# MRI Practical – MRI/Matlab Focus
# Non-Ionising Functional & Tissue Imaging

Dr. Neal Bangerter, February 2021

**Part 2: Using the 2D Fast Fourier Transform in Matlab**

In this section, we'll explore the 2D Fourier transform and its properties using the discrete 2D fast Fourier transform (2D FFT) available in Matlab.

Let's first try looking at the 2D Fourier transforms of some of the signals from Part 1. Fourier transform the Gaussian signal s1 first (whoops—I just gave away an earlier question… ☺):

```
S1 = fftshift(fft2(s1));
```
← fft2 computes the 2D Fourier Transform. The fftshift command is needed to move the origin of spatial frequency space to the center of the image.

```
imshow(abs(S1),[]);
```
← Remember the Fourier transform is in general complex so we'll often just look at the magnitude (or magnitude & phase)

Refer back to the table of 2D Fourier transforms in the February 4th lecture notes supplement on the 2D Fourier transform. Is your result consistent with what you expect?

(a) Compute the Fourier transforms of s1 (above), s3, and s4 (call them S1, S3, and S4). Using the subplot command generate an array of figures 2 wide and 3 high with each original signal (left column) and the magnitude of its Fourier transform (right column). Explain your results using the table of 2D Fourier transforms.

Now let's get some practice loading, manipulating, and saving a real image. Download the file "head_mri.jpg" from Blackboard and save it in your Matlab working directory. You can load the image into a Matlab array using the "imread" command:

```
f1 = imread('head_mri.jpg');
f1 = double(f1);
```
← Depending on the platform and image file, these sometimes load as uint8 or other types, so we cast to a double to be safe.

```
imshow(f1,[]);
```

Compute the 2D Fourier Transform F2 of the image, and display the logarithm of its absolute value and its phase.

```
F1 = fftshift(fft2(f1));
figure;
subplot(2,1,1);
imshow(log(abs(F1)),[]);
subplot(2,1,2);
imshow(angle(F1),[]);
```
← "angle" in Matlab computes phase of a complex number

Why the logarithm of the absolute value? Unlike some of the 2D signals we examined before, most real images (like MRI images) have the vast majority of the energy concentrated in the very low spatial frequencies (the center of the k-space image). If you don't take the logarithm, you'll often just see a bright spot in the middle of your 2D Fourier transform, and won't see any detail in the higher spatial frequencies. The bright spot in the center corresponds to the "DC component" of the image, which is average intensity across all of

the pixels in the image. (I'm ignoring scaling here, since the fast Fourier transform doesn't scale the result as we expect from our Fourier transform equations.)

(b) Using subplot, create a plot showing (1) the original image f1, (2) the log of the magnitude of F1 (the 2D Fourier transform of f1), and (3) the phase of F1. You can add titles to each graph with the "title" function.

Let's try low-pass filtering the image by multiplying F2 by a rect function. (You'll notice here I use a brute force method for creating a mask that serves as my 2D rect function. I'm trying to introduce you to some various tricks in Matlab…) NOTE: Low-pass filtering a signal is a way of "band limiting" the signal.

```
rect1 = zeros(256);
rect1(112:144,112:144) = 1;
imshow(rect1,[]);      ← Display your low-pass rect function to make sure you've done it correctly

F1_lowpass = F1.*rect1;
imshow(log(abs(F1_lowpass)),[]);   ← Display your k-space data after applying the low-pass filter

f1_lowpass = ifft2(fftshift(F1_lowpass));   ← Do the inverse 2DFT
imshow(abs(f1_lowpass),[]);        ← And display the low-pass filtered image
```

Is the result consistent with what you expect?

(c) Now try high-pass filtering the image. (Hint: try multiplying F2 by 1 - rect1.) Do the inverse Fourier Transform to get your high-pass filtered image. Explain your result. You can save the image using the "imwrite" command.

(d) Now try zeroing out every second line in k-space, going back into the image domain, and seeing what happens. Note that if you have a Fourier transform of the image called `F1_half_kspace`, you could use something like the following to accomplish this:

```
F1_half_kspace(1:2:256,:) = 0;
```

Explain your result in the image domain.

In the following two sections, we're going to demonstrate some of the properties of the Fourier transform using the 2D rect signal s4 that you generated previously.

(e) Translate (or shift) your rect image s4 using the "circshift" command in Matlab. Shift it 40 pixels in the x direction and 20 pixels in the y direction. Now compute the 2D Fourier transform of the **shifted** image, and compare its magnitude and phase to the magnitude and phase of the Fourier transform of the original image. **Explain your results using a property of the Fourier transform. Again, you may need to refer to supplemental lecture material note on the 2D Fourier Transform and its Properties in order to do so. (Hint: This has to do with the shift property of the Fourier transform.)**

(f) Now try rotating your original 2D rect s4 using the "imrotate" command in Matlab. Compare the Fourier transforms of the original image and the rotated image, and explain your results. (NOTE: Keep the rotation angle small, like maybe <20 degrees. Imrotate scales the image, and weird things will happen if you use, for example, 45 degrees—try it if you like.) ☺

(g) **OPTIONAL AND ADVANCED:** Finally, take the Fourier transform of your **shifted** 2D rect and zero out almost half of your data. For example, if the Fourier transform of your shifted rect is S4_shifted, you could type:

```
S4_shifted(1:80,:) = 0;     ← Our signals are 161x161 pixels, so this zeros out almost half the data
```

Inverse Fourier transform the zeroed-out version and look at the effect on the image. Can you devise a scheme (maybe using conjugate symmetry) to restore the zeroed-out values in the Fourier domain? Try it and see if you can get something that looks more like your original shifted rectangle back.

### Part 3: Effect of Point Spread Function (PSF) on Image Quality

In this section, you will explore the degrading effects on an ideal image from imaging systems with various point spread functions (PSFs).

(a) Load the Shepp-Logan phantom (file "shepp256.png" on Blackboard). You will probably want to convert it from a uint8 (which only allows integer values from 0 to 255) to a double prior to manipulating the image, as shown below.

```
f1 = imread('shepp256.png');
f1 = double(f1);
imshow(abs(f1),[]);
```

(b) Load the various PSFs that I have provided on the web site (files "h1.png", "h2.png", and "h3.png"). NOTE: You should cast each of the PSFs to a double as you did with f1 above, since they may load as uint8.

1. **Generate versions of the Shepp-Logan phantom showing how imaging systems with each of these PSFs would affect image quality. Note that this is most easily done by multiplying image matrices pixel-by-pixel in the Fourier domain. Remember: convolving an image in the image domain with the point-spread function (PSF) gives us the result of that PSF on that image. Convolution in the image domain is equivalent to multiplication in the Fourier domain (k-space). So we are effectively simulating what an imaging system does to an ideal image using these PSFs, but doing the math in the Fourier domain.**
2. Generate a version of the Shepp-Logan phantom corresponding to sending the image through a cascade of three systems with PSFs h1, h2, and h3 respectively (from the files).

(c) In MRI, we often experience exponential decay in the signal in the Fourier domain in one direction in k-space. This can be modeled with the "Transfer function" H4 found in the file 'H4.png'. A Transfer Function is the 2D Fourier Transform of the PSF (hence the upper-case H4 instead of a lower-case h4).

1. Load and display the Transfer function H4. Do you see why this would put an exponential decay weighting on the spatial frequency data? (NOTE: You may need to convert H4 to doubles.)
2. Produce the PSF h4 corresponding to the Transfer function H4. Display it. What effect do you expect a system with this PSF to have on image quality?
3. Produce a version of the Shepp-Logan phantom sent through a system with the transfer function H4.