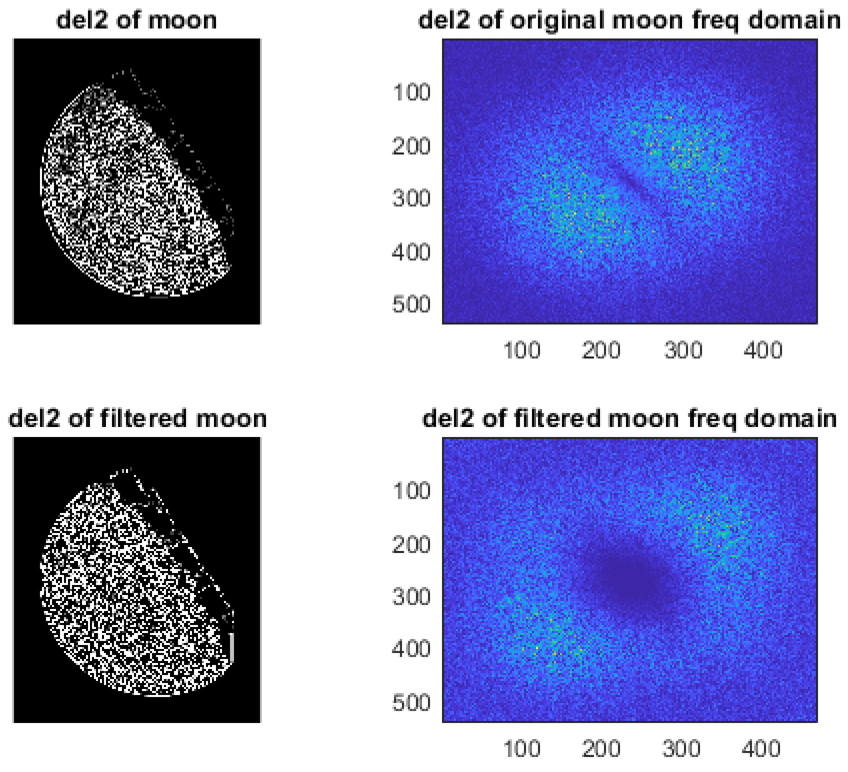
Prove:

$$L[\triangle f(x,y)] = L\left[\frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2}\right] = 2\pi\cdot 2\pi\cdot[(ju)^2F(u,v) + (jv)^2F(u,v)]$$
$$= 4\pi\text{^}2F(u,v)\cdot(-v^2 - u^2)$$

Because: $L\left[\frac{\partial^n f(x)}{\partial x^n}\right] = (ju)^nF(u)$

6.b.

We get:



We can see the frequency domain of the filtered image looks much ‘better’ (i.e. as we learned the higher frequencies are centered.)

Code:

```
%6
figure(6);
moon=imread("blurry_moon.tif");
lap_moon=del2(double(moon));
f_lap_moon=fftshift(abs(fft2(lap_moon)));
subplot(2,2,1);
imshow(lap_moon);
title("del2 of moon");
subplot(2,2,2);
imagesc(f_lap_moon);
title("del2 of original moon freq domain");
h=[0 1 0;1 -4 1; 0 1 0];
fil_moon=conv2(h,moon);
fil_lap_moon=del2(double(fil_moon));
fil_f_lap_moon=fftshift(abs(fft2(fil_lap_moon)));
subplot(2,2,3);
imshow(fil_lap_moon);
title("del2 of filtered moon");
subplot(2,2,4);
imagesc(fil_f_lap_moon);
title("del2 of filtered moon freq domain");
```