Convolution.

1a. The symmetry of Fourier Transform

The Fourier Transform is "almost" its own inverse. Lemma 6.1 in the course book states that if the function f is Fourier Transformed yielding the function F then Fourier Transforming F again does not yield a new function, but a center-mirrored version (rotated with 180 degrees) of f.

The first part of this lab is about reading an image ft and verifying the symmetry property.

```
produce_letters_image; %Read the original image. The routine
%delivers the image ft.
figure(1)           %we decide to work with figure 1 and show it
subplot(3,2,1);     %...we decide to divide it to 6 subimages to see
    % all at the same time. We work with the first quadrant in the
    %reading direction, the current quadrant
imshow(ft) %...we show the original image ft in the current quadrant

FTabs=abs(fft2(ft));
FTabsproc=fftshift(log(FTabs+eps)); %We take the discrete Fourier
%transform and process it for display purposes
subplot(3,2,2);     %...choose the second quadrant to display it....
imshow(FTabsproc, [min(FTabsproc(:)) max(FTabsproc(:))]); %The
%peculiar %syntax, with min and max is to make sure that the minimum
%of FTabs %maps to 0 and maximum to 255. We could have used imagesc
%without this %complicated call, but imagesc is older and has som
%undesired side %effects ...
```

Now take the fourier transform of ft, and reapply it again on the result, i.e. fill the missing statement for FTFT. Display what you have obtained in the third quadrant.

```
FTFT=…; %insert the missing statement BEFORE EXECUTION
```

and then display the result as follows…

```
subplot(3,2,3);imshow(FTFT) % We display it in the current work
%quadrant
%Let us see, the range of the imaginary part of FTFT...
display('Here is the range of the imaginary part of FTFT, ')
[min(min(imag(FTFT))), max(max(imag(FTFT)))]
display('...it shows that all pixels are practically zero, that is
FTFT has real valued pixels!!!')
display('...and fourier transform is "almost" its inverse...except
it is mirrored in the centre')
```

Likewise, fill the missing statement for FTFTFT to the effect that it is 4 times Fourier Transform of the image ft.

```
FTFTFTFT=…; %insert the missing statement BEFORE EXECUTION …i.e.
            %we take four times fourier transform of the
            %image ft...
```

and then display the result (next).

```
subplot(3,2,4);imshow(FTFTFTFT) % We display it in the current work
%quadrant
%Let us see the range of imaginary part of FTFTFTFT...
display('Here is the range of the imaginary part of FTFTFTFT, ')
[min(min(imag(FTFTFTFT))), max(max(imag(FTFTFTFT)))]
display('...it shows that alll pixels are practically zero, that is…
FTFTFTFT has real valued pixels!!!')
display('...and after 4 times Fourier Transform we have obtained…
original ...etc')
display('...This is illustrated also by the second and fourth…
image!')
```

1b. What does a translation in the spatial domain do in the Fourier Transform domain?

The Fourier Transform coefficients of a translated function, $\boldsymbol{f' = f(t - t_0)}$ are given by, see eq. 5.13 in the course book:

$$\boldsymbol{2\pi F'(\omega_m) = \int_{-T/2}^{T/2} f(t - t_0) exp(-it\omega_m) dt = \int_{-T/2}^{T/2} f(t) exp(-it\omega_m) exp(-it_0\omega_m) dt}$$

This means that the Fourier Transform coefficients are just multiplied with a complex function that just depends on $\boldsymbol{t_0\omega_m}$:

$$\boldsymbol{F'(\omega_m) = F(\omega_m) \exp(-it_0\omega_m)}$$

It also means that

$$|F'| = |F|$$

That is the magnitudes of Fourier Transform coefficients are invariant to (cyclical/modulo) translations!

Verify this fact with the image, ft, by first obtaining its translated version ft_p as follows.

```
%%We translate our original image ft with 35 pixels to the
%right...using modulo arithmetic
% Note that to move THE OBJECTS IN THE IMAGE TO THE RIGHT YOU
%NEED TO ADD
% -35 TO THE CORRESPONDING column INDEX
N=size(ft,2); %Image width
ci=0:N-1; %column index in C notation
ft_p=ft(:,mod(ci-35,N)+1); %dont forget +1 for matlab indices
subplot(3,2,5);
imshow(ft_p);
```

Fill now the missing statement to obtain the absolute value of the Fourier transform of ft_p.

```
FT_Pabs=…;   %FILL!!
FT_Pabsproc=fftshift(log(FT_Pabs+eps)); %We
%process the transformed image for display
%purposes
subplot(3,2,6);     %...choose the second quadrant
%to display it....
imshow(FT_Pabsproc, [min(FT_Pabsproc(:)…
max(FT_Pabsproc(:))]); %The peculiar syntax, with
%min and max is to
             % makes sure that the minimum of
%FTabs maps to 0 and maximum to 255
             %We could have used imagesc without
this complicated call, but imagesc is older and
having some bugs...                          %

diff=FT_Pabs-FTabs;
%Let us see the range of the difference between %
% |FT| and |FT_P|...
display(' ')
display('Here is the range of the difference…
between  |FT| and |FT_P|')
[min(min(diff)), max(max(diff))]
display('...it shows that all pixels are…
practically zero, that is |FT| and |FT_P| ...')
```

```
display('...has the same values at all pixels!!!')
```

1c. **What does a scaling of objects in the spatial domain do in the Fourier Transform domain?**
Scaling of an object can be normally achieved by replacing pixel coordinates by $\frac{x}{\alpha}, and \frac{y}{\alpha}$. However, in this exercise we generate the input image from scratch when rescaling.

```
%% section 1.C
%You can repeat to run this section by different circle sizes to
%see that
%if you reduce the radius in one domain the extension of the
%function in the other domain
%increases, and vice versa
figure(2)
cir=circle(10); %generate a white circle on a black background
subplot(3,2,1); imshow(cir); %show the circle and see that its
%centre is at the display centre...


% subplot(3,2,2); imshow(L); %show L now and see that it is a
%circle with centre at (1,1) as fft/ifft expects it...

CIR=fft2(cir);
CIRproc=fftshift(log(abs(CIR))+eps);
subplot(3,2,2); imshow(CIRproc, [min(CIRproc(:))…
max(CIRproc(:))]);

%Let us see the range of CIR...
display('Range of real part of CIR')
[min(min(real(CIR))), max(max(real(CIR)))]
display('Range of imaginary part of CIR)')
 [min(min(imag(CIR))), max(max(imag(CIR)))]
display('Range of absolute value of CIR)')
[min(min(abs(CIR))), max(max(abs(CIR)))]

%Repeat the above but with circle(20)...
```

QUESTIONS: What happens to the Fourier Transform of a function when you expand the radius of the circle with factor 2? Which α would this correspond if the image of the circle was obtained by coordinate substitution i.e. by replacing $x, and\ y$ with $x/\alpha, and\ y/\alpha$?

2. Convolution by DFT, and direct convolution

Convolution in the image domain can be performed by multiplication in the Fourier domain.
That is:

(f * g)(x) = IFT{ FT(f(x)) • FT(g(x)) }

Example a): synthetic images
Generate two images **f =ones(5,5);** and **g=ones(3,3);**.
Compute the convolution y = (f * g) in the spatial domain (**y=conv2(f,g);**).

Compute the convolution in the frequency domain by multiplication, by using DFT and inverse DFT (IDFT).
That is:

y = IDFT{ DFT(f) • DFT(g) }

%do convolution by DFT, multiplication, and IDFT
F=fft2(f,5,5); % DFT in 5 x 5 points (N=5)
G=fft2(g,5,5); % both DFT must be of equal size (later on multiplication!)
Y=F.*G; % point multiplication
yy=real(ifft2(Y)); % convolved image

Print the matrices **y** and **yy** on the screen (and print them on a paper).

Questions:
Explain the differences between the matrices y and yy (hint: periodic images, period=N, when using the DFT).
Generate by hand the matrix yy from y (hint: indices are calculated modulo N).

Choose the correct value on N. Do now the convolution of f and g by using DFT.
Check that y equals yy!

Example b): real images
Load the image 'flowers.jpg', crop it to 256 x 256 pixels and cast it to type double.

f=imread('flowers.jpg');
f=double(f(1:256,1:256));

Generate the (normalized) low pass filter **g=ones(21,21)/(21*21);**.
Compute the convolution y = (f * g) (**y=conv2(f,g);**).
Display the original image f and the convolved image y (**subplot(2,2,1); imshow(f/255);**).

Now compute the convolution in the frequency domain by multiplication. Use the DFT and the inverse DFT (IDFT) as follows:

yy = IDFT{ DFT(f) • DFT(g) }

First, compute yy1 by using 256 x 256 points in the DFT (which is the size of the original image). Display the convolved image yy1 (**subplot; imshow(yy1/255);**).

Second, compute yy2 by using 276 x 276 points in the DFT. Display the convolved image yy2 (**subplot; imshow(yy2/255);**).

Questions:
Compare the two images yy1 and yy2. Explain the differences between them and with respect to image y.
In the second case you used 276 x 276 points when calculating the DFT:s. Why just 276 x 276 points?

*"For fun": try if you manage to reconstruct yy1 from the image yy2 (refer to the simpler case in Example a) above). This gives a new image (let's call it "new"). Display the image "new" and compute the norm of the difference between the images "new" and "yy1" (**e=yy1−new; error = sum(sum(e.*e));**).*