Non-Cartesian Sampling and Image Reconstruction

MBP1400H

Mark Chiew (mark.chiew@utoronto.ca) **March 26, 2025**

Outline

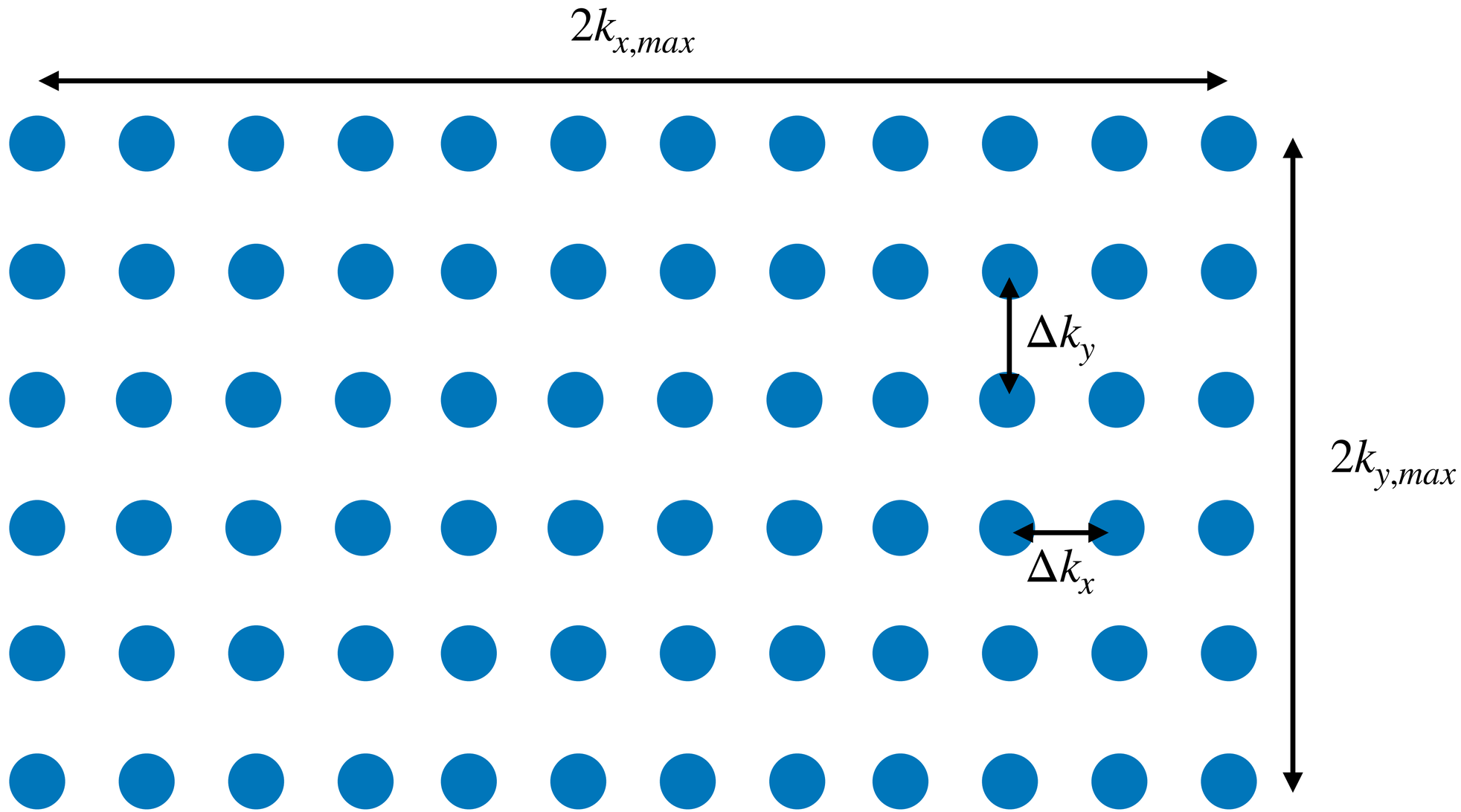
- **Part I – Review of Cartesian Sampling and Reconstruction**
  - Cartesian sampling “rules”
  - FFT-based reconstruction
- **Part II – Non-Cartesian Sampling**
  - Non-Cartesian trajectories
  - Point-spread functions
  - Defining image properties: FOV, resolution
- **Part III – Non-Cartesian Image Reconstruction**
  - Direct (Pseudo-)Inversion
  - Gridding/Non-uniform FFT
- **Part IV – Additional Considerations**
  - Density-compensation
  - Iterative Reconstruction
  - Noise Propagation & SNR Efficiency

Part I – Review of Cartesian Sampling and Reconstruction

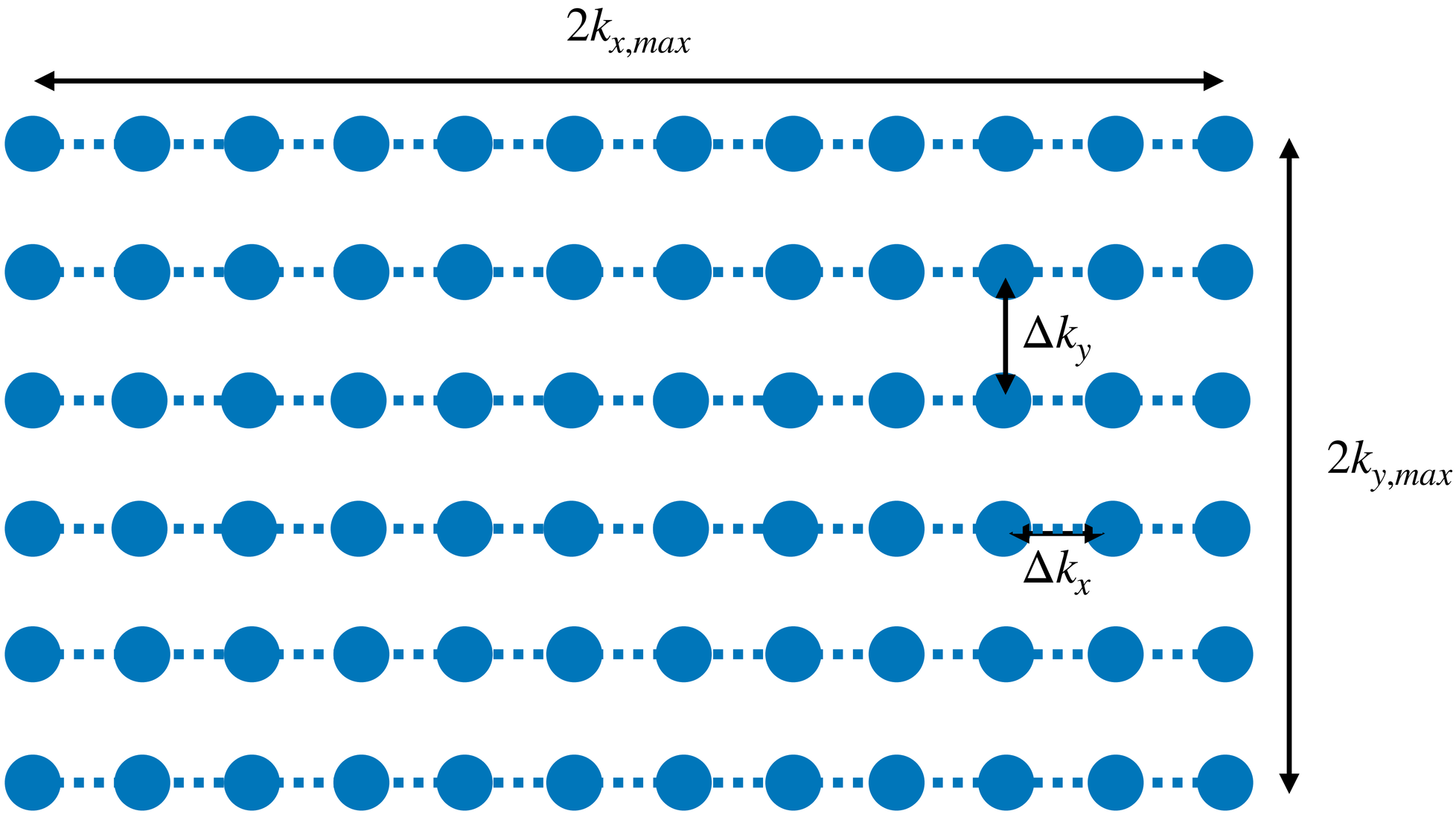
What is Cartesian Sampling?

- Cartesian sampling means:
  - Sampling on a grid with orthogonal (90º) axes
  - Samples are evenly spaced (in the same direction)
  - Samples can have different spacing on any axis
  - The sampling grid can have any dimensionality (typically 2D or 3D)
  - The grid is typically sampled by traversing straight lines in k-space

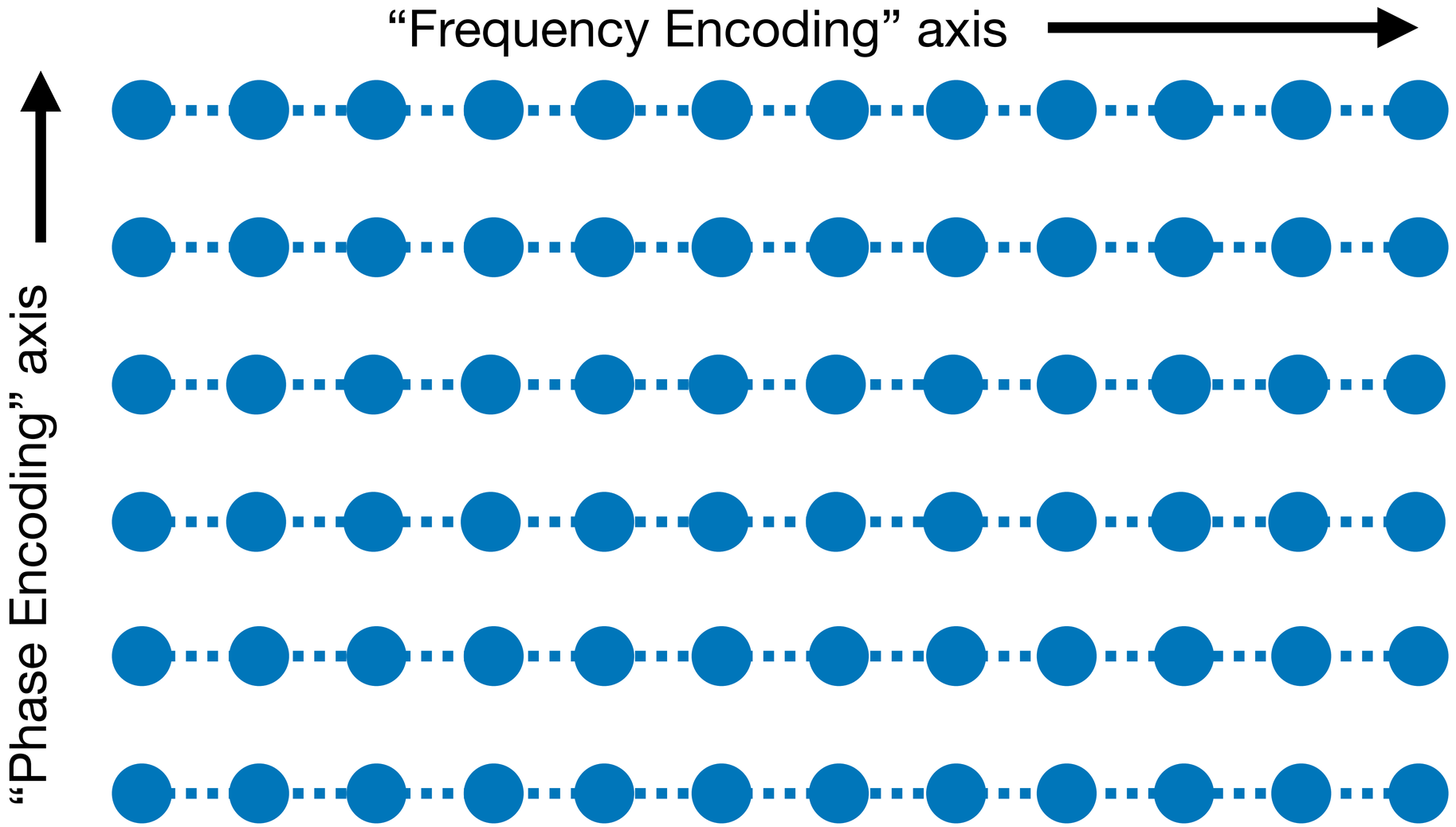
Cartesian Sampled k-space



Cartesian Sampled k-space



Cartesian Sampled k-space



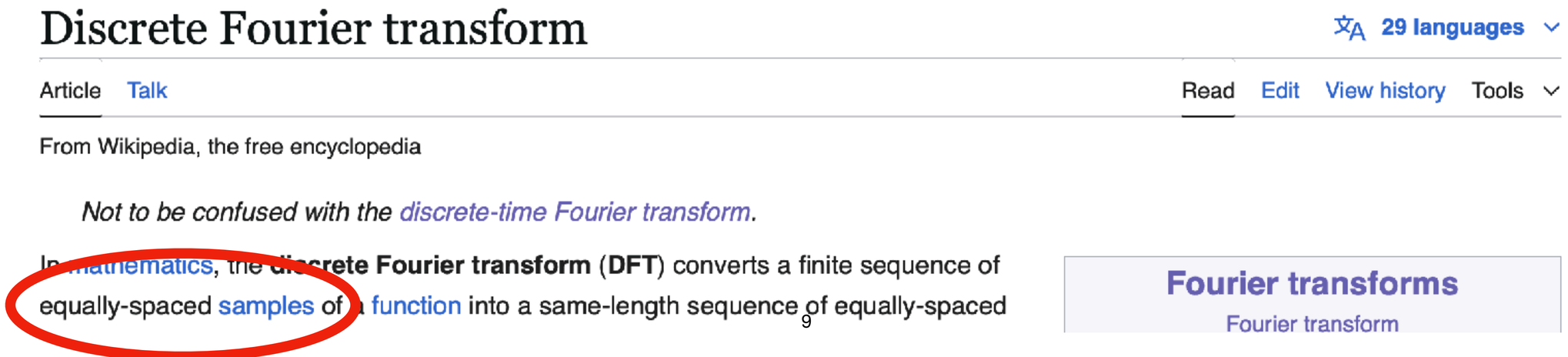
Cartesian k-space “rules”

- Image field-of-view:
  - $FOV_x = \frac{1}{\Delta k_x}$
  - $FOV_y = \frac{1}{\Delta k_y}$
- FOV in each direction is inversely proportional to the distance between samples in k-space

- Image resolution:
  - $\Delta x = \frac{1}{2k_{x,max}}$
  - $\Delta y = \frac{1}{2k_{y,max}}$
- Resolution is inversely proportional to the maximum extent of k-space sampling

Cartesian k-space reconstruction

- Use the inverse Fourier transform
  - What does that mean? The Fourier transform is for *continuous* functions
- It actually means use the inverse *discrete* Fourier transform (DFT)
  - What are the conditions necessary to be able to apply the DFT?
- To be able to apply the DFT (or inverse DFT) to a discretely sampled signal, the samples must be *equally spaced*


Discrete Fourier transform

29 languages

Article Talk Read Edit View history Tools

From Wikipedia, the free encyclopedia

Not to be confused with the discrete-time Fourier transform.

In mathematics, the **discrete Fourier transform** (**DFT**) converts a finite sequence of equally-spaced samples of a function into a same-length sequence of equally-spaced

Fourier transforms

Fourier transform

Cartesian k-space reconstruction

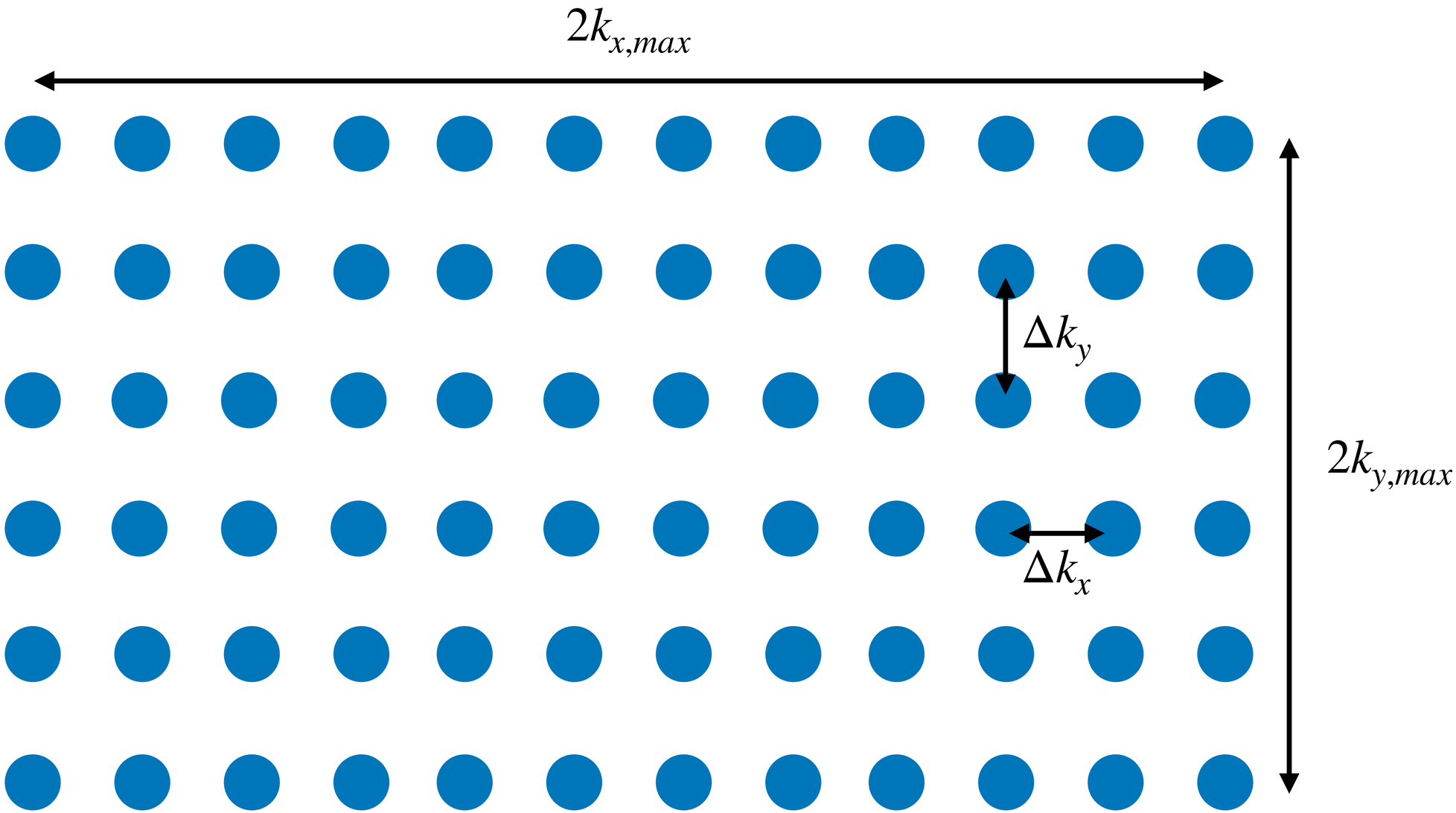
- The DFT is $O(N^2)$ complexity, which means computational cost scales very poorly as the k-space dimensionality goes up
- For a 256×256 sampled k-space, computing the DFT requires $256^4 \approx 4.3 \times 10^9$ operations (multiplications and additions)
- In practice, the “Fast Fourier Transform” (FFT) is used instead, which has $O(NlogN)$ complexity,
- For the same 256×256 k-space, computing the FFT requires only 1×10^6 operations, which is about 4000x faster!
- Because the FFT is a fast algorithm for computing the FFT, it has the exact same requirements of *equally spaced samples*

Cartesian k-space summary

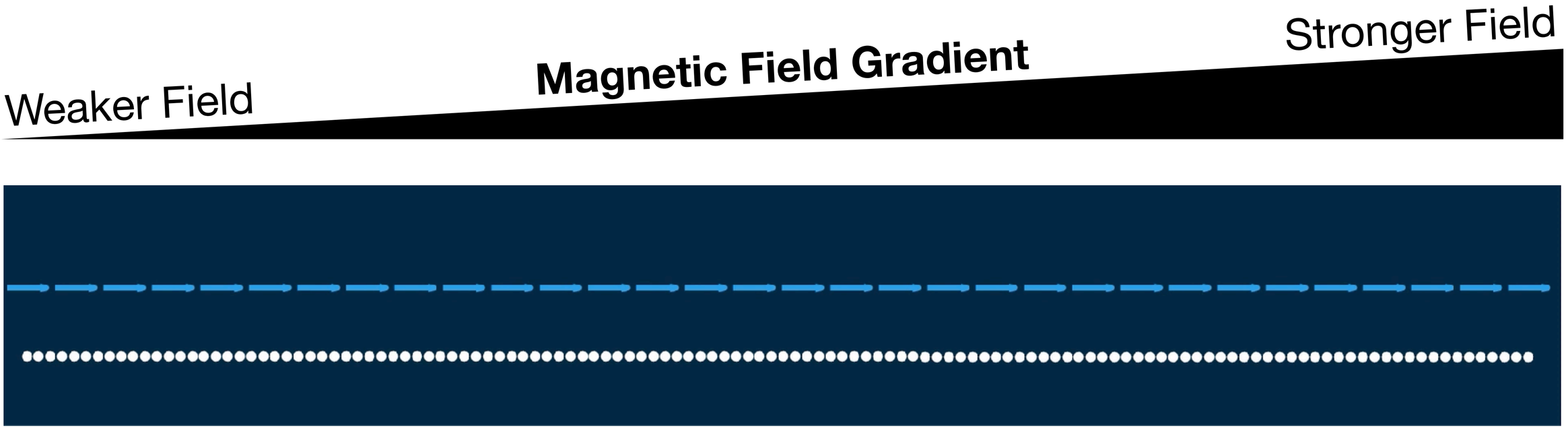
- Points are sampled on a regular grid
- Reconstruction can be performed with an inverse FFT
- Simple rules for image properties

Part II – Non-Cartesian Sampling

What is so special about these grid points?



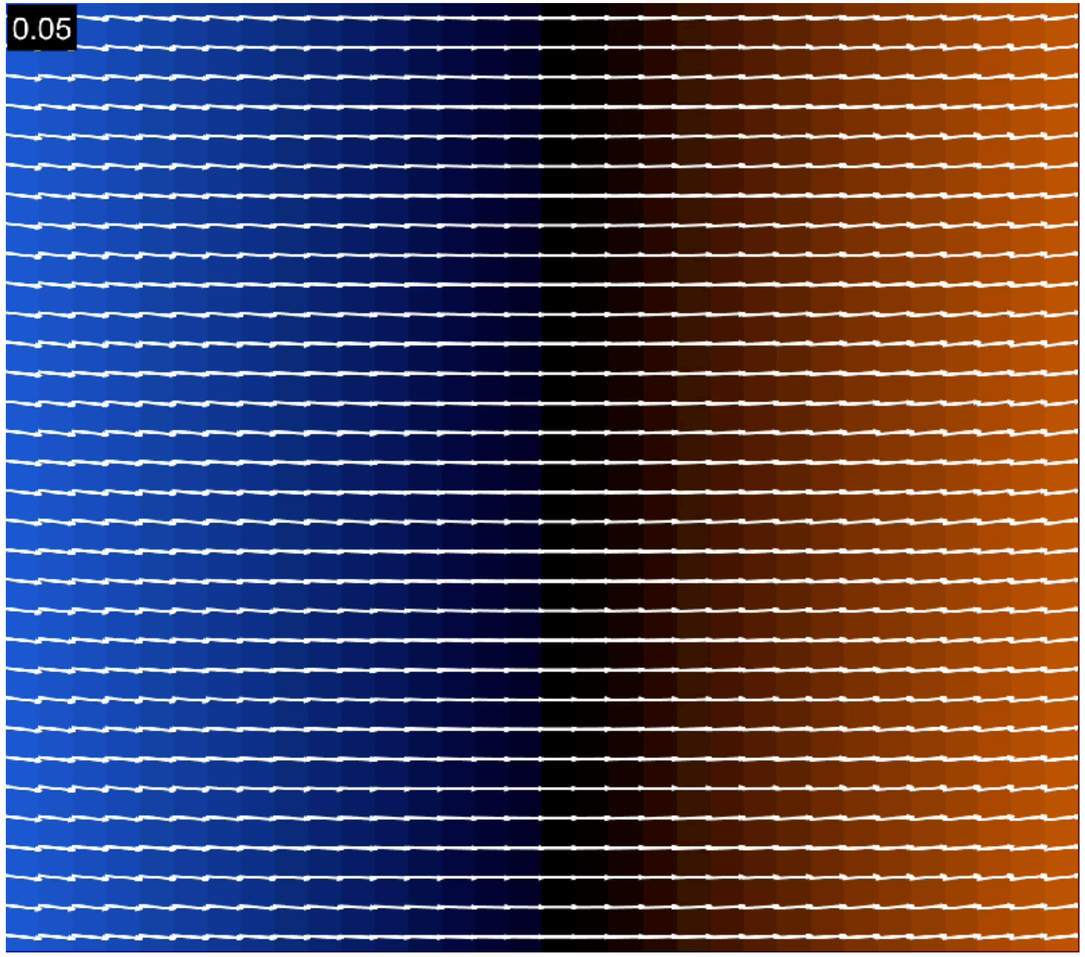
Recall: How k-space locations are reached

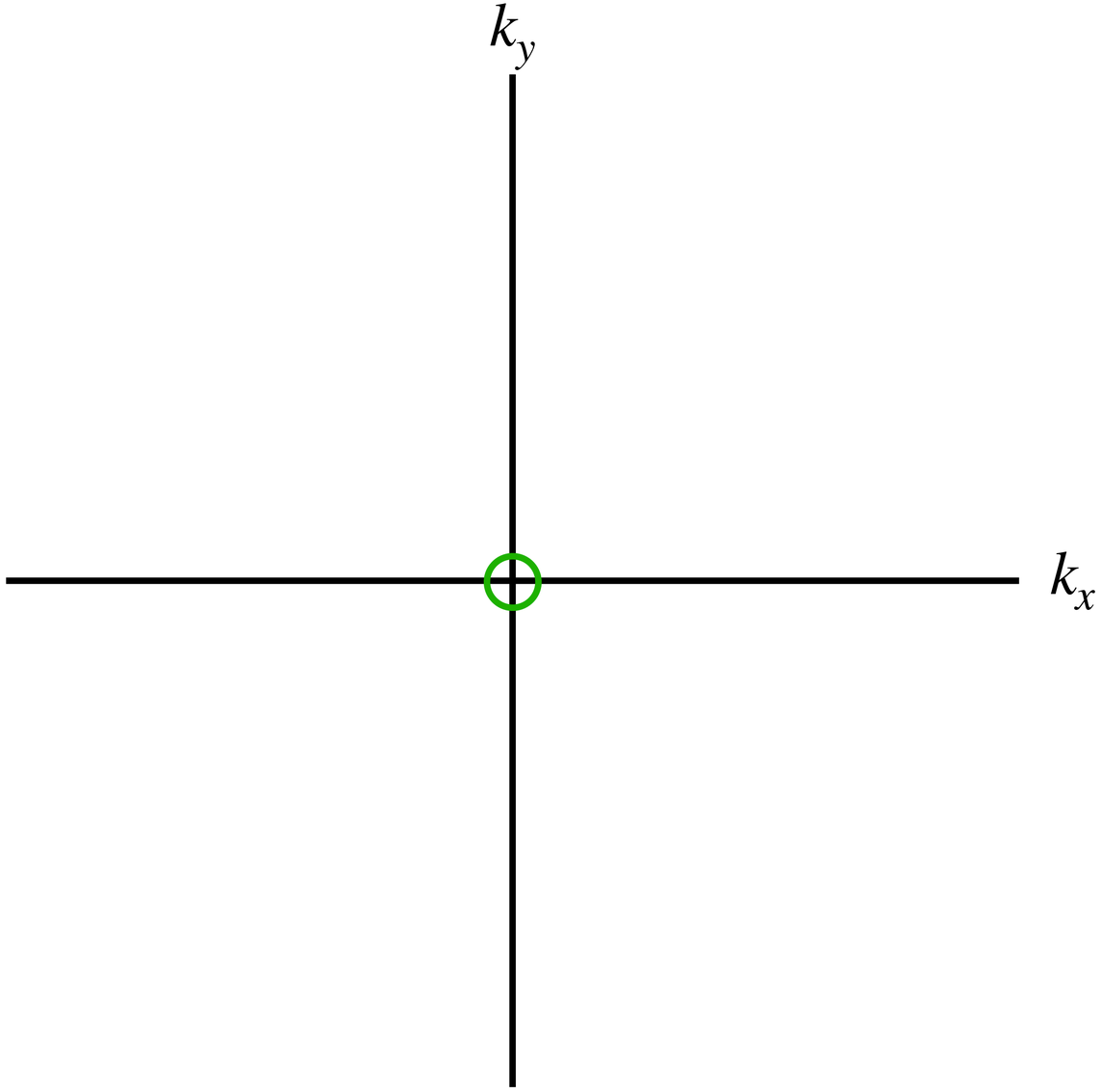


- The k-space co-ordinate space maps out spatial frequencies
- Larger “k” co-ordinates mean higher spatial frequencies (i.e. more cycles per unit length, equivalent to wave-number)

Recall: How k-space locations are reached

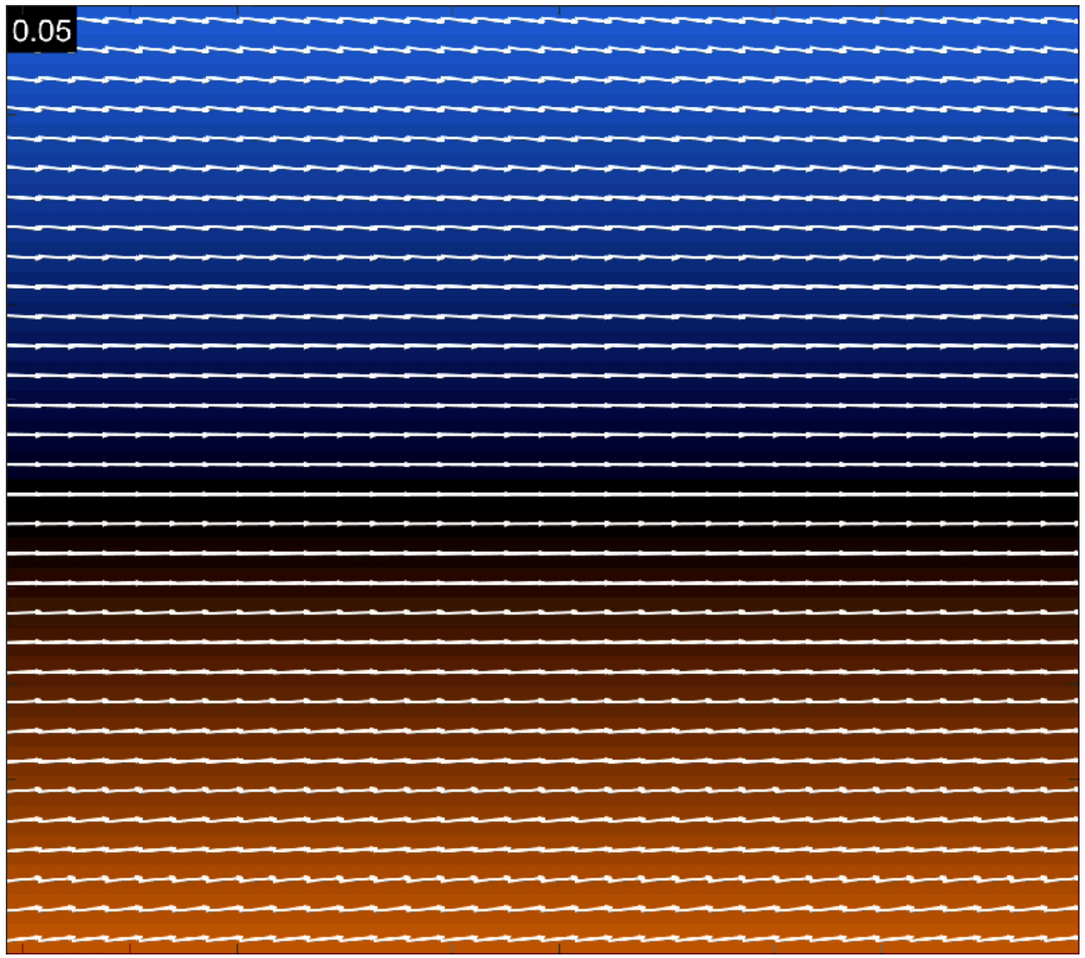
X-gradient

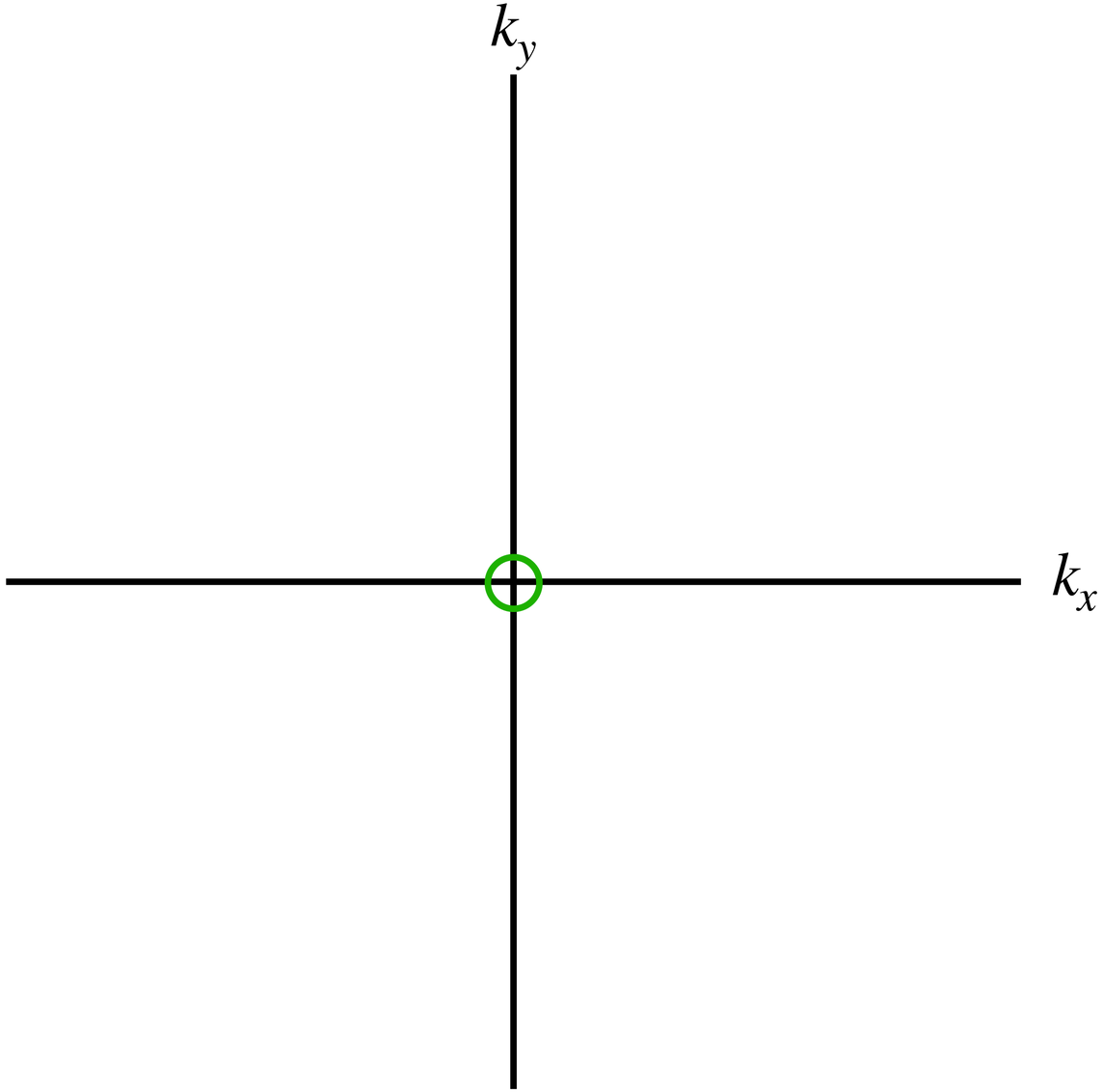




Recall: How k-space locations are reached

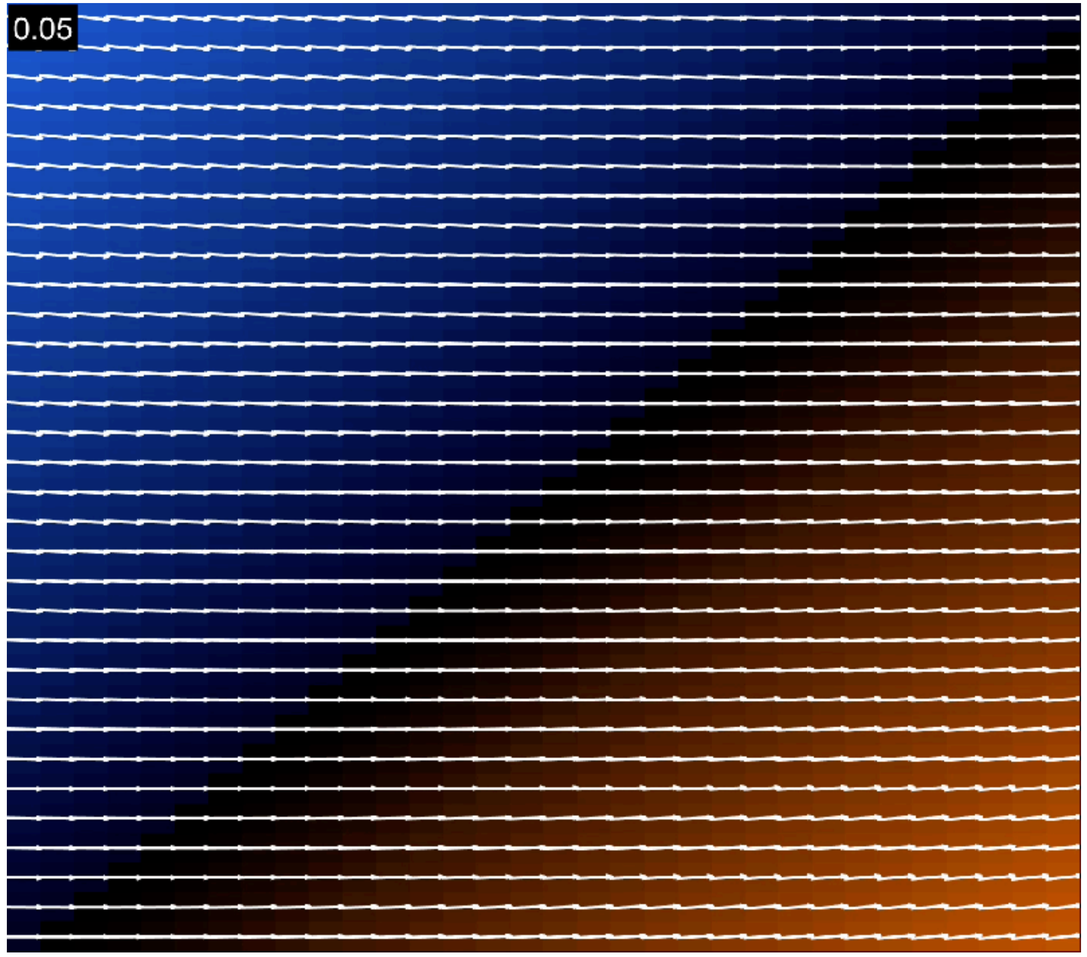
Y-gradient

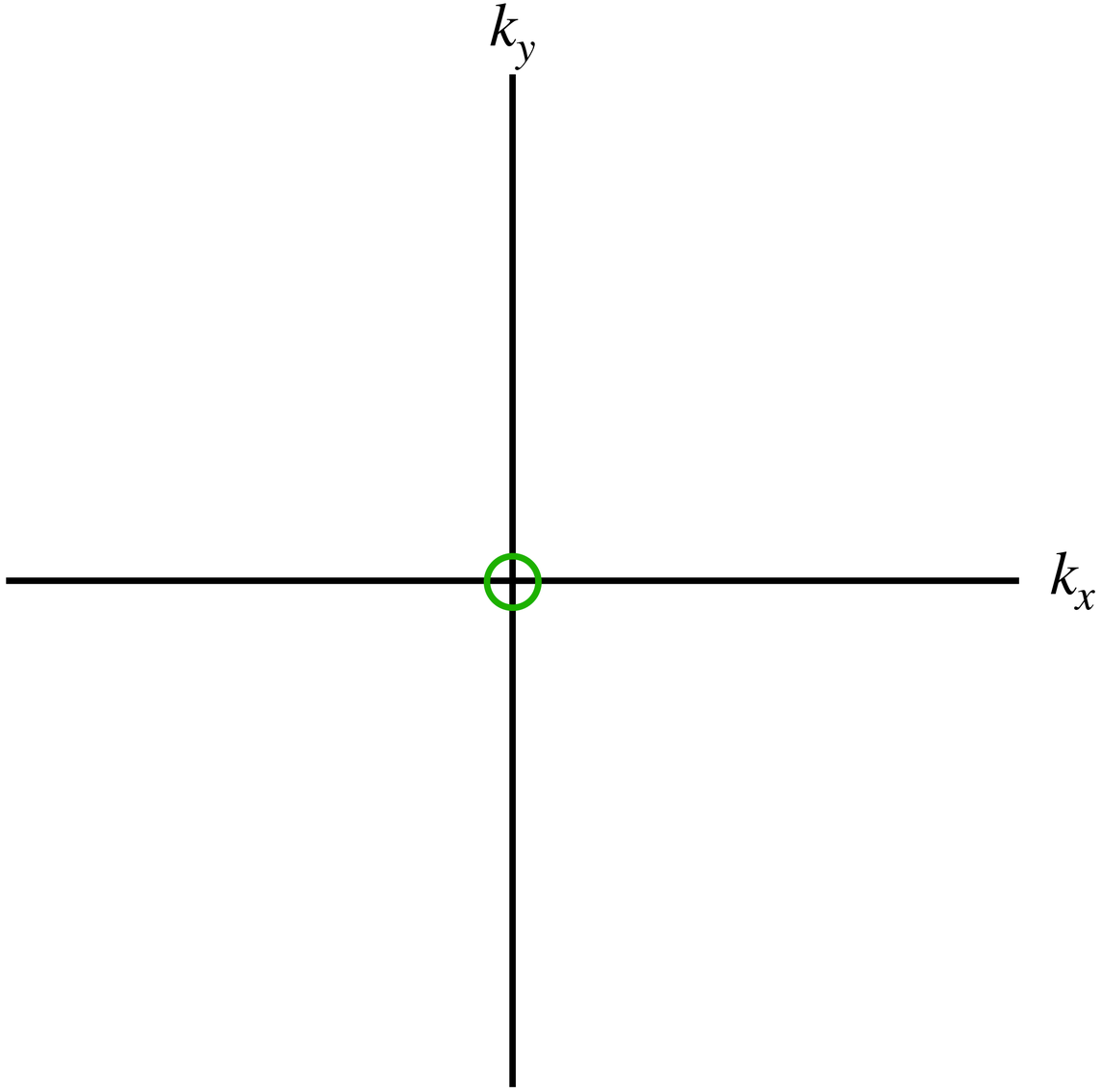




Recall: How k-space locations are reached

X+Y gradients





Recall: How k-space locations are reached

- Note that a given spatial frequency pattern can be reached by turning on a gradient, and waiting a certain amount of time
- You can also increase the gradient *strength* and decrease the time, or decrease the gradient strength and increase the time
- These achieve the same goal of creating some positionally dependent phase offset by some product of frequency by time ($\phi = \omega \Delta t$)

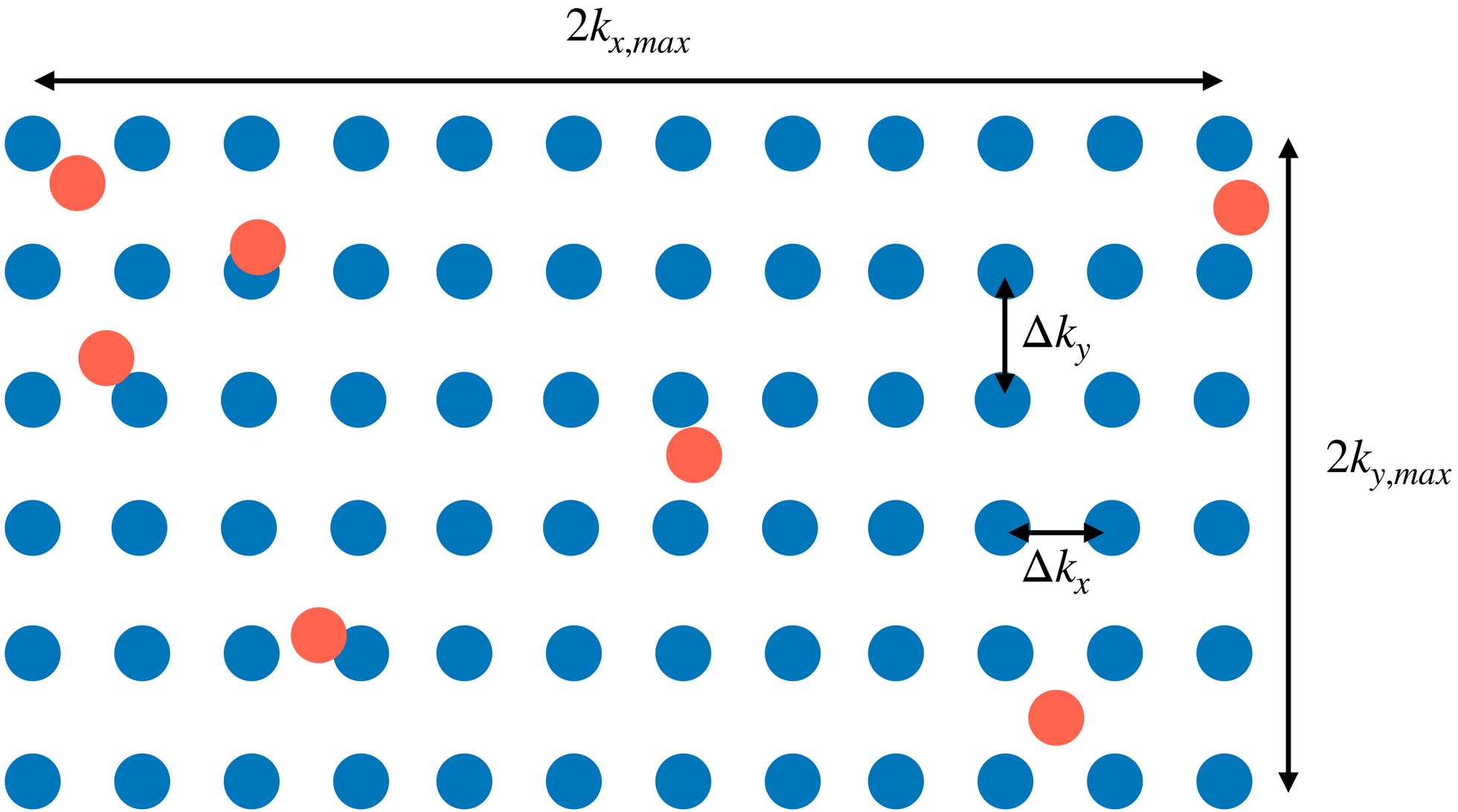
Recall: How k-space locations are reached

- Formally $k(t) = \frac{\gamma}{2\pi} \int_0^t G(\tau) d\tau$, or the area under the gradient amplitude
  - sometimes also referred to as the zeroth moment of the gradients
- So k-space *velocity* is directly related to gradient strength
  - Higher gradient strength means more area accumulation in a fixed time, and therefore a greater k-space displacement in the same time
  - Higher gradient amplitudes move us *faster* through k-space

K-space co-ordinates are continuous

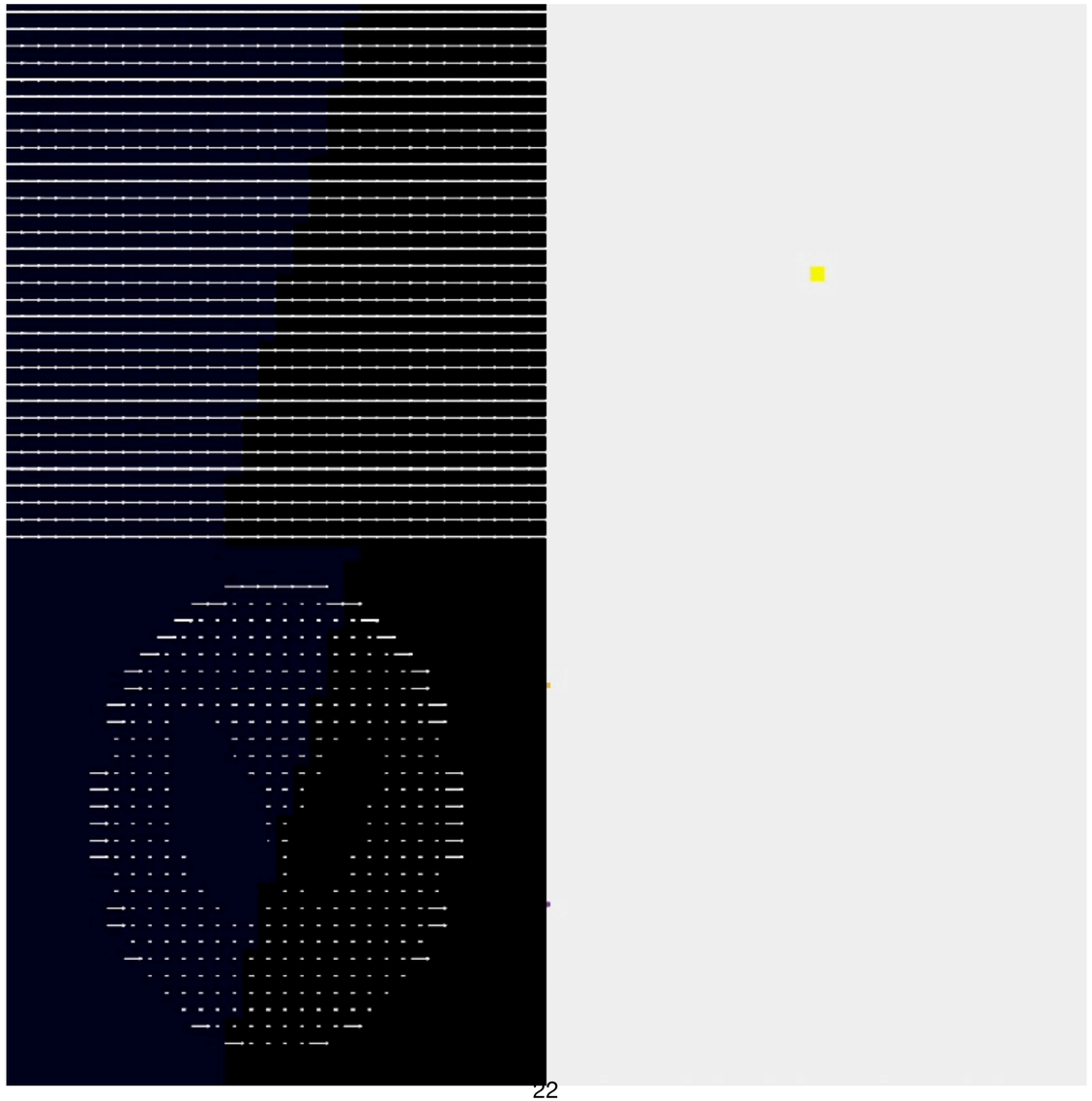
- K-space co-ordinates are *continuous* variables
- Even though we record our data in discrete samples, the *location* of those samples can be placed *anywhere in k-space*
  - $k_x = 0\ m^{-1}$, $k_x = -3.182\ m^{-1}$, $k_x = \frac{\pi}{7}\ m^{-1}$, $k_x = -\sqrt{3}\ m^{-1}$ all valid
- With control over the time-varying gradient strength and precisely when we choose to sample, we can sample essentially any location in k-space, on a grid or not

Nothing is special about these grid points



Spiral Sampling

Encoding Phase

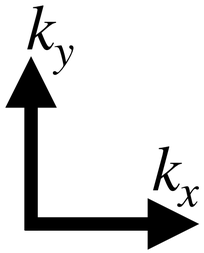


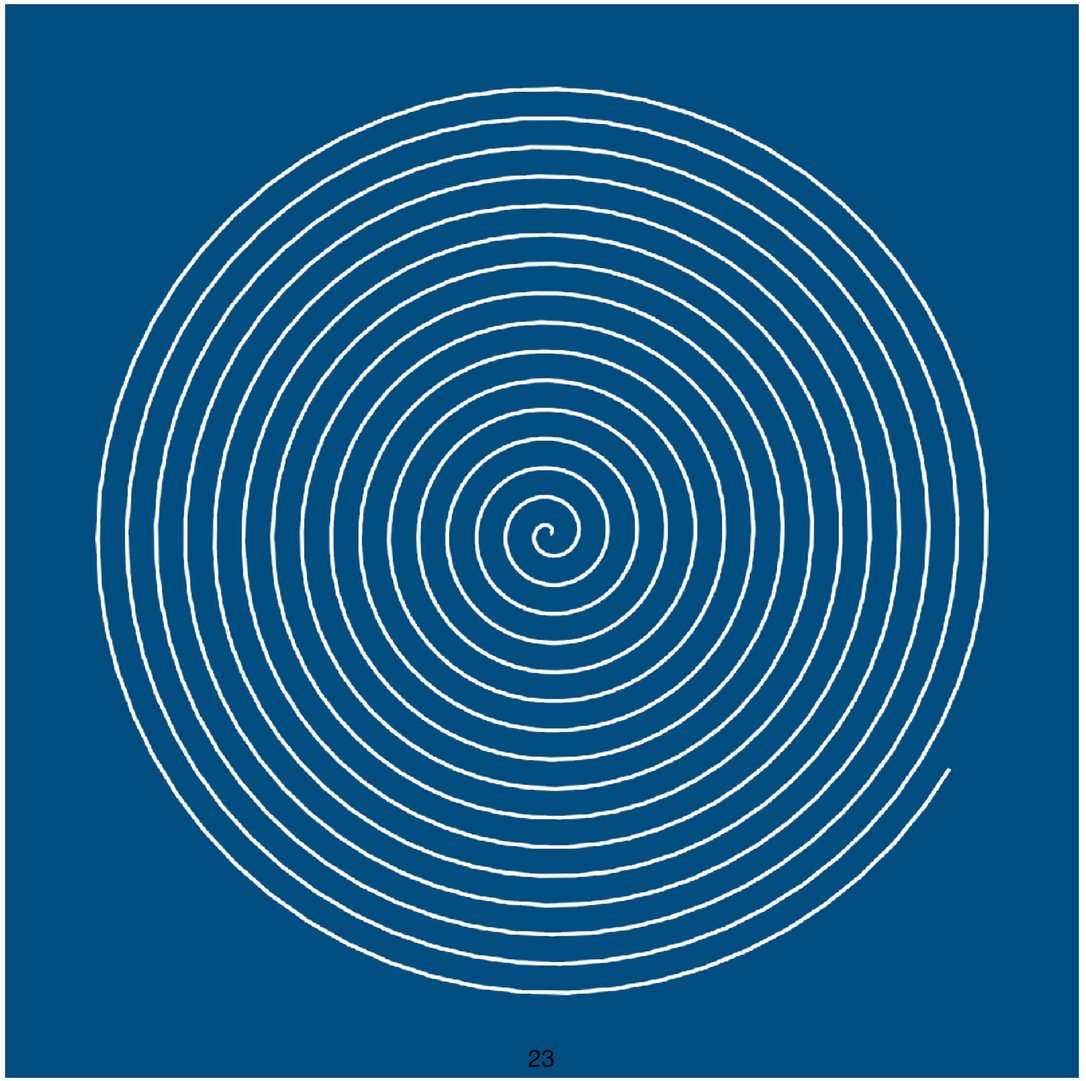
K-space magnitude

Encoding Phase
Imposed on Sample

Real + **Imag**
recorded signal

Where is the “frequency encoding” axis?

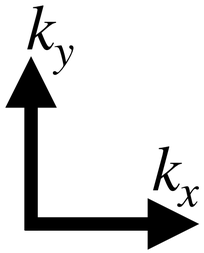




Frequency encoding is not fundamental

- All encoding is phase encoding
- Frequency encoding is essentially a special case of very fast, sequential phase encoding, when constant gradients are used for readout
- Incidentally, gradients are not even needed to readout data, you can make measurements in MRI with no gradients on at all (and therefore no frequency differentiation across any signals in space)
- Gradients are only needed to move us to locations in k-space – once we're there we don't *need* them anymore
- It's only for efficiency that we move and sample simultaneously

How do we determine resolution/FOV?



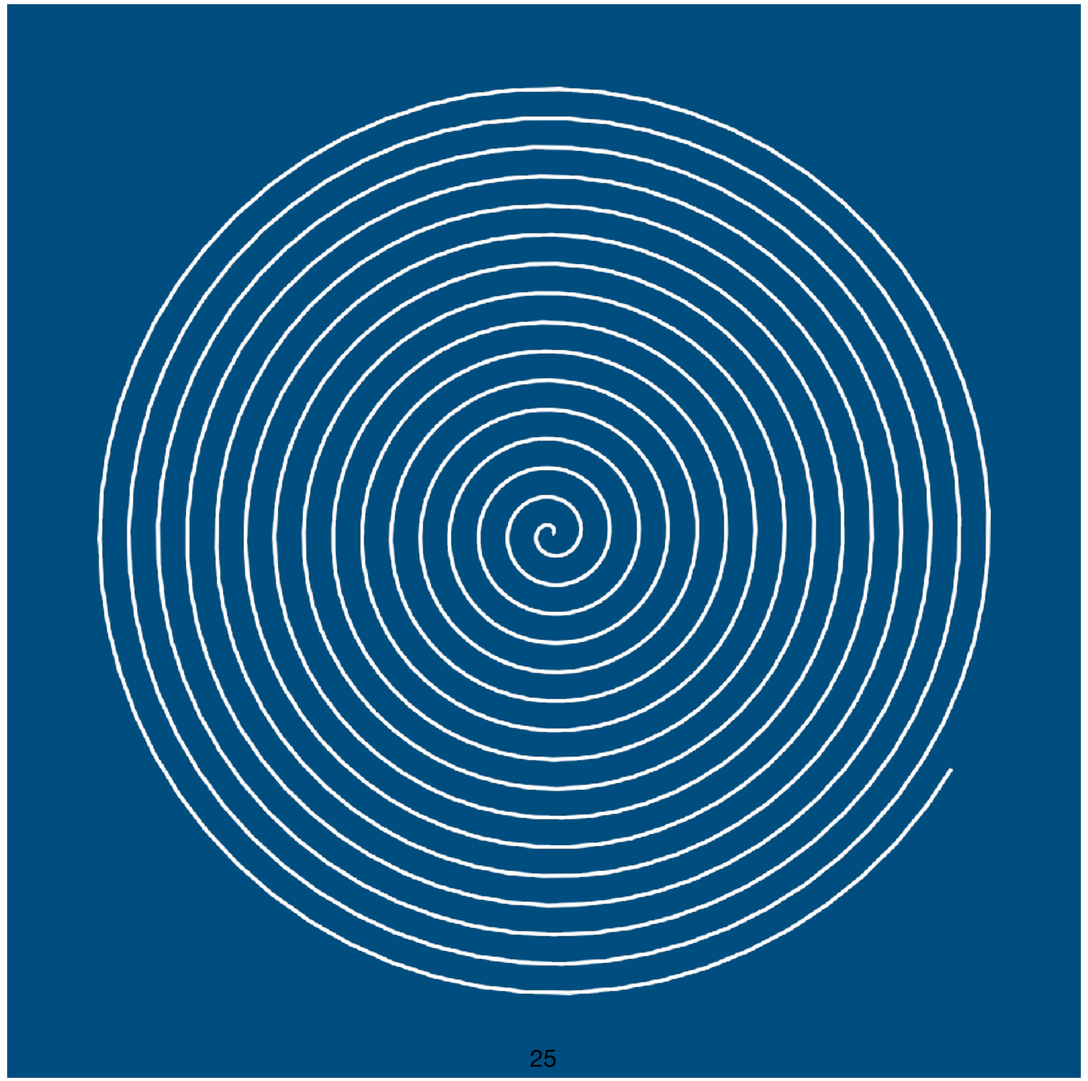


Image properties are determined by the PSF

Point Spread Function

- Non-Cartesian trajectories may not have a consistent Δk_x or Δk_y, so how do you determine the FOV?
- Similarly, these trajectories may not have a consistent $k_{x,max}$ or $k_{y,max}$, so how do you determine the voxel resolution?
- The Cartesian “rules” no longer apply
- Turns out that the simple Cartesian rules are derived from a more general principle:
  - The sampling point spread function

Point Spread Function

Sampling as a function in continuous space

- Recall that the magnetization is essentially a continuous function in real-space $\rho(x)$
- Through the (continuous) Fourier transform, the k-space representation of the magnetization is also a continuous function $\Psi(k) = \mathcal{F}(\rho(x))$
- When we choose to sample our k-space in discrete points, we are multiplying the continuous k-space $\Psi(k)$ by a sampling function $\text{III}(k)$:
- $$\text{III}(k) = \begin{cases} 1 & \text{if } k \text{ sampled} \\ 0 & \text{if } k \text{ not sampled} \end{cases} \quad \text{or} \quad \text{III}(k) = \prod_{i=1}^{N} \delta(k - k_i)$$

Point Spread Function

Fourier Convolution Theorem

- Recall that the Fourier convolution theorem states that a product in one domain is a convolution in the other domain
- So we can consider that the product $\Psi(k) \cdot \mathrm{III}(k)$ becomes $\rho(x) \circledast \mathrm{PSF}(x)$ for some function $\mathrm{PSF}(x) = \mathscr{F}^{-1}(\mathrm{III}(k))$
- Then we can analyze the impact of sampling choices by looking at the effect of the convolution of the $\mathrm{PSF}(x)$ with our underlying magnetization
  - The PSF does not depend at all on the magnetization, only on the sampling!

Point Spread Function

An impulse response function

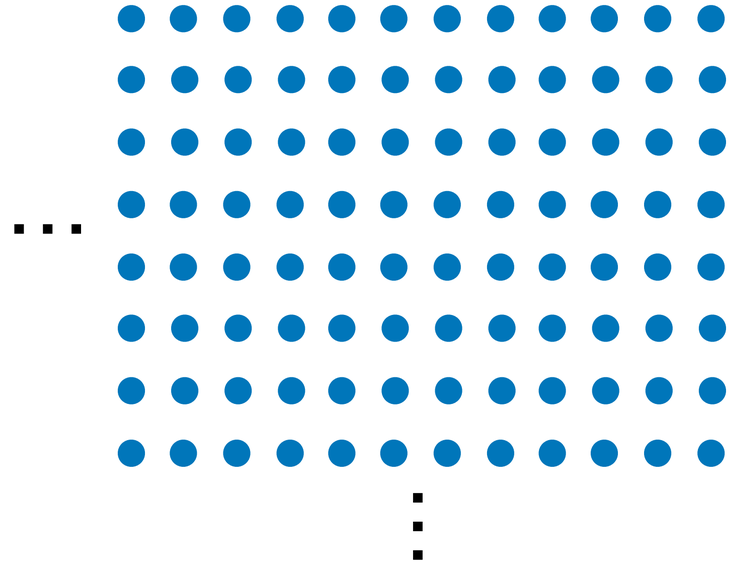
- One reason we call this a point-spread function (PSF) is because it describes the impact of sampling on a point source
- Imagine a point source $\rho(x) = \delta(x)$ located at the origin, zero everywhere else, with $\Psi(k) = 1$
- So $\Psi(k) \cdot \mathrm{III}(k) = \mathrm{III}(k)$, and therefore looking at $\mathrm{III}(k)$ alone is equivalent to looking at the impact of $\mathrm{III}(k)$ on a point source
- Then $\mathrm{PSF}(x) = \mathscr{F}^{-1}\,\mathrm{III}\,\mathscr{F}\delta(x)$
- More generally, the Fourier convolution theorem is related to linear shift-invariant systems, and the entire impact of sampling and conventional reconstruction is determined by the impulse response function, which is our PSF

Point Spread Function

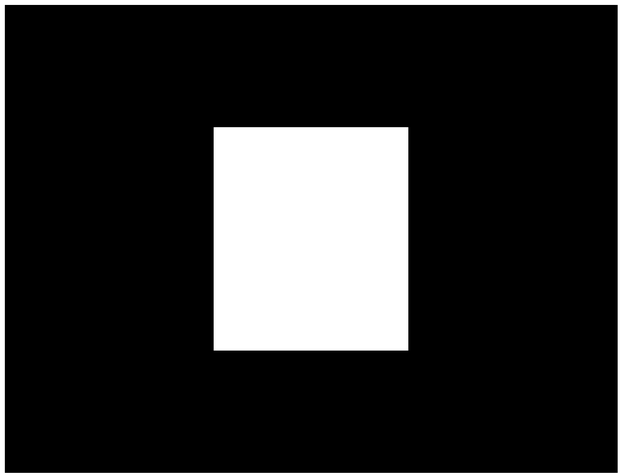
Cartesian PSF

- In the Cartesian case, we can analytically solve for the PSF
  - This is not true in general - for arbitrary sampling, the PSF needs to be estimated numerically

Infinite Dirac Comb Function



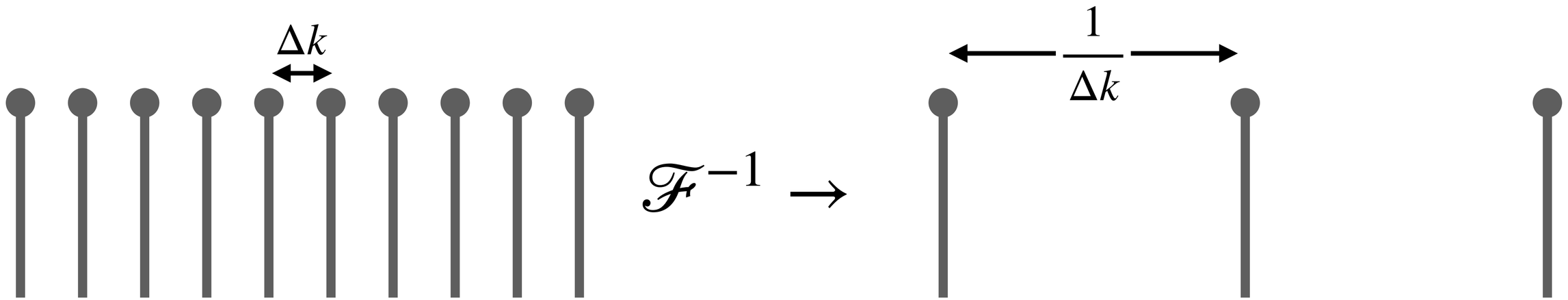
Rectangular Window Function



Point Spread Function

Cartesian PSF

- For simplicity, let's consider the 1D case, and each term 1 at a time
- The Fourier transform of a train of delta functions is another train of delta functions:
- $$\mathcal{F}^{-1}\left(\sum_n \delta(k - n\Delta k)\right) = \sum_n \delta(x - \frac{n}{\Delta k})$$

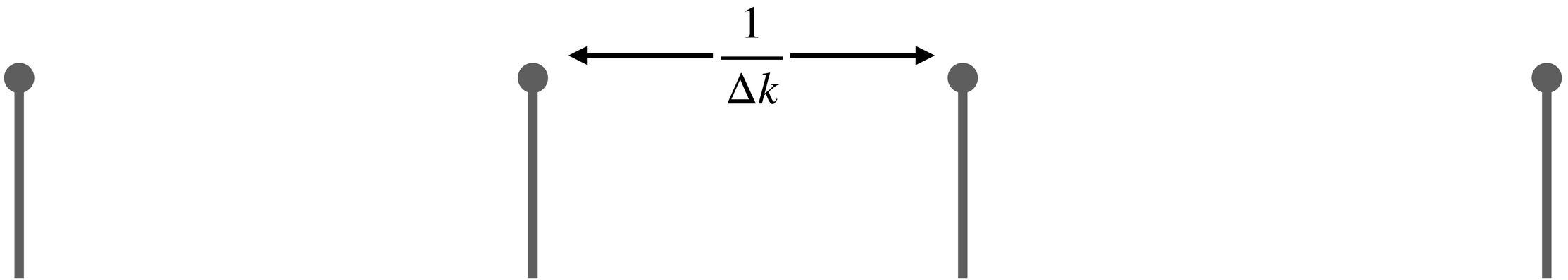


Point Spread Function

Cartesian PSF

Object to be imaged

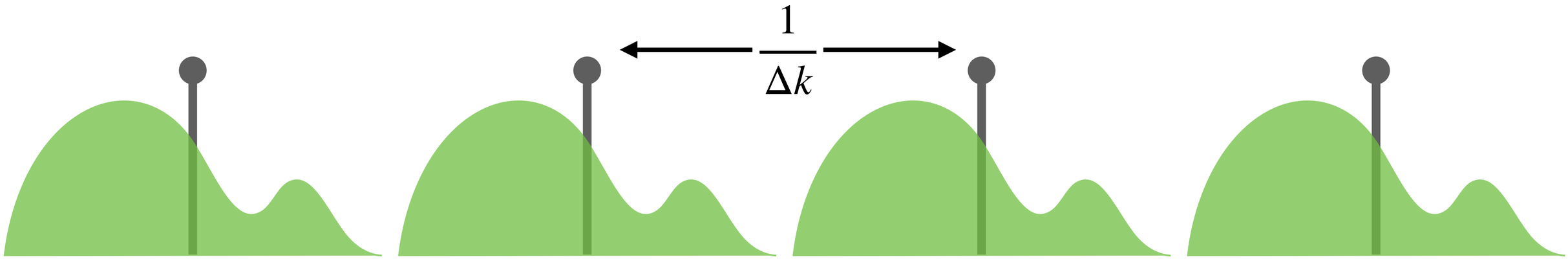
Multiplying with the sampling function in k-space is equivalent to convolving with the inverse Fourier transform of the sampling function in image space



Point Spread Function

Cartesian PSF

Object to be imaged



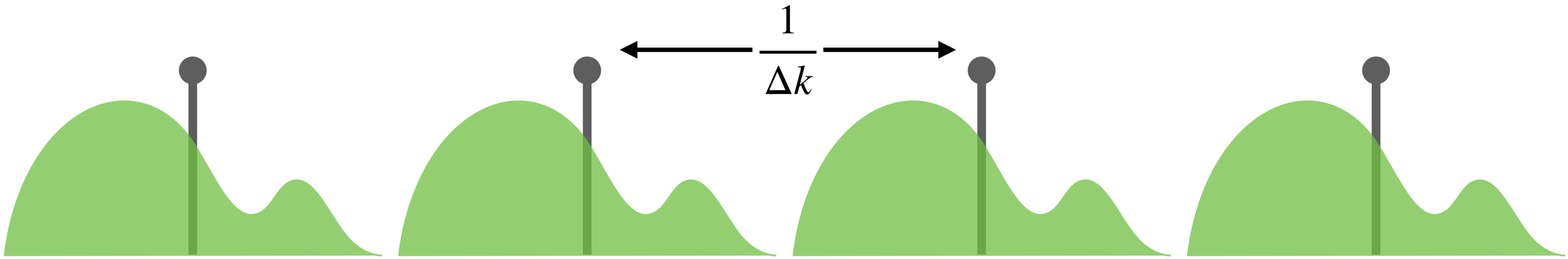
Point Spread Function

Cartesian PSF

The parameter $\frac{1}{\Delta k}$ is what controls the spacing of these copies

It also corresponds to the maximum width of the object before it overlaps with itself (aliasing)

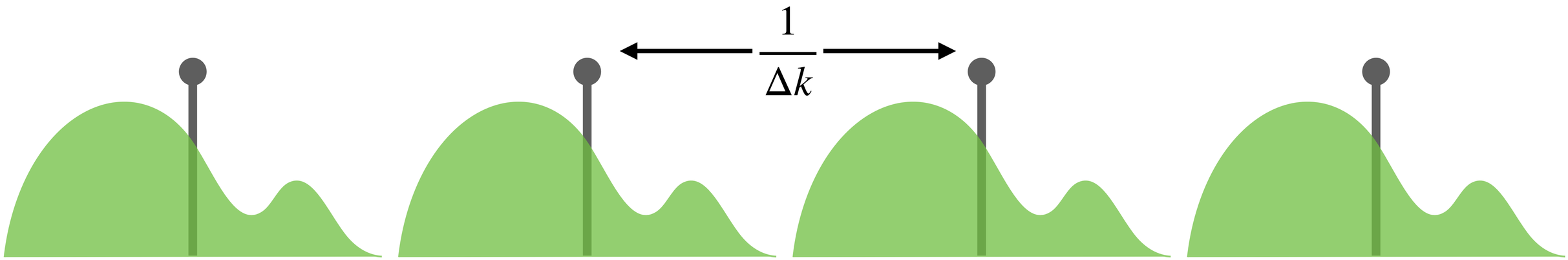
Equivalently, it's the largest window containing unique image information



Point Spread Function

Cartesian PSF

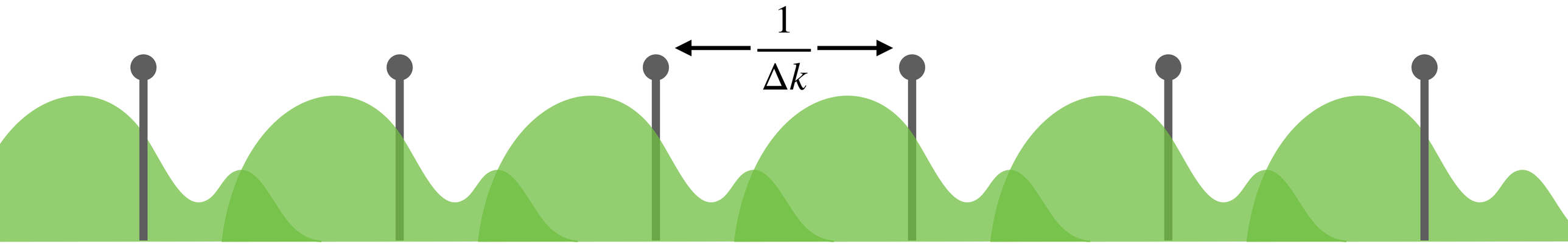
This is why we define the image $FOV = \frac{1}{\Delta k}$



Point Spread Function

Cartesian PSF

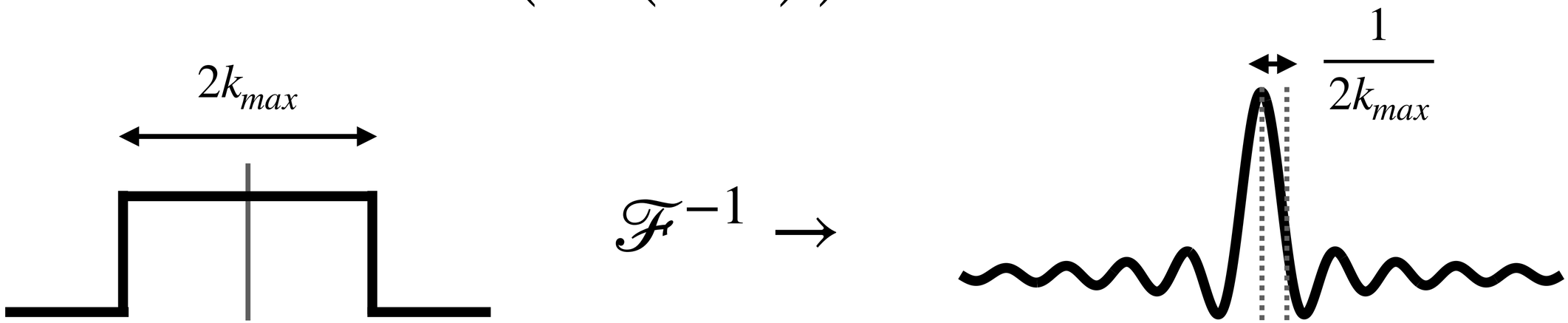
If Δk is too large, and $\frac{1}{\Delta k}$ too small, then the (infinite) copies of the object in discrete image space will overlap



Point Spread Function

Cartesian PSF

- The second term is the rect function, defining the window of sampling
- The Fourier transform of a rect function is a sinc:
- $$\mathcal{F}^{-1}\left(\text{rect}\left(\frac{k}{2k_{max}}\right)\right) = \text{sinc}(2k_{max}x)$$

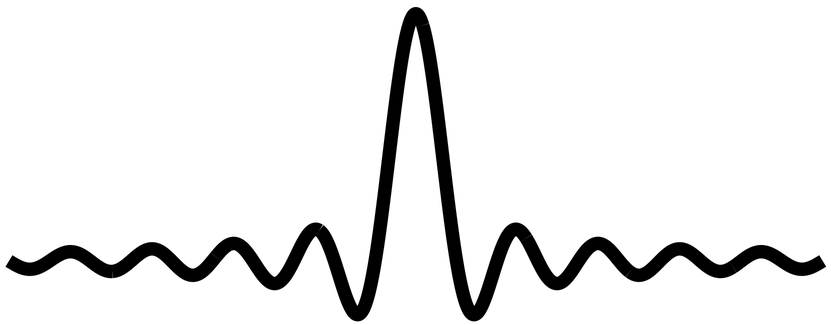


Point Spread Function

Cartesian PSF

Object to be imaged

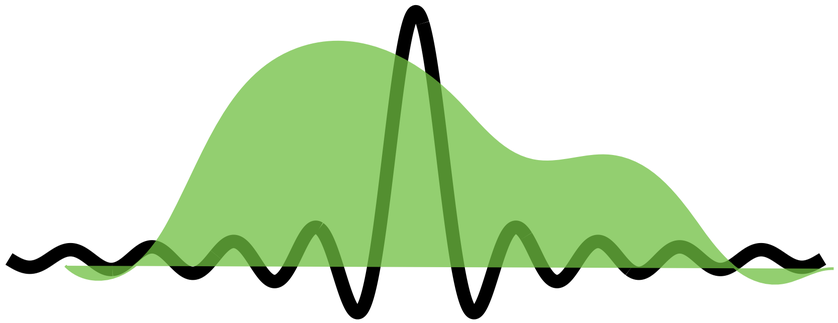
Multiplying with the rectangular window function in k-space is equivalent to convolving with the sinc



Point Spread Function

Cartesian PSF

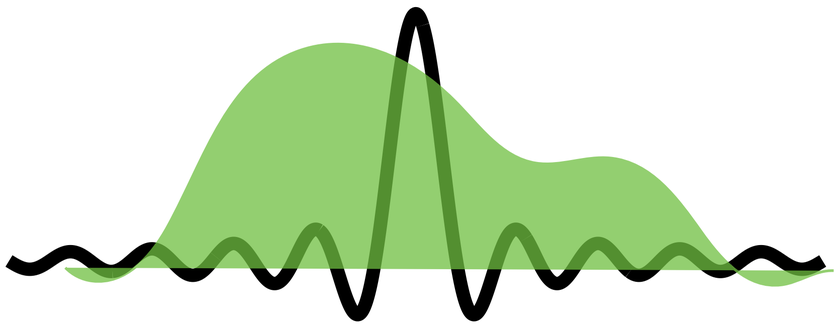
- The width of the rect function is inversely proportional to the width of the sinc
- Since $2k_{max}$ is the width of a symmetric k-space window, we then define the image resolution as $\Delta x = \frac{1}{2k_{max}}$
- This corresponds to the distance to the first zero-crossing of the sinc



Point Spread Function

Cartesian PSF

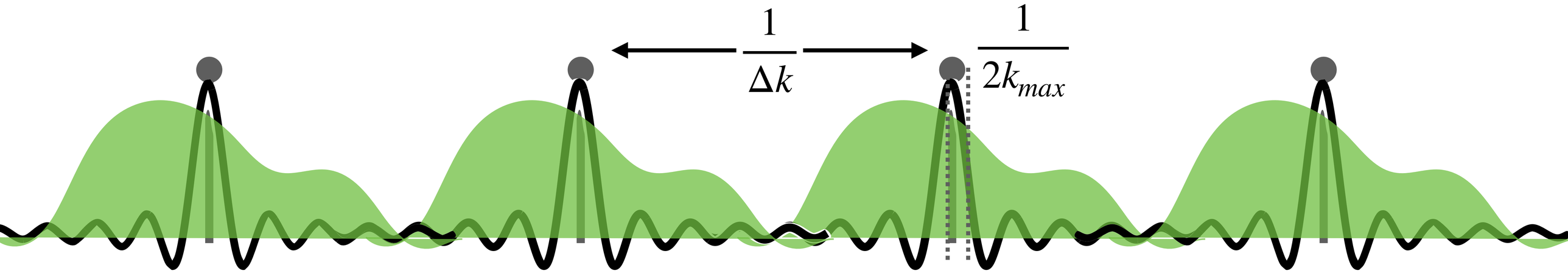
- This definition is a *choice*, but in reality the voxel shape is defined by the shape of the PSF
- The voxel is not a rectangular box of signal with well defined edges
- The value you associate with your image voxels is the integral of this PSF over all space centred on the voxel location of interest



Point Spread Function

Cartesian PSF

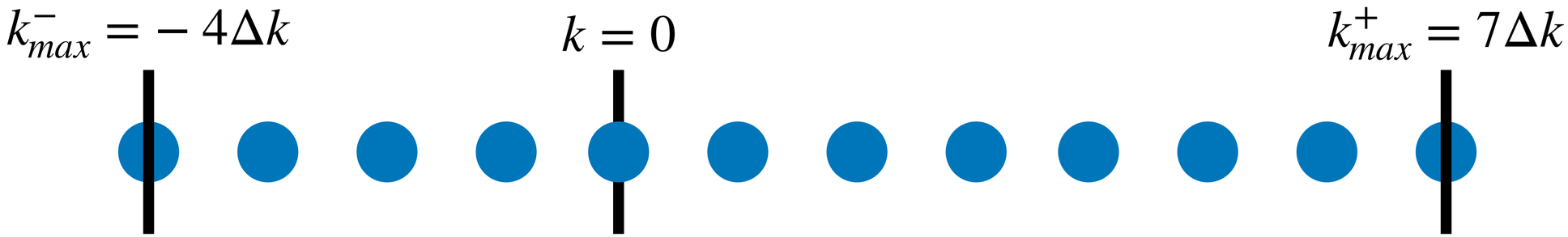
- The combined effect of both the sampling and the windowing in k-space are chained together to get the net effect of the aliasing and blurring



Point Spread Function

Deriving image properties from the PSF

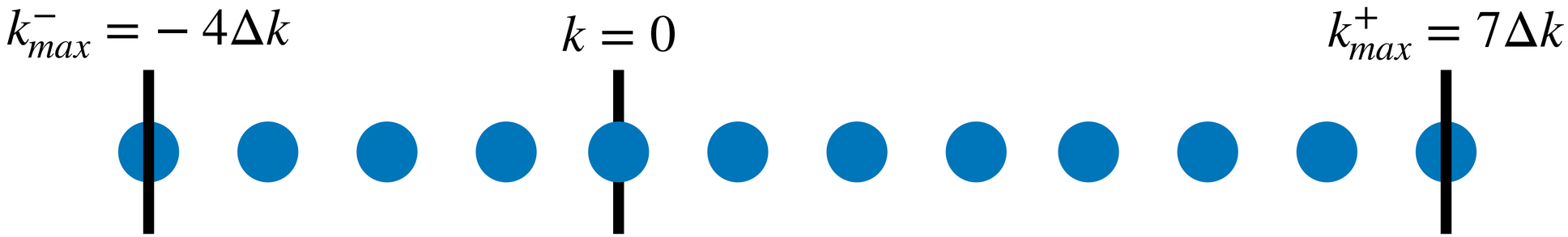
- So now we understand that the FOV and resolution are properties of the PSF
- In the special case of symmetric Cartesian sampling, we can associate these with the sampling parameters Δk and k_{max}, but in the general case we need to estimate resolution and FOV from the PSF itself
- Consider a slightly different type of cartesian sampling, such that the sampling window in k-space is not symmetric



Point Spread Function

Deriving image properties from the PSF

- What is the resolution of an image sampled like this, if there is no consistent k_{max}?
  - This type of sampling is common in Partial Fourier acquisitions
- The simple “rules” cannot give you the answer, but the PSF can
- The resolution is actually dictated by the *width* of the sampling window

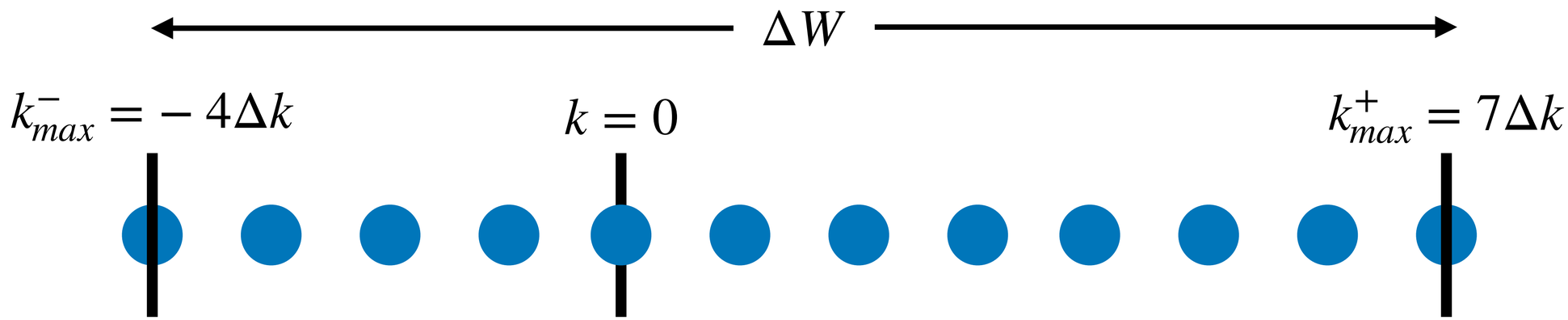


Point Spread Function

Deriving image properties from the PSF

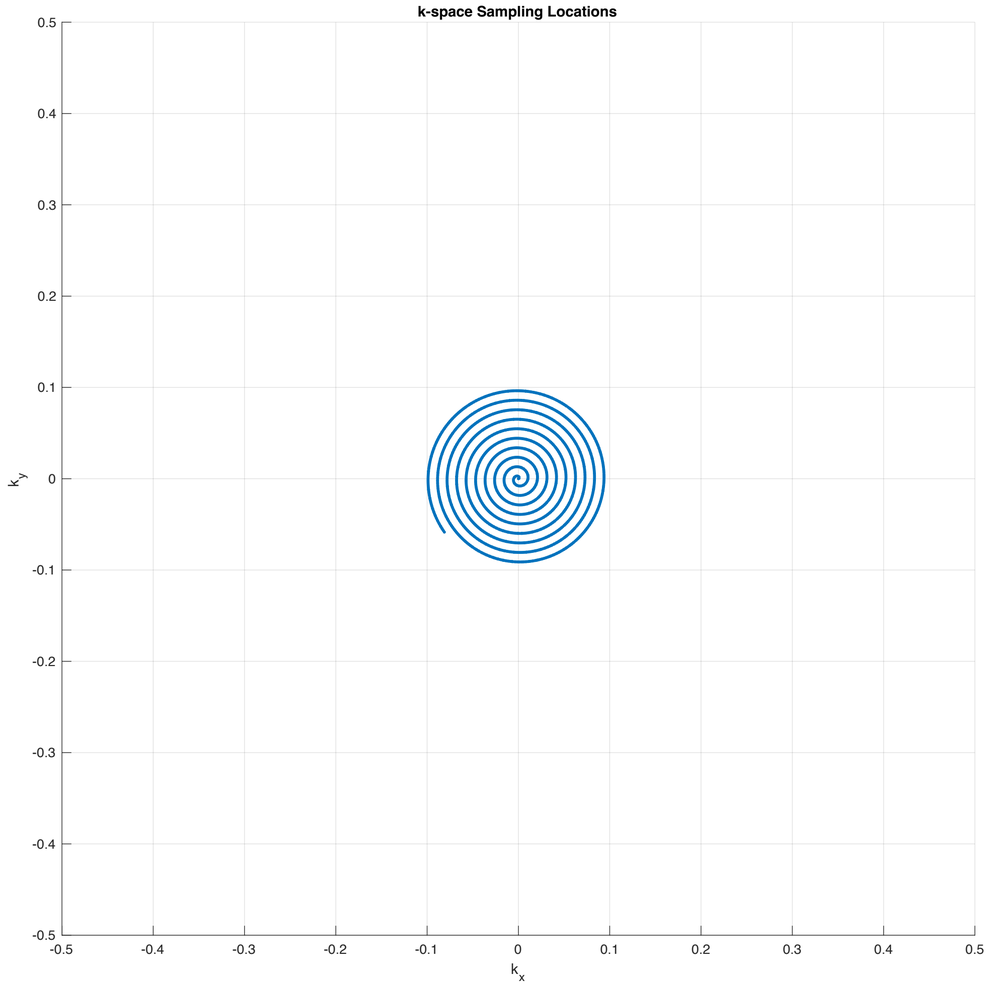
- When the sampling is symmetric, the width = $2k_{max}$, which is where $\Delta x = \frac{1}{2k_{max}}$ comes from

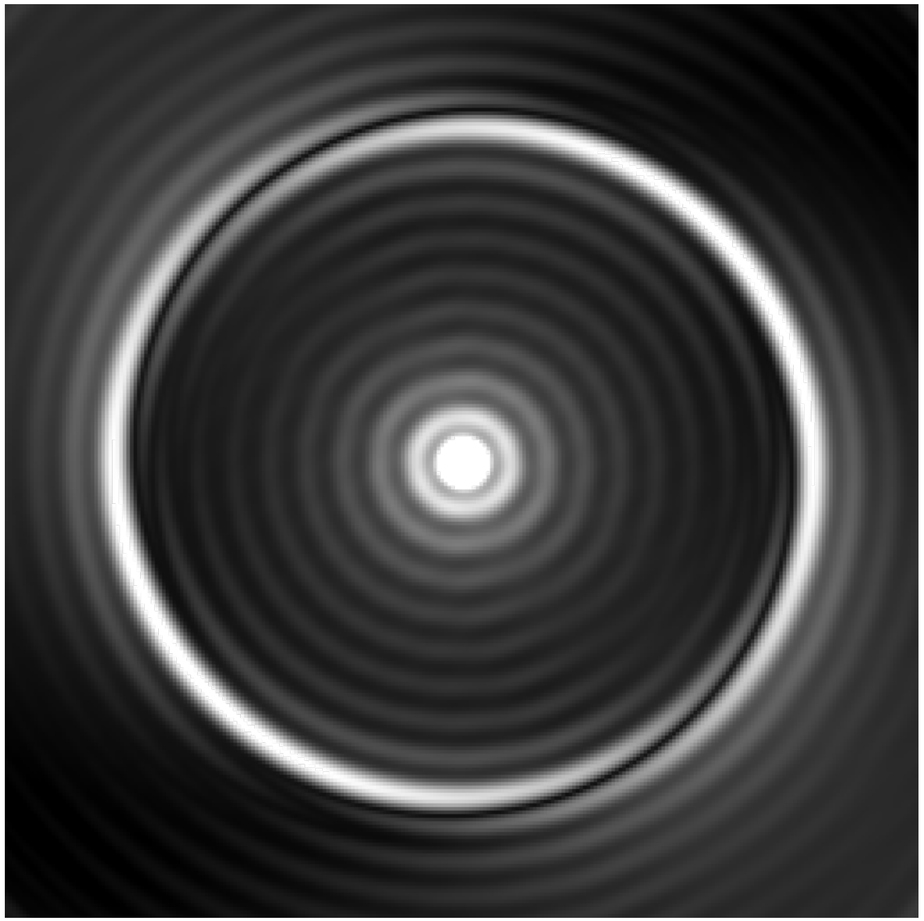
- When the sampling is not symmetric, $\Delta x = \frac{1}{\Delta W}$



Point Spread Function

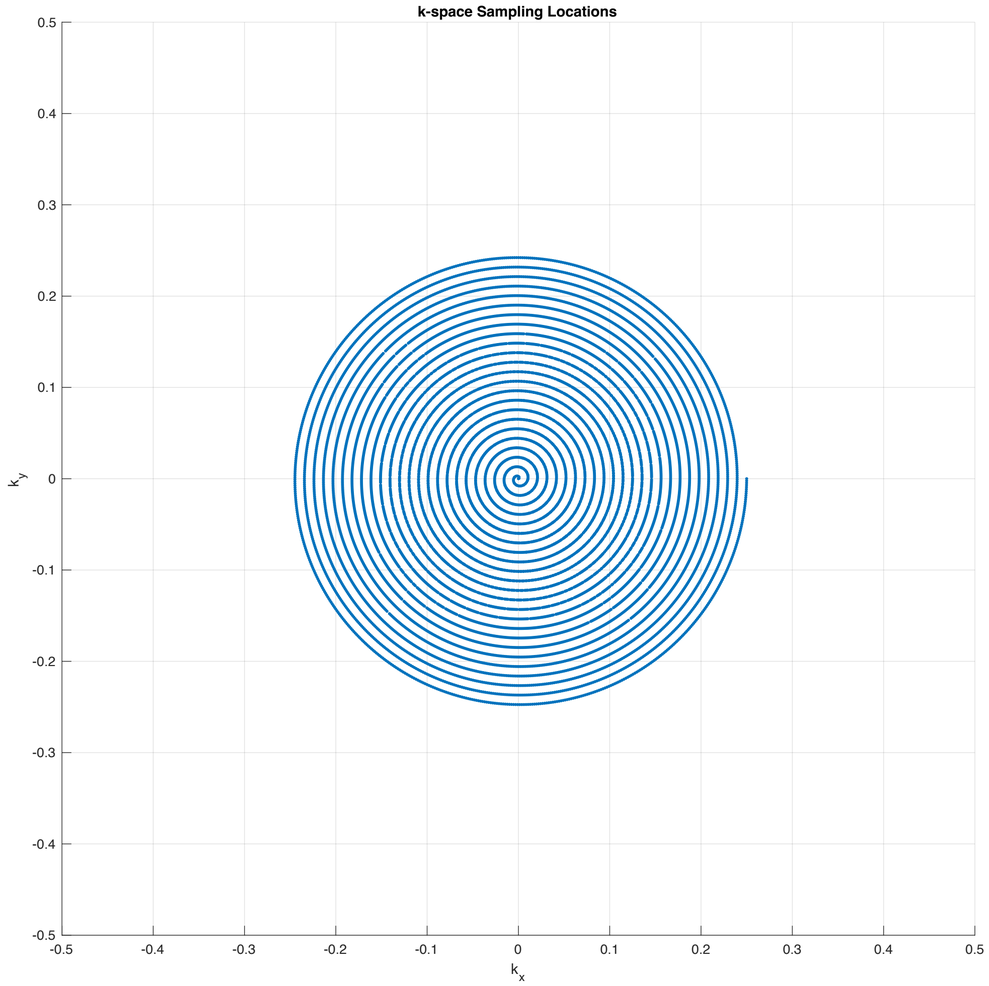
Spiral

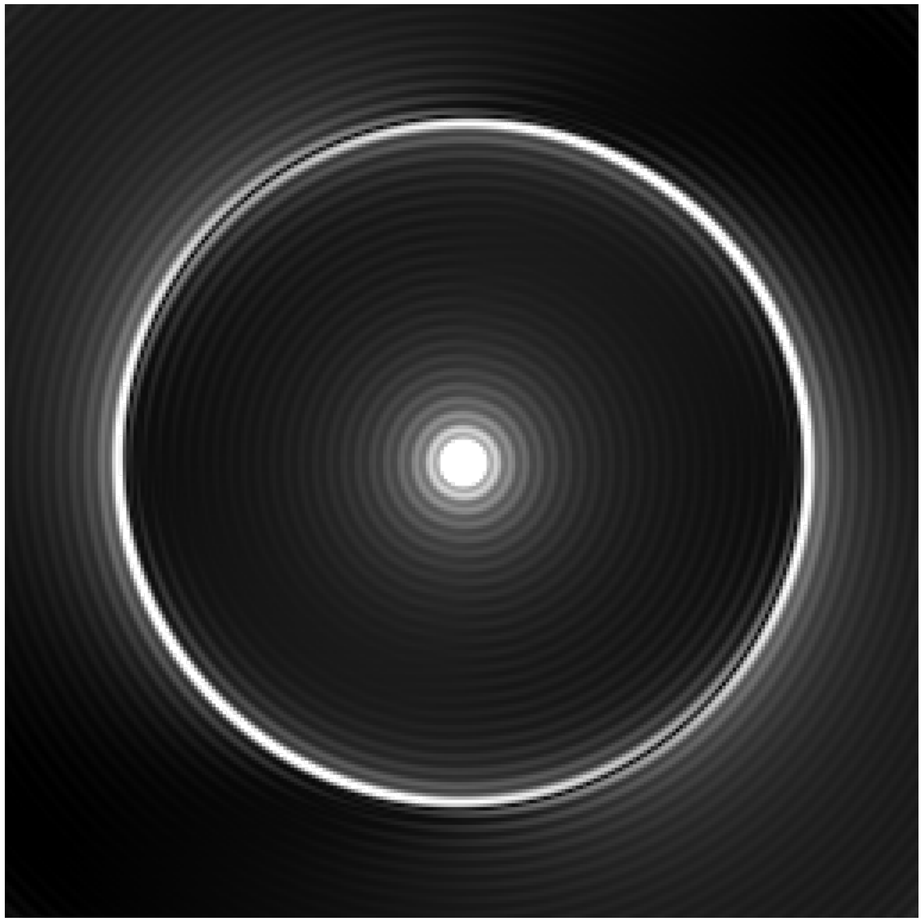




Point Spread Function

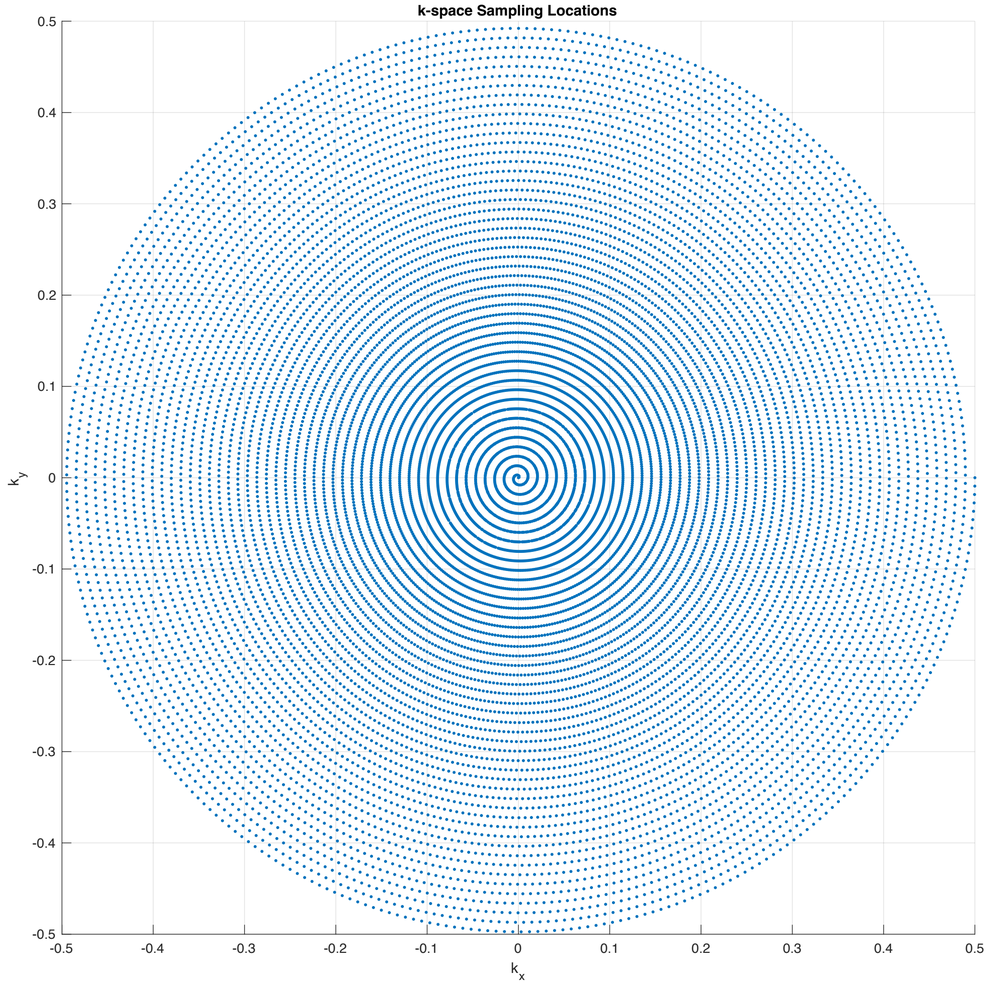
Spiral

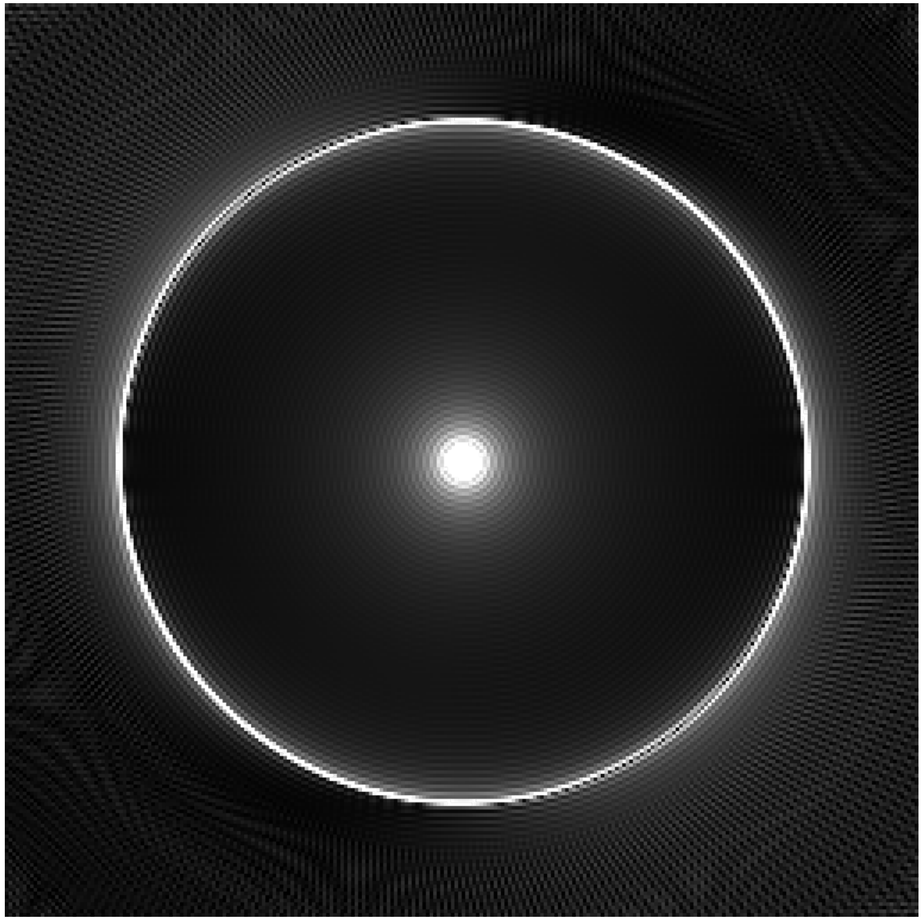




Point Spread Function

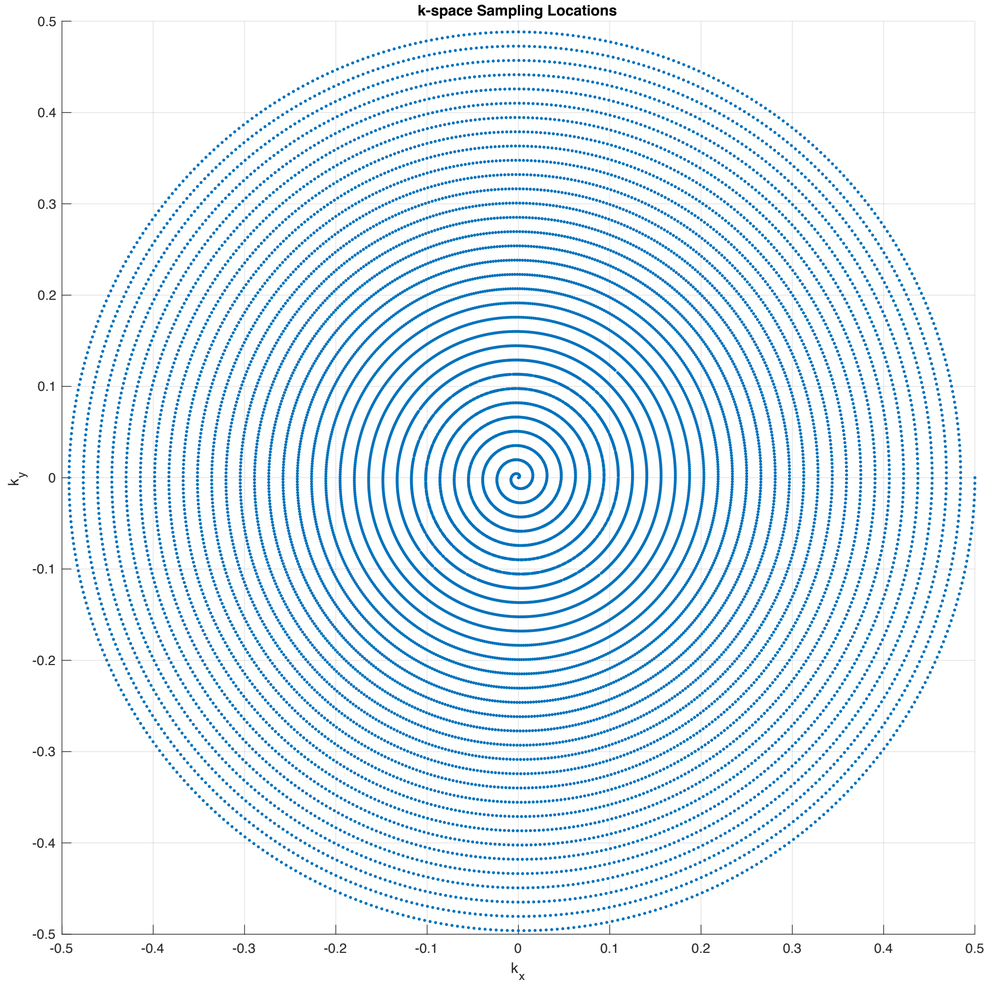
Spiral

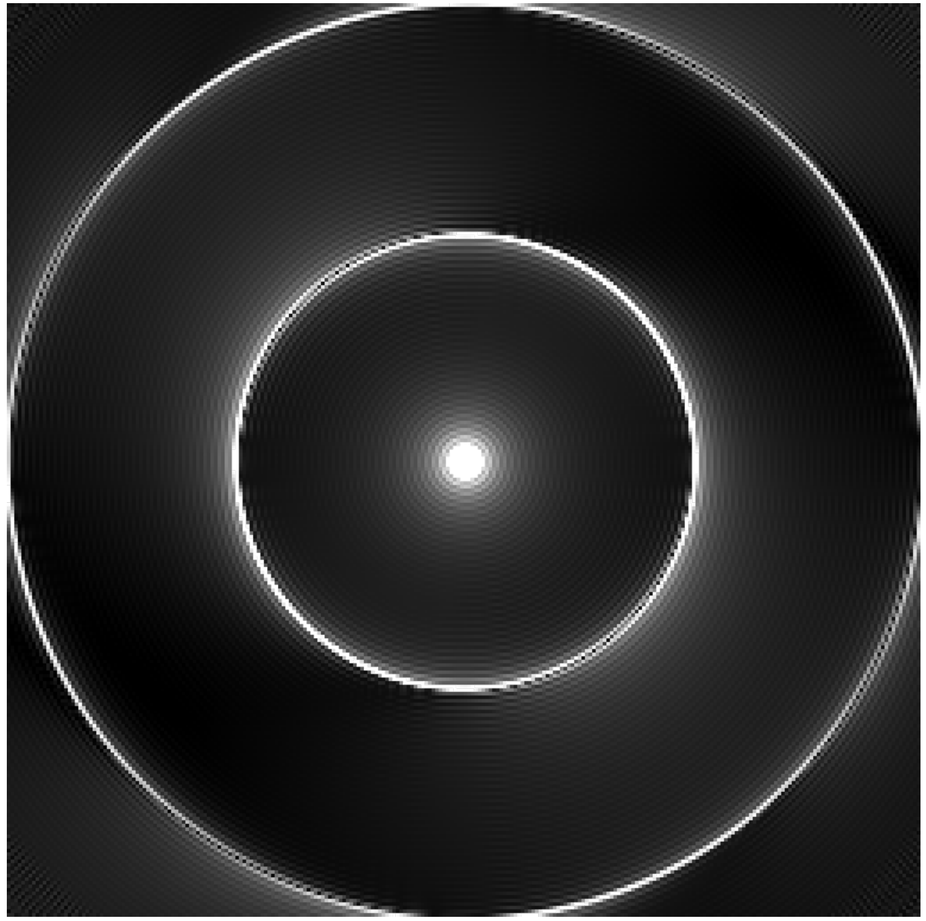




Point Spread Function

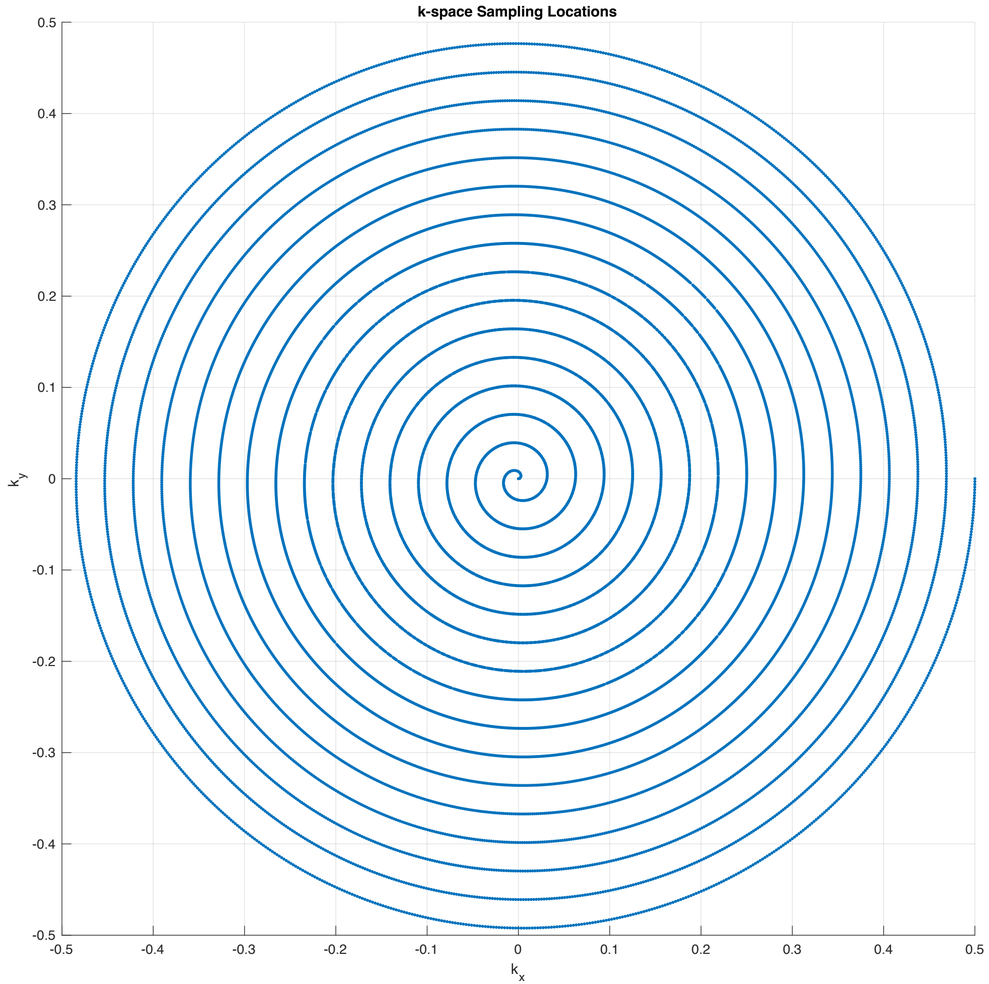
Spiral

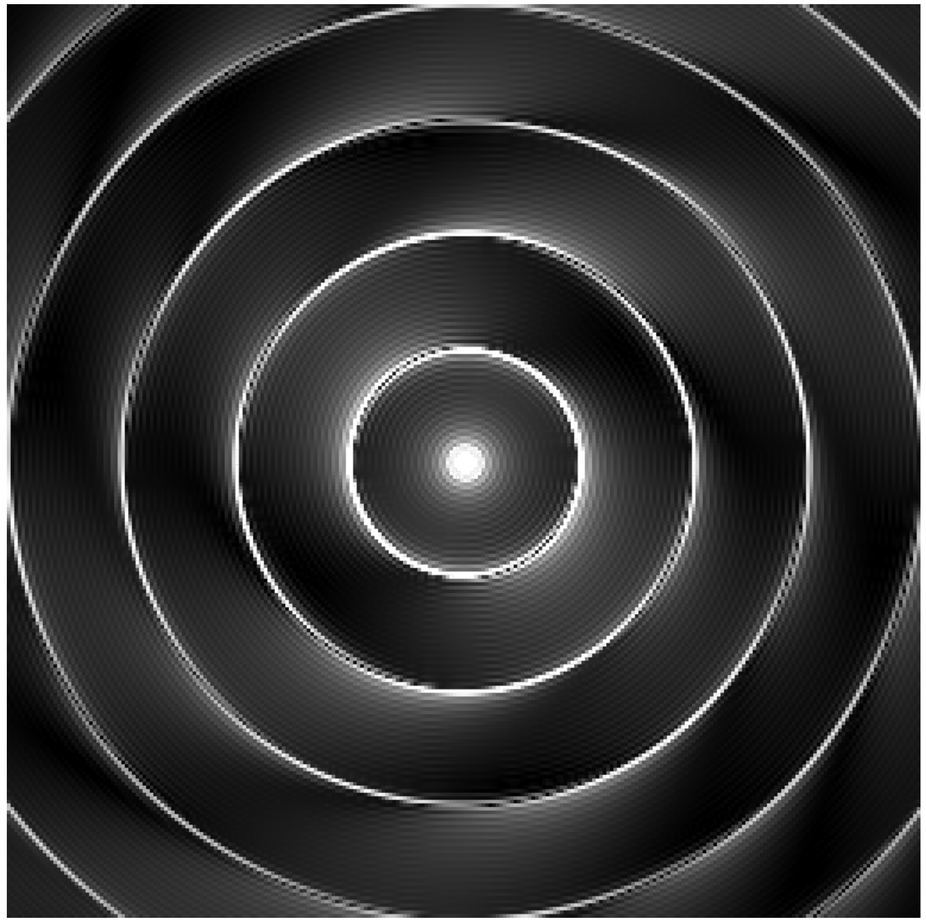




Point Spread Function

Spiral

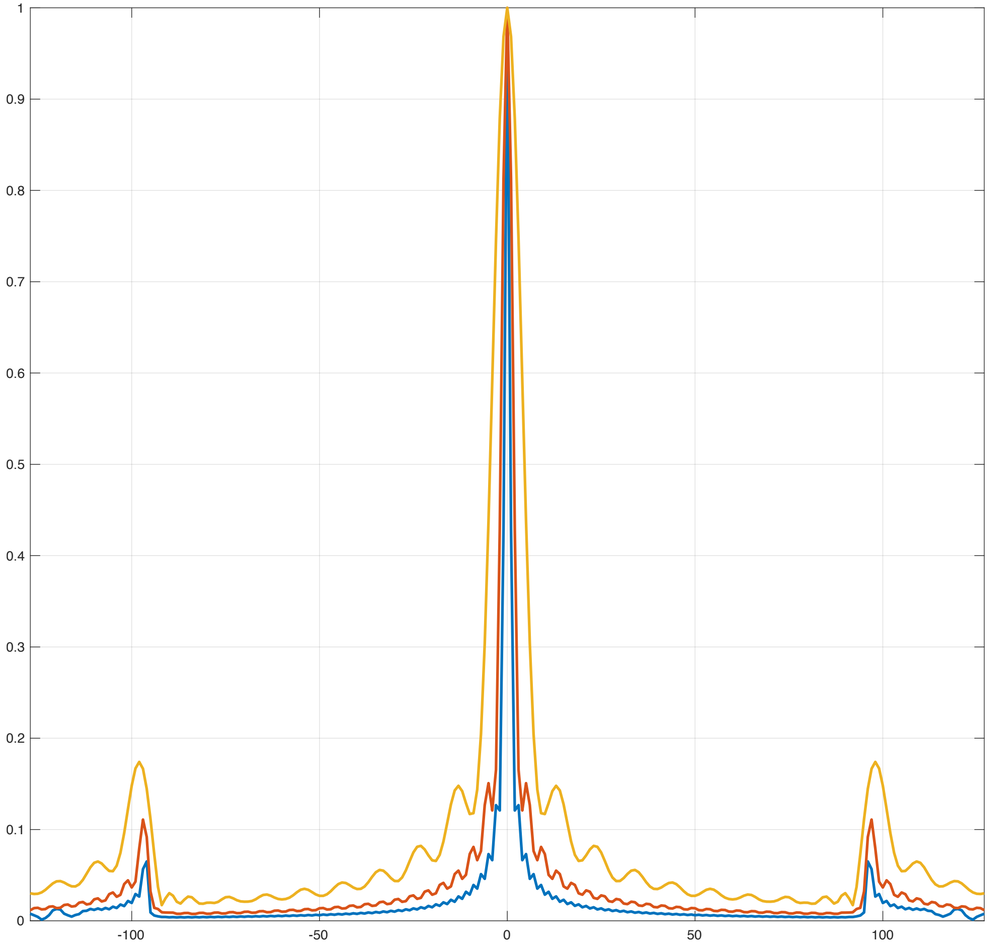


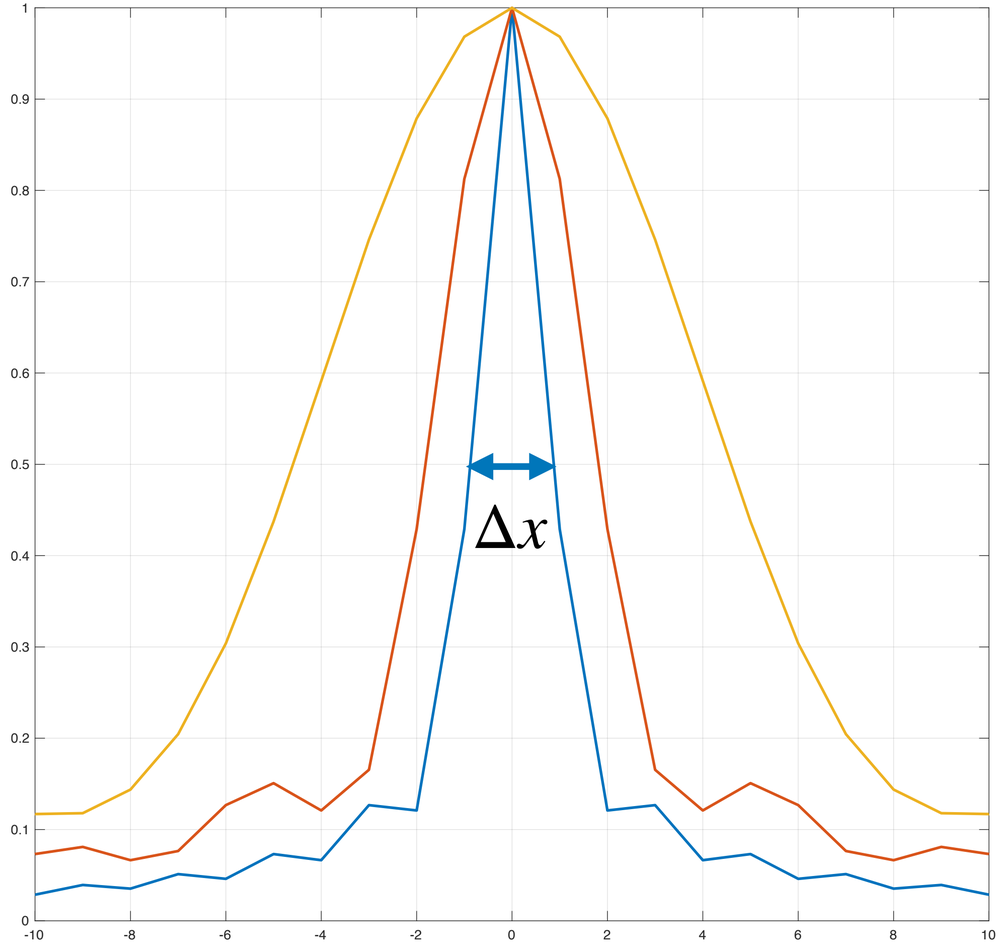


Point Spread Function

Spiral - resolution

1x
0.5x
0.2x

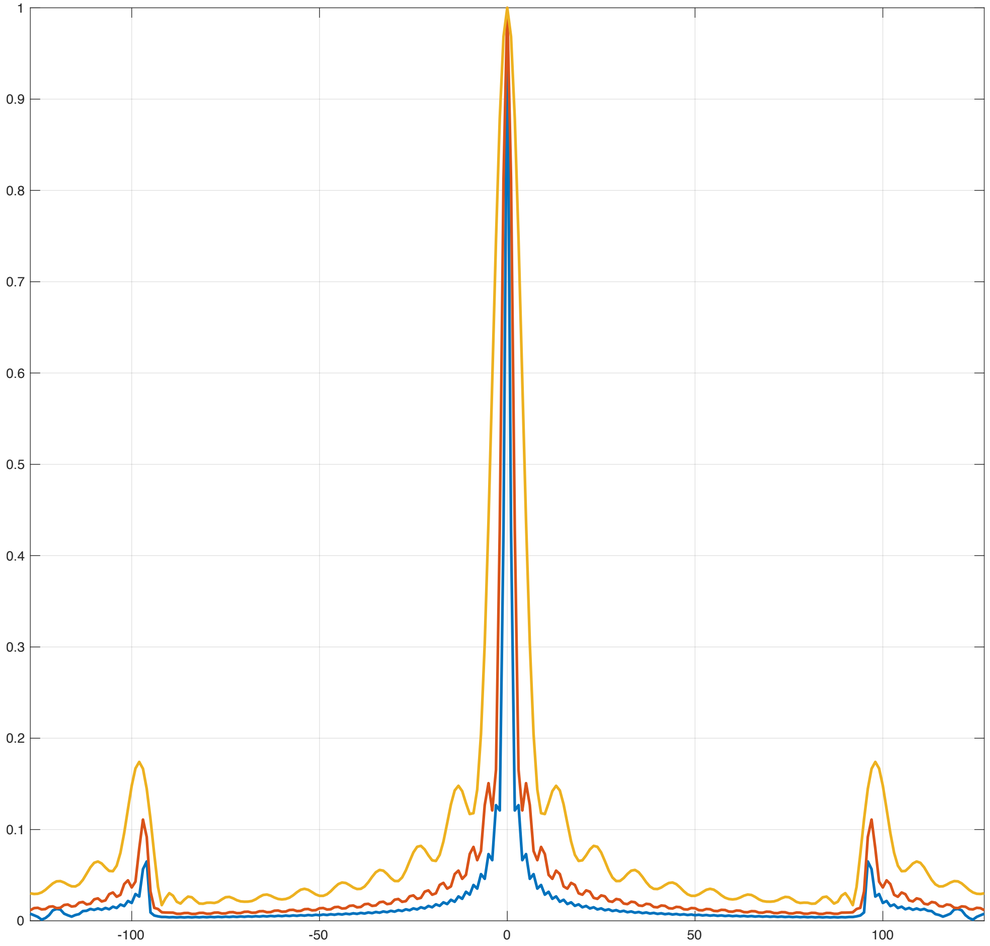


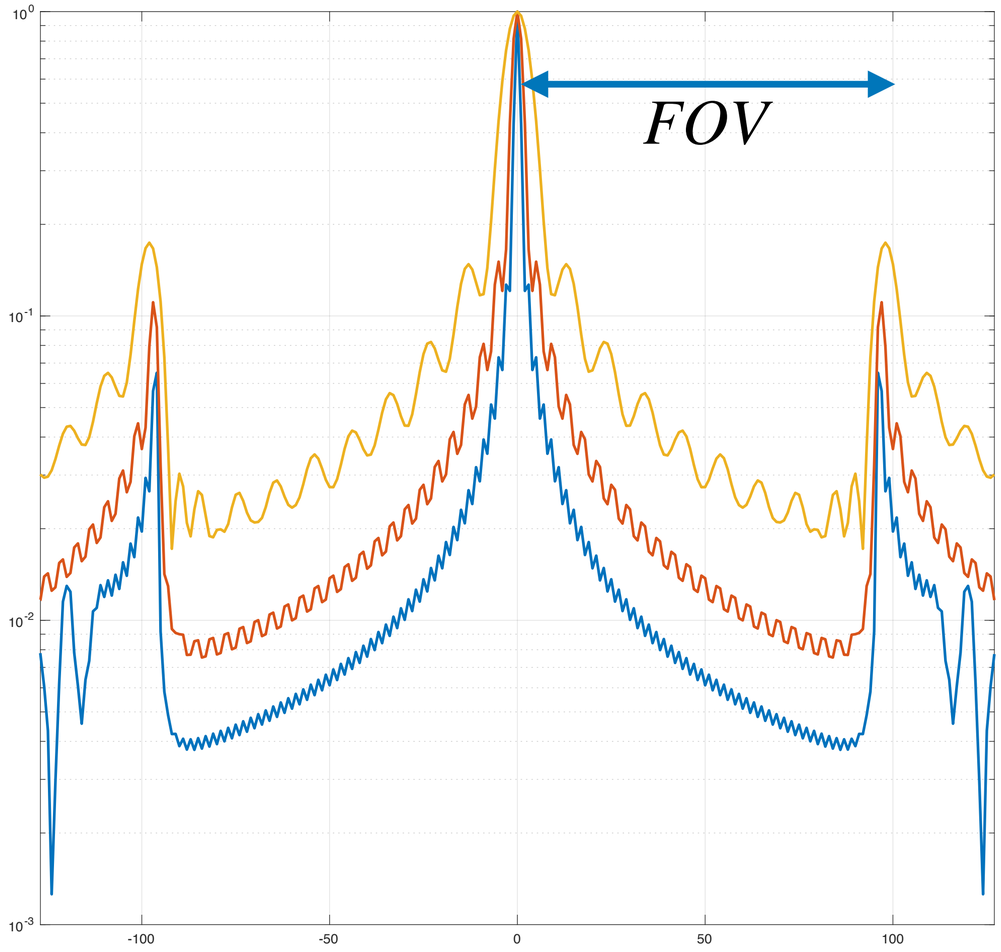


Point Spread Function

Spiral - FOV

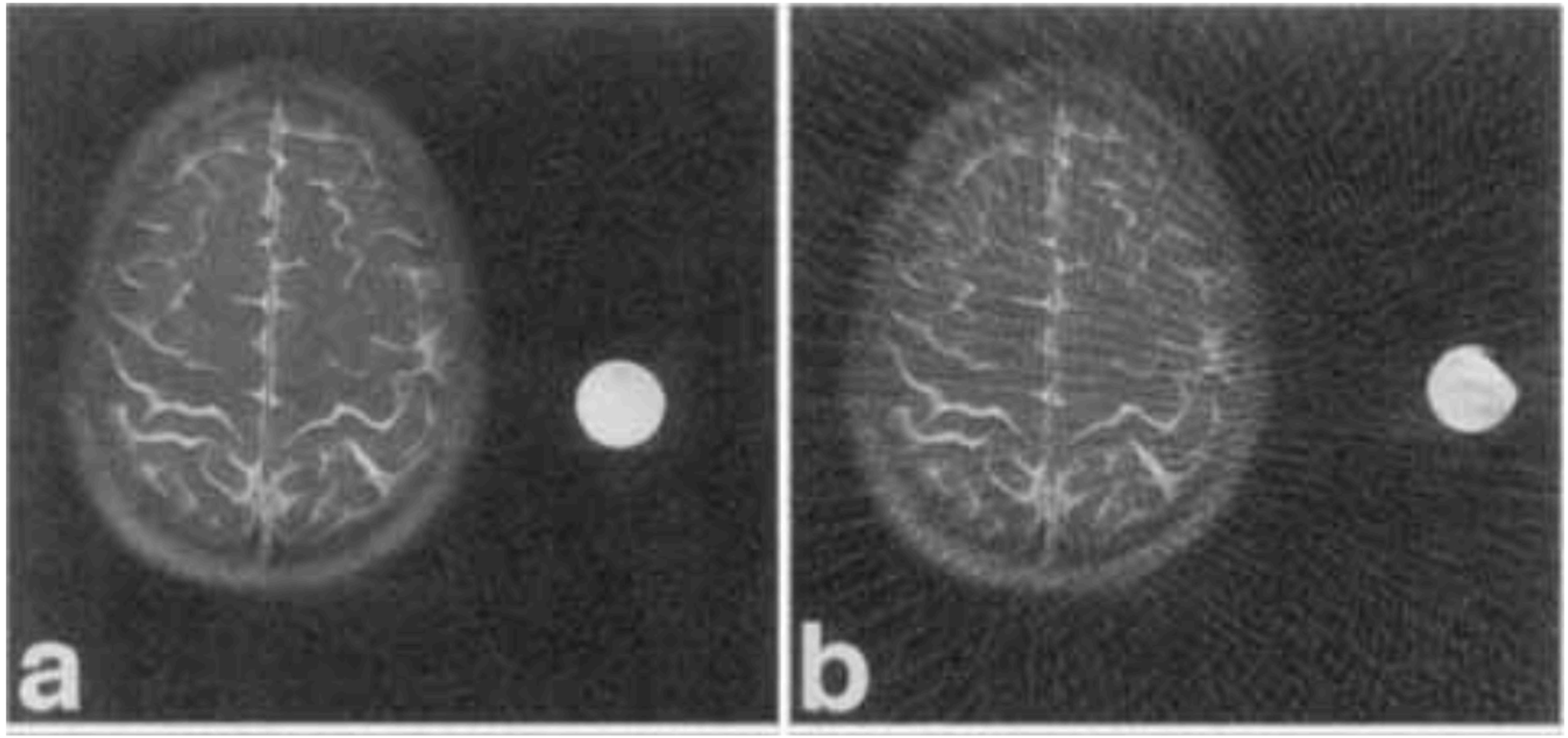
1x
0.5x
0.1x





Point Spread Function

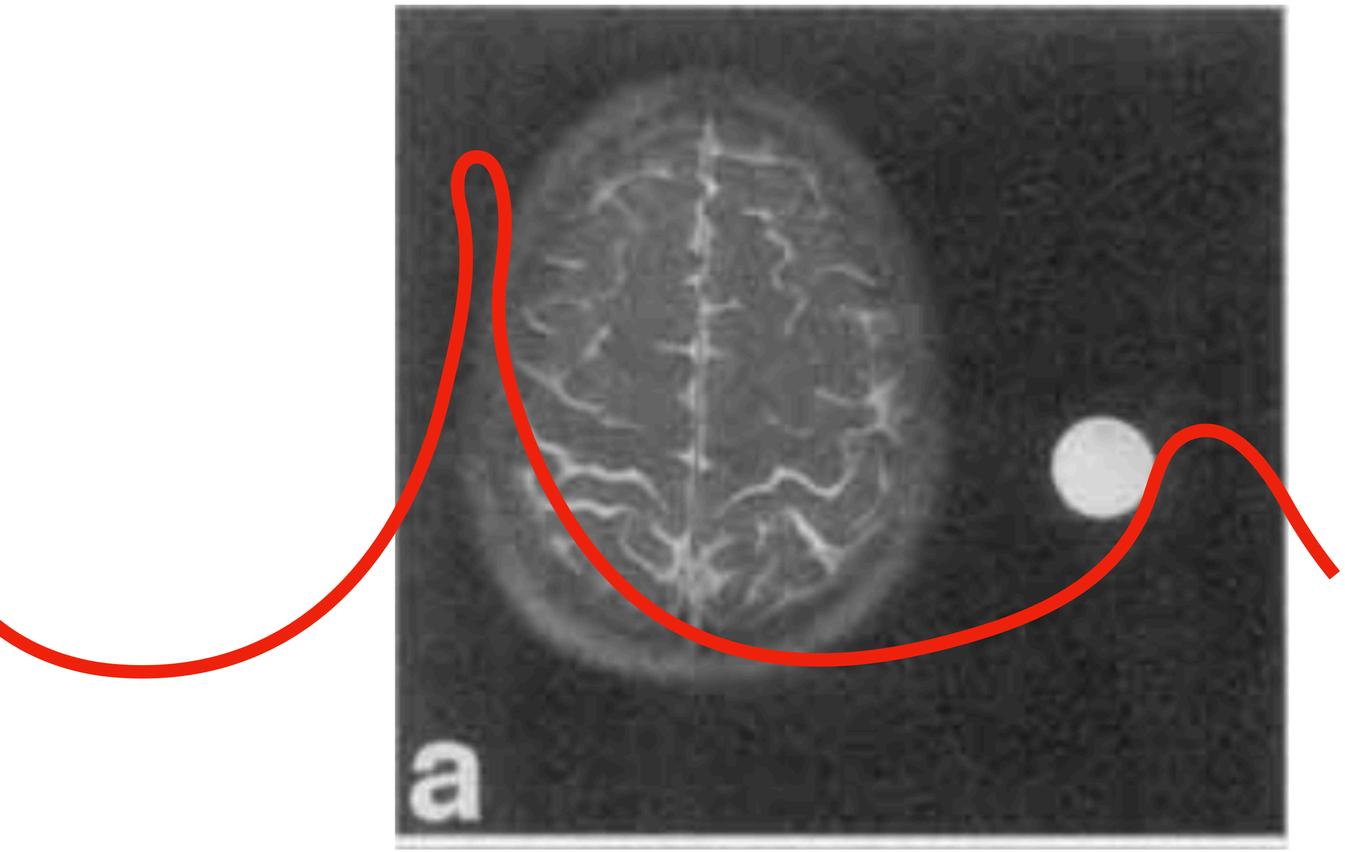
Spiral - FOV



Scheffler & Hennig MRM 1998

Point Spread Function

Spiral - FOV



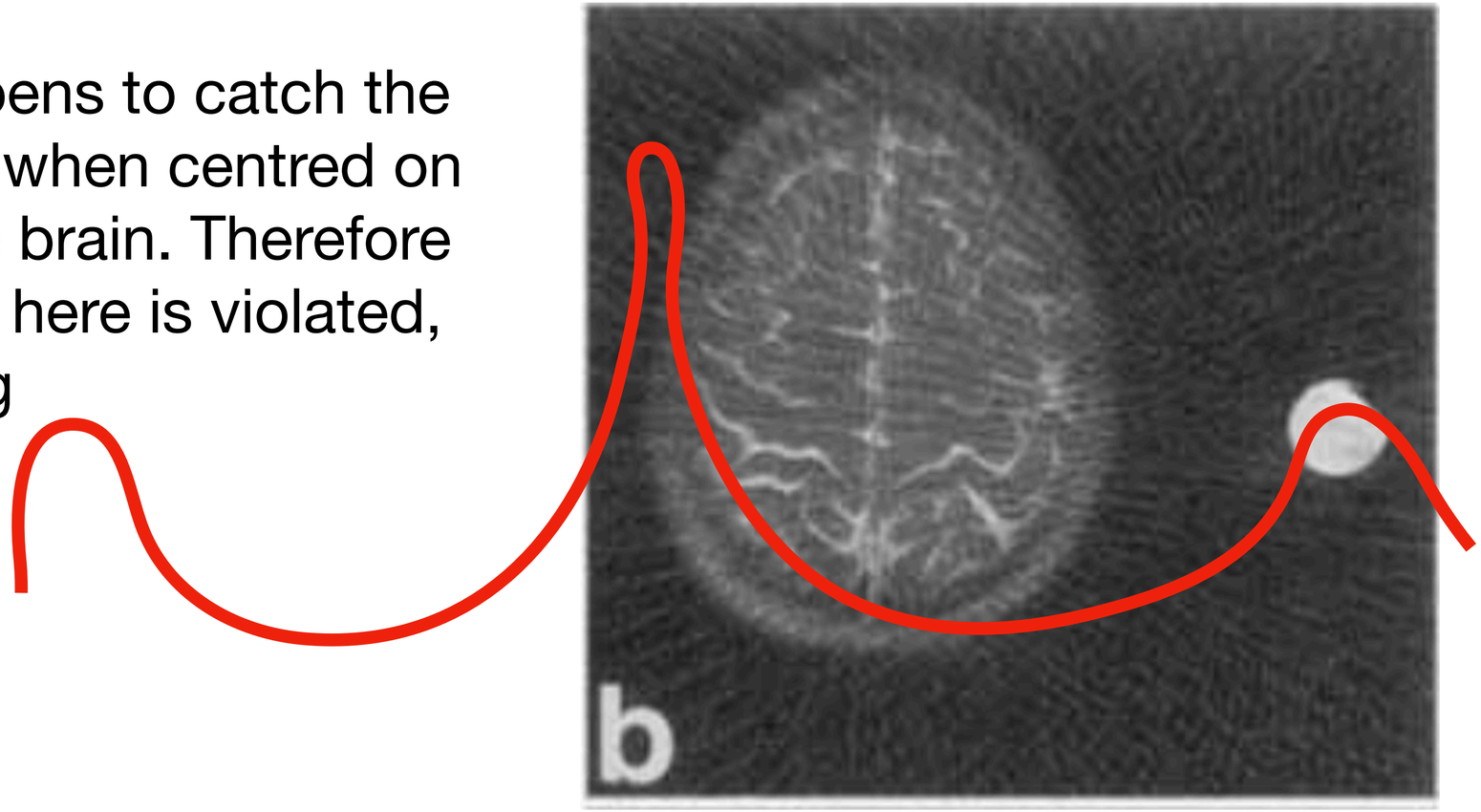
Point spread function doesn’t catch any extra stuff. It can be centred anywhere in the image and the sidelobe doesn’t see anything

Scheffler & Hennig MRM 1998

Point Spread Function

Spiral - FOV

Here the PSF happens to catch the object on the right when centred on the left edge of the brain. Therefore the maximum FOV here is violated, and we get aliasing



Scheffler & Hennig MRM 1998

PSF

Resolution and FOV

- Resolution can be thought of as related to the width or FWHM of the main lobe of the PSF
  - Think of this like convolution with a blurring function, the wider the blurring kernel, the less resolved the image is
  - Resolution is *not* the width of some rectangular voxel profile
- The FOV is the maximum window in image space that can be resolved without aliasing
  - This of this like convolution with side-lobes - if the object is wider than the distance from main lobe to side lobes, then those signals will be mixed (aliased)
  - FOV is not the size of your image – this is a choice you make at reconstruction time, which is often chosen to be equal to this peak-to-side-lobe distance

Part III – Non-Cartesian Image Reconstruction

What we know so far

- We know how to reconstruct images from regular/grid/Cartesian sampled k-space: use the inverse FFT (a faster version of the DFT)
- We also know how to acquire data in k-space in arbitrary (not necessarily aligned to any grid) sampling locations
- We understand how to analyze the impact of arbitrary sampling on image properties through the PSF
- **The last thing we need to understand is how to reconstruct images from non-Cartesian samples in k-space**

Direct Inversion

- Our (non-uniform) k-space measurements are linear functions of the input
  - Linear means $f(c \cdot x) = c \cdot f(x)$ and $f(a + b) = f(a) + f(b)$
  - You can try and prove this for yourself, that the MRI encoding process (either in discrete or continuous form) obeys linearity
  - Discrete linear operators can be represented by matrices (and vice versa)
- So the strategy of the direct inversion method is to construct the matrix equation corresponding to the non-Cartesian measurements, and then solve for the image using conventional linear algebra

Direct Inversion

Matrix Equation

- If the discretized image source is $\rho[x]$, and the non-uniform measurements are encoded in the matrix E to produce measurements $\Psi[k]$, we can write:
- $\Psi[k] = E\rho[x]$, or dropping the indexing variables, simply $\Psi = E\rho$
- The rows of E correspond to each measurement, which is the complex Fourier exponential with some arbitrary k-coordinate
  - $E_{i,:} = \begin{bmatrix} e^{-i2\pi k_i x_1} & e^{-i2\pi k_i x_2} & \dots & e^{-i2\pi k_i x_N} \end{bmatrix}$
- If there are N voxels in the image, and M k-space samples, then E is an $M \times N$ matrix:
  - $\Psi^{(M\times 1)} = E^{(M\times N)}\rho^{(N\times 1)}$

Direct Inversion

Solving

- If $M \geq N$ (and all the sampling points are unique), this has a least-squares optimal solution given by the pseudo-inverse:
  - $\hat{\rho} = (E^H E)^{-1} E^H \Psi$
- If $M < N$, this has a minimum-norm optimal solution given by the pseudo-inverse:
  - $\hat{\rho} = E^H (E E^H)^{-1} \Psi$

Direct Inversion

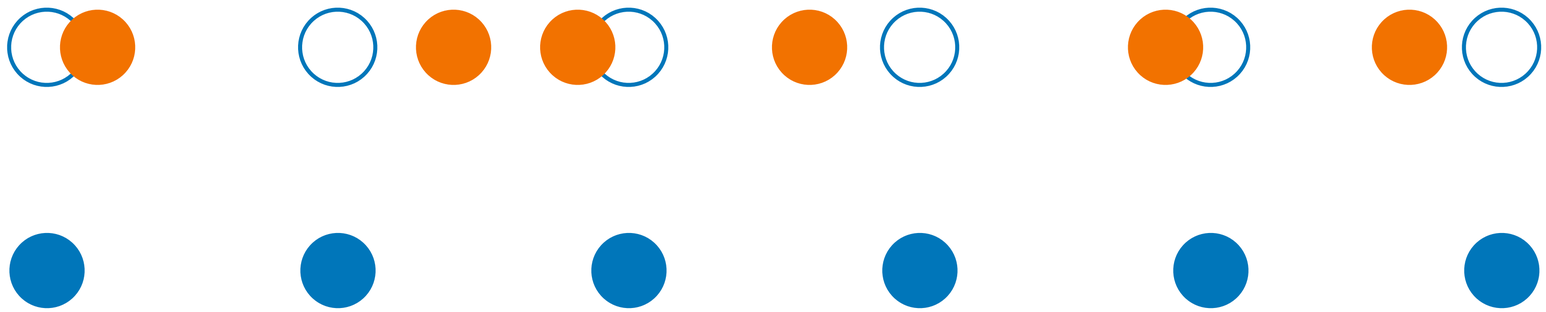
Challenges

- However, direct solving is not often recommended for the following reasons:
- It is slow, because it involves estimating inverses $O(N^3)$ or iterative optimization $O(N^2)$, and requires dense matrix operations
- It can also require massive amounts of memory if MN is large (e.g. 3D problems)
- If the matrix operator E is very close to singular (i.e. rows are nearly linearly dependent, or non-orthogonal) then this solution may have very poor performance due to both numerical instability (matrix conditioning) and noise propagation (noise amplification)
- Normally people don't do this, and I don't recommend it

Gridding

Overview

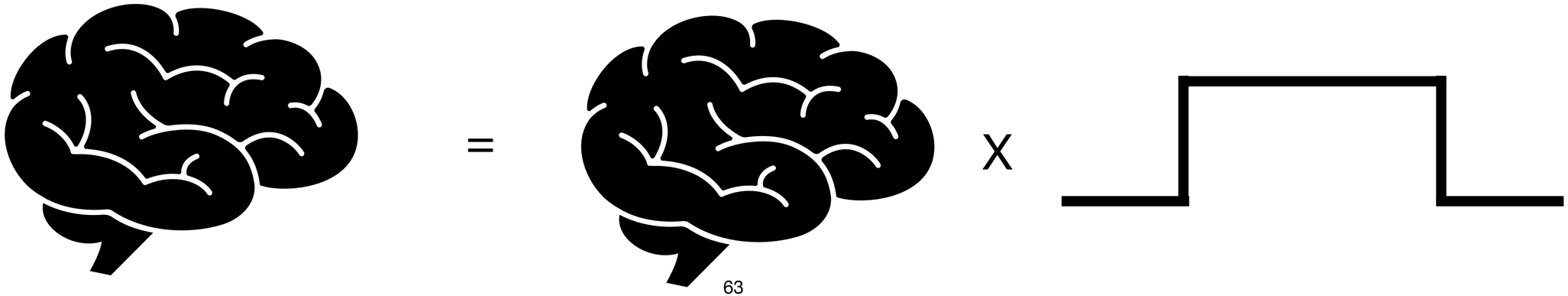
- Gridding allows us to solve the non-uniform sample problem efficiently by first transforming the problem to a uniform-sample problem, then using the FFT
- So we turn the reconstruction problem into a re-sampling problem, then proceed as usual with a Cartesian reconstruction (inverse FFT)
- The question is then *how*, and *why* is it possible to be able to take our non-Cartesian sampled points and turn them into Cartesian sampled points?



Gridding

Theoretical Basis

- Formally, the answer comes from the Shannon Sampling Theorem, which tells us something about properties of sampled band-limited signals
- This is fairly straightforward to understand intuitively:
- First, consider that we're always imaging objects with some finite extent
- Then, multiplying our object with a rect function does not change it



Gridding
Theoretical Basis

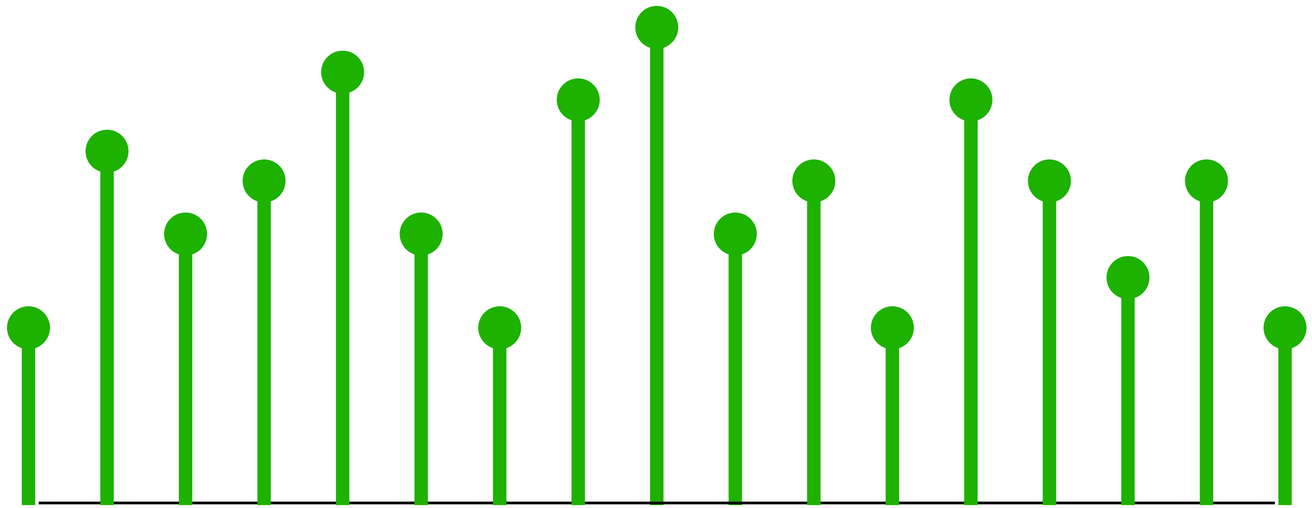
- So we have $\rho(x) = \rho(x) \cdot \text{rect}\left(\frac{x}{W}\right)$, where W is typically the FOV
- Taking the Fourier transform of both sides, we get $\Psi(k) = \Psi(k) \circledast \text{sinc}(\text{Wk})$
- The way we interpret this is that $\Psi(k)$ doesn't change when being convolved with $\text{sinc}(Wk)$
- So when we have some regularly sampled $\Psi[k]$, and convolve it with $\text{sinc}(Wk)$, we should just get our points back

Gridding
Theoretical Basis

- So we have $\rho(x) = \rho(x) \cdot \text{rect}\left(\frac{x}{W}\right)$, where W is typically the FOV
- Taking the Fourier transform of both sides, we get $\Psi(k) = \Psi(k) \circledast \text{sinc}(\text{Wk})$
- The way we interpret this is that $\Psi(k)$ doesn't change when being convolved with $\text{sinc}(Wk)$
- So when we have some regularly sampled $\Psi[k]$, and convolve it with $\text{sinc}(Wk)$, we should just get our points back

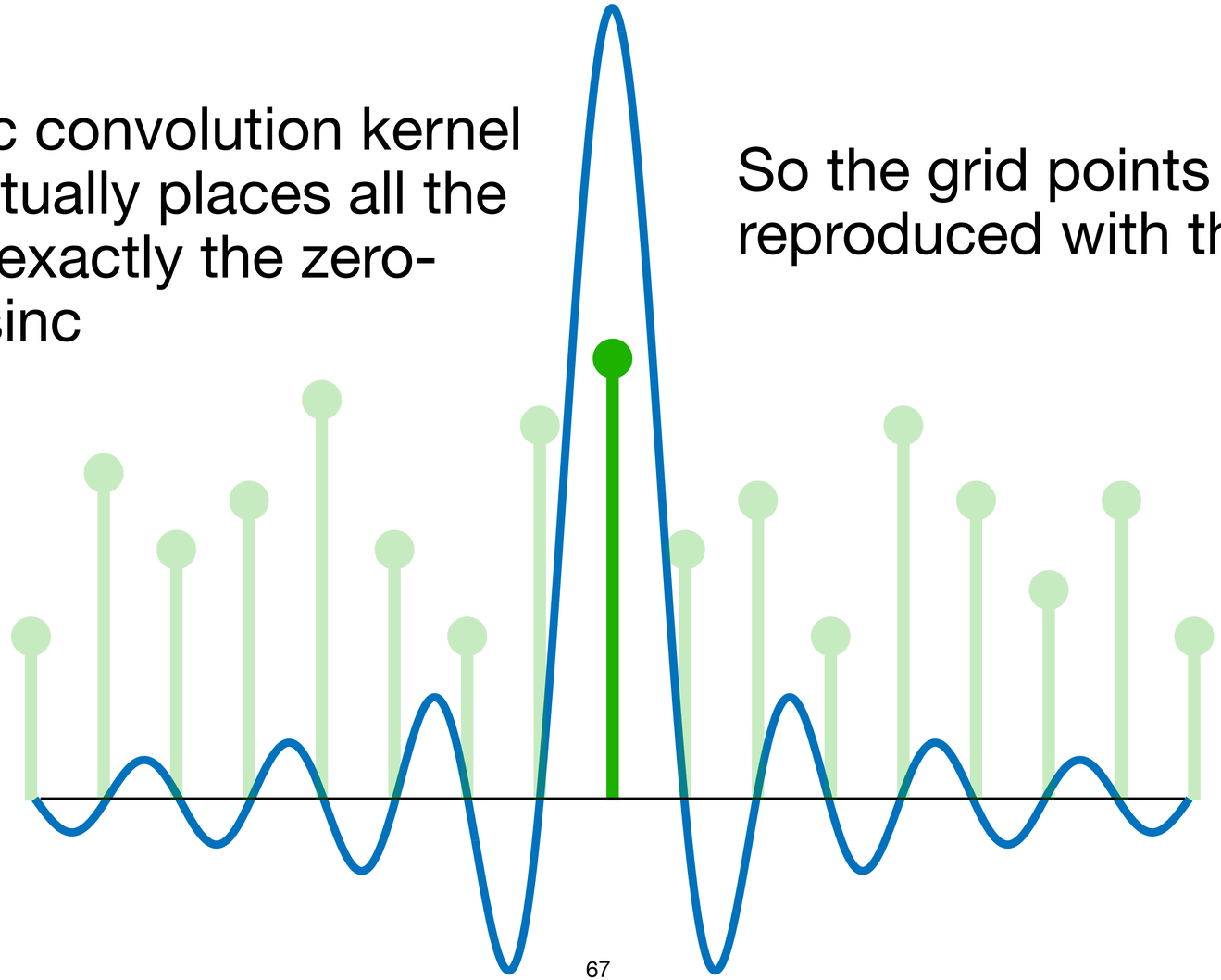
Gridding

Imagine some sampled k-space signal



Gridding

Centering the sinc convolution kernel on a grid point actually places all the other samples at exactly the zero-crossings of the sinc

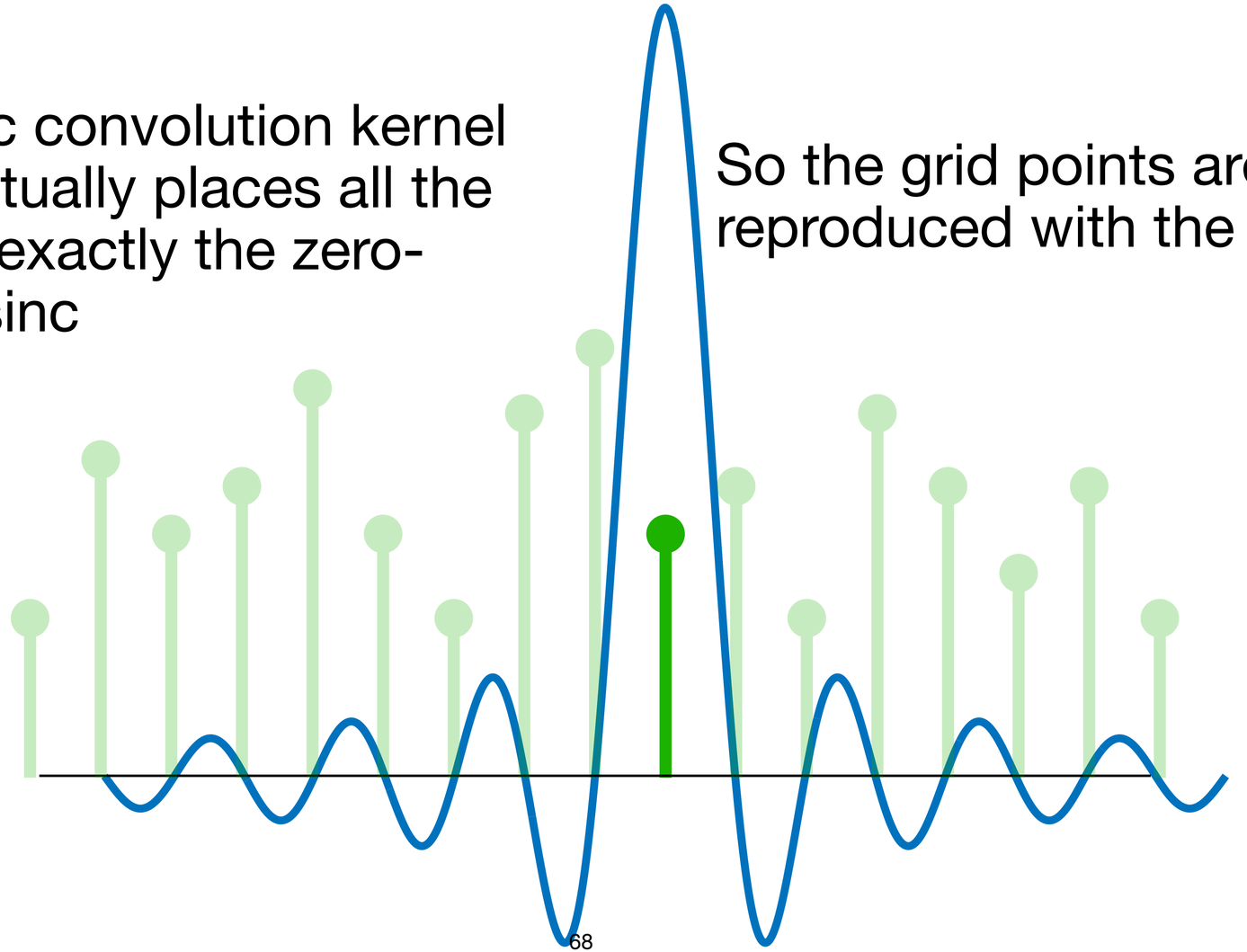


So the grid points are trivially reproduced with the sinc convolution

Gridding

Centering the sinc convolution kernel on a grid point actually places all the other samples at exactly the zero-crossings of the sinc

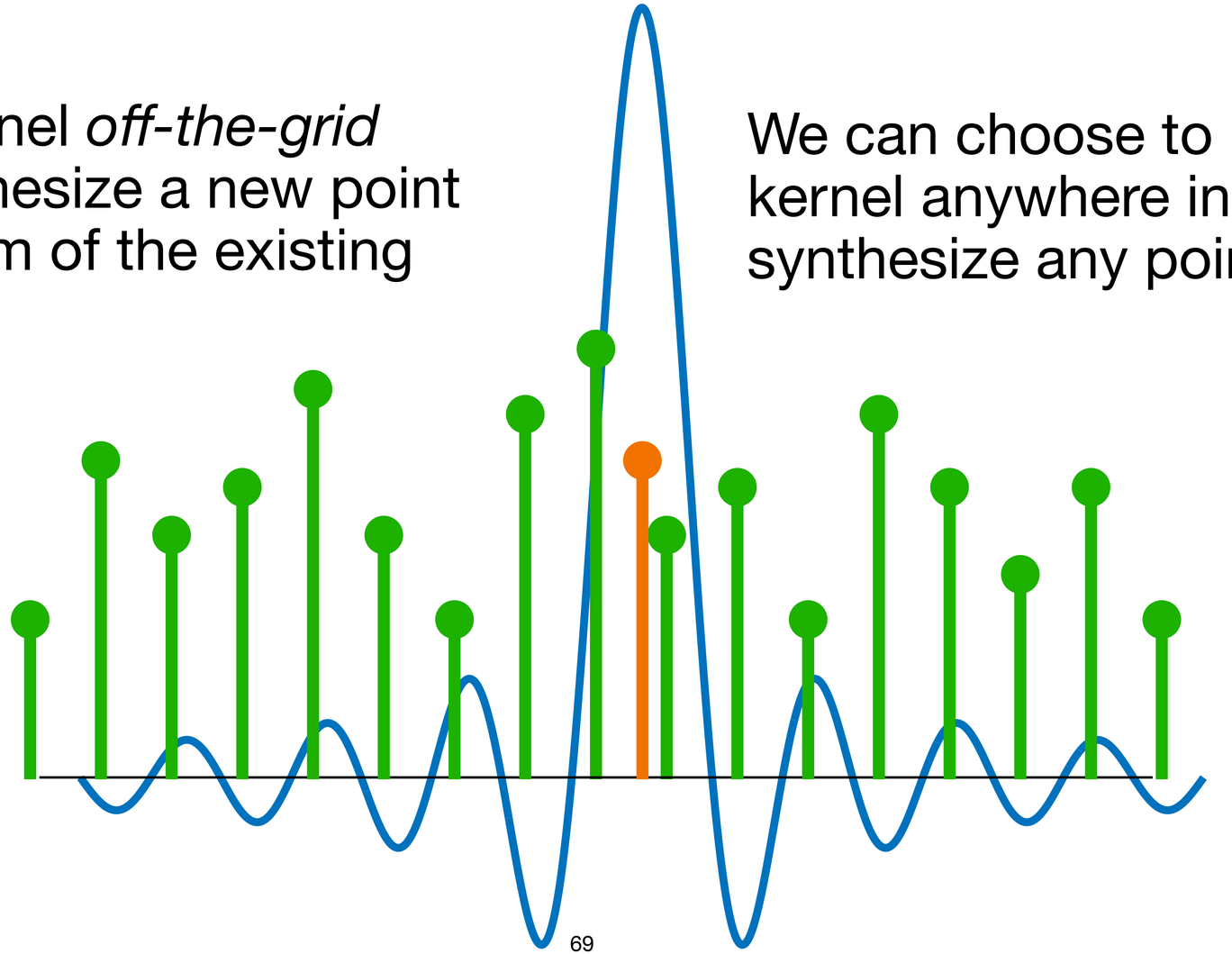
So the grid points are trivially reproduced with the sinc convolution



Gridding

Centering the kernel *off-the-grid* allows us to synthesize a new point as a weighted sum of the existing points

We can choose to place the sinc kernel anywhere in k-space to synthesize any point!



Gridding

Sinc Interpolation

- So from a grid of sampled points, we see that we can synthesize any other point using convolution with the sinc kernel
- This is called *sinc interpolation*
- We can turn this idea around and use it to synthesize grid points from non-uniformly sampled points!
- This is still the exact same convolution/interpolation process, simply synthesizing new points from existing points
- There was no requirement that the source points be regular
- We only require that the irregular points still satisfy the minimum aliasing distance rule (object size < FOV), otherwise the multiplication with the rect function is no longer an identity operation

Gridding

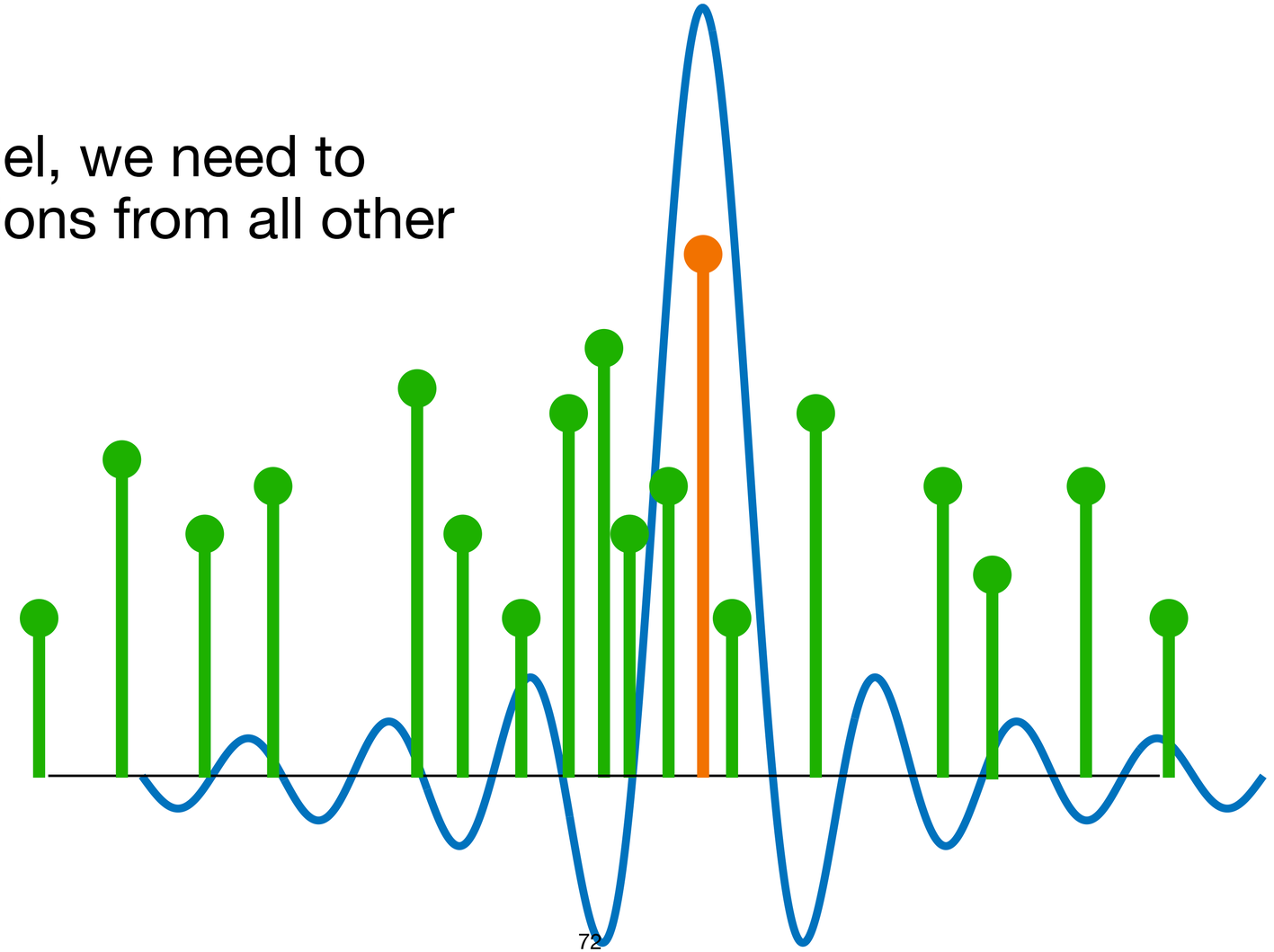
Sinc Interpolation

- However, sinc interpolation is not ideal since it technically requires convolution with an infinite width sinc to match the theoretical requirements
- This makes the interpolation slow and impractical, and the whole point of gridding is to speed things up
- The idea behind gridding is then to relax the theoretical requirements of sinc-interpolation, but using the inspiration of interpolating grid points from other points
- Want to find a compact kernel (small width) so we can estimate grid points from only a small neighbourhood of points nearby
    - This is more computationally efficient (smaller number of operations needed)

Gridding

Sinc Kernel

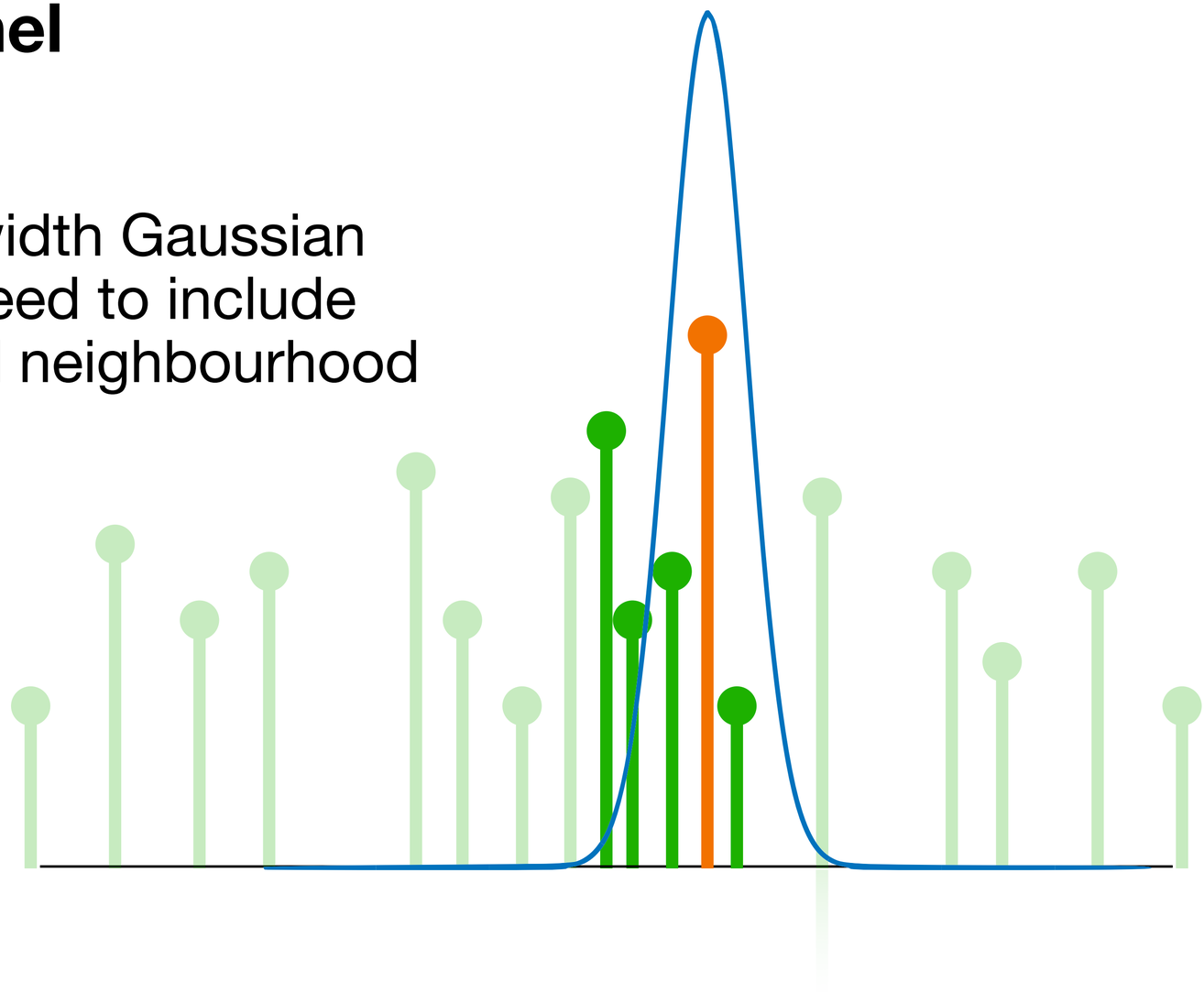
Using a sinc kernel, we need to include contributions from all other points



Gridding

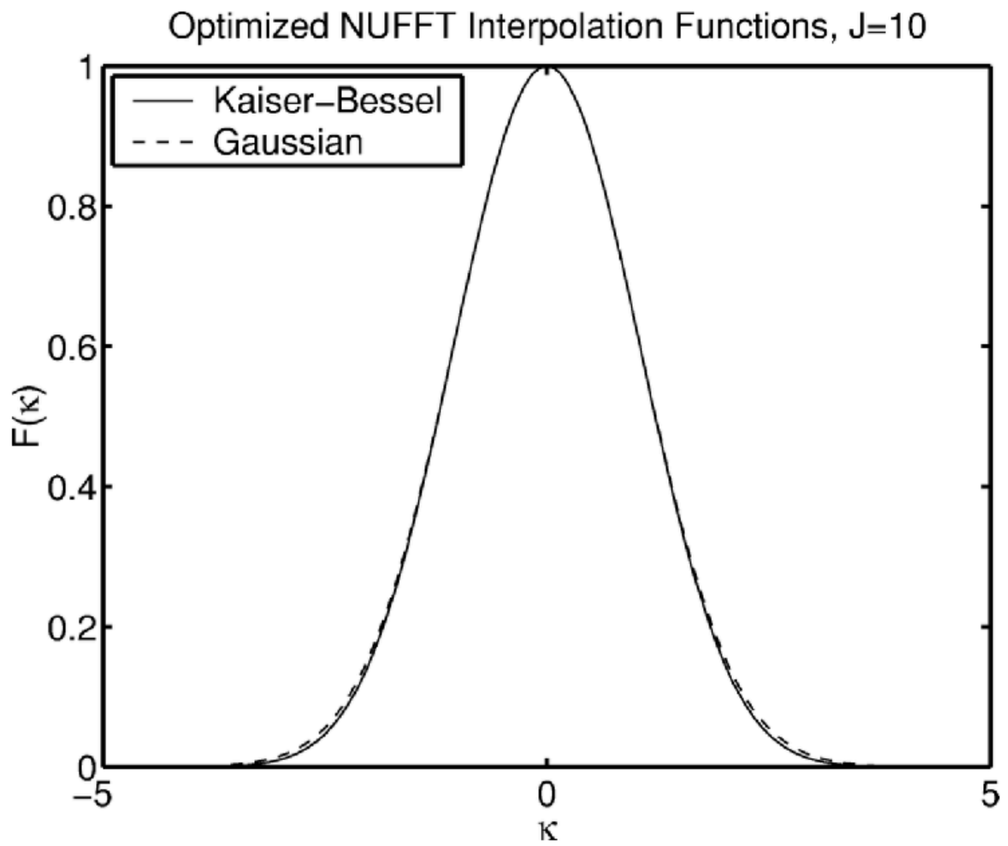
Gaussian Kernel

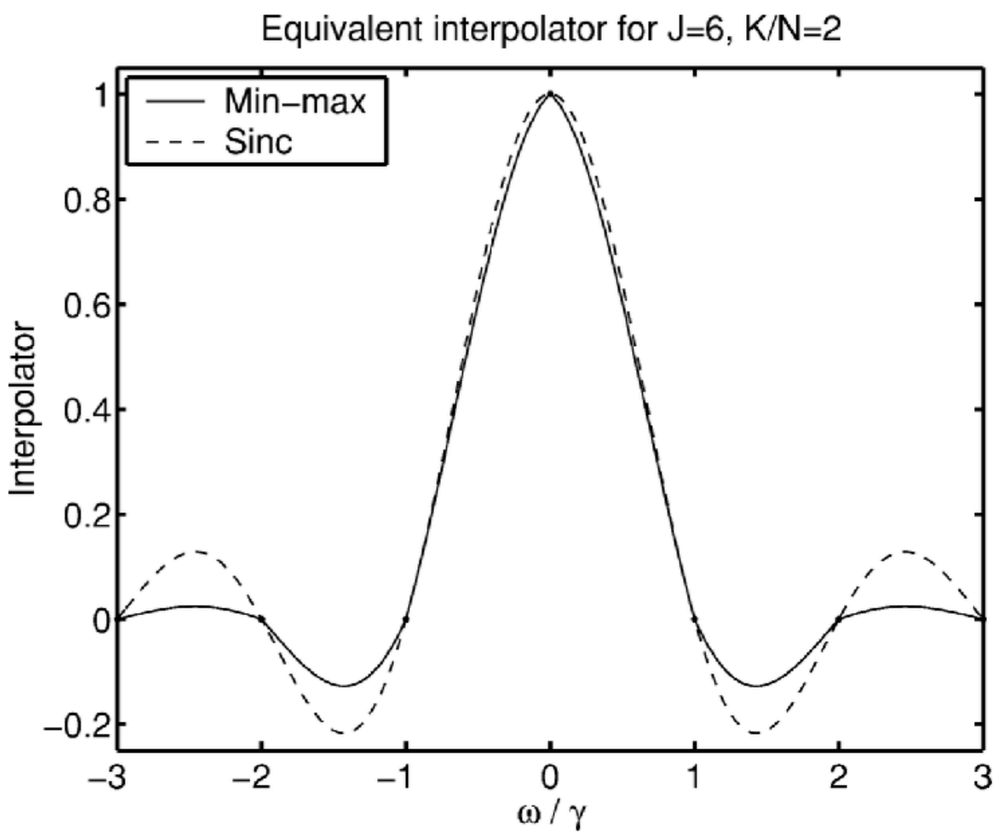
Using a smaller width Gaussian kernel, we only need to include points in the local neighbourhood



Gridding

Other interpolators





Fessler & Sutton IEEE-TSP 2003

Gridding
Interpolation

- So gridding is an approximation, not concerned with trying to be *exactly* correct, only as correct as possible
- The trade-off is speed for correctness
- The kaiser-bessel interpolator, or the min-max interpolators are the most popular, but the differences are marginal
- In practice, gridding often takes place onto an over-sampled grid (finer than strictly necessary given the target FOV), to avoid any interpolation-related aliasing issues
    - The image is typically cropped back afterwards to the desired FOV

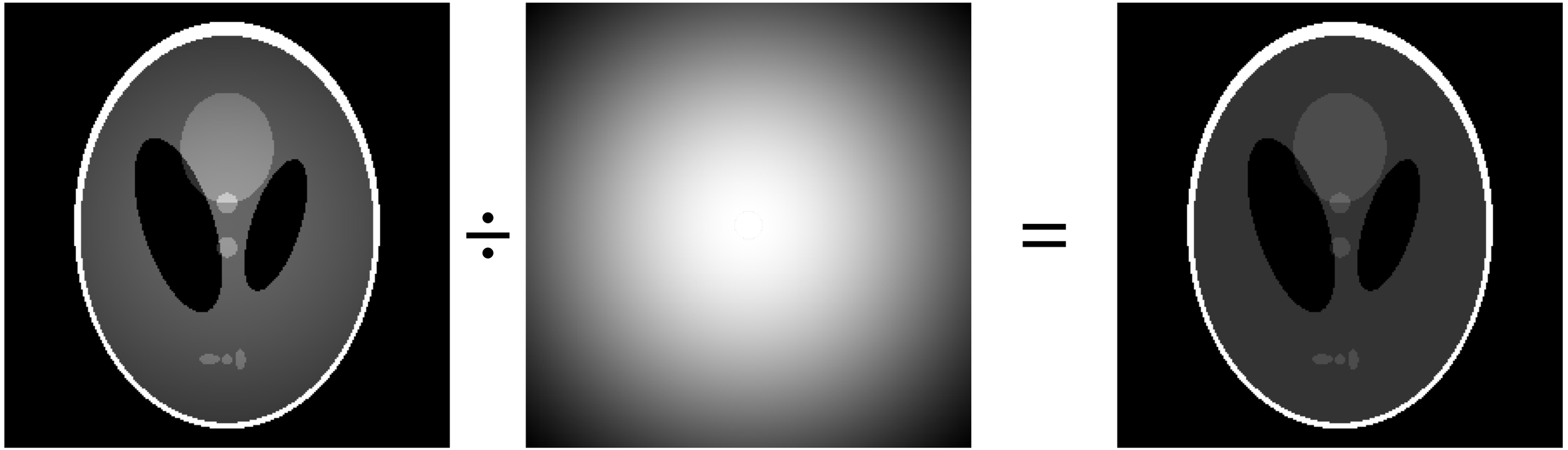
Gridding

Bias

- Recall that the ideal sinc-interpolator was motivated by the fact that it correponds to multiplication of the image by a rect function which doesn’t change the image
- But now, with these more efficient interpolators, we are convolving with a function that is not a sinc
- This corresponds to multiplication of the image by some spatially-dependent scaling function that biases the image
- This scaling function is the inverse Fourier transform of the interpolator

Gridding
Bias/Scaling

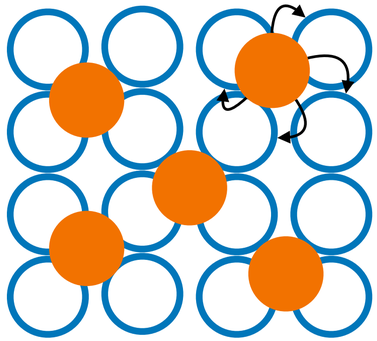
- Therefore, as a final step, the image output after inverse FFT must be divided by the scaling function to de-bias the image amplitude

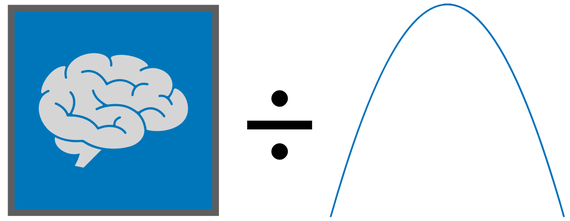


Gridding

Pipeline Overview

1. Perform interpolation using desired kernel, onto over-sampled grid
2. Once a regular k-space is estimated, perform an inverse FFT to generate an image
3. Crop the over-sampled image to the desired FOV
4. De-bias the image by dividing out the inverse FFT of the interpolation kernel





Non-Uniform FFT (NUFFT)

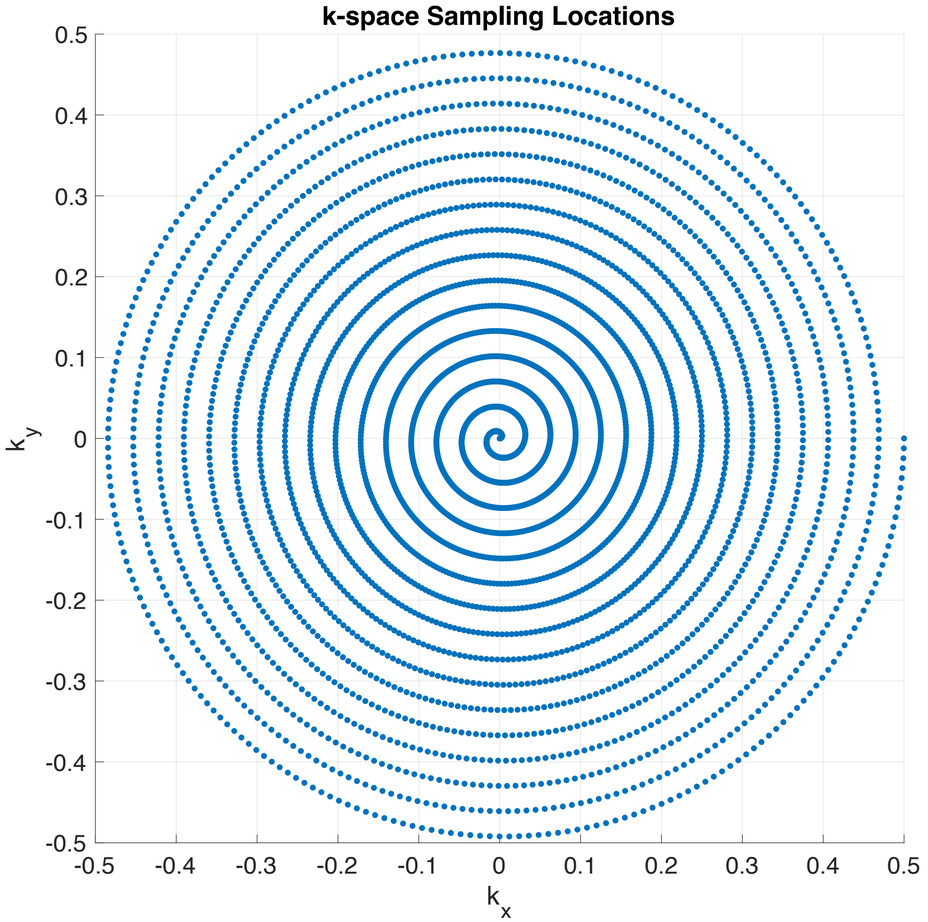
Gridding as a special case

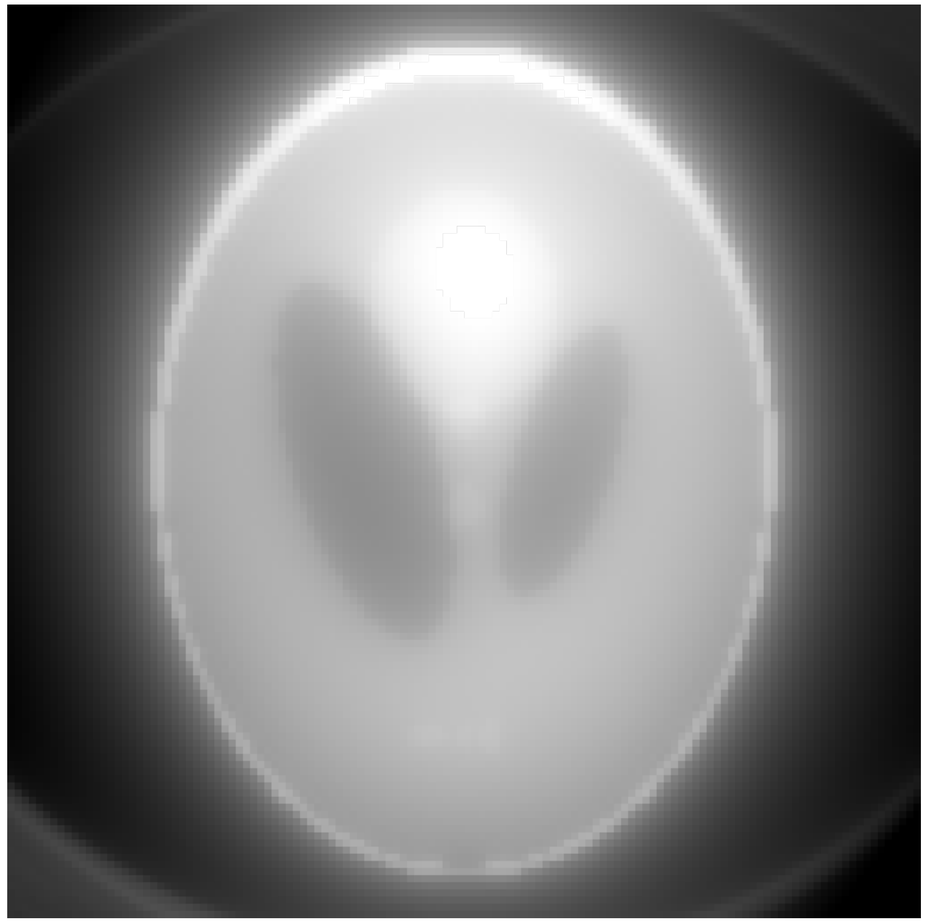
- Gridding is a particular set of operations that takes you from a non-uniformly sampled k-space to an image
- In fact, gridding as described here is essentially only the adjoint (conjugate transpose) non-uniform FFT (NUFFT)
  - i.e. the transform that takes you from non-uniform k-space to uniform image space
- The NUFFT includes this, but also the forward transform (mapping image space to non-uniform k-space)
- This allows for the use of iterative reconstruction schemes, which often need both the forward and reverse (adjoint) transforms

Part IV – Additional Considerations

Density Compensation

- It turns out, that the gridding process alone doesn't actually produce good looking images





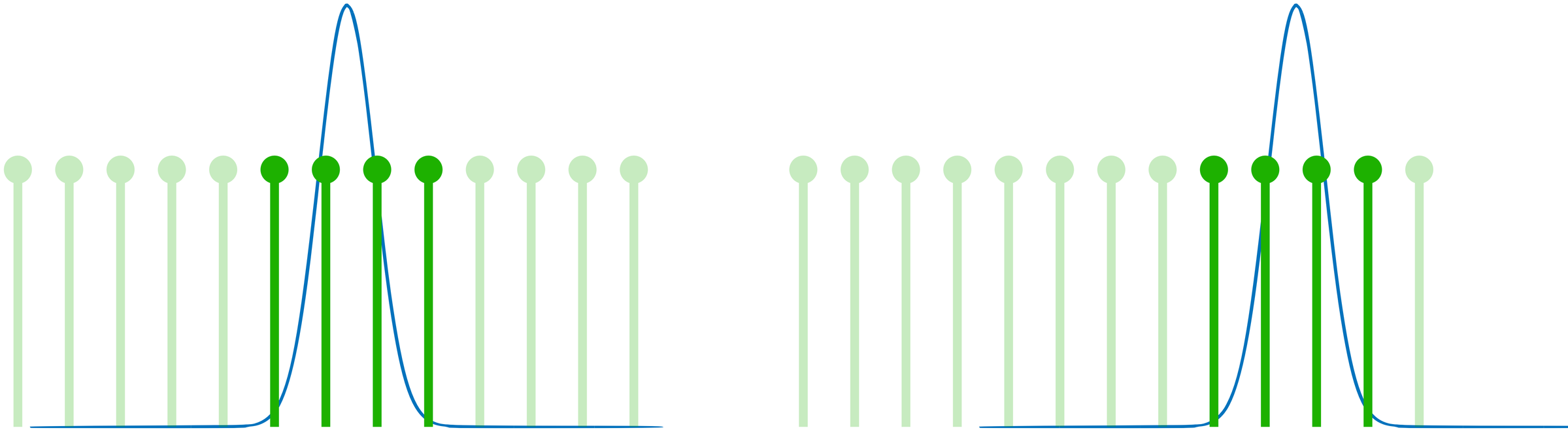
Density Compensation

Blurring

- Why does the output image look so blurry?
- It's because typically, non-Cartesian trajectories do not sample k-space with uniform density
- Some k-space locations have more samples per unit area, some have less
- This means that some k-space locations (typically central) are *overrepresented* relative to others
- That in turn means that those spatial frequency components (typically low spatial frequency) have more weight
- Ideally, to get a faithful representation of the image, we want every spatial frequency to be weighted equally

Density Compensation

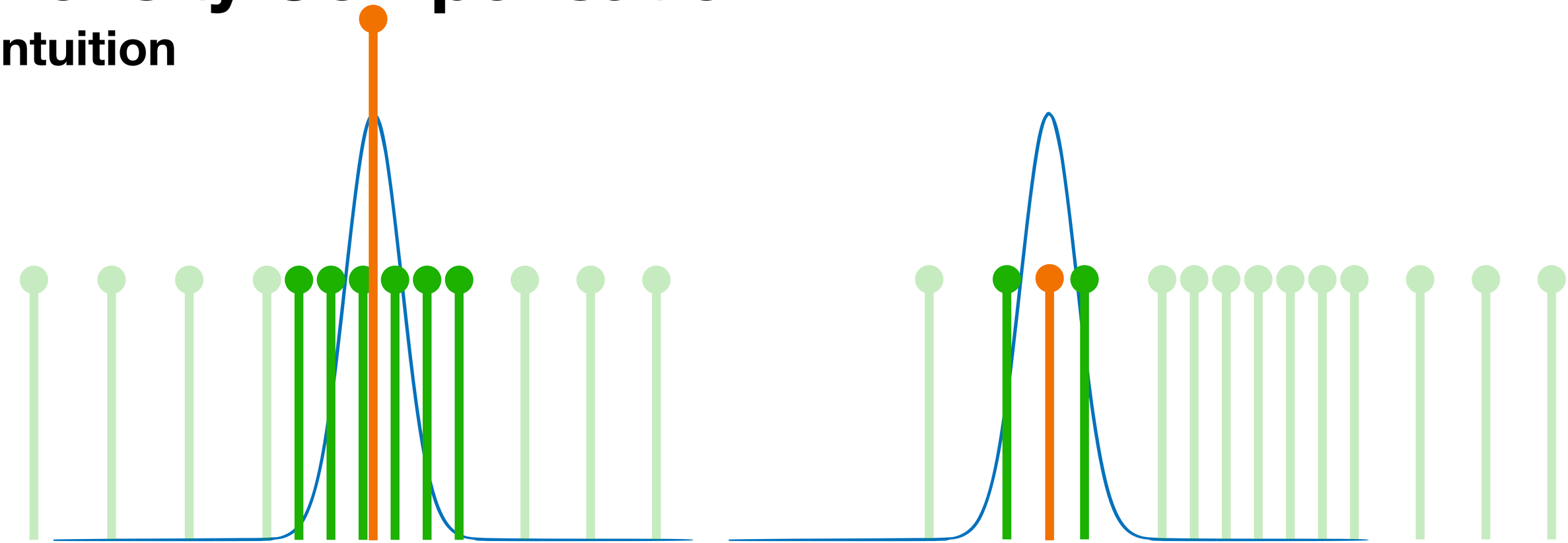
Intuition



With uniform density, the same number of points contribute to the interpolating convolution no matter where we are in k-space

Density Compensation

Intuition



With non-uniform density, the convolution kernel “sees” more points in denser sampling regions than in less dense regions

Since more points are being added together, the interpolated value actually scales with the density

Density Compensation

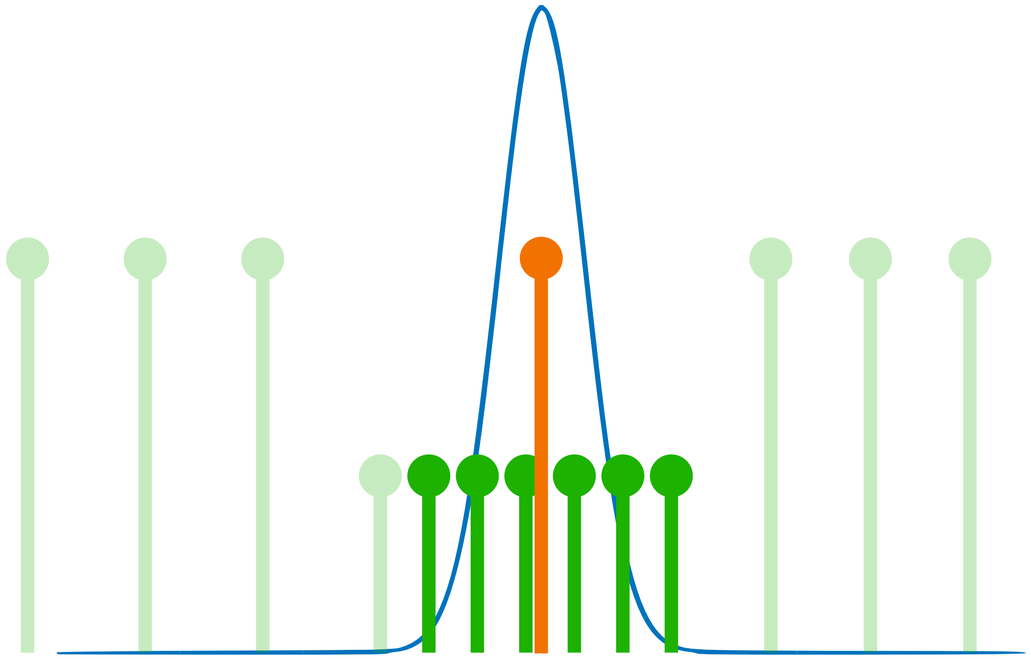
Effect of density

- Because the interpolated k-space ends up being scaled by a factor related to density, the image itself is *density weighted*
- That is, the k-space is multiplied with spatial frequency scaling factors: $\Psi[k] \rightarrow \Psi[k] \cdot D[k]$
- The impact of this on the image is a convolution with the inverse Fourier Transform of the density factor: $\mathscr{F}^{-1}(\Psi[k] \cdot D[k]) = \rho[x] \circledast \mathscr{F}^{-1}(D[k])$
- For most trajectories, D is high in the centre and lower at the periphery of k-space, so $\mathscr{F}^{-1}(D[k])$ looks like a blurring kernel
- Note this expression looks a lot like the PSF - these expressions are related!

Density Compensation

Correction

- So what we want to do is find some correction factor so we can pre-compensate our k-space to account for the density differences, so that after the gridding interpolation everything is equally weighted



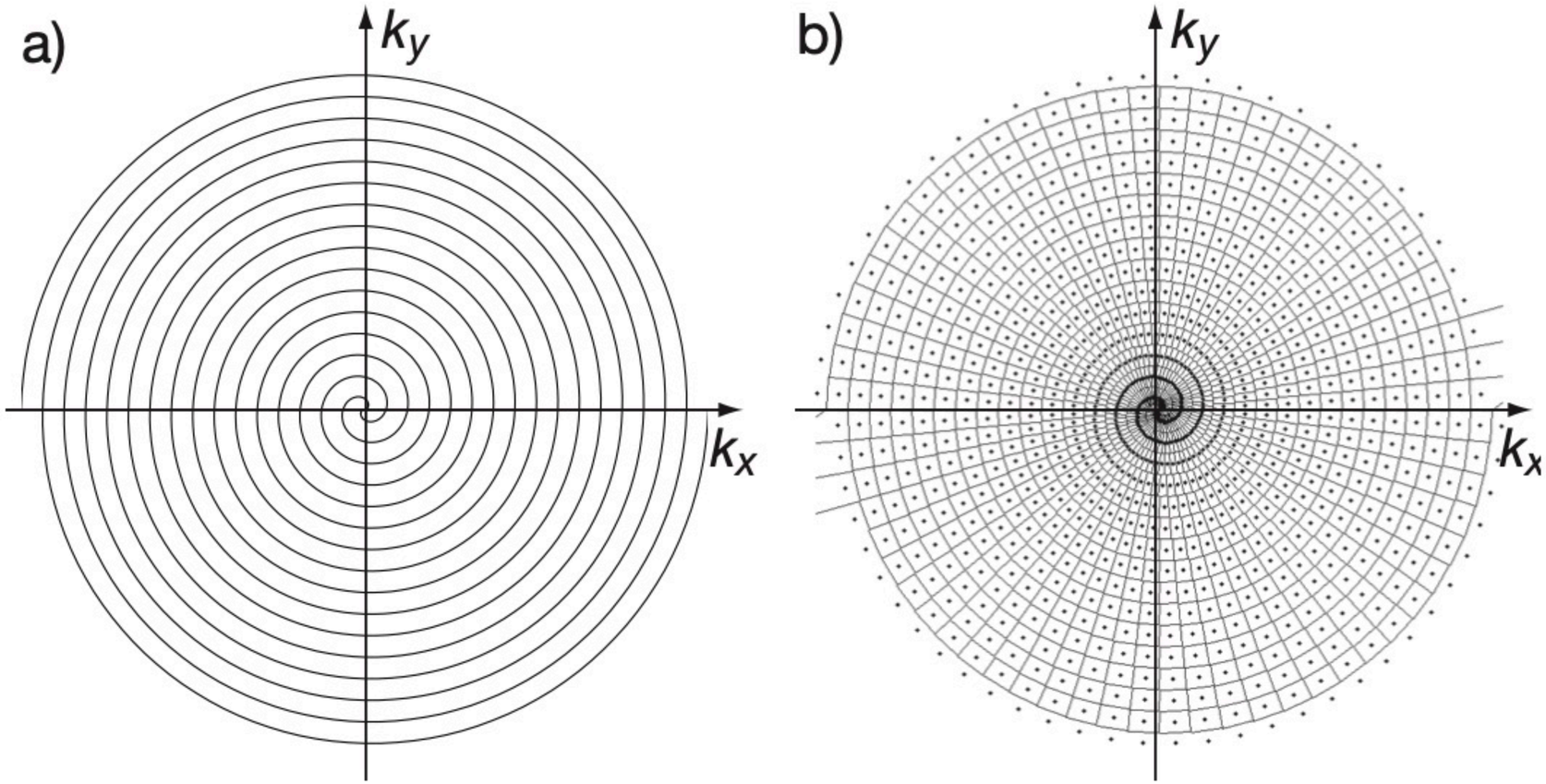
Density Compensation

Correction

- Ideally, we want to be able to “divide” out the density term D such that
- $$\left(\frac{\Psi[k]}{D[k]}\right) \cdot D[k] = \Psi[k]$$
- So then the question becomes one of estimating D
- This can be done analytically for some simple trajectories (radial, simple spirals)
- In most cases, this is estimated empirically

Density Compensation

Voronoi Cells



Pauly Lecture Notes

Density Compensation

Iterative schemes

- Recall that the effect of blurring is essentially the PSF of the sampling function
- So we can try to “undo” the PSF by finding some weights $w[k] \approx \frac{1}{D[k]}$
- We want then the product of the sampling function and the weights to result in an effective PSF that is as close to a delta function as possible
- Using the forward NUFFT, you could try to estimate the non-uniform k-space associated with the PSF, and define $w[k] = \frac{1}{\mathcal{F}(\mathrm{PSF[x]})}$
- However, this typically doesn’t work optimally due to approximation errors

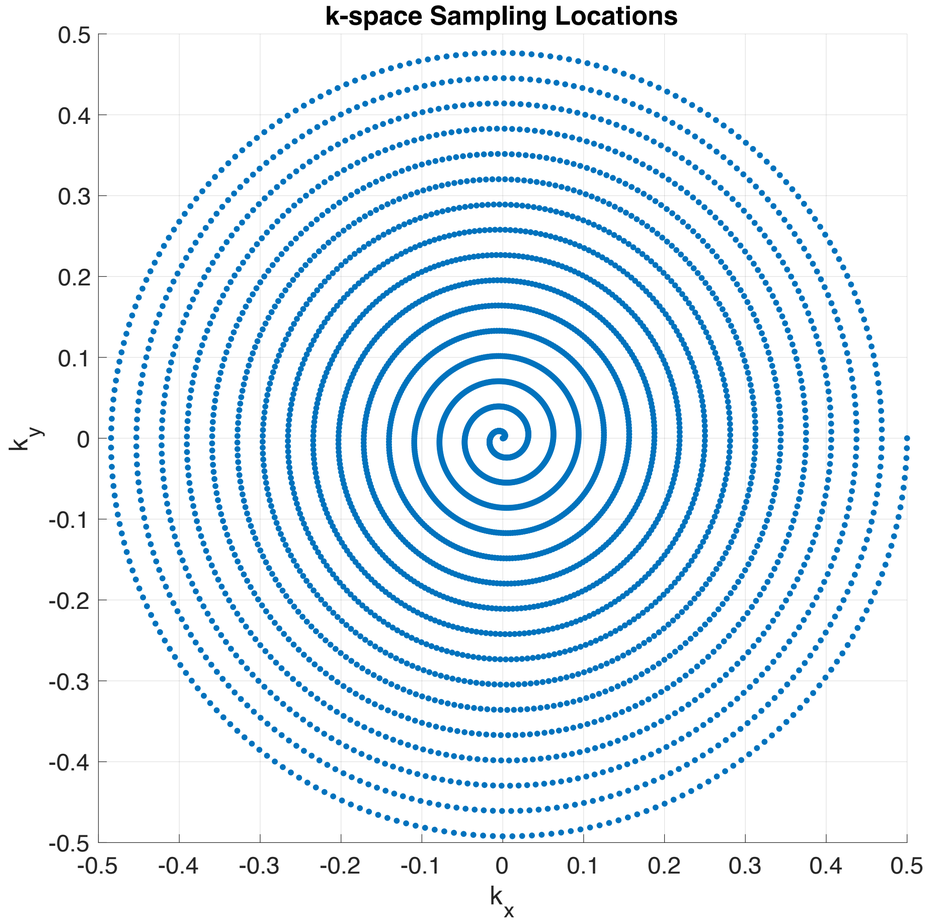
Density Compensation

Iterative schemes

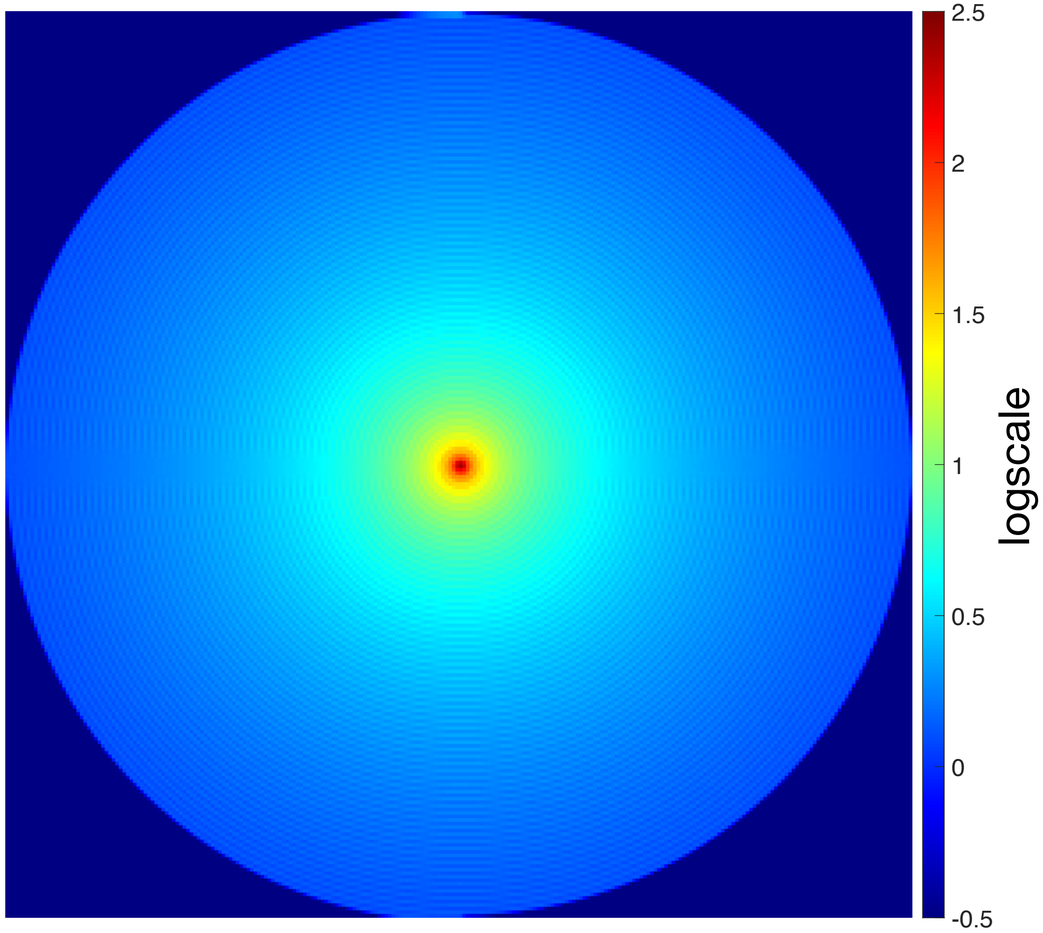
- Using an argument like to this, Pipe and Menon (MRM 1999) derived an iterative scheme for defining the weights:
- $w^{i+1}[k] = \frac{w^i[k]}{w^i[k] \circledast \psi[k]}$, where $\psi[k]$ is the gridding interpolation kernel
- The way to interpret this is to consider that the denominator “grids” the weights, where we want the result to be 1 ideally (for uniform weight, or the PSF to be a delta-function)
  - If the result is greater than 1, this reduces the weight, and if result is less than 1, this increases the weight
- This process should converge to weights that result in a delta-like PSF

Density Compensation

Pipe & Menon Weights

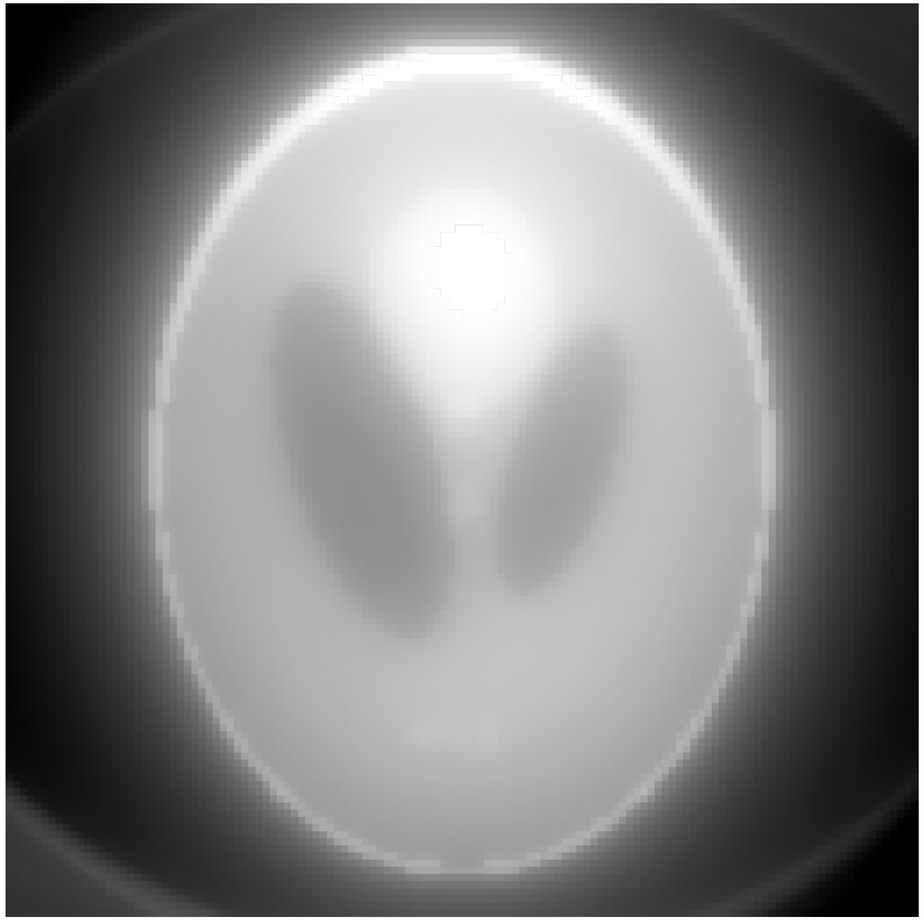


$$1/w[k] \sim D[k]$$



Density Compensation

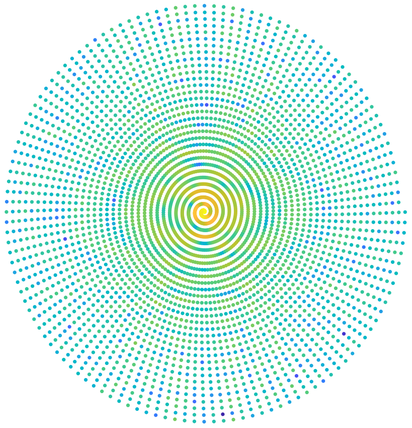
No Density Compensation



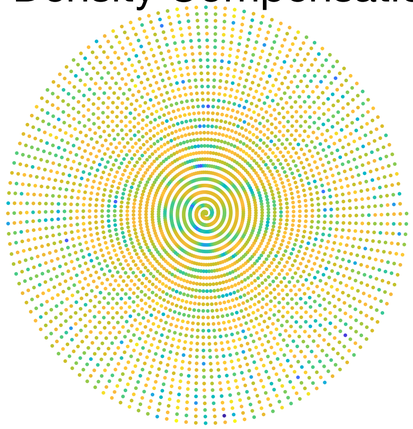
With Density Compensation

Gridding Pipeline

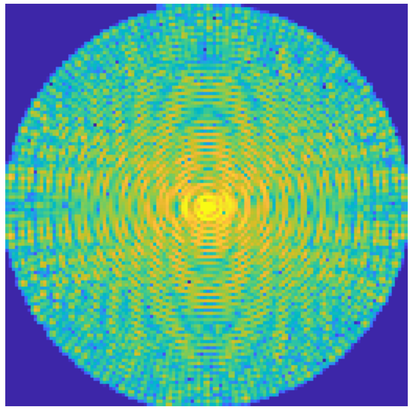
1. Non-Cartesian k-space



2. Density Compensation



3. Interpolation onto Oversampled Grid



4. Inverse FFT

5. Crop

5. Bias Correct

Density Compensation

Matrix interpretation

- Ultimately, since we are point-wise multiplying the k-space data by some correction factor, we can consider the full gridding reconstruction as:
- $$\hat{\rho}[x] = B\mathcal{F}^{-1}G_{\psi}W\Psi[k]$$
- Where $\mathcal{F}^{-1}$ is the inverse FFT, G_{ψ} is the gridding convolution operator with interpolation kernel ψ, W is a diagonal matrix of density compensation weights, and B is the scaling bias correction factor (also a diagonal matrix), both W and B typically real-valued
- So if we want our reconstruction $\hat{\rho}[x]$ to be as close to the original $\rho[x]$ as possible, we want:
- $$B\mathcal{F}^{-1}G_{\psi}WE \approx I$$

Iterative Reconstruction

Density compensation not always needed

- If you're doing an iterative reconstruction, you don't *need* to explicitly compensate for density
- That's because you're solving a different problem:
- Solving $\Psi[k] = E\rho[x]$ in the least-squares sense means solving the normal equations exactly:
- $$E^H\Psi[k] = (E^HE)\rho[x]$$
- If you solve the above iteratively, you *implicitly* density compensate by *effectively* inverting (but not explicitly inverting) E^HE

Iterative Reconstruction

Toeplitz Embedding

- This is actually more efficient than it seems, despite requiring iterations
- $E^H \Psi[k]$ only needs to be pre-computed once
- $E^H E$ turns out can be very efficiently evaluated since it has Toeplitz (or block-Toplitz) structure, and can be embedded within a circulant matrix C such that $C = \mathscr{F}^{-1} \Lambda \mathscr{F}$ with diagonal matrix Λ
- This is sometimes referred to as the “Toeplitz embedding” of $E^H E$
- So to compute $E^H E x$, one can instead compute $\mathscr{F}^{-1} \Lambda \mathscr{F} \tilde{x}$, where $\tilde{x}$ is a zero-padded version of x, and truncate/crop after evaluation
- This is much faster because $\mathscr{F}$ can be evaluated with FFTs, and the diagonal matrix Λ can be precomputed (in fact, there is a close relationship between Λ as the Fourier transform of the PSF)

Noise Propagation

- It turns out that while density compensation is beneficial for image fidelity, it has an impact on the image SNR
- Not all trajectories are created equal with respect to noise, even if they have similar signal properties
- Tsai and Nishimura (MRM 2000) showed that the SNR efficiency of a given sampling trajectory is related to how non-uniform the density is
- The more non-uniform a sampling trajectory is, the larger the density compensation weights need to be, and you can think of this as causing amplification of the noise

Noise Propagation

More Intuition

- The inverse Fourier transform performs a sort of “noise averaging” process because the signal + noise samples across all of k-space are combined to generate the image space signals
- Now imagine 2 noise sources n_1 and n_2, with equal variance σ^2
- Consider weighting these two noise sources with weights w_1 and w_2 respectively, such that $w_1 + w_2 = 1$
- Then the total variance is $\mathrm{Var}(w_1 n_1 + w_2 n_2) = (w_1^2 + (1 - w_1)^2)\sigma^2$
- This is minimized when $2w_1 - 2(1 - w_1) = 0$, or $w_1 = w_2 = 0.5$
- Total variance is always minimized when noise sources of equal variance are weighted equally
- Therefore we can understand that uniform weighting (i.e. uniform density) is SNR optimal

Noise Propagation

More formally

- For a gridding reconstruction $B\mathcal{F}^{-1}G_\psi W\Psi[k]$, where $\Psi[k] = E\rho[x] + n$ and n is some zero mean i.i.d. noise (identity covariance Σ_n)
- Then the variance of $\hat{\rho}[x]$ is the expression:
- $$(B\mathcal{F}^{-1}G_\psi W)\Sigma_n(B\mathcal{F}^{-1}G_\psi W)^H$$
- Which simplifies to
- $$B\mathcal{F}^{-1}G_\psi W^2 G_\psi^H \mathcal{F} B$$
- Which only depends indirectly on the trajectory through W^2, implying that the only important characteristic of the trajectory as far as noise is concerned is the density

References

- Jackson, J.I., C.H. Meyer, D.G. Nishimura, and A. Macovski. “Selection of a Convolution Function for Fourier Inversion Using Gridding (Computerised Tomography Application).” IEEE Transactions on Medical Imaging 10, no. 3 (September 1991): 473–78. https://doi.org/10.1109/42.97598.
- Scheffler, K, and J Hennig. “Reduced Circular Field-of-View Imaging.” Magnetic Resonance in Medicine 40, no. 3 (January 1, 1998): 474–80.
- Pipe, J G, and P Menon. “Sampling Density Compensation in MRI: Rationale and an Iterative Numerical Solution.” Magnetic Resonance in Medicine 41, no. 1 (January 1, 1999): 179–86.
- Tsai, C M, and D G Nishimura. “Reduced Aliasing Artifacts Using Variable-Density k-Space Sampling Trajectories.” Magnetic Resonance in Medicine 43, no. 3 (January 1, 2000): 452–58. https://doi.org/10.1002/(SICI)1522-2594(200003)43:3<452::AID-MRM18>3.0.CO;2-B.
- Fessler, Jeffrey A, and B.P Sutton. “Nonuniform Fast Fourier Transforms Using Min-Max Interpolation.” Ieee Transactions on Signal Processing 51, no. 2 (January 1, 2003): 560–74. https://doi.org/10.1109/TSP.2002.807005.