The Discrete Fourier Transform.

The Fourier Transform (FT) of an image f(row,col) is a representation of the image in the frequency domain, here called c(krow,kcol). In the Discrete Fourier Transform (DFT), the frequency representation of an image is estimated at a finite set of frequency values (krow,kcol), so that the DFT of f, called c here, is also an image. If the image f is of size N x N, the indexes, row, col, krow, and kcol are integers between 0 and N−1. The values krow, and kcol represent spatial frequencies of the image (frow=krow/N and fcol=kcol/N), in the range from 0 to N/2-1. *Negative* frequencies exist but are "shifted" in the DFT indices. They are to be found at index values from N/2 to N−1 (see Figure 1), this is, because images are matrices with fixed index ranges. Because of the periodicity of the DFT (period=N) you can however transform any values of krow and kcol to the range [0,N-1], by performing **modulo N calculations**. In MATLAB you use the **mod** command, mod(krow,N), to map krow (or kcol) values in any interval (including [−N/2, N/2-1]) to the interval [0, N−1], Fig. 1. However, Matlab matrix indices start from 1 (not 0) so that a final correction of +1 needs to be done to krow and kcol.

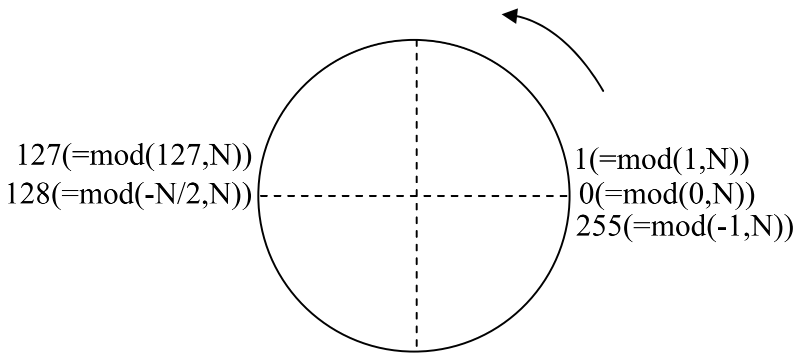

Fig. 1: Accessinng and mapping krow, and kcol, from integers beyond the interval [0,N-1] to [0,N-1], with N=256.

Examples:
krow=mod(**2**,256) +1 % krow takes the value **2** (see Fig. 1) plus **1** (correction for matlab indices).
kcol=mod(**−2**,256) +1 % kcol takes the value **254** (see Fig. 1) plus **1** (correction for matlab indices).

A practical notice:

We used row, col as indices to access images, e.g. f(row,col), because in Matlab images are matrices. The alternative was to use x, and y, which we did not. This is because, for example, x would then risk to increase confusion since it would have to be used for the 2nd index, as in f(y,x).

1. Modulo N calculations.
A modulo operation is an integer operation that consists in computing the remainder of the division by a given integer. For example, mod(7,5) = 2, because 7 divided by 5 is 1 with a remainder of 2. Likewise, mod(10,5) = 0, because 10 divided by 5 is 2 with no remainder.
In general, mod(n,N) with N a positive integer is an integer between 0 and N−1. In the case of negative n we can always require the remainder to be positive, for instance mod(−6,5) = 4 because −6=−2 x 5 + 4 (where 4 is the positive remainder).
Performing an addition modulo N corresponds to a translation along a circle, where the last step takes us back to the starting point, i.e. mod(255+1,256)=0 (see Figure 1 above when N=256).

Use Figure 1 above to find out the index values for krow=266, 522, and -10 by computing krow modulo 256.
Check your result by using the Matlab commands ***mod(266,256)***, ***mod(522,256)***, *and* ***mod(−10,256)*** *respectively.*

The mapping n −> n mod N is a **ring homomorphism** of the integer ring Z onto the ring Z_N={0,1,.., N−1} with respect to addition and multiplication modulo N. Among other things, this means that the following nice properties hold:

For addition:
If mod(n,N)=a and mod(m,N)=b, then mod(n+m,N) = mod(a+b,N).
Example:
mod(8,5)=3 and mod(11,5)=1, then mod(8+11,5) = mod(3+1,5) = 4.
Construct another example with N=7.

Likewise, for multiplication:
If mod(n,N)=a and mod(m,N)=b, then mod(n·m,N) = mod(a·b,N).
Example:
mod(21,19)=2 and mod(103,19)=8 then mod(21·103,19) = mod(2·8,19) = 16.
Construct an example with N=11.

As you see from the examples above, if we are interested in a result modulo N we can **keep our calculations simple** by applying the mod N operation both **before** and **after** the addition or multiplication.

2. The DFT of an image.

Load the image cameraman.tif of size 256 x 256 (use **imread**) and convert its type to **double,** allowing high precision floating point computation. Information on the variables that you defined in MATLAB is available by the command **whos.** Display the image.
(**subplot, imagesc, colormap(gray), axis image**).

Calculate the DFT of the "cameraman image" f (use **c=fft2(f)**) . Compute the inverse (**ifft2, real**) and display it.

Did you get the original image back?
Check it by computing the sum of square errors!

Display the centralized DFT. Use **subplot, imagesc, fftshift, log10, abs** (applying log10, or log will give you a logarithmic scale, which is convenient due to the high dynamic range of the DFT, that is there are very high as well as low magnitudes).
The centralized DFT has index (0,0) in the middle of the image. We call the vertical axis krow (pointing down) and the horizontal axis kcols (pointing right; see Figure 2).
An alternative to "log10 displaying" is to display the absolute value of the DFT raised to a positive number < 1, say 0.2.
Try this also!

Questions:
Why do you have to use the absolute value when displaying the DFT?
Where, in the centralized DFT, do you find the low frequency part and where do you find the high frequency part of the spectrum?
How do you interpret a point c(krow,kcol) in the centralized DFT?

3. Hermitian symmetry.

When working with real images there is a symmetry, called Hermitian symmetry, in the DFT image. This means that c(krow,kcol) = conj(c(−krow,−kcol)).

Verify this symmetry for a few c(krow,kcol) in the DFT.

Use the following MATLAB code (a "+1" appears in the modulo calculation because matrices in MATLAB are indexed starting from (1,1), not from (0,0)).
c(mod(30,256)+1,mod(50,256)+1) % c(30,50) is shown on the screen
c(mod(−30,256)+1,mod(−50,256)+1) % c(−30,−50) is shown on the screen

*Test other values of krow and kcol to verify the Hermitian symmetry. Find out the type of symmetry (even or odd) of the absolute value (**abs**), the angle (**angle**), the real part (**real**) and the imaginary part (**imag**) of each selected DFT point.*

4. Reconstruction of an image from half DFT using Hermitian symmetry.

By using the Hermitian symmetry property of the FT for *real images* you can reconstruct the entire DFT (and therefore, by inverse transform, the image) from only half of the DFT.
When the *real valued image* you start from is 256 x 256, you need only half the values, for example those in the range kcol=−127:128 and krow= 0:128 shown in Figure 2.

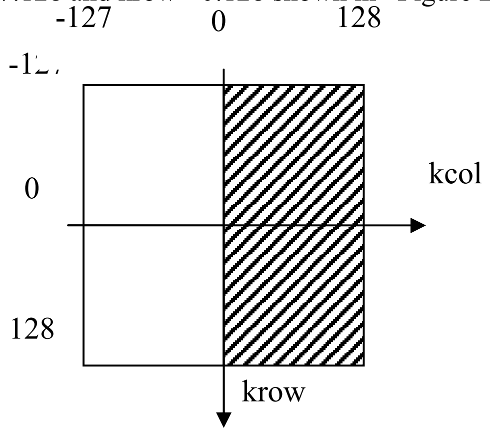

Figure 2: The half of coefficients of a centralized DFT of a real image is determined by the other half

The m-file hermsym.m is an example of how you can implement the reconstruction from the known "half of a DFT".

Study the code!
Note in particular that the MATLAB command, in the code:
c1(mod(−krow,256)+1,mod(−kcol,256)+1)=conj(c(mod(krow,256)+1,mod(kcol,256)+1)),
refills the destroyed half with values from the half that is in tact.
Run the code (hermsym.m)!
Check the outputs: figures, and the error measure.
Is the reconstruction successful?

*Check the size of the real image and the size of the DFT of the real image in bytes (**whos**). Does it make sense that we destroyed "half a DFT" but yet we did not destroy anything at all? Do you think that this would be possible if the image (blood1.tif) was a complex-valued (instead of real-valued)?*

5. Conservation of the scalar product.

The scalar product <f,g> is preserved by the Fourier Transform (except possibly for a normalization factor, depending on the definition of FT/DFT adopted).

That is: <f,g> = <F, G>/(N^2)

Study Eq. 3.27 in the course material, Bigun J. "Vision with Direction", 2006, and notice that in the definition of the scalar product there is a complex conjugation of the first argument. Evidently, complex conjugation has an effect only in case the first argument is complex valued. Also, the images entering a scalar product must have the same size. Here f, g are images of size N x N, F=DFT{f} and G=DFT{g}. Considering the special case when g=f, we see that the energy of the image is preserved by the Fourier Transform.

Verify the statements above for (<f,g> and for <f,f>).

Use the images f (blood1.tif) and g (enamel.tif). Load the images (**imread**), crop them to 256 x 256 (**g(1:256,1:256)**), and convert to **double**.
Compute the DFT of the images (**fft2**). Obtain the scalar products <f,g>, <f,f>, <F,G>, and <F,F>.

Example of MATLAB commands to be used:
fscalg=sum(sum(f.*g)) % <f,g>
cscald=sum(sum(conj(c).*d))/256/256 %<c,d>, c=DFT{f} and d=DFT{g}

6. The values of the DFT as projections on base images.
The DFT value c(krow,kcol) of an image f can be interpreted as the projections of f on the 2−dimensional base function (image) $E_{krow,kcol}$(row,col). Each projection is obtained as the scalar product between the image f and the base image E, i.e. c(krow,kcol) = <$E_{krow,kcol}$,f> .

Verify this by calculating c(5,−10) as the projection of f,
f=imread('cameraman.tif');
onto the base image, <$E_{5,-10,}$, *f*>.
You can benefit from the following code for this purpose:
E=base(256,5,−10); %generate the base image E of size 256x256, krow=5 and kcol=−10
proj=sum(sum(conj(E).*f)) %compute the projection
c=fft2(f); %the DFT of the image f of size 256x256
c(mod(5,256)+1,mod(−10,256)+1) %compare with the DFT value for krow=5 and kcol=−10

Test for other DFT points c(krow,kcol).
Use for example (krow,kcol)=(5,5), and (5,-5).
When testing these values plot the base image $E_{krow,kcol}$ *as well* ***(subplot(1,2,1), imagesc(real(E)), subplot(1,2,2), imagesc(real(E))*** *to see how the real part of the basis image changes with (krow,kcol).*

Explain what a point c(krow,kcol) in the DFT represents. In this, you can use the analogy of the expansion of a 2D vector in its base vectors e_1 *and* e_2*:* $v=c(1)e_1+ c(2)e_2$ *where* $e_1=[1,0]^t$ *and* $e_2=[0,1]^t$ *are the base vectors and* $c(1)=<v,e_1>$, $c(2)=<v,e_2>$ *are the projections. Now create the image*
*Cim=(3+2i)*E1+(2-2i)*E2+(3-2i)*E3 + (2+2i)*E4 ;*
where
E1=base(256,5,5), E2=base(256,10,-10), E3=base(256,-5,-5) and E4=base(256,-10,10).
Notice that Cim is a real image, since its imaginary part is essentially zero, (hint: imag, abs, max, max). Display the realpart of Cim (hint: real, imagesc, truesize). Take the DFT of Cim (or its realpart) in MATLAB and display the magnitude of the spectrum (hint: fft2, abs, fftshift, imagesc, truesize). How many spikes are there? Which (row,column) do you think these spikes are at?