Iterative Methods for Solving Linear Problems

When problems become too large (too many data points, too many model parameters), SVD and related approaches become impractical.

Iterative Methods for Solving Linear Problems

When problems become too large (too many data points, too many model parameters), SVD and related approaches become impractical.

Very popular alternatives utilize "iterative" methods to obtain **approximate** solutions.

Iterative Methods for Solving Linear Problems

When problems become too large (too many data points, too many model parameters), SVD and related approaches become impractical.

Very popular alternatives utilize "iterative" methods to obtain **approximate** solutions.

In many cases for large problems, the **G** matrix will be **sparse** (many many zero elements), so strategies to take advantage of this characteristic are important.

Simple Example - Kaczmarz's algorithm

Simple Example - Kaczmarz's algorithm

A class of techniques that attack the problem one row of **G** at a time.

In 2D, each row of **G** can be thought of as defining a line given by **G m**. In 3D **G m** is a plane, and above that it's a "hyperplane."

Simple Example - Kaczmarz's algorithm

A class of techniques that attack the problem one row of **G** at a time.

In 2D, each row of **G** can be thought of as defining a line given by **G m**. In 3D **G m** is a plane, and above that it's a "hyperplane."

Kaczmarz's algorithm starts with an initial guess (e.g., $\mathbf{m}_0 = 0$), and then steps through each row of **G** moving to the point on the $\mathbf{G}_{i,.}\mathbf{m}$ hyperplane closest to the current estimate of **m**. Amazingly, if **G m** = **d** has a unique solution, this simple approach will converge!

ALGORITHM:

To implement the algorithm, we need a formula to compute the projection of a vector onto the hyperplane defined by equation i. Consider the hyperplane defined by $\mathbf{G}_{i+1,\cdot}\mathbf{m} = d_{i+1}$. Because the vector $\mathbf{G}^T_{i+1,\cdot}$ is perpendicular to this hyperplane, the update to $\mathbf{m}^{(i)}$ from the constraint due to row $i+1$ of $\mathbf{G}$ will be proportional to $\mathbf{G}^T_{i+1,\cdot}$.

$$\mathbf{m}^{(i+1)} = \mathbf{m}^{(i)} + \beta \mathbf{G}^T_{i+1,\cdot}. \tag{6.2}$$

(6.3)

(6.4)

(6.5)

ALGORITHM:

To implement the algorithm, we need a formula to compute the projection of a vector onto the hyperplane defined by equation i. Consider the hyperplane defined by $\mathbf{G}_{i+1,\cdot}\mathbf{m} = d_{i+1}$. Because the vector $\mathbf{G}_{i+1,\cdot}^T$ is perpendicular to this hyperplane, the update to $\mathbf{m}^{(i)}$ from the constraint due to row $i+1$ of $\mathbf{G}$ will be proportional to $\mathbf{G}_{i+1,\cdot}^T$.

$$\mathbf{m}^{(i+1)} = \mathbf{m}^{(i)} + \beta \mathbf{G}_{i+1,\cdot}^T. \tag{6.2}$$

Using the fact that $\mathbf{G}_{i+1,\cdot}\mathbf{m}^{(i+1)} = d_{i+1}$ to solve for β, we obtain

$$\mathbf{G}_{i+1,\cdot}\left(\mathbf{m}^{(i)} + \beta \mathbf{G}_{i+1,\cdot}^T\right) = d_{i+1} \tag{6.3}$$

$$\mathbf{G}_{i+1,\cdot}\mathbf{m}^{(i)} - d_{i+1} = -\beta \mathbf{G}_{i+1,\cdot}\mathbf{G}_{i+1,\cdot}^T \tag{6.4}$$

$$\beta = -\frac{\mathbf{G}_{i+1,\cdot}\mathbf{m}^{(i)} - d_{i+1}}{\mathbf{G}_{i+1,\cdot}\mathbf{G}_{i+1,\cdot}^T}. \tag{6.5}$$

FORMULA:

Thus the update formula is

$$\mathbf{m}^{(i+1)} = \mathbf{m}^{(i)} - \frac{\mathbf{G}_{i+1,\cdot}\mathbf{m}^{(i)} - d_{i+1}}{\mathbf{G}_{i+1,\cdot}\mathbf{G}_{i+1,\cdot}^{T}}\mathbf{G}_{i+1,\cdot}^{T}$$

Simple Example - Kaczmarz's algorithm

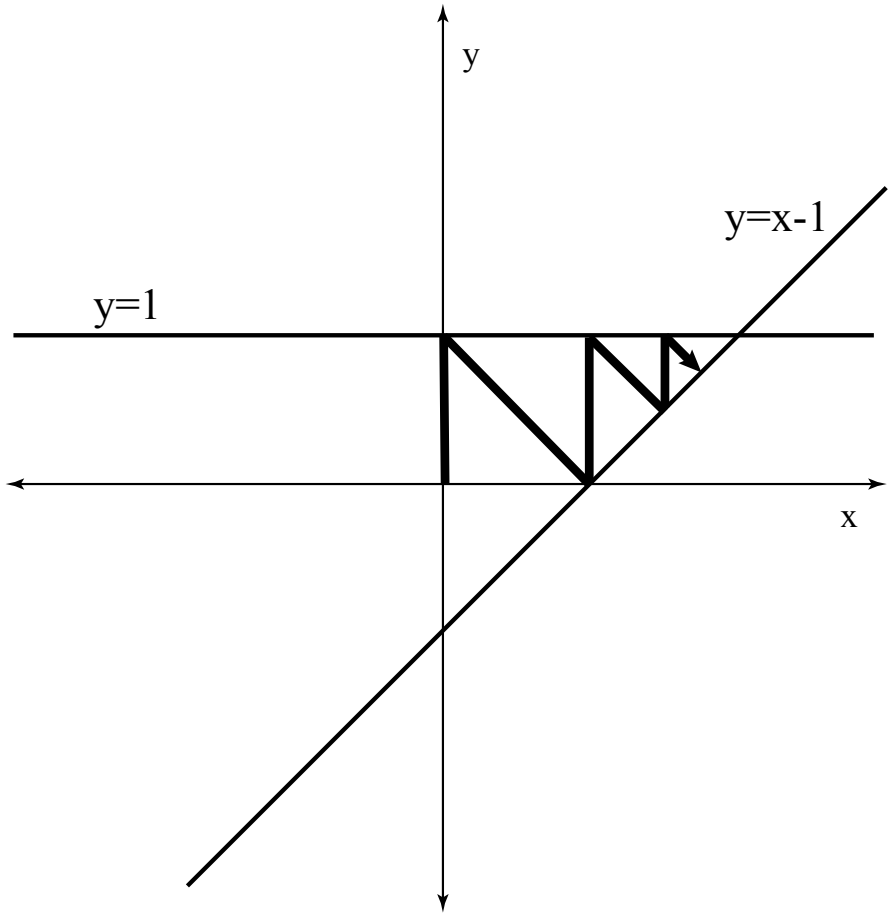


Figure 6.1: Kaczmarz's algorithm on a system of two equations.

MATLAB
Examples

QUESTIONS?

ART and SIRT

These two methods, Algebraic Reconstruction Technique and Simultaneous Iterative Reconstruction Technique, were developed for tomography applications.

For simplest version of ART, take Kaczmarz's formula

$$\mathbf{m}^{(i+1)} = \mathbf{m}^{(i)} - \frac{\mathbf{G}_{i+1,\cdot}\mathbf{m}^{(i)} - d_{i+1}}{\mathbf{G}_{i+1,\cdot}\mathbf{G}_{i+1,\cdot}^{T}}\mathbf{G}_{i+1,\cdot}^{T}$$

and replace all non-zero entries in $\mathbf{G}$ with 1's, assuming the length of the ray path in each cell is equal! What are the numerator and denominator then?

ART and SIRT

These two methods, Algebraic Reconstruction Technique and Simultaneous Iterative Reconstruction Technique, were developed for tomography applications.

For simplest version of ART, take Kaczmarz's formula

$$\mathbf{m}^{(i+1)} = \mathbf{m}^{(i)} - \frac{\mathbf{G}_{i+1,\cdot}\mathbf{m}^{(i)} - d_{i+1}}{\mathbf{G}_{i+1,\cdot}\mathbf{G}_{i+1,\cdot}^{T}}\mathbf{G}_{i+1,\cdot}^{T}$$

and replace all non-zero entries in $\mathbf{G}$ with 1's, assuming the length of the ray path in each cell is equal! What are the numerator and denominator then?

An estimate of the residual and the number of "hit" cells!

ART and SIRT

A slight improvement to ART can be made by scaling things by the equations to reflect the fact that different ray paths have different lengths (and note error on page 147 - "cell to cell" should be "path to path"!).

The approximation can be improved by taking into account that the ray path lengths actually will vary from cell to cell. If L_{i+1} is the length of ray path $i+1$, the corresponding improved update formula from (6.6) for the tomography problem is

$$m_j^{(i+1)} = \begin{cases} m_j^{(i)} + \frac{d_{i+1}}{L_{i+1}} - \frac{q_{i+1}}{lN_{i+1}} & \text{cell } j \text{ in ray path } i+1 \\ m_j^{(i)} & \text{cell } j \text{ not in ray path } i+1. \end{cases} \tag{6.10}$$

ART solutions tend to be noisy because the model is "jumping around" at every update.

ART and SIRT

SIRT uses the same "update formula"

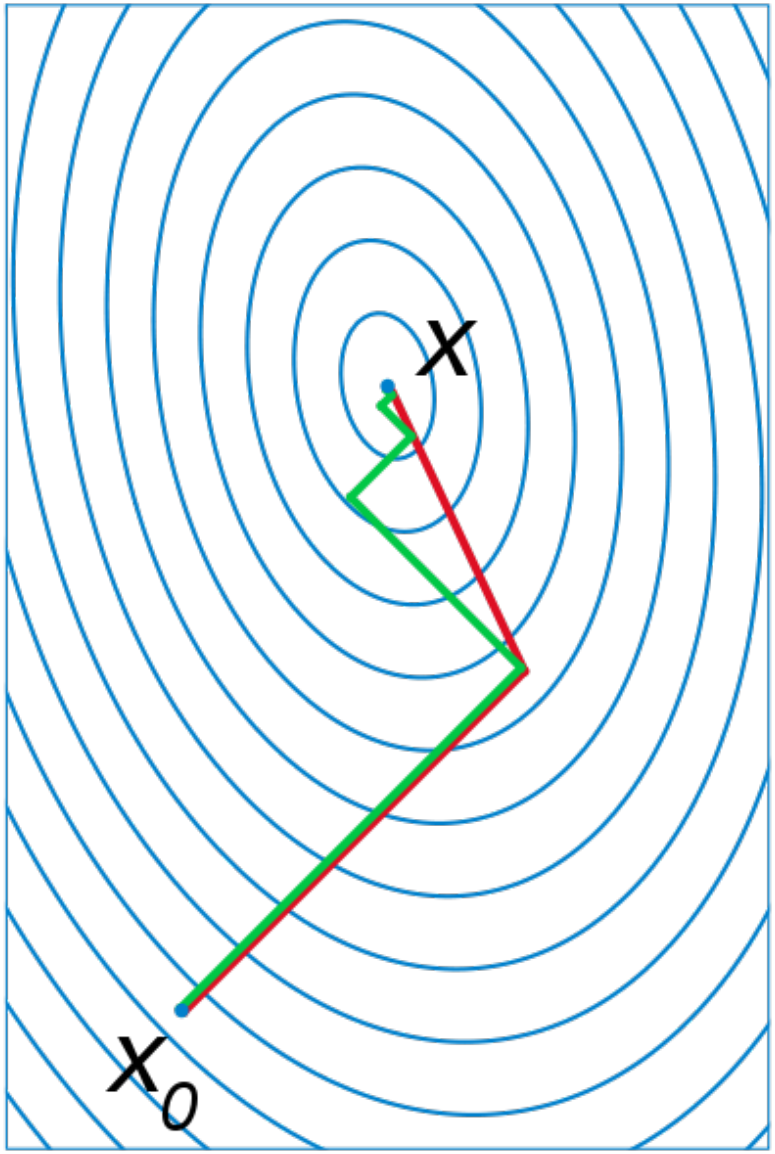
$$m_j^{(i+1)} = \begin{cases} m_j^{(i)} + \frac{d_{i+1}}{L_{i+1}} - \frac{q_{i+1}}{lN_{i+1}} & \text{cell } j \text{ in ray path } i+1 \\ m_j^{(i)} & \text{cell } j \text{ not in ray path } i+1. \end{cases}$$

but all model updates are computed before applying any of them to the model! The updates are averaged to obtain the model perturbation for that iteration step. As a result, SIRT results tend to be less noisy, due to the averaging effect.

QUESTIONS?

Conjugate Gradient and CGLS

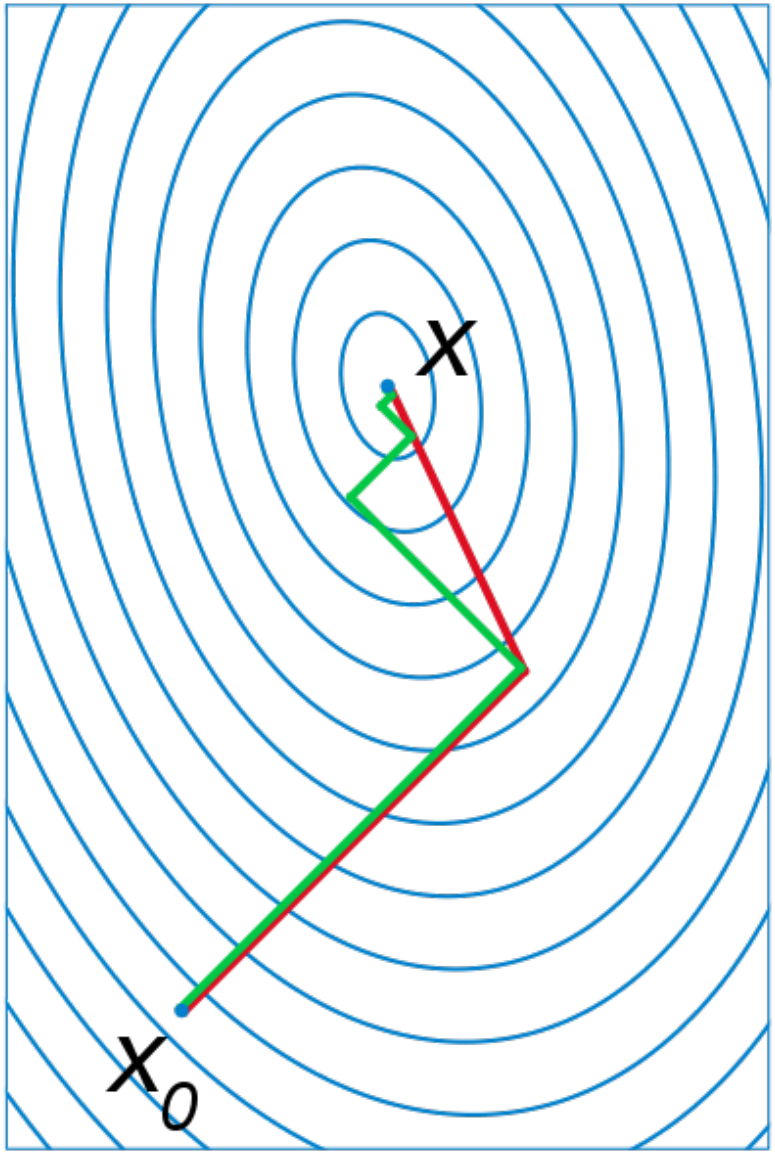
Goal - in N dimensions, get to the minimum in N steps!



Conjugate Gradient and CGLS

Goal - in N dimensions, get to the minimum in N steps!

Gradient method - go in the "downhill" direction (-g).

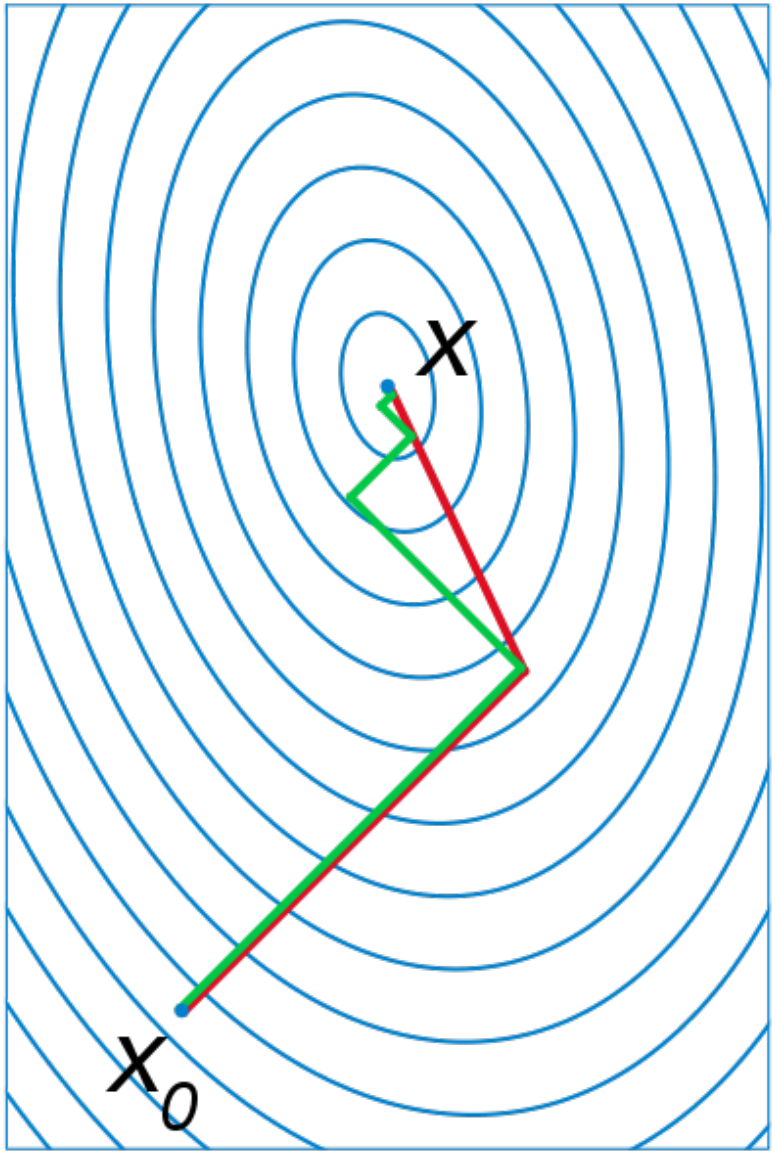


Conjugate Gradient and CGLS

Goal - in N dimensions, get to the minimum in N steps!

Gradient method - go in the "downhill" direction (-g).

Inefficient in a similar way that Kaczmarz's method is inefficient - bounce back and forth.



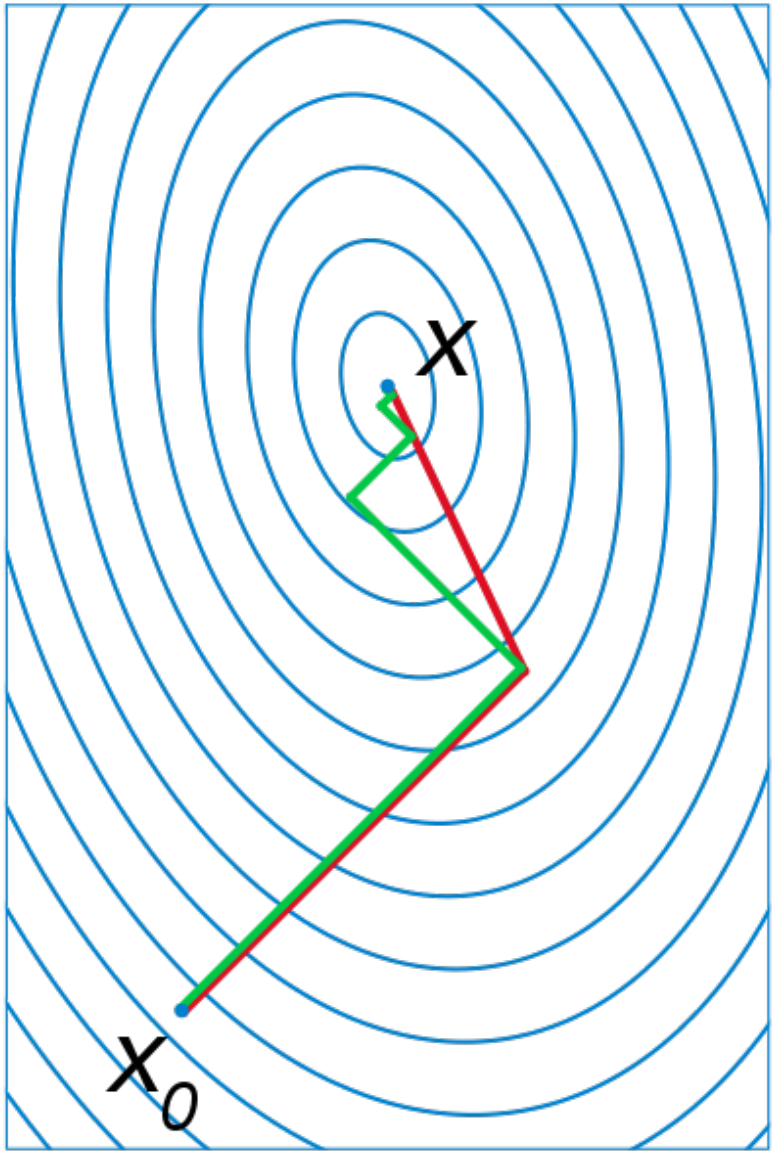
Conjugate Gradient and CGLS

Goal - in N dimensions, get to the minimum in N steps!

Gradient method - go in the "downhill" direction (-g).

Inefficient in a similar way that Kaczmarz's method is inefficient - bounce back and forth.

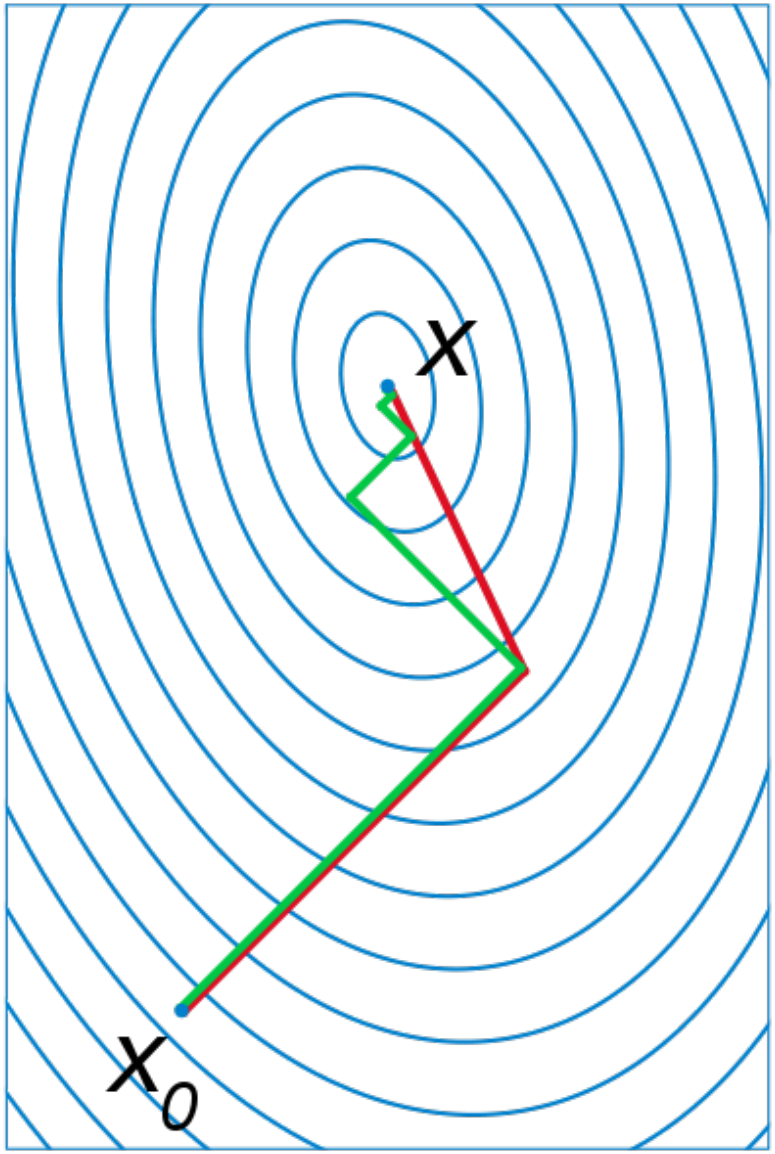
CG - zoom directly to the minimum! How??



Conjugate Gradient

In the figure, the green line is the optimal gradient (steepest descent) steps and the red line is the CG steps. Note that the second CG step can be thought of as a linear combination of the first two gradient steps.

How can we find the right linear combination?

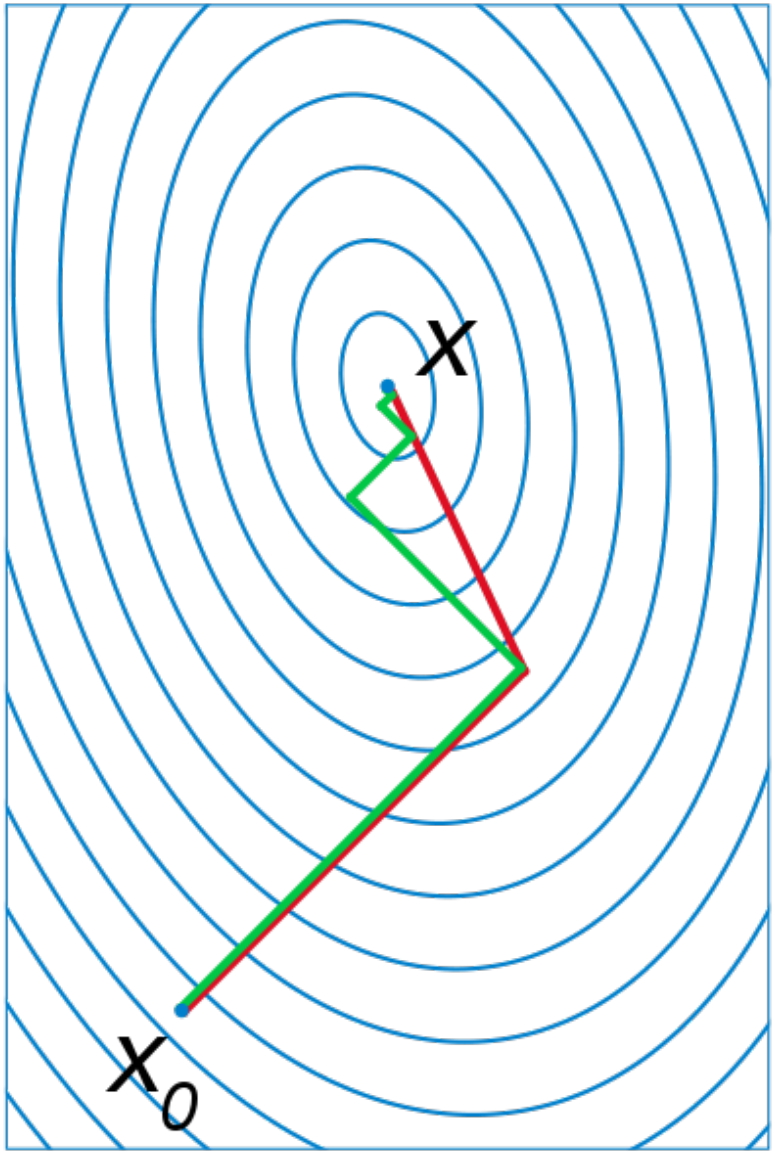


Conjugate Gradient

For the system $\mathbf{A}\,\mathbf{x} = \mathbf{b}$, vectors $\mathbf{p}_i$ and $\mathbf{p}_j$ are said to be mutually conjugate with respect to $\mathbf{A}$ if

$$\mathbf{p}_i^T\,\mathbf{A}\,\mathbf{p}_j = 0$$

If we build the CG steps so that they satisfy this relationship, then it turns out that these steps do what we want!



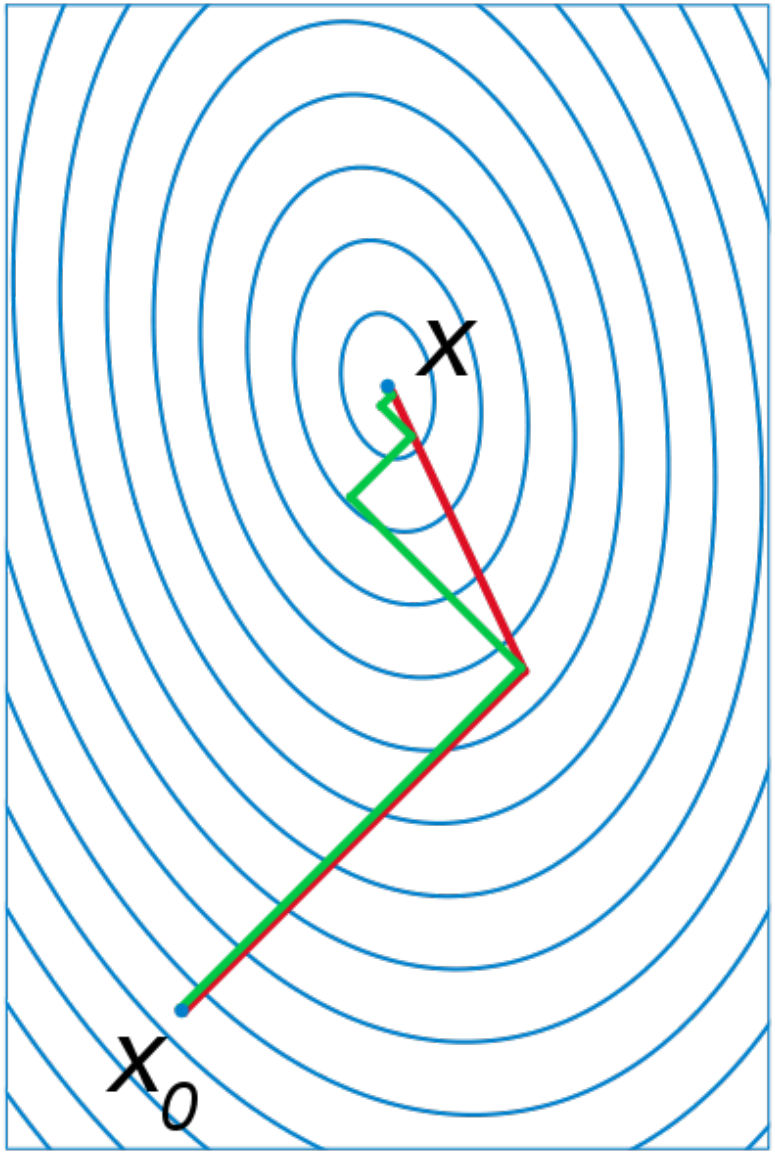
Conjugate Gradient

For the system $\mathbf{A}\,\mathbf{x} = \mathbf{b}$, vectors $\mathbf{p}_i$ and $\mathbf{p}_j$ are said to be mutually conjugate with respect to $\mathbf{A}$ if

$$\mathbf{p}_i^T\,\mathbf{A}\,\mathbf{p}_j = 0$$

If we build the CG steps so that they satisfy this relationship, then it turns out that these steps do what we want!

Note - CG works on symmetric positive definite matrices **A**. What do we do for a general **G**?



Conjugate Gradient Least Squares

Basically, we need to turn our general problem

$$\mathbf{G\,m = d}$$

into a symmetric positive definite system so that we can solve it with the Conjugate Gradient method.

Conjugate Gradient Least Squares

Basically, we need to turn our general problem

$$\mathbf{G}\,\mathbf{m} = \mathbf{d}$$

into a symmetric positive definite system so that we can solve it with the Conjugate Gradient method.

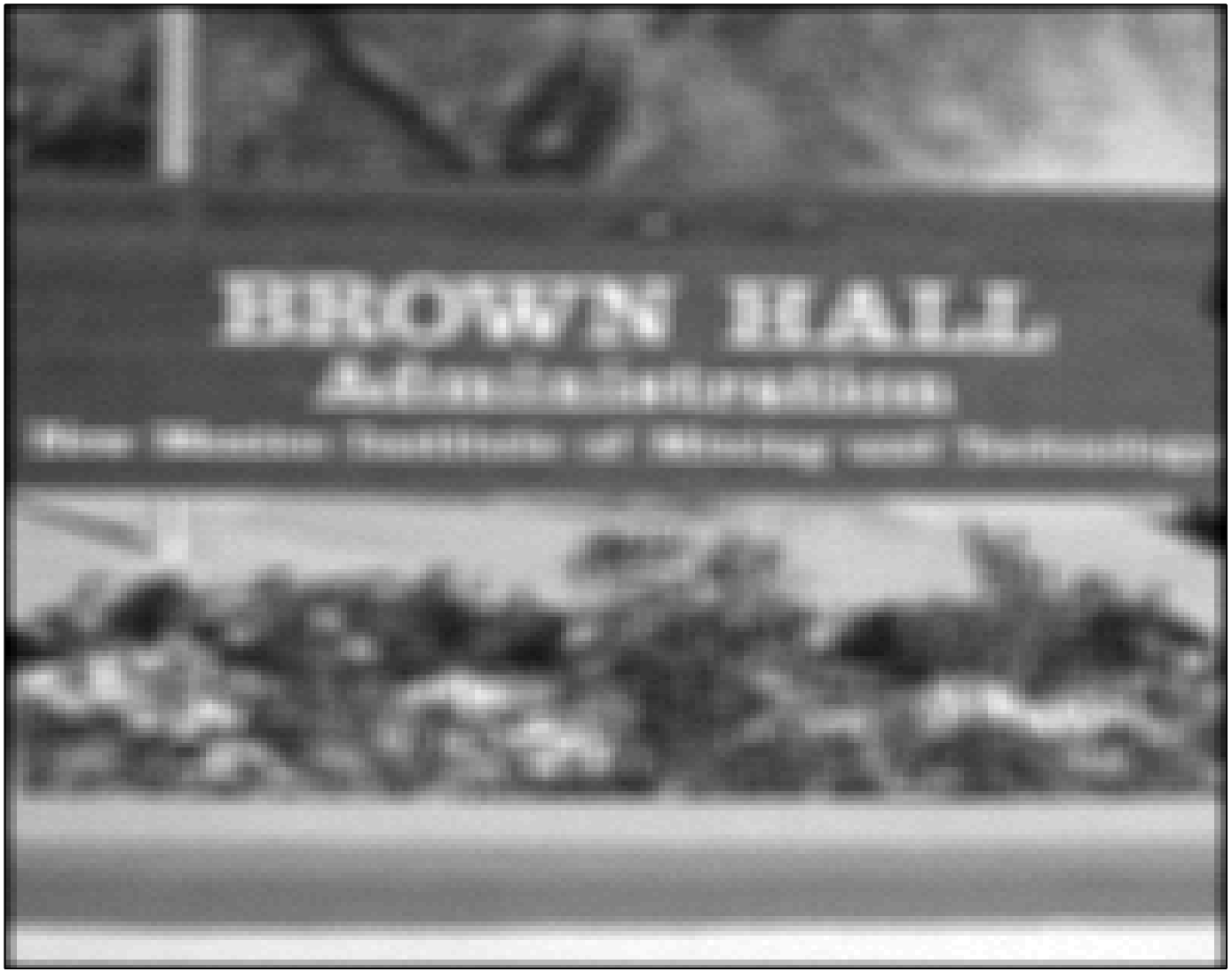
Two options:

1. Form the normal equations, $\mathbf{G}^T\,\mathbf{G}\,\mathbf{m} = \mathbf{G}^T\,\mathbf{d}$

2. Solve a regularized system such as $\min \left\| \begin{bmatrix} \mathbf{G} \\ \alpha\mathbf{L} \end{bmatrix} \mathbf{m} - \begin{bmatrix} \mathbf{d} \\ \mathbf{0} \end{bmatrix} \right\|_2^2$

QUESTIONS?

Conjugate Gradient Least Squares - Application to image deblurring



Conjugate Gradient Least Squares - Application to image deblurring

Example 6.3

A commonly used mathematical model of image blurring involves the two-dimensional convolution of the true image $I_{\text{true}}(x, y)$ with a **point spread function**, $\Psi(u, v)$ [14]:

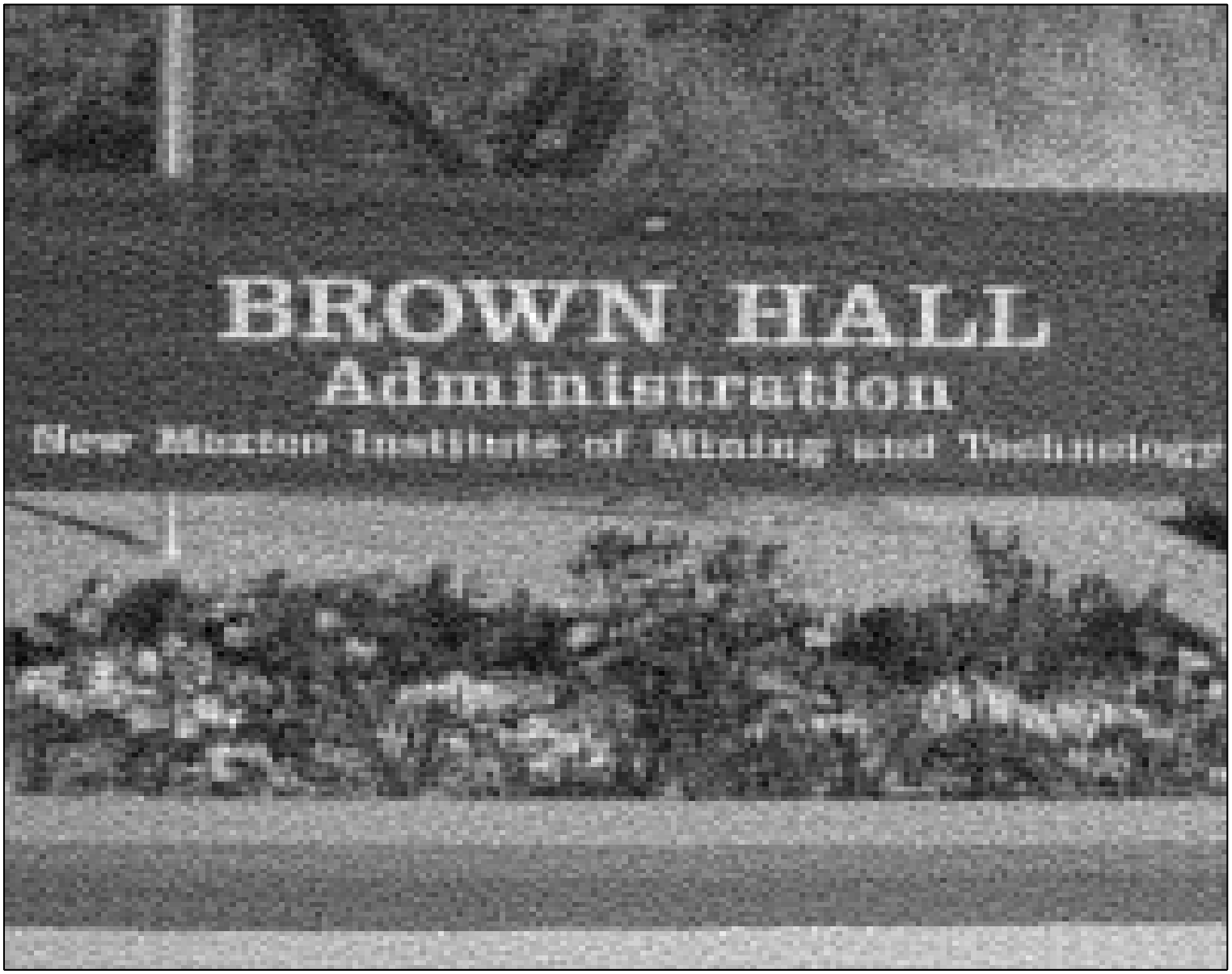
$$I_{\text{blurred}}(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} I_{\text{true}}(x-u, y-v)\Psi(u, v)\, du\, dv. \tag{6.77}$$

Here the point spread function shows how a point in the true image is altered in the blurred image. A point spread function that is commonly used to represent the blurring that occurs because an image is out of focus is the **Gaussian point spread function**

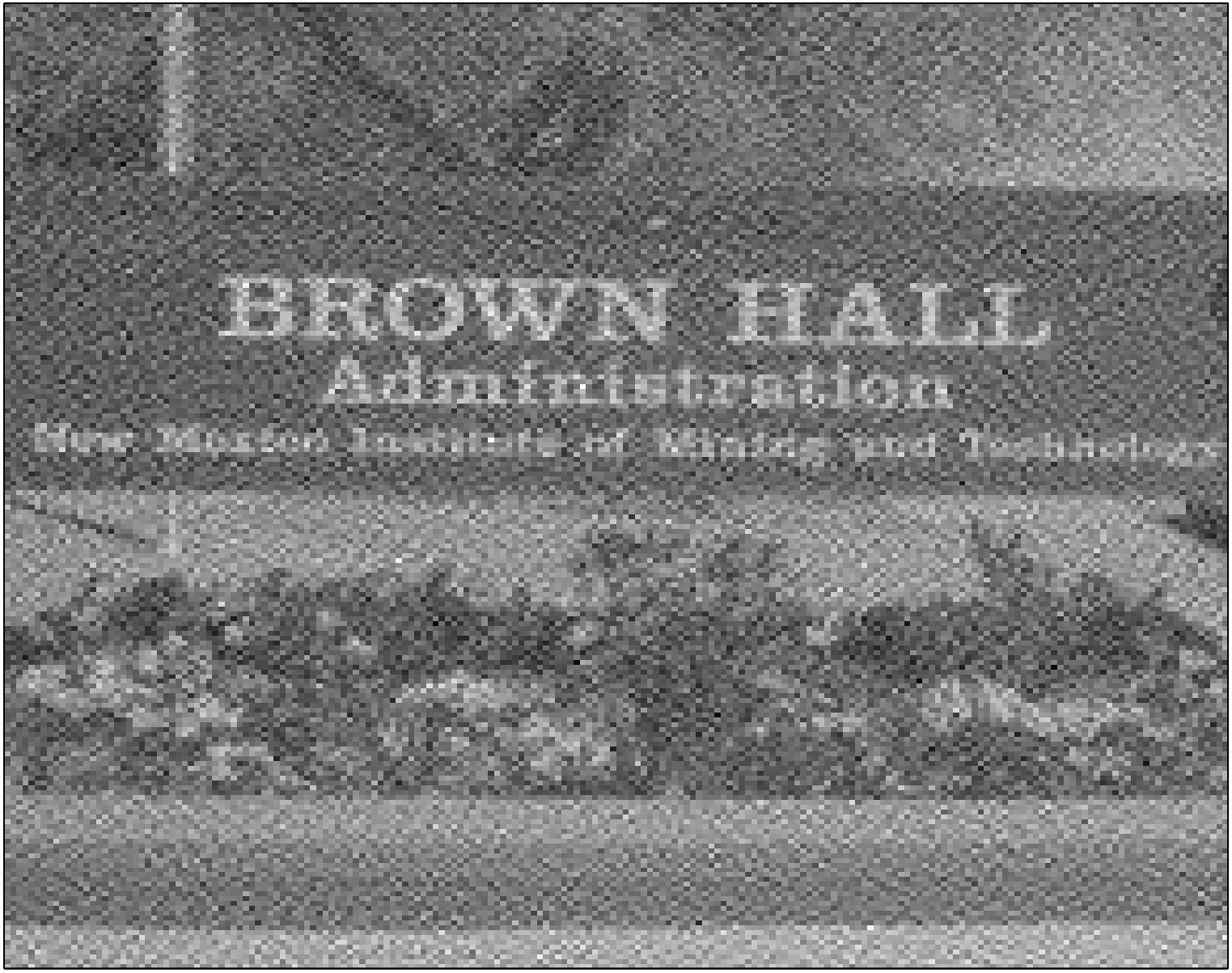
$$\Psi(u, v) = e^{\frac{u^2+v^2}{2\sigma^2}}. \tag{6.78}$$

Here the parameter σ controls the relative width of the point spread function. In practice, the blurred image and point spread function are discretized into pixels. In theory, Ψ is nonzero for all values of u and v. However, it becomes small quickly as u and v increase. If we set small values of Ψ to 0, then the $\mathbf{G}$ matrix in the discretized problem will be sparse.

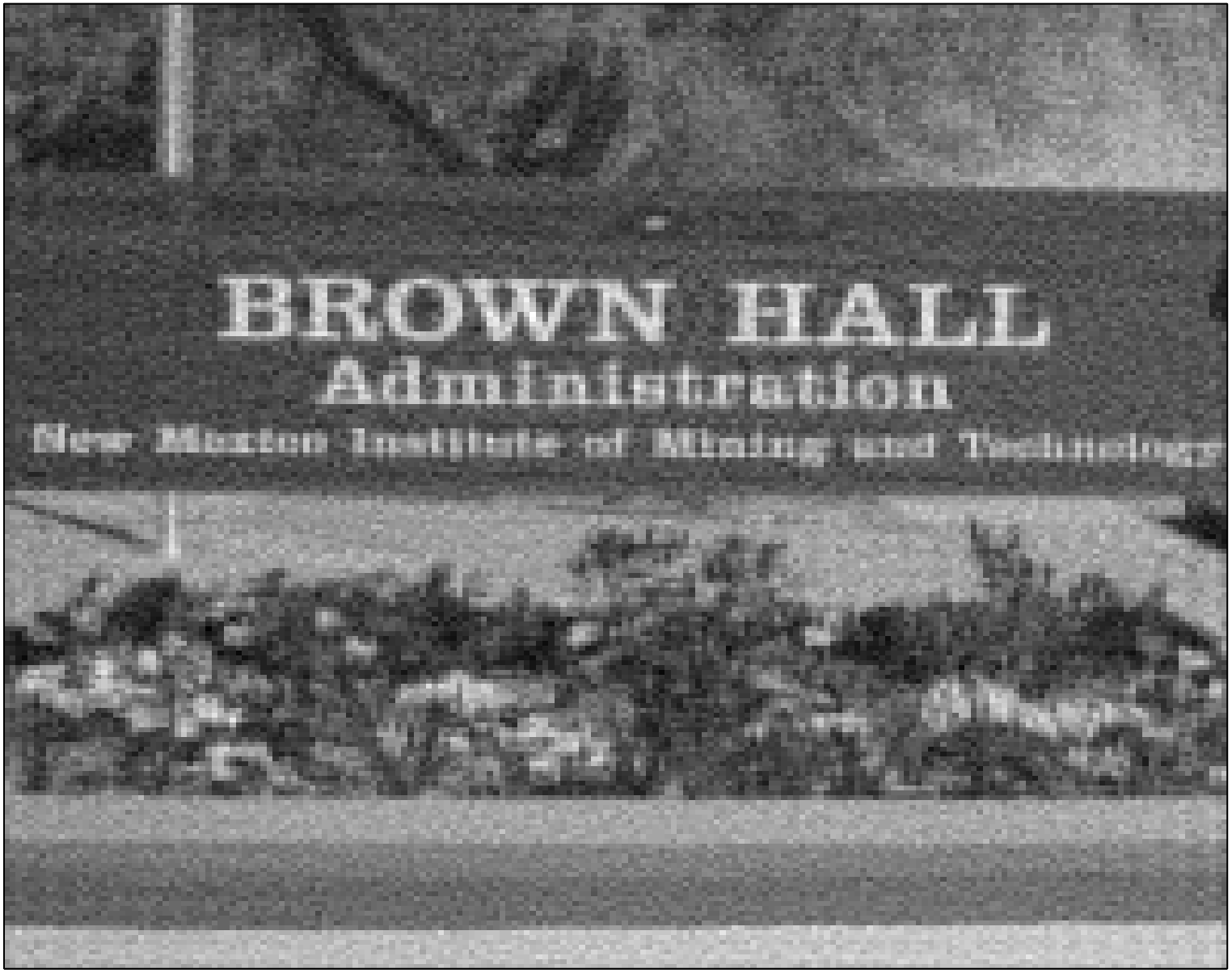
CGLS, 30 iterations, no explicit regularization



CGLS, 100 iterations, no explicit regularization



CGLS, 200 iterations, with explicit regularization



QUESTIONS?