Physics Tutorial 1: Exercises in Image Formation

Originally written by Karla Miller
Last Modified: Mark Chiew, Nadine Graedel, Tom Okell, Oct 2014

In this tutorial, you will be exploring different aspects of image formation using MATLAB with your tutor. Interspersed throughout the tutorial, there will be questions for you consider. Questions without any asterisks are those that should be attempted by everyone, whereas more challenging questions are marked with one (*) or two (**) asterisks, and should be considered optional. Take these opportunities to think about these questions, and then discuss your answers with your tutor.

At the end of the tutorial period, you will receive a "take-home tutor", which is an annotated version of this tutorial guide that will help you complete the tutorial at home if you don't manage to make it all the way through with your tutor, or if you missed the tutorial session.

Attendance will be taken by the tutors, and marks will be given by participation. If you would like additional feedback or clarification on the tutorial material, you are welcome to submit your questions or comments to Weblearn, and a tutor will provide written feedback for you. Those unable to attend the tutorial must submit answers to all unstarred questions to receive credit for the tutorial.

Part 0 – Getting Started

0.1 Starting MATLAB

Download and unzip the file containing the tutorial resources from Weblearn, or if you have access to the FMRIB internal network, copy them into your current directory from here:

```
~mchiew/GradCourse/1_Image_Formation
```

Start MATLAB, and make sure you're inside the tutorial directory (i.e., the folder containing all the tutorial files).

Note to jalapeno00 users: please start with the -nojvm option, "matlab -nojvm"; *this should reduce server CPU load if lots of people are trying to use jalapeno00 simultaneously*

Part 1 – K-Space Sampling

This section uses data contained in the file `brain.mat` which you can load using:

```
>> load brain.mat
```

1.1 Properties of k-space

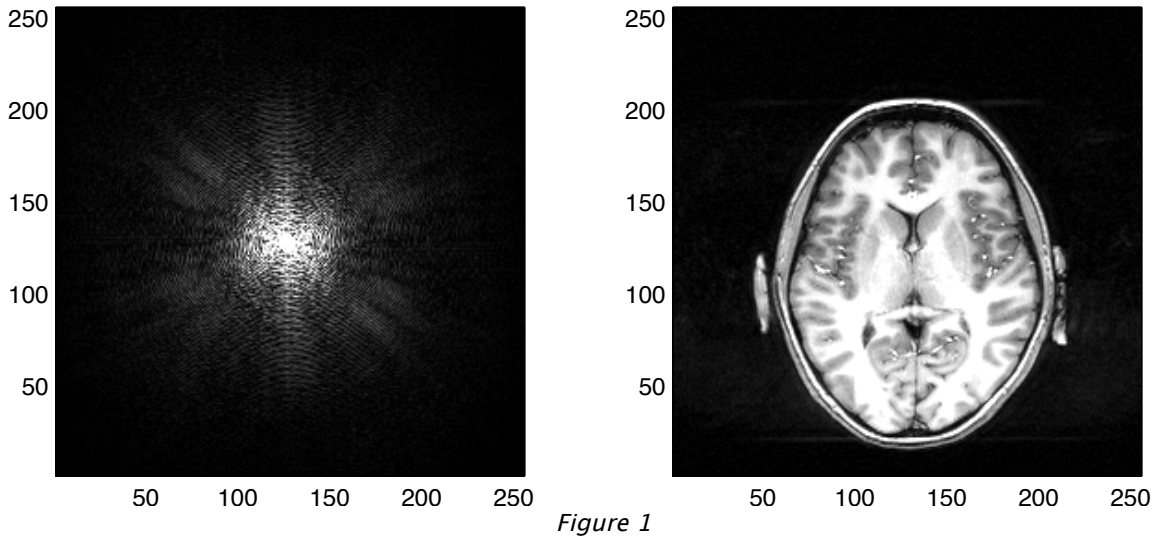

Figure 1

Figure 1 shows an image of a single slice through the brain on the right, and its corresponding k-space magnitude image on the left. The relationship between k-space and image space is governed by Fourier transform properties/relationship.

Some properties of k-space:

- The central point in k-space (**k**=0) is typically the greatest in magnitude
- The k-space of a magnitude image (or any real image) has (conjugate) symmetry
- Shifting an image around has no effect on the magnitude of k-space (and vice versa)
- The size of a k-space matrix is the same as the size of the image matrix

Question 1.1.1*
What do you think happens to k-space when you *rotate* an image, instead of shifting it?

Question 1.1.2**
Describe an image that could have a k-space that does not peak in magnitude at **k**=0.

1.2 Δk and Field-of-View (FOV)

$$FOV \propto \frac{1}{\Delta k}$$

The image field-of-view (FOV), and the spacing between k-space points have an inverse relationship. That is, *reducing* the space between k-space samples *widens* the FOV, and *increasing* the space between samples *shrinks* the FOV.

It is important to note that Δk is a property of the sampling performed by the imaging pulse sequence (as you'll see in Part 2), and not something that is evident from the k-space matrices themselves. That is, two k-space matrices with the same number of rows and columns can have very different Δk, and hence represent images that have very different FOVs.

We can simulate the effect of changing Δk by sub-sampling some k-space data. For example, selecting every other point in k-space effectively doubles Δk, which should halve the effective FOV in the corresponding image.

Question 1.2.1
What happens to the image FOV when every third line is selected? Or every fourth?

Extra Information
In reality the physical object in the scanner obviously does not suffer from the overlap that you see in the reduced FOV images. Clearly then, the FOV does not actually refer to an actual physical interval over which the image is captured, otherwise you would not see the "aliasing", or unwanted overlap. What the FOV actually refers to, and what Δk actually controls, is the distance between successive *copies* of the image. This distance dictates the extent over which points in space are uniquely spatially encoded.

This is why, practically speaking, the phase-encoding FOV must be set at least as large as the object is in that direction, because you cannot exclude portions of the object by modifying sampling alone.

Question 1.2.2*
We typically only observe aliasing to occur along the phase-encoding direction. Given what you know about the difference between how the phase-encoding and frequency-encoding (readout) data are collected, why might this be?

1.3 k_{max} and Resolution

$$\Delta x \propto \frac{1}{k_{max}}$$

If Δk is the distance between samples in k-space, then k_{max} is the maximum extent to which k-space is sampled. What k_{max} represents is the highest spatial frequency measured, which in turn dictates the resolution (or voxel size) in the corresponding image.

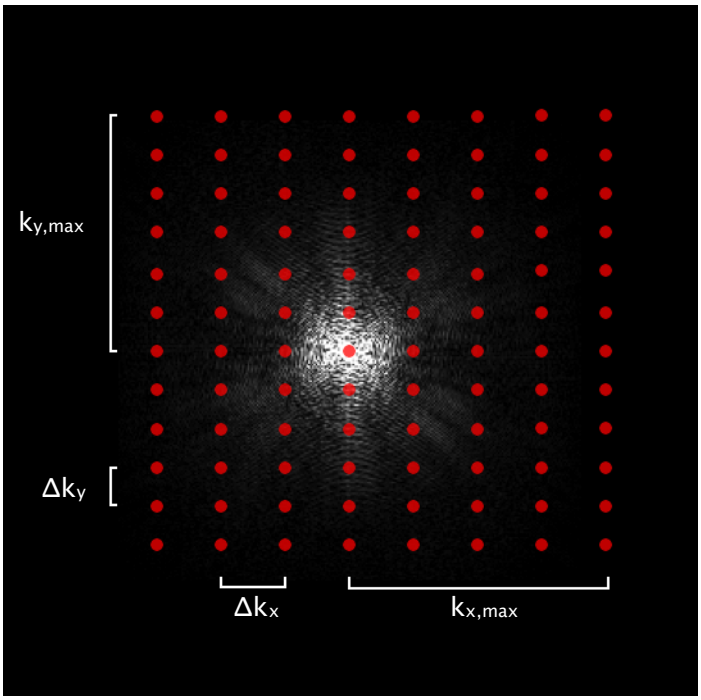

Figure 2

Figure 2 illustrates how the continuous and infinite 2D k-space is discretely sampled to form a 2D image. Different sampling parameters along the x- and y-directions lead to different image properties along these directions.

We can again simulate the effect of sampling to different values of k_{max} by sub-sampling a pre-sampled k-space, and observe how the image changes with the different sampling patterns.

An important thing to remember is that the resolution in an image is related to the amount of spatial frequency content sampled in k-space, and not simply the number of pixels or voxels in the image. For example, we can "zero-pad" our k-space to any desired matrix size. In zero-padding, empty or zero-filled k-space entries are generated to pad out the matrix, without adding any information about the sample. The resulting image is has more pixels, but no additional resolution.

Question 1.3.1
Consider a zero-padded image. What features do you notice appearing near the sharp edges and boundaries? What are these features called, and what causes them?

Extra Information
Another feature of sampling that affects resolution is the total width of the k-space sampling window. Two sampling patterns with identical k_{max}, but different sampling windows can have different effective image resolutions. Larger sampling windows lead to higher image resolutions. Consider the extreme example of 2 k-spaces, one with *N* samples extending to k_{max}, and the second with a single sample at k_{max} itself. While both sampling patterns have the same k_{max}, the second has an infinitesimally small window (a single point), and conveys no spatial information (along that direction).

Question 1.3.2
Since k-space sampling takes time, we would often like to reduce the amount of sampling required to form an image to reduce acquisition times. What property of k-space can be exploited to enable reduced k-space coverage without a corresponding loss of image resolution? What is this technique called?

Part 2 – K-Space Trajectories

This section uses data contained in the file `epi_traj.mat`

2.1 Gradient Waveforms

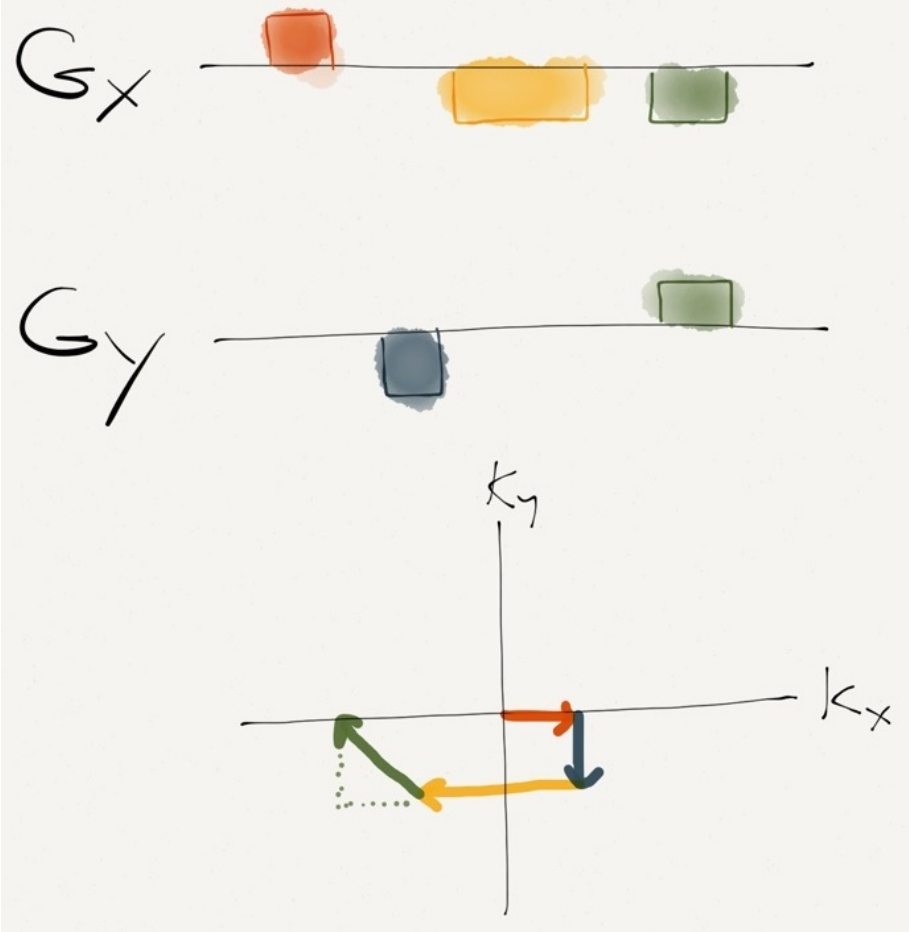

Figure 3

Sampling in k-space is achieved through the use of magnetic field gradients. Mathematically, the position you occupy in k-space is equal to the net area under the magnetic field gradient waveforms. This rule dictates the position along each direction independently, i.e., the x-gradient controls k_x position, the y-gradient controls k_y. Note that these waveforms are not what the magnetic fields look like, rather they are the time-varying amplitudes of the spatially linear gradient magnetic fields.

Question 2.1.1
Consider the G_y gradient waveform provided (see Fig. 4), and divide its amplitude by 2. What would result from a scan that used this new Gy gradient, in terms of k-space and the resulting image FOV and resolution?

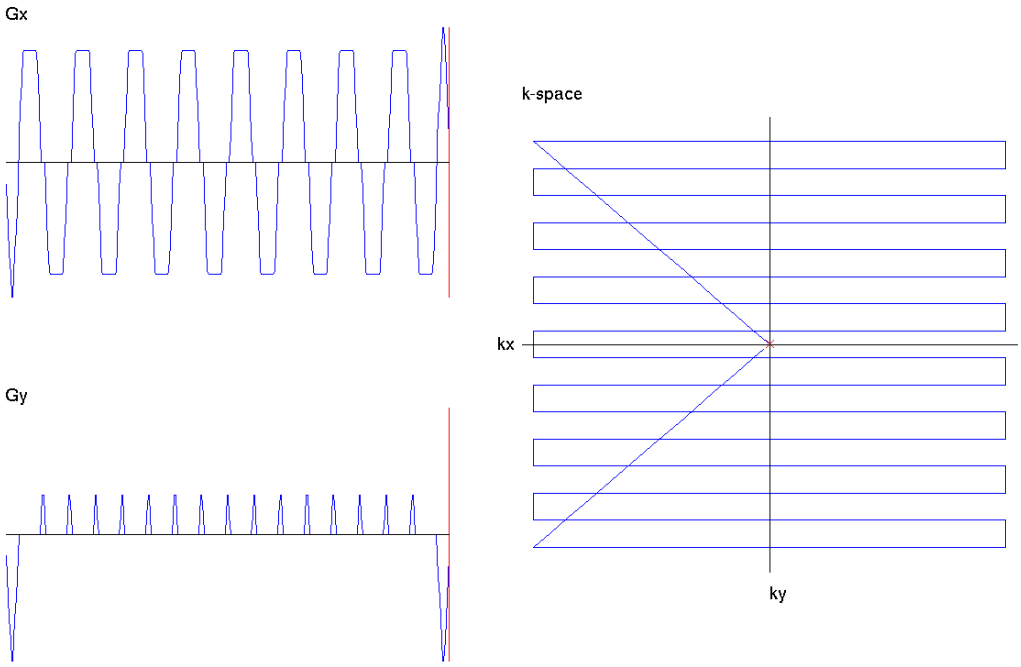

Figure 4

Nothing in the k-space formalism requires you to travel in straight lines across k-space. In fact, although the geometry of the k-space sampling patterns theoretically completely dictates image properties, in practice, the way you go from sample-to-sample (i.e. the trajectory) in k-space also contributes to different image features or artefacts. Spiral trajectories, for instance, cover 2D k-space in a completely different way than standard (Cartesian) trajectories do.

Extra Information
This relationship between gradient waveform amplitude and k position is not a neat trick or coincidence, but is actually linked to how k space is defined. The time average gradient waveform dictates the spatial frequency of a sinusoidal signal variation, and moving around in k space amounts to imposing various different sinusoidal patterns on the object before sampling. Remember that data is acquired as a sum of the signals across the entire object (or slice) to produce a single number. If we did not impose different gradients prior to sampling, that number would never change, and we would not be able to spatially resolve objects at all. Technically, the signal variations we impose on our object need not be sinusoidal (although they must satisfy other mathematical properties), but fortuitously, we find that the sinusoidal variations produced by linear gradients correspond exactly to Fourier basis functions, which enables us to use the Fourier transform to reconstruct our images easily.

Question 2.1.2**
Compare MRI acquisition with x-ray and CT (computed tomography) imaging. Why are you unable to get 3D data from a simple x-ray? What feature of CT imaging allows 3D information to be recovered? How might you replicate an x-ray projection type image using MRI? *NB. If you are unfamiliar with CT, it is essentially a scanner that obtains multiple x-ray images at different angles around the subject.*

Appendix 1 – MATLAB Primer

A.1 Some Useful MATLAB Commands

`ifft2c`	This function calculates the inverse fast Fourier transform (FFT) of a 2D matrix (e.g. a 2D image). For example, the following calculates the inverse FFT of a 2D matrix k and puts the result in the 2D matrix `im`: `>> im = ifft2c(k);`
`show_pair`	Displays a pair of matrices side-by-side. For example, the following will show `im1` on the left and `im2` on the right: `>> show_pair(im1,im2);`
`zeros`	Creates a matrix of a specified size containing all zeros. For example, to create a matrix of zeros, **A**, with 64 rows and 32 columns: `>> A = zeros(64,32);`
`show_traj`	Given gradient waveforms, displays both the waveforms and the corresponding k-space trajectory. For example, to display the trajectory traversed by waveforms Gx and Gy: `>> show_traj(Gx,Gy);`

Question A.1.1*
Given that `ifft2c` is the *inverse* fast Fourier transform, what do you think the forward fast Fourier transform (`fft2c`) does?

Question A.1.2*
The "c" in `(i)fft2c` stands for "centric". What is the difference between the centric transforms, and the built-in default `(i)fft2` MATLAB functions?

0.2 MATLAB matrix indexing

Accessing a single element	The element of a matrix M, in the i^{th} row and j^{th} column is M(i,j)
Accessing a row or column	To refer to the (entire) i^{th} row or j^{th} column, MATLAB uses the shorthand: M(i,:) or M(:,j). The colon (:) represents "all elements along this dimension"
Accessing intervals	You can create an index for matrices with the notation `min:step:max`, which creates an array of numbers between min and max in increments of step. For example, to access the odd columns between 30 and 40 in a matrix M, you would use M(:,31:2:39).

Question A.2.1**
If `M` is a square matrix, how would you get all the entries along the diagonal in a single command?