Spatial Localization, Linear Gradients, and Fourier Encoding: Solving an Inverse Problem

Magnetic Resonance in Medicine, Fall 2014
Johns Hopkins Department of Biomedical Engineering

The following exercise is designed to clarify the relationship between the image recovery problem in MRI, the use of linear gradients and Fourier encoding. As you work on this document, follow along with the three separate scripts designed to help elucidate the problem and the proposed solution.

The Imaging Problem

We have a set of pixels in the x and y imaging directions. We have just performed perfect slice selection using a long sinc-shaped RF pulse envelope, making our slice profile ideally rectangular. Also, all our pixels are perfectly on resonance, i.e. they all resonate at ω_0 and are therefore stationary in the rotating frame of reference. We need to figure out the intensity of each pixel at the various locations on the x-y plane. And since we are doing MRI, we are going to use our linear gradients to do so.

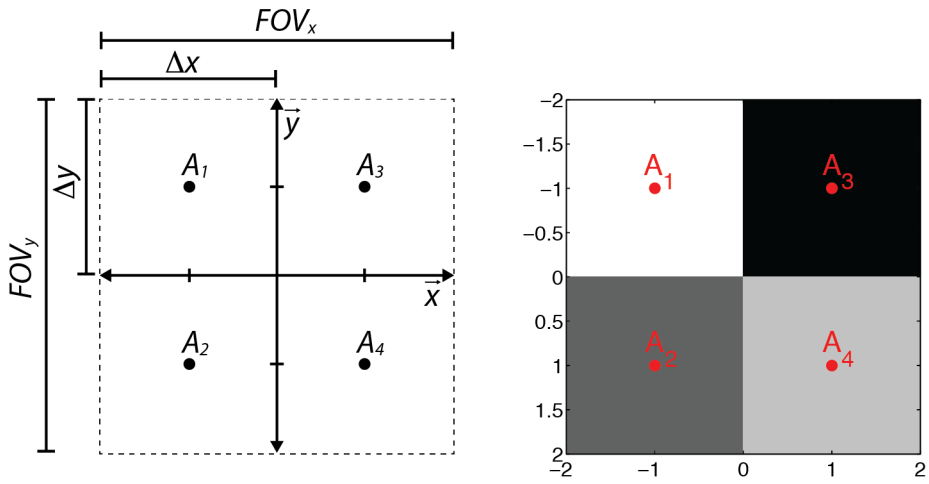
By the end of this demonstration / exercise you should be able to:

1) understand how frequency encoding is used to spatially localize
2) understand how phase encoding is used to spatially localize
3) understand why we use Fourier encoding in MRI
4) understand how the imaging prescription parameters affect how we encode

Assumptions

- ignore relaxation (no T_1, T_2, T_2*, etc)
- encoding as shown here takes place after ideal slice selection using a 90 degree pulse along the x' direction
- all gradient waveforms are ideal (infinite slew rate), so they can start and finish instantaneously (rectangular in shape). This means we can estimate the time-integrals of the gradients using rectangles.
- the B_0 field is perfectly homogeneous so all spins resonate at ω_0 and demodulation leads to perfect removal of the carrier frequency.

The image



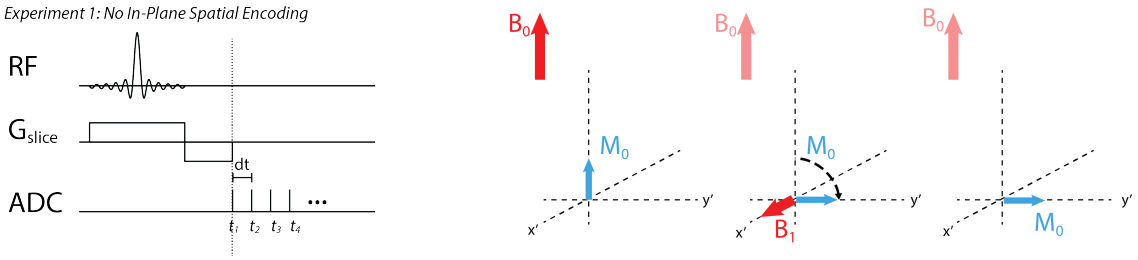
The image we are trying to recover is composed of an array of pixels $N_{pixels} = N_x * N_y$ which we will represent as a series of delta functions, each with amplitude A_p. In this example, N_{pixels}=4=2x2. The image spans the field of view (FOV) and the pixels evenly divide the FOV in each dimension. The resolution of the image, as measured in mm, is the width of a pixel in x or in y. The image and FOV do not need to be square ($FOV_x \neq FOV_y$) and the resolution does not need to be isotropic ($\Delta x \neq \Delta y$). We can also visualize the image directly with intensity reflected by grayscale. Note that for the purposes of this exercise, the amplitude coefficients are numbered in a way compatible with MATLAB (row priority). In this example, FOV = 40 mm, $\Delta x = \Delta y = 20\ \ mm$.

In the accompanying MATLAB script, ***ImagingExperiment.m***, you have the choice of **A**. An **A**Ny_Nx represents a matrix with $N_x * N_y$ pixels. If the variable name terminates in an "i", the coefficients in the generated **A** are complex, possessing both real and imaginary components.

Experiment #1: No spatial localization

If we were to apply no spatial localization and perform the following pulse sequence, what would we observe? Note that we are neglecting all relaxation. We begin by noting that after the RF pulse, at time t_1, all magnetization vectors (one per pixel) are on the transverse plane, along the y' axis in the rotation frame of reference. In the figure, the dotted vertical line represents the start of data collection as part of our imaging experiment (for the purposes of this exercise).

To calculate the detected magnetization (we only have one coil for now), we use the following general equation, which is valid at any point in time:



$$\vec{M}_{total}(t) = \sum_{p=1}^{N_{pixels}} \overrightarrow{M_p}(t) = \sum_{p=1}^{N_{pixels}} A_p \, e^{-i\phi_p(t)}$$

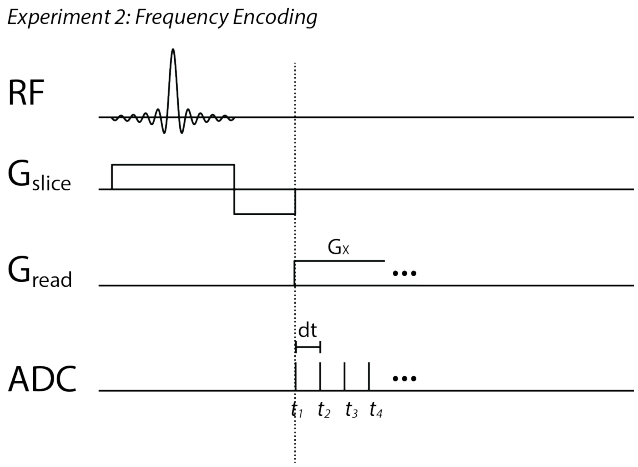
and we define each magnetization vector as $\overrightarrow{M_p} = A_p e^{i\phi_p}$. Since at the beginning of the experiment no phase has been accrued, we assume that the spins have zero phase relative to the arbitrary y' axis ($\phi_p = 0$), as demonstrated in the last panel of the drawing. Therefore, at t_1 the magnetization measured can be expressed as:

$$\overrightarrow{M_{total}}(t_1) = \sum_{p=1}^{N_{pixels}} A_p$$

which is just the sum of the amplitudes of the coefficients. What happens to the measured magnetization as time progresses? **Question 1: What is $\overrightarrow{M_{total}}(t_n)$ for n ϵ {2,3,4}? Can we calculate the A_p? Why?**

Experiment #2: Frequency Encoding

We saw from the previous exercise, that given the information, we cannot individually determine the amplitude coefficients. In fact, at best we can determine their combined amplitude. Now we apply frequency encoding using the following pulse sequence.



At t_1 we apply an ideal readout gradient *Gx* for frequency encoding. We will now attempt to determine the coefficients (A_p) from this single experiment. For now, we assume we have to determine $N_{pixels}=4$ unknowns, or equivalently, $N_{pixels}=4$ real-valued coefficients. We begin by measuring the magnetization at time t_1. We know from Experiment #1 that this value is given by:

$$\overrightarrow{M_{total}}(t_1) = \sum_{p=1}^{N_{pixels}} A_p$$

since there has been no phase accrual at this time point. To estimate the net magnetization at t_2, we need to know how much each magnetization vector has precessed due to the presence of $G_x \equiv \frac{\delta B_Z}{\delta x}$, the gradient in B_z along the *x*-axis. Hence we first calculate the rate of precession:

$$\Delta\omega = \gamma \Delta B(t) = \gamma(x \cdot G_x(t))$$

$\Delta\omega$ tells us how much phase is accrued per unit time: $\Delta\omega = \frac{\Delta\phi}{\Delta t}$. More specifically, we know that in our pulse sequence we sample the signal at intervals of *dt* (typically defined in usec), which implies that we can calculate the phase accrued per time step (as a function of x-coordinate):

$$\Delta\phi(t) = -\Delta\omega \cdot \Delta t$$

This last equation demonstrates that phase is accrued linearly under a constant gradient, with positive phase being defined (from the right hard rule) as counterclockwise rotation. We can now formulate an equation for the net magnetization as a function of sampling time

$$\vec{M}_{total}(t) = \sum_{p=1}^{N_{pixels}} A_p \, e^{-i\phi_g(t)}$$

$$= \sum_{p=1}^{N_{pixels}} A_p \, e^{-i\cdot\Delta\omega_p\cdot\Delta t}$$

$$\vec{M}_{total}(t) = \sum_{p=1}^{N_{pixels}} A_p \, e^{-i\cdot\gamma\left(x_p\cdot G_x(t)\right)\cdot\Delta t}$$

where x_p represents the x coordinate of the pth pixel, and ϕ_g represents the phase accrued due to the gradient. For this example, we can simplify this expression since two pixels share each x-coordinate.

$$\vec{M}_{total}(t_m)$$

$$= \sum_{p=1,2} A_p \, e^{-i\cdot\gamma(x_1\cdot G_x(t))\cdot(dt\cdot(m-1))} + \sum_{p=3,4} A_p \, e^{-i\cdot\gamma(x_3\cdot G_x(t))\cdot(dt\cdot(m-1))}$$

$$\vec{M}_{total}(t_m)$$

$$= (A_1 + A_2)e^{-i\cdot\gamma(x_1\cdot G_x)\cdot(dt\cdot(m-1))} + (A_3 + A_4)e^{-i\cdot\gamma(x_3\cdot G_x)\cdot(dt\cdot(m-1))}$$

The above equation is valid for any sampling time point, if we sample multiple times, e.g. N_{dt}=5, we can construct a system of equations of the form

$$\vec{M} = \boldsymbol{E_{FE}}\vec{A}$$

where $\vec{M}$ and $\vec{A}$ represent vectors holding the measurements of net magnetization we have , and the desired coefficients. $\boldsymbol{E}$ is the encoding matrix, which we will have to invert in order to generate $\vec{A}$ from our complete set of m measurements $\vec{M}$. The encoding matrix will hold all of the exponential phase factors that we have imparted with the use of the gradients. For this experiment, these factors are given by:

$$E_{m,p} = e^{-i\cdot\gamma(x_p\cdot G_x)\cdot dt(m-1)}$$

where p represents the pth magnetization vector ($A_1 \ldots A_p$) and m represents the m^{th} point sampled during frequency encoding (a.k.a. the readout). In other words, for every pixel, we accrue phase based on the ΔB at that position (given by $x_p \cdot G_x$) multiplied by the amount of time for accrual. Note that $\boldsymbol{E_{FE}}$ does not have to be square (and it won't be for these initial experiments).

We populate the equations

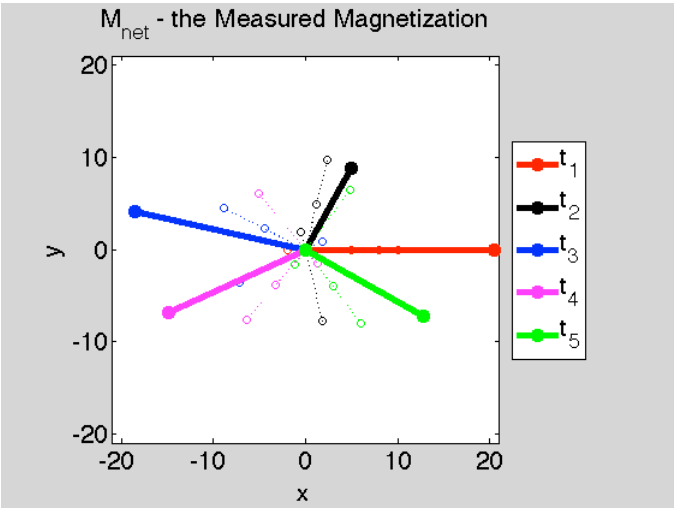
$$\begin{bmatrix} M_1 \\ M_2 \\ M_3 \\ M_4 \\ M_5 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ e^{-i\cdot\gamma(x_1\cdot G_x)\cdot(dt)} & \blacksquare & \blacksquare & e^{-i\cdot\gamma(x_4\cdot G_x)\cdot(dt)} \\ \blacksquare & \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare & \blacksquare \\ e^{-i\cdot\gamma(x_1\cdot G_x)\cdot(dt\cdot 4)} & \blacksquare & \blacksquare & e^{-i\cdot\gamma(x_4\cdot G_x)\cdot(dt\cdot 4)} \end{bmatrix} \begin{bmatrix} A_1 \\ A_2 \\ A_3 \\ A_4 \end{bmatrix}$$

where M_m correponds to the measurement of the net magnetization at time t_m. The ■ represent other coefficients omitted for clarity. The matrix representation should make it easier to solve for the system, since we can now recover $\vec{A}$ (the image) by inverting $\boldsymbol{E}$ and multiplying it against our measurements:

$$\vec{A} = \boldsymbol{E_{FE}^{+}}\vec{M}$$

$\boldsymbol{E_{FE}^{+}}$ is the Moore-Penrose pseudoinverse given by$(E^H E)^{-1}E^H$ (*pinv()* in MATLAB) that is used for inversion when a matrix isn't square. It may not be hard to guess that for this experiment, the resulting inversion and multiplication does not yield the original $\vec{A}$.

After executing the MATLAB code provided for the A2_2 matrix, we see the following figure. In this figure, bold (thicker) vectors represent net magnetization measured at each time point. The dotted thinner vectors represent the individual magnetization vectors precessing. As expected, at the first time point, the magnetization aligned with the *y'* axis. As time progresses the two groups (at the –x and +x coordinates) separate as they precess in opposite directions.



However, because the vectors at a given x-coordinate precess together they cannot be distinguished! From a more mathematical perspective, the matrix $\boldsymbol{E_{FE}}$ ill-conditioned for inversion.

If we look at the parameters that characterize $\boldsymbol{E_{FE}}$, we see that 1) it is has a low rank = 2 (implying that there are only two linearly

independent rows or columns) 2) it has a very high condition number (the ratio of the largest to the smallest eigenvector) which implies ill-conditioning for inversion. As a rule of thumb the condition number should be <20-30 for a good change of successful matrix inversion. These two facts can be clarified by the row-reduced version of the matrix:

$$rref(\boldsymbol{E_{fe}}) = \begin{matrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{matrix}$$

where *rref()* is the MATLAB function that returns the row-reduced echelon form of $\boldsymbol{E_{FE}}$. Note that last 3 rows of the matrix are completely empty, implying that the row vectors of $\boldsymbol{E}_{FE}$ are not linearly independent. How can this be? We made 5 different measurements but it seems that only two are contributing. If we look at the original set of equations, we see that in reality there are only two different relationships:

$$\vec{M}_{total}(t_1) = A_1 + A_2 + A_3 + A_4 = A_1' + A_3'$$

and

$$\vec{M}_{total}(t_m) = A_1' e^{-i\cdot\gamma(x_1\cdot G_x)\cdot(dt\cdot(m-1))} + A_3' e^{-i\cdot\gamma(x_3\cdot G_x)\cdot(dt\cdot(m-1))}$$

$$= A_1' e^{-i\cdot c\cdot x_1(m-1)} + A_3' e^{-i\cdot c\cdot x_3(m-1)}$$

using $A_1' = A_1 + A_2$ and $A_3' = A_3 + A_4$ and $c = \gamma G_x dt$ The second expression is valid for all *t*, but closer inspection for t_2 to t_m should show that no matter the value of *m*, the entries of the row entries of the encoding matrix have a form of [*a a b b*] which implies that the first and second column vectors and the third and fourth column vectors are exactly the same.

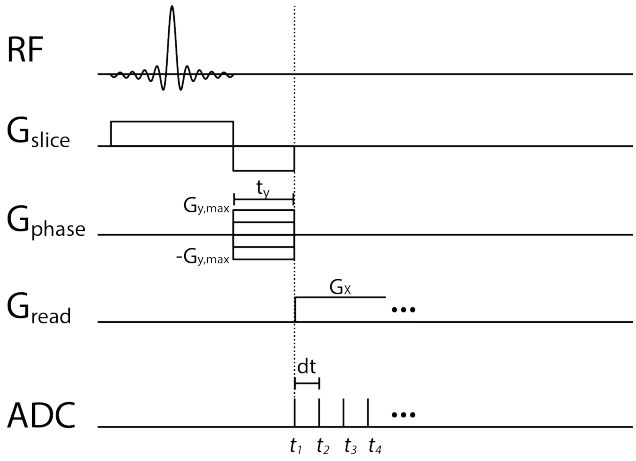
Question 2: Demonstrate (using the encoding matrix) that no matter how many samples we take for $\vec{\boldsymbol{M}}_{\boldsymbol{total}}(\boldsymbol{t_m})$, the solution does not improve. What do you think would happen if we did have relaxation incorporated into our model?

Experiment #3: Phase and Frequency Encoding

From the previous experiment we should know understand that frequency encoding alone will not be sufficient to determine the intensity (amplitude coefficient) for all pixels. There is now way to discriminate between pixels that share the same *x*-coordinate. We will add phase encoding to the pulse sequence

Phase encoding applies a gradient with maximum amplitude $G_{y,max}$ before the onset of frequency encoding, effectively imparting different phase to magnetization at different *y*-coordinates. We will include phase encoding into our expressions.

Experiment 3: Frequency and Phase Encoding



The change in frequency due to a phase encoding gradient is given by:

$$\Delta\omega = \gamma\Delta B(t) = \gamma\left(y \cdot G_y\right)$$

since the action of the phase encoding gradient is complete even before we begin sampling it is possible to determine the complete phase accrual due to the gradient:

$$\phi_g(y) = -\Delta\omega \cdot t_y$$
$$\phi_g(y) = -\gamma\left(y \cdot G_y\right) \cdot t_y$$

Note that this phase is a function of y-coordinate. It is not hard to discern that if we apply a single phase encoding gradient. We will run into the same problem as before (Try it – set N_{PE}=1). Hence, we require multiple measurements (N_{PE} of them), each with a different phase encoding gradient amplitudes. Each of those experiments will yield a complex-valued $\vec{M}_{total}$. We have to integrate the additional phase into our encoding matrix to attempt to recover $\vec{A}$.

First, we generate a matrix that represents the phase encoding process $\boldsymbol{E_{PE}}$. Each row of $\boldsymbol{E_{PE}}$ corresponds to one phase encode (e.g. phase accrual due to the m^{th} phase encoding gradient):

$$\boldsymbol{E_{PE}}(m) = e^{-i\cdot\gamma(y_p\cdot G_{y,m})\cdot t_y}$$

where y_p represents the y-coordinate of pth magnetization vector ($A_1 \ldots A_p$) and m represents the m^{th} phase encoding gradient. We will determine the amplitude of the applied phase encoding gradient as a fraction of the maximum phase encoding (f_{PE}) gradient, $G_{y,max}$, such that $f_{PE} \in \{-1 \ldots 1\}$ and the net gradient for m^{th} PE step is $f_{PE,m} * G_{y,max}$.

$$\boldsymbol{E_{PE}} = \begin{bmatrix} e^{-i\cdot\gamma(y_1\cdot f_{PE1}\cdot G_{y,max})\cdot t_y} & \blacksquare & \blacksquare & e^{-i\cdot\gamma(y_4\cdot f_{PE1}\cdot G_{y,max})\cdot t_y} \\ e^{-i\cdot\gamma(y_1\cdot f_{PE2}\cdot G_{y,max})\cdot t_y} & \blacksquare & \blacksquare & e^{-i\cdot\gamma(y_4\cdot f_{PE2}\cdot G_{y,max})\cdot t_y} \\ e^{-i\cdot\gamma(y_1\cdot f_{PE3}\cdot G_{y,max})\cdot t_y} & \blacksquare & \blacksquare & e^{-i\cdot\gamma(y_4\cdot f_{PE3}\cdot G_{y,max})\cdot t_y} \end{bmatrix}$$

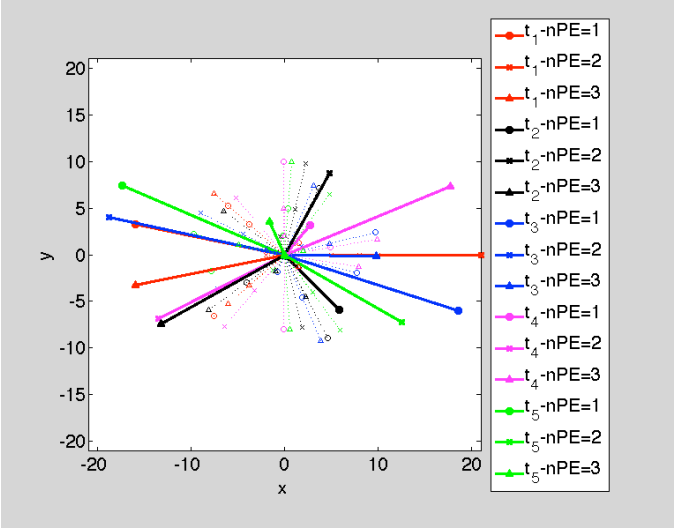
Again, the ■ represent other coefficients omitted for clarity. For the example above with N_{PE}=3, $f_{PE} \in \{-1,0,1\}$.

Now that we have determined the encoding matrix derived with phase encoding we have to combine it with the encoding matrix derived from frequency encoding, $\boldsymbol{E_{FE}}$. In other words we need to fill in the expression:

$$\vec{M} = \boldsymbol{E_{FE}}\boldsymbol{E_{PE}}\vec{A}$$

Unfortunately, if we use the definitions above, $\boldsymbol{E_{FE}}$ is a N_{dt} x N_{pixels} matrix, while $\boldsymbol{E_{PE}}$ is a N_{PE} x N_{pixels} matrix. To combine them we take into account the fact that for every phase encode, we have a complete set of frequency encodes. As shown in the MATLAB code provided, we replicate $\boldsymbol{E_{PE}}$ and concatenate along the row dimension N_{dt} times, and replicate and reshape $\boldsymbol{E_{FE}}$ to match appropriately. We finish we a combined encoding matrix that is (pseudo)inverted:

$$\vec{A'} = \boldsymbol{E^{+}_{FE,PE}}\vec{M}$$



the 'plus' superscript denotes the Moore-Penrose inverse which provides the least square solution to the system of equations, and the prime denotes the estimate of $\vec{A}$.

If examine the individual magnetization vectors after both phase and frequency encoding (figure), we observe that there are many more combinations of the vectors which allow for a solution to the system of equations to be found, and for and estimate of $\vec{A}$ to be made. If we look at the example with N_{dt}=5 and N_{PE}=3 (15

complex data samples), we see that the rank of the encoding matrix has increased to 4, and that the error in the recovered matrix is < 10^{-14} (execute ***ImagingExperiment.m***). The row-reduced echelon form now has a value in each of N_{pixel} columns, and the condition number is quite small (1 is the smallest it can be).

$$rref(\boldsymbol{E}) = \begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \end{matrix}$$

Question 3: what happens to the rank of the encoding matrix as we a) increase and b) decrease the total number of samples (N_{dt}xN_{PE})? How low can you go and still recover $\vec{A}$? Use A2_2 and A3_3 to investigate. How a) low or b) high do you need to go to recover a 10x10 complex (A10_10i) matrix?

Additional Questions

Q4: What do you think would happen if you set ndt=1? (e.g. you remove frequency encoding?)

Q5: What happens when you use A2_2i – the complex version of the matrix? This would be the case in which the assumption of a real image breaks down, which is in fact a true as we can't really guarantee a perfectly homogeneous B_0 field. A3_3i? Do the limits of how small you can make NPE and Ndt change?

Q6: What happens when we add noise of increasing standard deviation? Test your findings with A2_2 and 3 phase and frequency encodes. Increase the standard deviation of the noise (sigma) in a stepwise fashion and observe the errors reported after image intensity coefficient recovery.

Q7: Can we exchange between N_{PE} and N_{dt}? For example, can you recover A4_4 with NPE=2 and Ndt=8?

Q8: What happens when we increase the number of elements in A? Try A10_10. How large does the combined encoding matrix get? What if we had an A256_256? A512_256?

Q9: What happens as we increase N_{PE} and N_{dt} simultaneously? Set each to 21, and use A10_10. Try 31! Is this going to be a problem?

Q10: What do you think would happen to our simulation if we dealt with continuous tissue as opposed to pixels with homogeneous intensity that we can treat as delta functions? Think of the amount of phase twist being imparted by a combination of $(\gamma * G * \Delta t)$ within a single pixel.

Experiment #4: How to choose an encoding matrix

In the previous experiments (simulations!) we used pseudo-random (manually chosen) values for the gradient amplitudes and durations. This lead to uncontrolled phase accrual due to each gradient, something we cannot tolerate in real life due to limitations in gradient strength and in the rate of change of gradient amplitude. Relaxation also interferes, demanding certain speed in acquisition. The use of large image matrices (e.g. N_{pixels}=256x256=65536!) makes creating very these matrices unwieldy and maybe even untenable. Finally, the presence of noise destabilizes inversions as seen in the previous sections, making the recovery of the image coefficients in the presence of noise unnecessarily difficult and unstable.

In this next section we try and figure out how to best choose these values. In particular, given the limitations of the pseudoinverse for image recovery in the face of large encoding matrices needed to recover realistic images, we will seek better methods.

One way to approach this problem is to treat it as a standard linear inversion problem: $\vec{d} = \boldsymbol{G}\vec{a}$, where $\vec{d}$ is (observed) data or measurement vector, $\boldsymbol{G}$ is the model of the system we are examining and $\vec{a}$ is a vector of values we are interested in recovering (the unknown complex image coefficients). The standard solution would have $\vec{a} = \boldsymbol{G}^{-1}\vec{d}$. However, there is no guarantee that $\boldsymbol{G}$ is invertible, as it may not contain sufficient information to uniquely determine $\vec{a}$. If $\boldsymbol{G}$ were composed of linearly independent rows, the inverse would be possible and a solution to the system would be easier and obtainable by simple inversion. In other words, if $\boldsymbol{G}$ is rank deficient, it is not invertible and we cannot recover $\vec{a}$ robustly, a fact we have observed empirically in the previous exercise. If the system is overdetermined (more rows than columns in $\boldsymbol{G}$) we may find $\vec{a}$ through the use of the pseudoinverse, a least-squares solution. Nevertheless, it would nice to design $\boldsymbol{G}$ (or in our case the encoding matrix $\boldsymbol{E}$) such that we could simply invert it.

Either way, we need to find a way to construct our encoding matrix to allow for easier inversion and recovery of the missing coefficients.

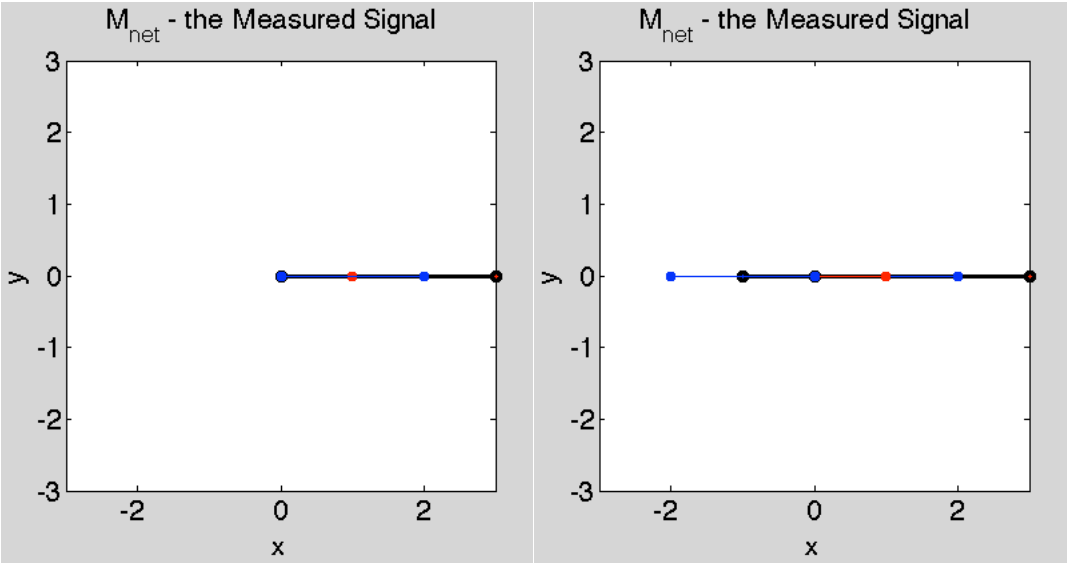
To start simply we consider the system of two coefficients. It is not hard to see that the best way to discriminate between these two vectors, is to add them, and to subtract them. Equivalently, we would want to encode such that the vectors were aligned in one encoding experiment, and 180° apart in the other. Hence, our encoding matrix would look like:

$$\boldsymbol{E} = \begin{bmatrix} \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{-1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{e}^{i\frac{2\pi}{2}\cdot\mathbf{0}} & \boldsymbol{e}^{i\frac{2\pi}{2}\cdot\mathbf{0}} \\ \boldsymbol{e}^{i\frac{2\pi}{2}\cdot\mathbf{0}} & \boldsymbol{e}^{i\frac{2\pi}{2}\cdot\mathbf{1}} \end{bmatrix}$$

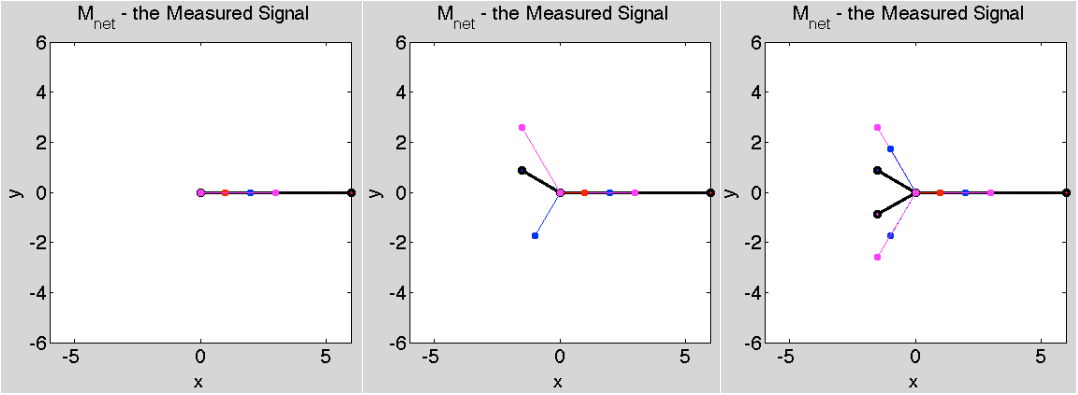
The second matrix is interesting because each complex exponential represents a rotation about an angle. In this case, we divide 2π into 2 as there are two samples. In the first row, neither magnetization vector is rotated. Encoding with the second row induces the 180° rotation of the second vector.

When dealing with ideal measurements, the recovery of our image coefficients could be done with any angle between the two vectors though in the presence of noise, we should be able to recover our coefficients most robustly if they are separated as far as possible. Explore the script ***separatevectors.m***. You should see that as the angle of separation deviates from ideal, the error in our recovery increases. Remember, we are dealing with a square matrix, which means that we are taking exactly the minimum number of measurements to recover our coefficients (unlike before when we took extra measurements).

Graphically, this can be seen in the following figures, where net magnetization measurements can be seen in thick black lines and individual vectors (A = {1;2} x={0,-1}) visible in the thinner dotted lines (red and blue). The figure on the left shows the effect of the first row of the encoding matrix: the two values individual vectors are added. The net magnetization vector (black) is what is measured. The figure on the right shows the effect of the second row of the encoding matrix superimposed on that of the first row (Net magnetization = -1).



We can repeat the exercise with 3 coefficients to be recovered. Intuitively, it makes sense to separate out the 3 individual magnetization vectors in the x-y plane in angular dimension. We do so with 120° ($2\pi/3$) degree rotations. The third row of the encoding matrix (and we need three) requires a little more thought: if we assume that our gradient must be constant, that is the rate of phase accrual ($\Delta\omega$) is fixed, then as much phase as was accrued during the first encoding must be accrued in the second encoding. This is certainly true for frequency encoding, during which we cannot turn the gradient on/off or change its amplitude while we are acquiring data. We assume it must also hold constant for phase encoding.



The figure on the left is exactly as expected for a test set of coefficients $A_3 = \{1,2,3\}$ located at x-coordinates $X_3 = \{0, 1, -1\}$. The three vectors with maintain alignment and add constructively ($M_{net} = 6$) after the application of the first row of the encoding matrix. After the second row of the encoding matrix is applied, the second (length 2) and third (length 3) magnetization vectors have rotated +120° and -120° degrees, respectively. For the third row, we repeat that rotation adding another +120° for the second vector and -120° for the third vector. Note that negative rotation is counterclockwise. This leaves us with an encoding matrix that looks like:

$$\boldsymbol{E} = \begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \boldsymbol{e}^{-i\frac{2\pi}{3}} & \boldsymbol{e}^{+i\frac{2\pi}{3}} \\ \mathbf{1} & \boldsymbol{e}^{+i\frac{2\pi}{3}} & \boldsymbol{e}^{-i\frac{2\pi}{3}} \end{bmatrix}$$

This encoding matrix will maximize the angular distance between all three vectors for each encoding other than the first (with no encoding). We can convert this matrix to have a more general description:

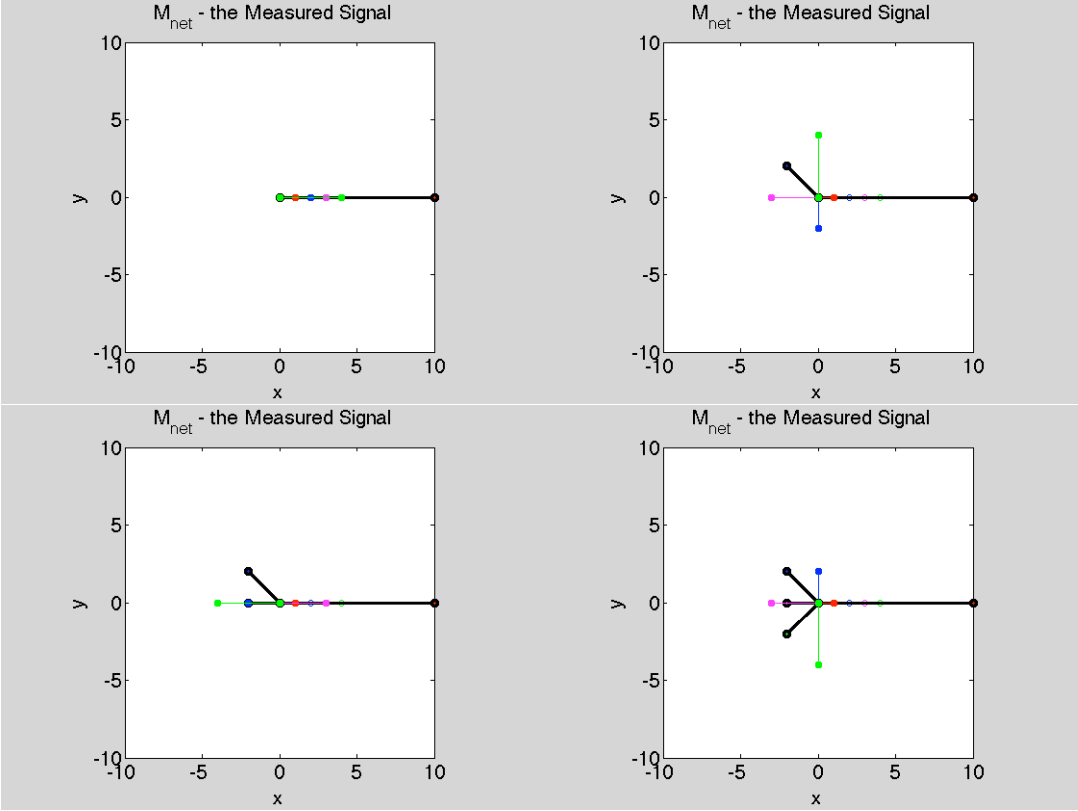
$$\boldsymbol{E} = \begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \boldsymbol{e}^{-i\frac{2\pi}{3}} & \boldsymbol{e}^{-i\frac{4\pi}{3}} \\ \mathbf{1} & \boldsymbol{e}^{-i\frac{4\pi}{3}} & \boldsymbol{e}^{-i\frac{8\pi}{3}} \end{bmatrix} = \begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \boldsymbol{e}^{-i\frac{2\pi}{3}\cdot\mathbf{1}} & \boldsymbol{e}^{-i\frac{2\pi}{3}\cdot\mathbf{2}} \\ \mathbf{1} & \boldsymbol{e}^{-i\frac{2\pi}{3}\cdot\mathbf{2}} & \boldsymbol{e}^{-i\frac{2\pi}{3}\cdot\mathbf{4}} \end{bmatrix}$$

Closer examination will yield that each element of the encoding matrix can be described as

$$E_{m,n} = e^{-i\frac{2\pi}{3}m\cdot n}$$

where m and n are the row and column indexes and $m, n \epsilon \{0 \dots N-1\}$.

Lets examine one more example. For A_4={1 2 3 4}, located at coordinates { -2, -1, 0, 1} we imagine spreading the vectors out in 90° intervals. The following four figures depict the realignment of the magnetization vector after the application of each row of the encoding matrix. Unlike before, the previous individual vectors have been removed for clarity.



The encoding matrix for the 4 sample case.

$$\boldsymbol{E} = \begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \boldsymbol{e}^{-i\frac{\pi}{2}\cdot 1} & \boldsymbol{e}^{-i\frac{\pi}{2}\cdot 2} & \boldsymbol{e}^{-i\frac{\pi}{2}\cdot 3} \\ \mathbf{1} & \boldsymbol{e}^{-i\frac{\pi}{2}\cdot 2} & \boldsymbol{e}^{-i\frac{\pi}{2}\cdot 4} & \boldsymbol{e}^{-i\frac{\pi}{2}\cdot 6} \\ \mathbf{1} & \boldsymbol{e}^{-i\frac{\pi}{2}\cdot 3} & \boldsymbol{e}^{-i\frac{\pi}{2}\cdot 6} & \boldsymbol{e}^{-i\frac{\pi}{2}\cdot 9} \end{bmatrix} = \begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \boldsymbol{e}^{-i\frac{\pi}{2}} & \boldsymbol{e}^{-i\pi} & \boldsymbol{e}^{+i\frac{\pi}{2}} \\ \mathbf{1} & \boldsymbol{e}^{-i\pi} & \mathbf{1} & \boldsymbol{e}^{-i\pi} \\ \mathbf{1} & \boldsymbol{e}^{+i\frac{\pi}{2}} & \boldsymbol{e}^{-i\pi} & \boldsymbol{e}^{-i\frac{\pi}{2}} \end{bmatrix}$$

Question 11: write out the encoding matrix for a six vector experiment. The matrix should spread out the magnetization vectors in intervals of 60°.

It should not surprise you to know that this form of a matrix is well known. This matrix is actually the encoding matrix of the discrete Fourier Transform, defined in MATLAB as:

$$X(m) = \sum_{n=1}^{N} e^{-i\frac{2\pi}{N}(n-1)(m-1)} \quad for\ 1 \le m \le N$$

The script *CreateEncodingMatrix.m* should help you explore what the Fourier Matrix looks like and its relationship to the *fft()* function in MATLAB.

Question 12: how fast can you generate the encoding (forward) and inverse matrices for larger numbers (e.g. 256, 512), relative to the pseudoinverse approach?

This exercise demonstrates that to get the best chance of recovering the complex coefficients that make up our image intensities using the fewest coefficients, we should use a matrix composed of linearly independent rows (or columns), and that the Fourier Matrix, the matrix representation of the Discrete Fourier Transform, does exactly that and matches what we can achieve using the linear gradients available in MRI. Therefore, we use Fourier encoding and the Fourier Transform to recover our images!

Experiment #5: How to choose FE and PE values

Now that we know we should be using the Fourier Transform to improve the recovery of our image coefficients, we need to decide what physical values to use (i.e. gradient amplitudes and durations). We begin by observing that for both FE and PE, the phase accrued can be given by

$$\phi_g(\vec{r},t) = -\gamma\left(\vec{r}\cdot\vec{G}(t)\right)\cdot t_{gradient}$$

where we have switched to a generic coordinate (given by $\vec{r}$) and generic gradient (given by $\vec{G}$), and $t_{gradient}$ represents the time the gradient is on. For the case of combined frequency and phase encoding, the net phase accrued at a given point in space at time point n·dt is given by:

$$\phi_g(x,y) = -\gamma\left(x\cdot G_x\cdot n\cdot dt + y\cdot G_y\cdot t_y\right)$$

Note that for our ideal gradients (with infinite slew rate that can go form 0 to *G* in no time), the product of $G\cdot t$ represents the area under the gradient curve, which is in fact the area of a rectangle of amplitude *G* and duration *t*. We can generalize this expression such that

$$\phi_g(\vec{r},t) = -\gamma\int_0^t \vec{r}\cdot\vec{G}(t')\,dt'$$

$$\phi_g(\vec{r},t) = -\gamma\,\vec{r}\int_0^t \vec{G}(t')\,dt'$$

The resultant expression does not require that gradients be ideal. It also assumes that spins start accruing phase from *t=0*, defined as the time at the 'middle' of the RF pulse. In fact, we can define a new quantity, *k*, which incorporates the gradient area in relationship to phase accrual:

$$\vec{k}(t) = \gamma\int_0^t \vec{G}(t')\,dt'$$

which results in the phase accrual expression

$$\phi_g(\vec{r},t) = -2\pi\cdot\vec{r}\cdot\vec{k}(t)$$

$\vec{k}(t)$ has units of spatial frequency (cycles/mm^{-1} or cycles/cm^{-1}). The magnetization we can detect can now be expressed as

$$\vec{M}_{total}(t) = \sum_{p=0}^{N_{pixels}-1} A_p\, e^{-i2\pi\cdot\vec{r_p}\cdot\vec{k}(t)}$$

If we expand to separate the coordinate systems, we get

$$\vec{M}_{total}(t) = \sum_{p=0}^{N_{pixels}-1} A_p \, e^{-i2\pi\cdot\left(x_p\cdot k_x(t)+y_p\cdot k_y(t)\right)}$$

If we look more closely at this equation we can see that it represents the discrete Fourier Transform (DFT) of our complex coefficients. The coordinates in k-space, which are determined by the areas of the respective gradients, determine which coefficients of the DFT we are sampling at any given time. Hence, we choose which coefficients we sample based by changing parameters, and we can choose those gradient parameters based on standard sampling theory.

There are established relationships between Fourier space, which we will call k-space, and image space. We begin by defining the sampling intervals in k-space as δk_x and δk_y. We can express the sampled magnetization as:

$$\widehat{M}(k_x, k_y) = M(k_x, k_y) \cdot \left(\frac{1}{\delta k_x \cdot \delta k_y}\right)^2 \cdot \mathrm{III}^{2D}\left(\frac{k_x}{\delta k_x}, \frac{k_y}{\delta k_y}\right) \cdot \prod^{2D}\left(\frac{k_x}{W_{kx}}, \frac{k_y}{W_{ky}}\right)$$

$\widehat{M}(k_x, k_y)$ represents the sampled magnetization, while $M(k_x, k_y)$ is the continuous signal sampled as function of k-space coordinate (Nishimura Chapters 2& 5). These samples are multiplied by a 2D comb function with samples every δk and the measurement is windowed with a 2D rect function, which represents the fact that we can only sample k-space in a limited fashion. The window widths are defined as:

$$W_{kx} = 2\left(k_{x,max} + \frac{\delta k_x}{2}\right)$$
$$W_{ky} = 2\left(k_{y,max} + \frac{\delta k_y}{2}\right)$$

which is simply the number of samples in k-space along a given dimension multiplied by δk. The boundaries of the rectangular window extend to half a pixel outside the limits defined by k_{max}. The resultant Fourier Transform of the sampled signal $\widehat{M}(k_x, k_y)$ is then

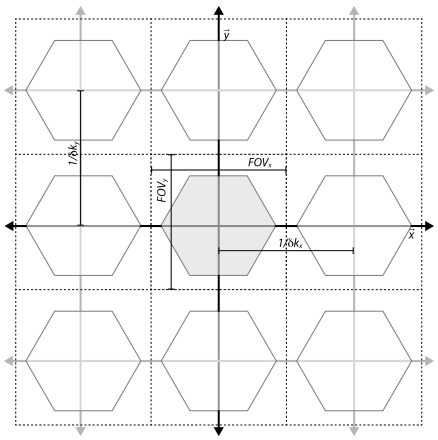
$$\widehat{m}(x, y) = m(x, y) * \mathrm{III}^{2D}(\delta k_x x, \delta k_y y) * (W_{kx} \cdot W_{ky}) sinc(W_{kx} x) \cdot sinc(W_{ky} y)$$

with these relationships, we can now establish the links between our simulation parameters and the coverage and sampling of k-space!

First, let's examine the relationship between the sampling in k-space and FOV. If we assume infinite sampling (no windowing), then the transformed sampled magnetization reduced to

$$\widehat{m}(x,y) = m(x,y) * \text{III}^{2D}\left(\delta k_x x, \delta k_y y\right)$$

The 2D convolution that results from sampling implies that *m(x,y)* is replicated in intervals of $1/\delta k_x$. We aim to separate these replicants, which means that to avoid overlap, the spacing must be the same as the FOV. Therefore,



$$FOV_x = \frac{1}{\delta k_x}, FOV_y = \frac{1}{\delta k_y}$$

The figure shows that sampling introduces replicates spaced $1/\delta k$. If the repeating pattern is such that the object is not contained within an interval $1/\delta k$, aliasing will occur. Note that $1/\delta k_x$ and $1/\delta k_y$ are independent, as are FOV_x and FOV_y.

Relating back to our simulation, the user will specify the FOVs, which means we have to calculate the δks that are needed to avoid aliasing. Using the incremental gradient areas to determine the δks we get:

$$\delta k_x = \gamma \cdot G_x \cdot dt$$

and

$$\delta k_y = \gamma \cdot G_{y,i} \cdot dt_y = \gamma \cdot (f_n - f_{n+1}) G_{y,max} \cdot t_y$$

where *m* is the phase encoding index and $(f_n - f_{n+1})$ represents the difference in amplitude from one phase encode to the next. If we rearrange the above equations, we can make one final observation:

$$\gamma \cdot G_x \cdot dt \;\; FOV_x = 2\pi$$
$$\gamma \cdot \Delta G_y \cdot t_y \;\; FOV_y = 2\pi$$

which implies that every additional encode increases the phase across the FOV by 2π. This fact should be useful in determining whether the encoding you are using is truly Fourier encoding.

Finally, we must find a relationship that helps us achieve our desired image resolution. Returning to the expression for the sampled object $\widehat{m}(x,y)$, we see that if we examine each of the replicates that results from sampling in k-space (i.e. ignore the comb function), we see that it is not simply *m(x,y)* but

$$\widehat{m}(x,y) = m(x,y) * \left(W_{kx} \cdot W_{ky}\right) sinc(W_{kx}x) \cdot sinc(W_{ky}y)$$

In this case, the desired function is convolved with a weighted 2D sinc. The sinc functions blur the underlying object, which will limit our ability to visualize small or highly detailed structures. Given and FOV, and a number of samples desired, we can use the following relationship to define resolution:

$$\delta_x = \frac{FOV_x}{N_{pixels,x}} = \frac{1}{\delta k_x \cdot N_{pixels,x}} = \frac{1}{W_{kx}}$$
$$\delta_y = \frac{FOV_y}{N_{pixels,y}} = \frac{1}{\delta k_y \cdot N_{pixels,y}} = \frac{1}{W_{ky}}$$

In other words, the resolution in image space is determined by the sampling width in Fourier space. Typically, the maximum sampled frequency is specified k_{max}, which then leads to

$$\delta_x = \frac{1}{2k_{x,max}} \text{ and } \delta_y = \frac{1}{2k_{y,max}}$$

The following image depicts these relationships. Note that dt and t_y are chosen using other restrictions, so for now we will use 4 us and 500 us respectively. We will also use $|G_{y,max}|$ = 30 mT/m. Given these parameters, and an input matrix (e.g. A10_11i), we should be able to calculate the coordinates of the k-space matrix we wish to sample (for the given resolution and FOV), as well as the number of frequency encodes and phase encodes, and the amplitudes of the gradients needed to achieve the desired sampling pattern.