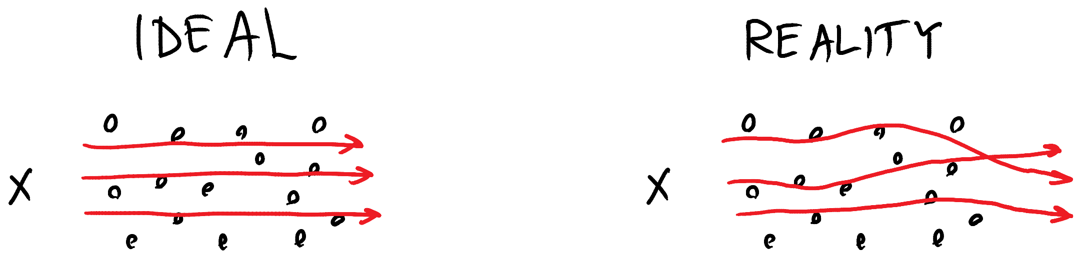
Deblurring

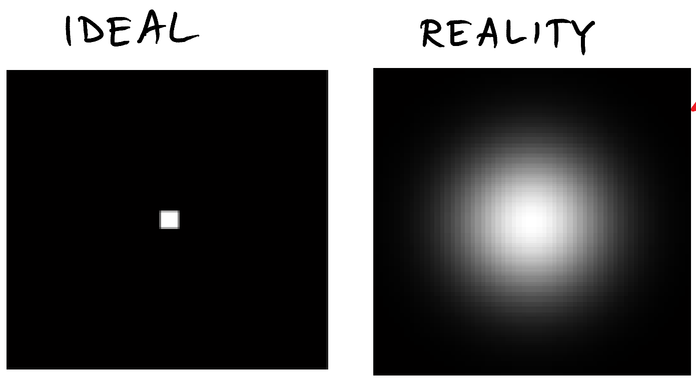
$\boxed{L21}$

Goal: To demonstrate the fundamental difficulties with deblurring, but learn some techniques that we can still use to make some progress.

Why deblurring in medical imaging?
eg. CT: Rayleigh scattering results in a "diffusion" of the x-ray beam.



As a result, a tightly-packed beam gets spread.



This blurred version of an ideal beam is called a point-spread function and can be thought of as a blurring kernel.

If convolving by a Gaussian blurs and image, it stands to reason that **deconvolving** deblurs an image.

$$\underbrace{g = f * h}_{\text{conv.}} \quad \Longleftrightarrow \quad \underbrace{G = F \cdot H}_{\text{mult.}}$$

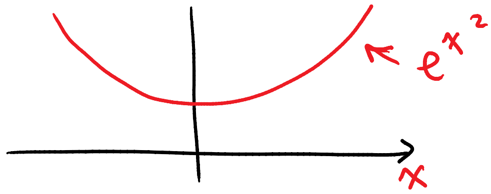
$$? \quad \Longleftrightarrow \quad F = \frac{G}{H}$$

?

deconvolution $\Longleftrightarrow$ $F = \frac{G}{H}$

division

If h is a Gaussian, then so is H.
And $\frac{1}{H}$ is something like this…



This de-weights the low frequencies, and accentuates the high freqs.

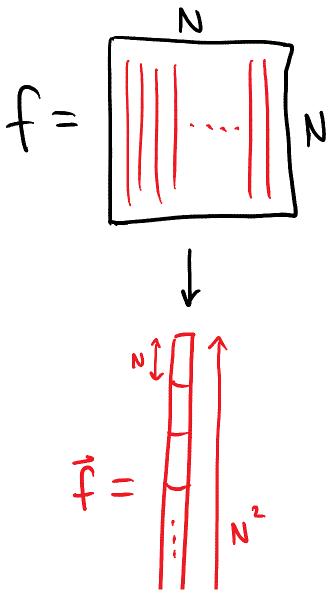
(deconvolution demo)

A problem arises when one of the Fourier coefs. *of G* is zero, or near zero. In such cases, and in general, deblurring is **ill-conditioned** . The resulting "deblurred" image is extremely sensitive to small perturbations. Hence, any noise in the blurry image will be blown up when you try to deblur it.

Another view… we can represent the convolution operation as a matrix operator.

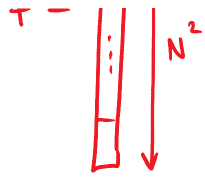
eg. blur kernel =

	1	
1	2	1
	1	



To deal with images using matrix operators, we convert the 2D image to a 1D column vector, $N^2 \times 1$.

Placing the kernel on the image then involves pixels that are adjacent in the column, and adjacent in rows.

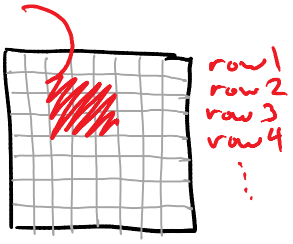


the column, and adjacent in rows.

Adjacent in cols of *f* $\Rightarrow$ adjacent in $\vec{f}$

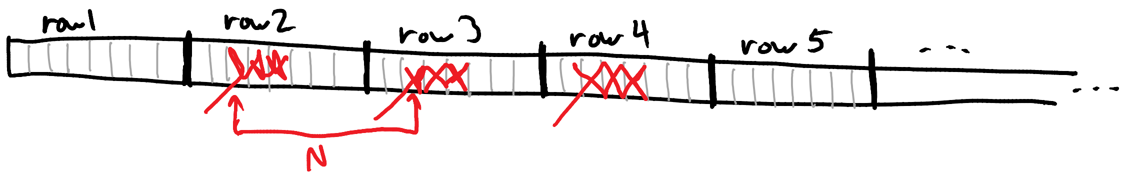
Adjacent in rows of *f* $\Rightarrow$ separated by N elements in $\vec{f}$

$D_i =$



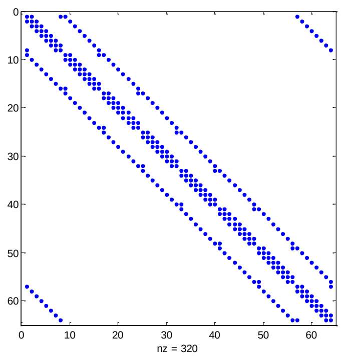
This is one row of the matrix convolution operator, D.

$\vec{D}_i =$



The matrix containing the appropriate weights is the convolution operator. We're going to use it as a blurring operator, so it is a blurring matrix.

$D =$



It has a multi-diagonal structure called "block circulant".

$$\vec{g} = D\vec{f} \Rightarrow \vec{f} = D^{-1}\vec{g}$$

(Matlab demo)

This matrix is ill-conditioned, which means its condition number is large.

$$\mathrm{cond}(D) = \kappa(D) = \|D\|\,\|D^{-1}\|$$

The condition number gives you an idea of how perturbations in your blurred image translate to perturbations in your deblurred image. In particular,

if $Df = g$ and $D(f + \Delta f) = g + \Delta g$ (getting rid of the vector arrows $f = \vec{f}$)

then $\dfrac{\|\Delta f\|}{\|f\|} \leq \text{cond}(D) \dfrac{\|\Delta g\|}{\|g\|}$

That is, the relative perturbation in the blurred image can be multiplied by as much as **cond(*D*).**
Thus, deblurring using $f = D^{-1} g$ is not a good idea.

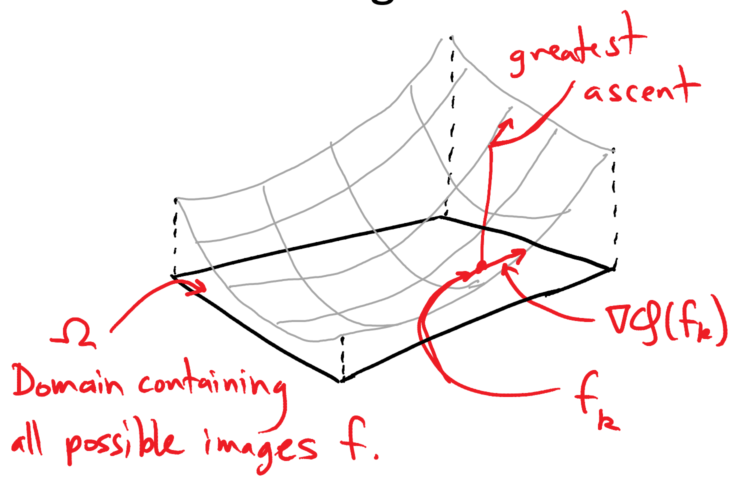
Instead, consider the "Reblurring" approach.

Minimize $\varphi(\tilde{f})$ where $\varphi(\tilde{f}) = \|g - D\tilde{f}\|^2$

In other words, instead of trying to invert the blurring operation, we try to find an image $\tilde{f}$ that, when blurred, looks like our original blurry image, g .

To minimize this, we can use an iterative gradient descent method. We want to find the vector $\tilde{f} \in \Omega$ that minimizes $\varphi(\tilde{f})$. The gradient of φ, denoted $\nabla \varphi$, is a vector in Ω that points in the direction of greatest increase of φ .

Gradient descent means take a step from where you are in the direction opposite the gradient.



$$f_{k+1} = f_k - \underbrace{\beta}_{\text{step size}} \nabla \varphi(f_k)$$

But how do we compute $\nabla \phi$?

$$\phi(f) = \| g - Df \|^2 = (g-Df)^T(g-Df)$$

$$= (g^T - f^T D^T)(g - Df)$$

$$= g^T g - g^T D f - f^T D^T g + f^T D^T D f$$

All these terms are scalar, so these two are equal (transposes of each other)

$$= \|g\|^2 - 2 f^T D g + f^T D^T D f$$

$$\Rightarrow \nabla \phi(f) = -2Dg + 2D^T D f$$

$$= -2D(g - Df)$$

$$\boxed{\therefore\ f_{k+1} = f_k + \beta D^T(g - Df), \quad f_0 = 0}$$

(reblurring demo)