Algebraic Methods for Computed Tomography

Per Christian Hansen

DTU Compute
Department of Applied Mathematics and Computer Science
Technical University of Denmark

About Me

- Professor of Scientific Computing (since 1996).
- Research: inverse problems, tomography, regularization algorithms, matrix computations, image deblurring, Matlab software, . . .
- Author of several Matlab software packages.
- Author of four books.

Rank-Deficient and Discrete Ill-Posed Problems
Numerical Aspects of Linear Inversion
Per Christian Hansen
siam

Fundamentals of Algorithms
Deblurring Images
Matrices, Spectra, and Filtering
Per Christian Hansen
James G Nagy
Dianne P O'Leary
siam

Fundamentals of Algorithms
Discrete Inverse Problems
Insight and Algorithms
Per Christian Hansen
siam

Least Squares Data Fitting
WITH APPLICATIONS
Per Christian Hansen, Victor Pereyra, and Godela Scherer

- Head of the project High-Definition Tomography, funded by an ERC Advanced Research Grant (www.compute.dtu.dk/∼pcha/HDtomo).

HDtomo
HIGH - DEFINITION TOMOGRAPHY

Plan for Today

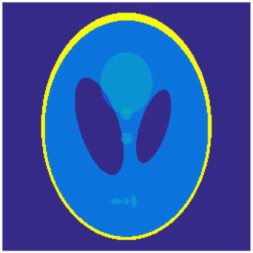
1. A bit of motivation.
2. The algebraic formulation.
3. Matrix notation and interpretation.
4. Kaczmarz's method (also known as ART) – fully sequential.
5. Cimmino's method and vatiants – fully simultaneous.
6. More linear algebra: null space and least squares.
7. The optimization viewpoint.

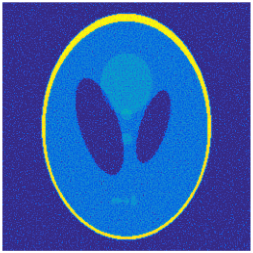
Points to take home today:

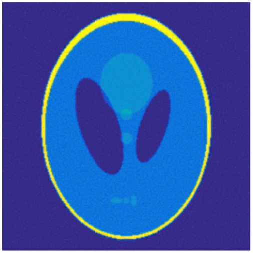
- Linear algebra provides a concise framework for formulating the algorithms.
- Convergence analysis of iterative algebraic methods:
  - Kaczmarz's method = ART converges for consistent problems only.
  - Cimmino's method always converges.
- Be careful with the null space.
- Least squares problems always have a solution.
- The optimization viewpoint leads to a broad class of iterative methods.

FBP: Filtered Back Projection

- This is *the classical* method for 2D reconstructions.
- There are similar methods for 3D, such as FDK.
- Many year of use $\rightarrow$ lots of *practical experience*.
- The FBP method is *very fast* (it uses the Fast Fourier Transform)!
- The FBP method has *low memory requirements*.
- With many data, FBP gives very good results.
- Example with 3% noise:

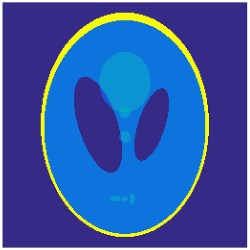

Phantom

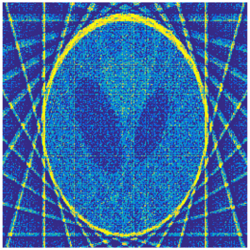

FBP 180 projections

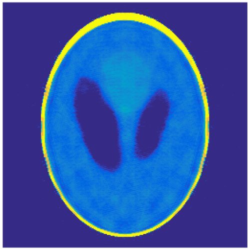

FBP 1000 projections

FBP Versus Algebraic Methods

- Limited data, or nonuniform distribution of projection angles or rays $\rightarrow$ *artifacts* appear in FBP reconstructions.
- Difficult to incorporate constraints (e.g., nonnegativity) in FBP.
- Algebraic methods are more flexible and adaptive.
- Same example with 3% noise and projection angles $15^\circ, 30^\circ, \ldots, 180^\circ$:



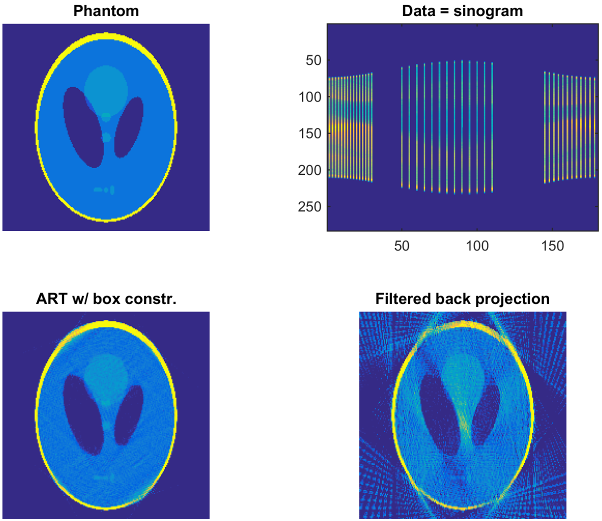




Algebraic Reconstruction Technique, box constraints (pixel values $\in$ [0,1]).

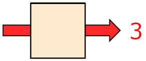
Another Motivating Example: Missing Data

Irregularly spaced angles & "missing" angles also cause difficulties for FBP:



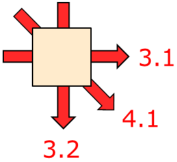
The Simplest Algebraic Problem

One unknown, no noise:



$$1 \cdot x \ = 3$$

Now with noise in the data – compute a weighted average:



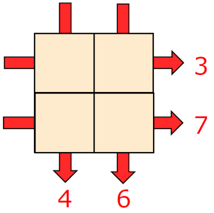
$$\begin{pmatrix} 1 \\ 1 \\ \sqrt{2} \end{pmatrix} x = \begin{pmatrix} 3.1 \\ 3.2 \\ 4.1 \end{pmatrix} \qquad x = 3.025$$

We know from statistics that solution's variance is inversely proportional to the number of data. So more data is better.

Let us immediately continue with a 2×2 image . . .

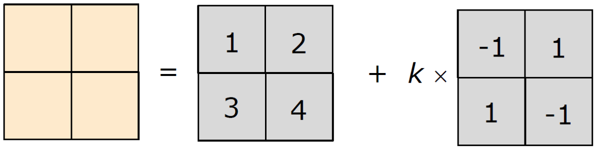
A “Sudoku” Problem

Four unknowns, four rays → system of linear equations $\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{b}$:



$$\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 3 \\ 7 \\ 4 \\ 6 \end{pmatrix}$$

Unfortunately there are infinitely many solutions, with $k \in \mathbb{R}$:

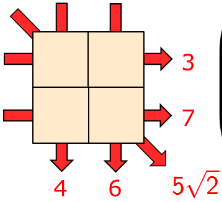


(There is an arbitrary component in the null space of the matrix $\boldsymbol{A}$.)

More Data Gives a Unique Solution

With *enough rays* the problem has a unique solution.

Here, one more ray is enough to ensure a full-rank matrix:



$$\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ \sqrt{2} & 0 & 0 & \sqrt{2} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 3 \\ 7 \\ 4 \\ 6 \\ 5\sqrt{2} \end{pmatrix}$$

The "difficulties" associated with the discretized tomography problem are closely linked with properties of the coefficient matrix $\boldsymbol{A}$:

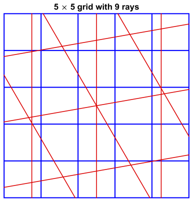
- The *sensitivity* of the solution to the data errors is characterized by the **condition number** $\kappa = \|\boldsymbol{A}\|_2 \cdot \|\boldsymbol{A}^{-1}\|_2$ (not discussed today).
- The *uniqueness* of the solution is characterized by the **rank** of $\boldsymbol{A}$, the number of linearly independent row or columns.

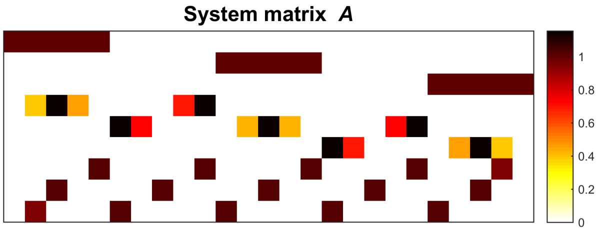
A Look at the System Matrix

Recall that our algebraic model is a system of linear equations

$$\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{b}$$

with a very **sparse** *system matrix* $\boldsymbol{A}$. Example: 5 × 5 pixels and 9 rays:



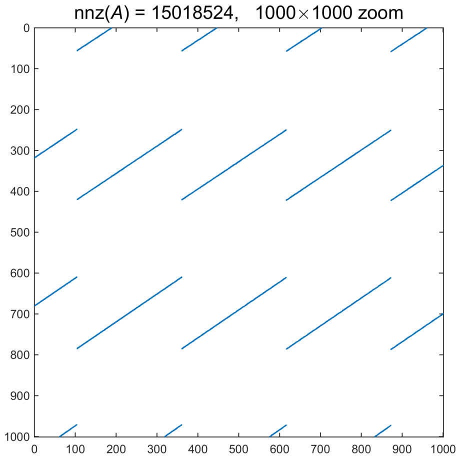


The System Matrix is Very Sparse

Another example: 256×256 pixels and 180 projections with 362 rays each.

The system matrix $\boldsymbol{A}$ is $65,160 \times 65,536$ and has $\approx 4.27 \cdot 10^9$ elements.

There are $15,018,524$ nonzero elements corresponding to a fill of 0.35%.



Algebraic Reconstruction Methods

- In principle, all we need to do in the algebraic formulation is to solve the large sparse linear system $\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{b}$:

  $$\text{Math: } \boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}, \qquad \text{MATLAB: } \texttt{x} = \texttt{A}\backslash\texttt{b}.$$

  How hard can that be?
- Actually, this can be a formidable task if we try do use a traditional approach such as Gaussian elimination.
- Researchers in tomography have therefore focused on the use of *iterative solvers* – and they have rediscovered many methods developed by mathematicians . . .
- In tomography they are called **algebraic reconstruction methods**. They are much more flexible than FBP, but at a higher computational cost!

Some Algebraic Reconstruction Methods

Fully Sequential Methods

- Kaczmarz's method + variants.
- These are row-action methods: they update the solution using one row of $\boldsymbol{A}$ at a time.
- Fast convergence.

Fully Simultaneous Methods

- Landweber, Cimmino, CAV, DROP, SART, SIRT, ...
- These methods use all the rows of $\boldsymbol{A}$ simultaneously in one iteration (i.e., they are based on matrix multiplications).
- Slower convergence.

Krylov subspace methods (rarely used in tomography)

- CGLS, LSQR, GMRES, ...
- These methods are also based on matrix multiplications.

Matrix Notation and Interpretation

Notation:

$$\boldsymbol{A} = \begin{pmatrix} | & | & & | \\ \boldsymbol{c}_1 & \boldsymbol{c}_2 & \cdots & \boldsymbol{c}_n \\ | & | & & | \end{pmatrix} = \begin{pmatrix} \text{---} & \boldsymbol{r}_1 & \text{---} \\ & \vdots & \\ \text{---} & \boldsymbol{r}_m & \text{---} \end{pmatrix},$$

The matrix $\boldsymbol{A}$ maps the discretized absorption coefficients (the vector $\boldsymbol{x}$) to the data in the detector pixels (the elements of the vector $\boldsymbol{b}$) via:

$$\boldsymbol{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix} = \boldsymbol{A}\,\boldsymbol{x} = \underbrace{x_1\,\boldsymbol{c}_1 + x_2\,\boldsymbol{c}_2 + \cdots + x_n\,\boldsymbol{c}_n}_{\text{linear combination of columns}} = \begin{pmatrix} \boldsymbol{r}_1 \cdot \boldsymbol{x} \\ \boldsymbol{r}_2 \cdot \boldsymbol{x} \\ \vdots \\ \boldsymbol{r}_m \cdot \boldsymbol{x} \end{pmatrix}.$$

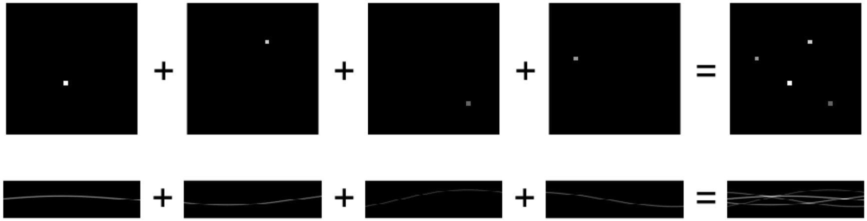
The ith row of $\boldsymbol{A}$ maps $\boldsymbol{x}$ to detector element i via the ith ray:

$$b_i = \boldsymbol{r}_i \cdot \boldsymbol{x} = \sum_{j=1}^{n} a_{ij}\,x_j, \qquad i = 1, 2, \ldots, m.$$

Example of Column Interpretation

A 32×32 image has four nonzero pixels with intensities 1, 0.8, 0.6, 0.4. In the vector $\boldsymbol{x}$ these four pixels correspond to entries 468, 618, 206, 793. Hence the sinogram, represented as a vector $\boldsymbol{b}$, takes the form

$$\boldsymbol{b} = 0.6\, \boldsymbol{c}_{206} + 1.0\, \boldsymbol{c}_{468} + 0.8\, \boldsymbol{c}_{618} + 0.4\, \boldsymbol{c}_{793}.$$



Forward and Back Again

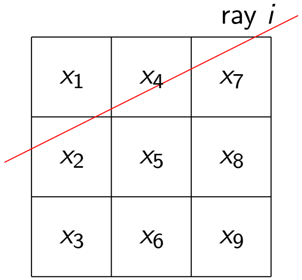
Forward projection. Consider ray i:

a_{ij} = length of ray i in pixel j

$\boldsymbol{r}_i = [\ a_{i1}\ a_{i2}\ 0\ a_{i4}\ 0\ 0\ a_{i7}\ 0\ 0\ ]$

$b_i = \boldsymbol{r}_i \cdot \boldsymbol{x} = a_{i1}x_1 + a_{i2}x_2 + a_{i4}x_4 + a_{i7}x_7$

The forward projection $\boldsymbol{b} = \boldsymbol{A}\,\boldsymbol{x}$ maps all image pixels x_j along the ray to the detector data b_i.

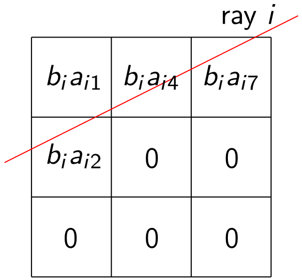


Back projection. Another name for multiplication with $\boldsymbol{A}^T$:

$$\boldsymbol{A}^T\boldsymbol{b} = \sum_{i=1}^{m} b_i\,\boldsymbol{r}_i = \sum_{i=1}^{m} \begin{pmatrix} b_i a_{i1} \\ b_i a_{i2} \\ \vdots \end{pmatrix}$$

Each data b_i is "distributed back" along the ith ray to the pixels, with "weights" $= a_{ij}$:

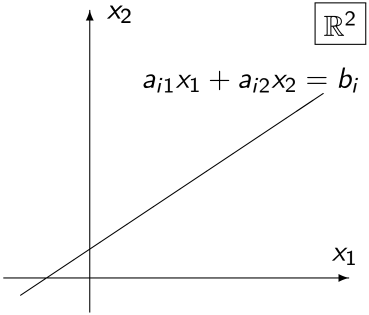
$b_i\,\boldsymbol{r}_i = [\ b_i a_{i1}\ b_i a_{i2}\ 0\ b_i a_{i4}\ 0\ 0\ b_i a_{i7}\ 0\ 0\ ]^T$

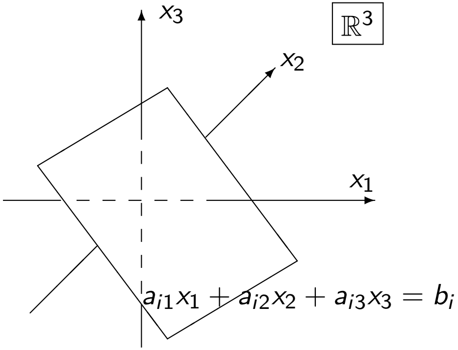


Geometric Interpretation of $\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{b}$

$$\begin{aligned}
\boldsymbol{r}_1 \cdot \boldsymbol{x} &= a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\
\boldsymbol{r}_2 \cdot \boldsymbol{x} &= a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\
&\vdots \\
\boldsymbol{r}_m \cdot \boldsymbol{x} &= a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &= b_m.
\end{aligned}$$

Each equation $\boldsymbol{r}_i \cdot \boldsymbol{x} = b_i$ defines an *affine hyperplane* in $\mathbb{R}^n$:

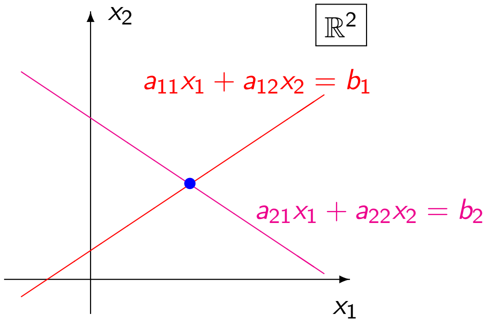




Geometric Interpretation of the Solution

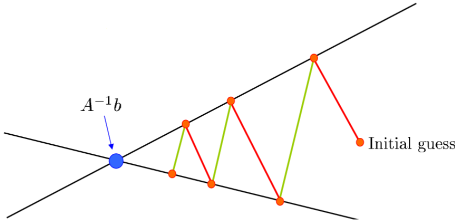
Assuming that the solution to $\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{b}$ is unique, it is the point $\boldsymbol{x} \in \mathbb{R}^m$ where all the m affine hyperplanes intersect.

Example with $m = n = 2$:



Kaczmarz's Method = Algebraic Reconstruction Technique

A simple, efficient iterative method based on the geometric interpretation.

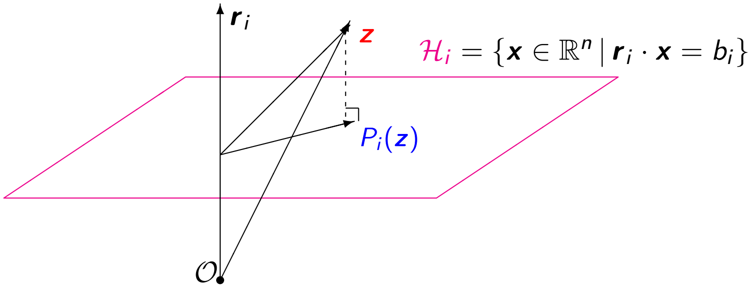


In each iteration, and in a *cyclic fashion*, compute the new iteration vector such that precisely one of the equations is satisfied.

This is achieved by projecting the current iteration vector $\boldsymbol{x}$ on one of the hyperplanes $\boldsymbol{r}_i \cdot \boldsymbol{x} = b_i$ for $i = 1, 2, \ldots, m, 1, 2, \ldots, m, 1, 2, \ldots$

Originally proposed in 1937, and independently suggested under the name ART by Gordon, Bender & Herman in 1970 for tomographic reconstruction.

Orthogonal Projection on Affine Hyperplane



$\boldsymbol{r}_i$
$\boldsymbol{z}$
$\mathcal{H}_i = \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{r}_i \cdot \boldsymbol{x} = b_i\}$
$P_i(\boldsymbol{z})$
$\mathcal{O}$

The orthogonal projection $P_i(\boldsymbol{z})$ of an arbitrary point $\boldsymbol{z}$ on the affine hyperplane $\mathcal{H}_i$ defined by $\boldsymbol{r}_i \cdot \boldsymbol{x} = b_i$ is given by:

$$P_i(\boldsymbol{z}) = \boldsymbol{z} + \frac{b_i - \boldsymbol{r}_i \cdot \boldsymbol{z}}{\|\boldsymbol{r}_i\|_2^2}\,\boldsymbol{r}_i, \qquad \|\boldsymbol{r}_i\|_2^2 = \boldsymbol{r}_i \cdot \boldsymbol{r}_i.$$

In words, we scale the row vector $\boldsymbol{r}_i$ by $(b_i - \boldsymbol{r}_i \cdot \boldsymbol{z})/\|\boldsymbol{r}_i\|_2^2$ and add it to $\boldsymbol{z}$.

Kaczmarz's Method

We thus obtain the following algebraic formulation:

```
Basic Kaczmarz algorithm
$\boldsymbol{x}^{(0)}$ = initial vector
for $k = 0, 1, 2, \ldots$
    $i = k \pmod{m}$
    $\boldsymbol{x}^{(k+1)} = P_i\big(\boldsymbol{x}^{(k)}\big) = \boldsymbol{x}^{(k)} + \dfrac{b_i - \boldsymbol{r}_i \cdot \boldsymbol{x}^{(k)}}{\|\boldsymbol{r}_i\|_2^2}\, \boldsymbol{r}_i$
end
```

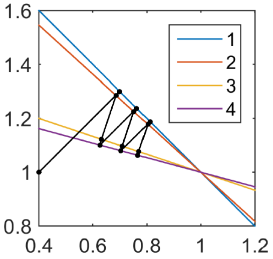
Each time we have performed m iterations of this algorithm, we have performed one *sweep* over the rows of $\boldsymbol{A}$. Other choices of sweeps:

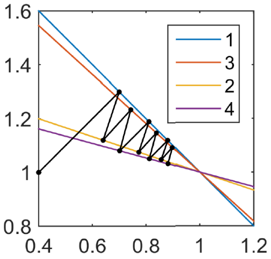
- **Symmetric Kaczmarz:** $i = 1, 2, \ldots, m{-}1, m, m{-}1, \ldots, 3, 2.$
- **Randomized Kaczmarz:** select row i randomly, possibly with probability proportional to the row norm $\|\boldsymbol{r}_i\|_2$.

Convergence Issues

The convergence of Kaczmarz's method is quite obvious from the graph on slide 19 – but can we say more?

Difficulty: the *ordering* of the rows of $\boldsymbol{A}$ influences the *convergence rate:*





$$\begin{pmatrix} 1.0 & 1.0 \\ 1.0 & 1.1 \\ 1.0 & 3.0 \\ 1.0 & 3.7 \end{pmatrix} \boldsymbol{x} = \begin{pmatrix} 2.0 \\ 2.1 \\ 4.0 \\ 4.7 \end{pmatrix}$$

The ordering 1–3–2–4 is preferable: almost twice as fast.

Convergence of Kaczmarz's Method

One way to avoid the difficulty associated with influence of the ordering of the rows is to assume that we *select the rows randomly*.

For simplicity, assume that $\boldsymbol{A}$ is invertible and that all rows of $\boldsymbol{A}$ are scaled to unit 2-norm. Then the expected value $\mathcal{E}(\cdot)$ of the error norm satisfies:

$$\mathcal{E}\Big(\|\boldsymbol{x}^{(k)} - \bar{\boldsymbol{x}}\|_2^2\Big) \leq \left(1 - \frac{1}{n\,\kappa^2}\right)^k \|\boldsymbol{x}^{(0)} - \bar{\boldsymbol{x}}\|_2^2, \quad k = 1, 2, \dots\ ,$$

where $\bar{\boldsymbol{x}} = \boldsymbol{A}^{-1}\boldsymbol{b}$ and $\kappa = \|\boldsymbol{A}\|_2\,\|\boldsymbol{A}^{-1}\|_2$. This is **linear convergence**.

When κ is large we have

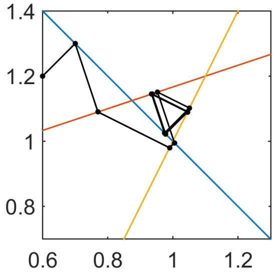
$$\left(1 - \frac{1}{n\,\kappa^2}\right)^k \approx 1 - \frac{k}{n\,\kappa^2}.$$

After $k = n$ steps, corresp. to one sweep, the reduction factor is $1 - 1/\kappa^2$. Note that there are often orderings for which the convergence is faster!

Cyclic Convergence

So far we have assumed that there is a unique solution $\bar{\boldsymbol{x}} = \boldsymbol{A}^{-1}\boldsymbol{b}$ that satisfies $\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{b}$, i.e., all the affine hyperplanes associated with the rows of $\boldsymbol{A}$ intersect in a single point.

What happens when this is not true? $\rightarrow$ *cyclic convergence:*



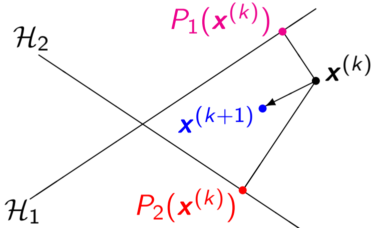
$m = 3, \; n = 2$

Kaczmarz's method can be brought to converge to a unique point, and we will discuss the modified algorithm later today.

From Sequential to Simultaneous Updates

Karzmarz's method accesses the rows sequentially. **Cimmino's method** accesses the rows *simultaneously* and computes the next iteration vector as the average of the all the projections of the previous iteration vector:

$$\begin{aligned} \boldsymbol{x}^{(k+1)} &= \frac{1}{m}\sum_{i=1}^{m} P_i\big(\boldsymbol{x}^{(k)}\big) = \frac{1}{m}\sum_{i=1}^{m}\Big(\boldsymbol{x}^{(k)} + \frac{b_i - \boldsymbol{r}_i \cdot \boldsymbol{x}^{(k)}}{\|\boldsymbol{r}_i\|_2^2}\,\boldsymbol{r}_i\Big) \\ &= \boldsymbol{x}^{(k)} + \frac{1}{m}\sum_{i=1}^{m}\frac{b_i - \boldsymbol{r}_i \cdot \boldsymbol{x}^{(k)}}{\|\boldsymbol{r}_i\|_2^2}\,\boldsymbol{r}_i . \end{aligned}$$



Matrix formulation of Cimmino's Method

We can write the updating in our matrix-vector formalism as follows

$$
\begin{aligned}
\boldsymbol{x}^{(k+1)} &= \boldsymbol{x}^{(k)} + \frac{1}{m}\sum_{i=1}^{m} \frac{b_i - \boldsymbol{r}_i \cdot \boldsymbol{x}^{(k)}}{\|\boldsymbol{r}_i\|_2^2}\, \boldsymbol{r}_i \\
&= \boldsymbol{x}^{(k)} + \frac{1}{m} \begin{pmatrix} \frac{\boldsymbol{r}_1}{\|\boldsymbol{r}_1\|_2^2} & \cdots & \frac{\boldsymbol{r}_m}{\|\boldsymbol{r}_m\|_2^2} \end{pmatrix} \begin{pmatrix} b_1 - \boldsymbol{r}_1 \cdot \boldsymbol{x}^{(k)} \\ \vdots \\ b_m - \boldsymbol{r}_m \cdot \boldsymbol{x}^{(k)} \end{pmatrix} \\
&= \boldsymbol{x}^{(k)} + \frac{1}{m} \begin{pmatrix} \boldsymbol{r}_1 \\ \vdots \\ \boldsymbol{r}_m \end{pmatrix}^T \begin{pmatrix} \|\boldsymbol{r}_1\|_2^{-2} & & \\ & \ddots & \\ & & \|\boldsymbol{r}_m\|_2^{-2} \end{pmatrix} \left(\boldsymbol{b} - \begin{pmatrix} \boldsymbol{r}_1 \\ \vdots \\ \boldsymbol{r}_m \end{pmatrix} \boldsymbol{x}^{(k)} \right) \\
&= \boldsymbol{x}^{(k)} + \boldsymbol{A}^T \boldsymbol{M}^{-1} \big(\boldsymbol{b} - \boldsymbol{A}\, \boldsymbol{x}^{(k)}\big),
\end{aligned}
$$

where we introduced the diagonal matrix $\boldsymbol{M} = \mathrm{diag}\big(m\|\boldsymbol{r}_i\|_2^2\big)$.

Cimmino's Method

We thus obtain the following formulation:

```
Basic Cimmino algorithm
$x^{(0)}$ = initial vector
for $k = 0, 1, 2, \ldots$
    $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \boldsymbol{A}^T \boldsymbol{M}^{-1} \big(\boldsymbol{b} - \boldsymbol{A}\,\boldsymbol{x}^{(k)}\big)$
end
```

Note that one iteration here involves all the rows of $\boldsymbol{A}$, while one iteration in Kaczmarz's method involves a single row.

Therefore, the computational work in one Cimmino iteration is equivalent to m iterations (a sweep over all the rows) in Kaczmarz's basic algorithm.

The issue of finding a good row ordering is, of course, absent from Cimmino's method.

Convergence Study

Assume $\boldsymbol{x}^{(0)} = \boldsymbol{0}$ and let $\boldsymbol{I}$ denote the $n \times n$ identity matrix; then:

$$\begin{aligned} \boldsymbol{x}^{(k+1)} &= \sum_{j=0}^{k} (\boldsymbol{I} - \boldsymbol{A}^T \boldsymbol{M}^{-1} \boldsymbol{A})^j \boldsymbol{A}^T \boldsymbol{M}^{-1} \boldsymbol{b} \\ &= \left(\boldsymbol{I} - (\boldsymbol{I} - \boldsymbol{A}^T \boldsymbol{M}^{-1} \boldsymbol{A})^{k+1} \right) (\boldsymbol{A}^T \boldsymbol{M}^{-1} \boldsymbol{A})^{-1} \boldsymbol{A}^T \boldsymbol{M}^{-1} \boldsymbol{b}. \end{aligned}$$

If $\boldsymbol{A}$ is invertible then

$$(\boldsymbol{A}^T \boldsymbol{M}^{-1} \boldsymbol{A})^{-1} \boldsymbol{A}^T \boldsymbol{M}^{-1} \boldsymbol{b} = \boldsymbol{A}^{-1} \boldsymbol{M} \boldsymbol{A}^{-T} \boldsymbol{A}^T \boldsymbol{M}^{-1} \boldsymbol{b} = \boldsymbol{A}^{-1} \boldsymbol{b}.$$

Moreover, the largest eigenvalue of the symmetric matrix $\boldsymbol{I} - \boldsymbol{A}^T \boldsymbol{M}^{-1} \boldsymbol{A}$ is strictly smaller than one, and therefore

$$\left(\boldsymbol{I} - (\boldsymbol{I} - \boldsymbol{A}^T \boldsymbol{M}^{-1} \boldsymbol{A})^{k+1} \right) \rightarrow \boldsymbol{I} \qquad \text{for} \qquad k \rightarrow \infty.$$

Hence the iterates $\boldsymbol{x}^{(k)}$ converge to the solution $\bar{\boldsymbol{x}} = \boldsymbol{A}^{-1} \boldsymbol{b}$.

Convergence of Cimmino's Method

To simplify the result, assume that $\boldsymbol{A}$ is invertible and that the rows of $\boldsymbol{A}$ are scaled such that $\|\boldsymbol{A}\|_2^2 = m$. Then

$$\|\boldsymbol{x}^{(k)} - \bar{\boldsymbol{x}}\|_2^2 \le \left(1 - \frac{2}{1+\kappa^2}\right)^k \|\boldsymbol{x}^{(0)} - \bar{\boldsymbol{x}}\|_2^2$$

where $\bar{\boldsymbol{x}} = \boldsymbol{A}^{-1}\boldsymbol{b}$, $\kappa = \|\boldsymbol{A}\|_2 \, \|\boldsymbol{A}^{-1}\|_2$, and we have **linear convergence**.

When $\kappa \gg 1$ then we have the approximate upper bound

$$\|\boldsymbol{x}^{(k)} - \bar{\boldsymbol{x}}\|_2^2 \lesssim (1 - 2/\kappa^2)^k \, \|\boldsymbol{x}^{(0)} - \bar{\boldsymbol{x}}\|_2^2,$$

showing that in each iteration the error is reduced by a factor $1 - 2/\kappa^2$.

This is almost the same factor as in one sweep through the rows of $\boldsymbol{A}$ in Kaczmarz's method.

Rectangular and/or Rank Deficient Matrices

In tomography, $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ is almost always a rectangular matrix: $m \neq n$.

It is also very common that $\boldsymbol{A}$ does not have full rank.

We need to set the stage for treating such matrices.

The **rank** r of $\boldsymbol{A}$ is the number of linearly independent rows (equal to the number of linearly independent columns), and $r \leq \min(m, n)$.

The **range** $\mathcal{R}(\boldsymbol{A})$ is the linear subspace spanned by the columns of $\boldsymbol{A}$:

$$\mathcal{R}(\boldsymbol{A}) \equiv \{\boldsymbol{u} \in \mathbb{R}^m \,|\, \boldsymbol{u} = \alpha_1 \boldsymbol{c}_1 + \alpha_2 \boldsymbol{c}_2 + \cdots + \alpha_n \boldsymbol{c}_n, \text{ arbitrary } \alpha_j\}. \tag{1}$$

The **null space** $\mathcal{N}(\boldsymbol{A})$ is the linear subspace of all vectors mapped to zero:

$$\mathcal{N}(\boldsymbol{A}) \equiv \{\boldsymbol{v} \in \mathbb{R}^n \,|\, \boldsymbol{A}\,\boldsymbol{v} = \boldsymbol{0}\}. \tag{2}$$

The dimensions of the two subspaces are r and $n-r$, respectively.

A Small Example

Consider the 3×3 matrix

$$\boldsymbol{A} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}.$$

This matrix has rank $r = 2$ since the middle row is the average of the first and third rows which are linearly independent.

The range $\mathcal{R}(\boldsymbol{A})$ and null space $\mathcal{N}(\boldsymbol{A})$ consist of all vectors of the forms

$$\alpha_1 \begin{pmatrix} 1 \\ 4 \\ 7 \end{pmatrix} + \alpha_2 \begin{pmatrix} 3 \\ 6 \\ 9 \end{pmatrix} = \begin{pmatrix} \alpha_1 + 3\alpha_2 \\ 4\alpha_1 + 6\alpha_2 \\ 7\alpha_1 + 9\alpha_2 \end{pmatrix} \qquad \text{and} \qquad \alpha_3 \begin{pmatrix} 1 \\ -2 \\ 1 \end{pmatrix},$$

respectively, for arbitrary α_1, α_2, and α_3.

A Small Example, Continued

Consider two linear systems with the matrix $\boldsymbol{A}$ from the previous example:

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \boldsymbol{x} = \begin{pmatrix} 14 \\ 20 \\ 50 \end{pmatrix} \qquad \text{and} \qquad \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \boldsymbol{x} = \begin{pmatrix} 6 \\ 15 \\ 24 \end{pmatrix} .$$

The left system has no solution because $\boldsymbol{b} \notin \mathcal{R}(\boldsymbol{A})$; no matter which linear combination of the columns of $\boldsymbol{A}$ we create, we can never form this $\boldsymbol{b}$.

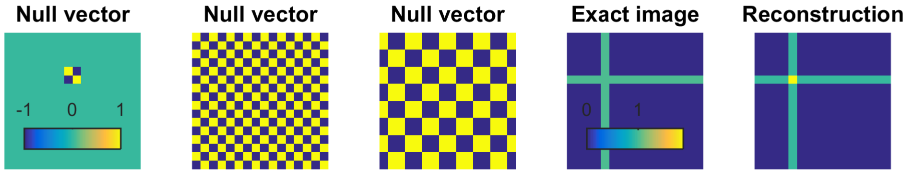
The right system has infinitely many solutions; any vector of the form

$$\boldsymbol{x} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} + \alpha \begin{pmatrix} 1 \\ -2 \\ 1 \end{pmatrix} , \quad \alpha \text{ arbitrary}$$

satisfies this equation. The arbitrary component is in the null space $\mathcal{N}(\boldsymbol{A})$.

Null Space Artifacts in Tomography I

Image has 16×16 pixels and we use 16 horizontal and 16 vertical X-rays $\rightarrow$ very under-determined system.



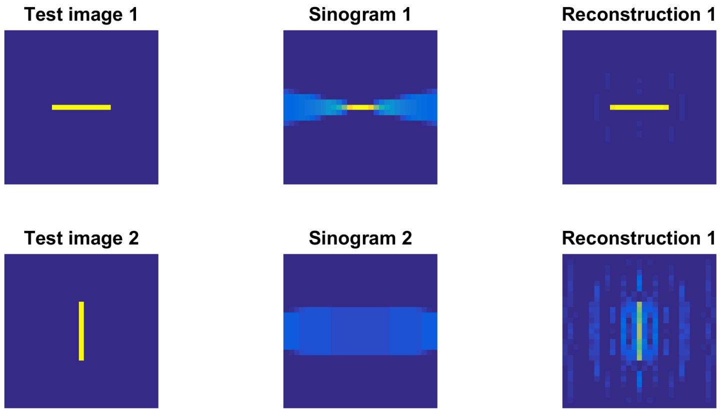
Left: three different vectors in the null space $\mathcal{N}(\boldsymbol{A})$.

Right: the exact ("ground truth") image $\bar{\boldsymbol{x}}$ and the reconstruction.

One pixel at the intersection of the vertical and horizontal "strips" has a large and incorrect value, and the values of the horizontal and vertical "strips" are slightly too low.

Null Space Artifacts in Tomography II

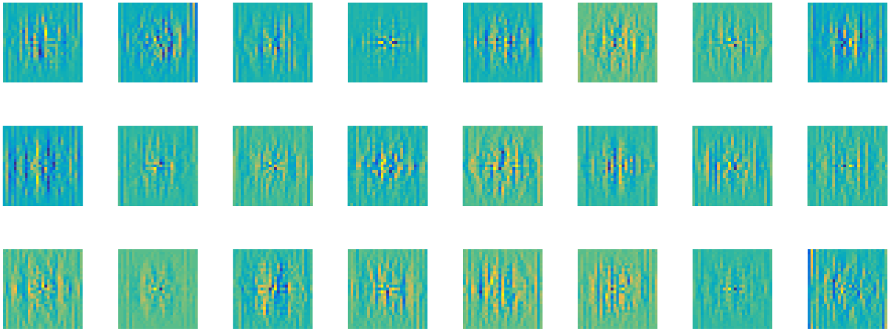
The mage has 29×29 pixels and we use projection angles in $[50^\circ, 130^\circ]$
$\rightarrow$ $\boldsymbol{A}$ is 841×841 and rank deficient; the dimension of $\mathcal{N}(\boldsymbol{A})$ is 24.



Both reconstructions are imperfect, and the vertical structure of the second test image is almost completely lost in the reconstruction.

Null Space Artifacts in Tomography III

Same example – the 24 images that span the null space $\mathcal{N}(\boldsymbol{A})$ represent information about missing vertical structures in the reconstruction:



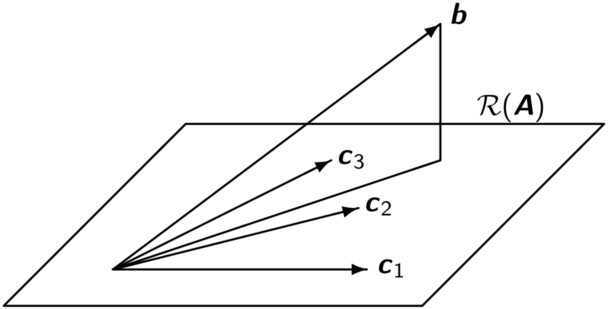
This illustrates that intuition and mathematics go hand-in-hand.

Consistent and Inconsistent Systems

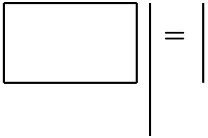
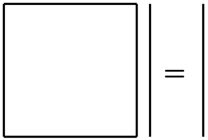
A system is *consistent* if there exists at least one $\boldsymbol{x}$ such that $\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{b}$, i.e., such that $\boldsymbol{b}$ is a linear combination of the columns $\boldsymbol{c}_i$ of $\boldsymbol{A}$.

This is equivalent to the requirement $\boldsymbol{b} \in \mathcal{R}(\boldsymbol{A})$.

Otherwise the system is *inconsistent*, $\boldsymbol{b} \notin \mathcal{R}(\boldsymbol{A})$, as shown below.

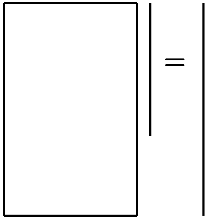


Overview of Systems: $m \leq n$

	Full rank	Rank deficient
$m < n$ Underdetermined 	$r = m$ Always consistent. Always infinitely many solutions.	$r < m$ Can be inconsistent. No solution or infinitely many solutions.
$m = n$ Square 	$r = m = n$ Always consistent. Always a unique solution.	$r < m = n$ Can be inconsistent. No solution or infinitely many solutions.

The system is *inconsistent* when $\boldsymbol{b} \notin \mathcal{R}(\boldsymbol{A})$.
There is a unique solution only if $r = m = n$.

Overview Of Systems: $m > n$

	Full rank	Rank deficient
$m > n$	$r = n$	$r < n$
Overdetermined 	Can be inconsistent. No solution or a unique solution.	Can be inconsistent. No solution or ininitely many solutions

The system is *inconsistent* when $\boldsymbol{b} \notin \mathcal{R}(\boldsymbol{A})$.

There is a unique solution only if $r = n$ and the system is consistent.

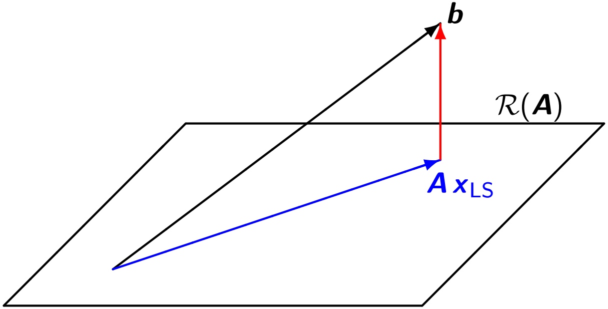
The Least Squares Solution

We must define a unique solution for inconsistent systems!

Assume that $\boldsymbol{b} = \boldsymbol{A}\,\bar{\boldsymbol{x}} + \boldsymbol{e}$ and $\boldsymbol{e}$ is zero-mean Gaussian noise. The best linear unbiased estimate of $\bar{\boldsymbol{x}}$ is the solution to the **least squares problem**:

$$\boldsymbol{x}_{\text{LS}} = \arg\min_{\boldsymbol{x}} {}^1\!/_2\, \|\boldsymbol{b} - \boldsymbol{A}\,\boldsymbol{x}\|_2^2,$$

and $\boldsymbol{x}_{\text{LS}}$ is unique when $r = n$. Geometrically, this corresponds to finding $\boldsymbol{x}_{\text{LS}}$ such that $\boldsymbol{A}\,\boldsymbol{x}_{\text{LS}}$ is orthogonal to the residual vector $\boldsymbol{b} - \boldsymbol{A}\,\boldsymbol{x}_{\text{LS}}$.


$\boldsymbol{b}$
$\mathcal{R}(\boldsymbol{A})$
$\boldsymbol{A}\,\boldsymbol{x}_{\text{LS}}$

Computing the Least Squares Solution

The requirement that $\boldsymbol{A}\,\boldsymbol{x}_{\mathrm{LS}} \perp (\boldsymbol{b} - \boldsymbol{A}\,\boldsymbol{x}_{\mathrm{LS}})$ leads to:

$$\left(\boldsymbol{A}\,\boldsymbol{x}_{\mathrm{LS}}\right)^T\left(\boldsymbol{b} - \boldsymbol{A}\,\boldsymbol{x}_{\mathrm{LS}}\right) = 0 \quad \Leftrightarrow \quad \boldsymbol{x}_{\mathrm{LS}}^T\left(\boldsymbol{A}^T\boldsymbol{b} - \boldsymbol{A}^T\boldsymbol{A}\,\boldsymbol{x}_{\mathrm{LS}}\right) = 0$$

which means that $\boldsymbol{x}_{\mathrm{LS}}$ is the solution to the *normal equations:*

$$\boldsymbol{A}^T\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{A}^T\boldsymbol{b} \qquad \Rightarrow \qquad \boldsymbol{x}_{\mathrm{LS}} = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{b}.$$

$\boldsymbol{x}_{\mathrm{LS}}$ exists and is unique when $\boldsymbol{A}^T\boldsymbol{A}$ is invertible, which is the case when $r = n$ (i.e., the system is over-determined and $\boldsymbol{A}$ has full rank).

Bonus info: the matrix $\boldsymbol{A}^\dagger = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T$ is called the *pseudoinverse* (or Moore-Penrose inverse) of $\boldsymbol{A}$.

The Minimum-Norm Least Squares Solution

If $r < n$ we can define a unique *minimum-norm least squares solution* by:

$$\boldsymbol{x}_{\mathrm{LS}}^0 = \arg\min_{\boldsymbol{x}} \|\boldsymbol{x}\|_2 \qquad \text{subject to} \qquad \boldsymbol{A}^T\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{A}^T\boldsymbol{b}.$$

Example. Consider again the problem

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \boldsymbol{x} = \begin{pmatrix} 14 \\ 20 \\ 50 \end{pmatrix} \quad \text{with} \quad r = 2 < n = 3,$$

$\boldsymbol{x}_{\mathrm{LS}}$ is not unique, and all least squares solutions have the form

$$\boldsymbol{x}_{\mathrm{LS}} = \begin{pmatrix} 3 \\ 2 \\ 1 \end{pmatrix} + \alpha \begin{pmatrix} 1 \\ -2 \\ 1 \end{pmatrix}, \qquad \alpha \text{ arbitrary.}$$

The minimum-norm least squares solution $\boldsymbol{x}_{\mathrm{LS}}^0$ is obtained by setting $\alpha = 0$.

Weighted Least Squares Solutions

Recall our definition of the diagonal matrix $\boldsymbol{M} = \mathrm{diag}\big(m\|\boldsymbol{r}_i\|_2^2\big)$.

We also define the *weighted least squares problem*

$$\min_{\boldsymbol{x}} {}^1\!/_2 \, \|\boldsymbol{M}^{-1/2}(\boldsymbol{A}\,\boldsymbol{x} - \boldsymbol{b})\|_2^2 \qquad \Leftrightarrow \qquad (\boldsymbol{A}^T\boldsymbol{M}^{-1}\boldsymbol{A})\,\boldsymbol{x} = \boldsymbol{A}^T\boldsymbol{M}^{-1}\boldsymbol{b}$$

and the corresponding solution $\boldsymbol{x}_{\mathrm{LS},\boldsymbol{M}} = (\boldsymbol{A}^T\boldsymbol{M}^{-1}\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{M}^{-1}\boldsymbol{b}$.

Similarly we define the minimum-norm weighted least squares solution

$$\boldsymbol{x}^0_{\mathrm{LS},\boldsymbol{M}} = \arg\min_{\boldsymbol{x}} \|\boldsymbol{x}\|_2 \quad \text{subject to} \quad \boldsymbol{A}^T\boldsymbol{M}^{-1}\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{A}^T\boldsymbol{M}^{-1}\boldsymbol{b}.$$

The full picture of Cimmino's method:

- $r = n = m$: convergence to $\boldsymbol{A}^{-1}\boldsymbol{b}$.
- $r = n < m$ and $\boldsymbol{b} \in \mathcal{R}(\boldsymbol{A})$: convergence to $\boldsymbol{x}_{\mathrm{LS}}$.
- $r = n < m$ and $\boldsymbol{b} \notin \mathcal{R}(\boldsymbol{A})$: convergence to $\boldsymbol{x}_{\mathrm{LS},\boldsymbol{M}}$.
- $r < \min(m, n)$: convergence to $\boldsymbol{x}^0_{\mathrm{LS},\boldsymbol{M}}$.

The Optimization Viewpoint

Karczmarz, Cimmino and similar algebraic iterative methods as usually considered as solvers for systems of linear equations.

But it is more convenient to consider them as **optimization methods**.

Within this framework we can easily handle common extensions:

- We can introduce a *relaxation parameter* – or step length parameter – in the algorithm which controls the "size" of the updating and, as a consequence, the convergence of the method:
  - a constant λ, or
  - a parameter λ_k that changes with the iterations.
- We can also, in each updating step, incorporate a *projection* $P_{\mathcal{C}}$ on a suitably chosen convex set $\mathcal{C}$ that reflects prior knowledge, such as
  - the positive orthant $\mathbb{R}^n_+ \to$ nonnegative solutions,
  - the *n*-dimensional box $[0,1]^n \to$ solution elements in [0,1].
- We can introduce *other norms* than the 2-norm $\|\cdot\|_2$, which can improve the robustness of the method.

Example: Robust Solutions with the 1-norm

The 1-norm is well suited for handling "outliers" in the data:

$$\min_{\boldsymbol{x}} \|\boldsymbol{A}\,\boldsymbol{x} - \boldsymbol{b}\|_1, \qquad \|\boldsymbol{A}\,\boldsymbol{x} - \boldsymbol{b}\|_1 = \sum_{i=1}^{m} |\boldsymbol{r}_i \cdot \boldsymbol{x} - b_i|.$$

Consider two over-determined noisy problems with the same matrix:

$$\boldsymbol{A} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \\ 1 & 4 & 16 \\ 1 & 5 & 25 \\ 1 & 6 & 36 \end{pmatrix}, \quad \boldsymbol{A}\,\bar{\boldsymbol{x}} = \begin{pmatrix} 6 \\ 17 \\ 34 \\ 57 \\ 86 \\ 121 \end{pmatrix}, \quad \boldsymbol{b} = \begin{pmatrix} 6.0001 \\ 17.0285 \\ 33.9971 \\ 57.0061 \\ 85.9965 \\ 120.9958 \end{pmatrix}, \quad \boldsymbol{b}^{\mathrm{o}} = \begin{pmatrix} 6.0001 \\ 17.2850 \\ 33.9971 \\ 57.0061 \\ 85.9965 \\ 120.9958 \end{pmatrix}.$$

Least squares solutions: $\boldsymbol{x}_{\mathrm{LS}}$ and $\boldsymbol{x}_{\mathrm{LS}}^{\mathrm{o}}$; 1-norm solutions: $\boldsymbol{x}_1$ and $\boldsymbol{x}_1^{\mathrm{o}}$:

$$\boldsymbol{x}_{\mathrm{LS}} = \begin{pmatrix} 1.0041 \\ 2.0051 \\ 2.9989 \end{pmatrix}, \quad \boldsymbol{x}_{\mathrm{LS}}^{\mathrm{o}} = \begin{pmatrix} 1.0811 \\ 2.0151 \\ 2.9943 \end{pmatrix}, \quad \boldsymbol{x}_1 = \begin{pmatrix} 0.9932 \\ 2.0087 \\ 2.9986 \end{pmatrix}, \quad \boldsymbol{x}_1^{\mathrm{o}} = \begin{pmatrix} 0.9932 \\ 2.0088 \\ 2.9986 \end{pmatrix}.$$

The Least Squares Problem Revisited

The objective function and its gradient

$$\mathcal{F}(\boldsymbol{x}) = {}^1\!/_2 \, \|\boldsymbol{b} - \boldsymbol{A}\,\boldsymbol{x}\|_2^2, \qquad \nabla\mathcal{F}(\boldsymbol{x}) = -\boldsymbol{A}^T(\boldsymbol{b} - \boldsymbol{A}\,\boldsymbol{x}).$$

The method of steepest descent for $\min_{\boldsymbol{x}} \mathcal{F}(\boldsymbol{x})$, with starting vector $\boldsymbol{x}^{(0)}$, performs the updates

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \lambda_k \, \nabla\mathcal{F}(\boldsymbol{x}) = \boldsymbol{x}^{(k)} + \lambda_k \, \boldsymbol{A}^T(\boldsymbol{b} - \boldsymbol{A}\,\boldsymbol{x}),$$

where λ_k is a *step-length parameter* that can depend on the iteration.

Also known as *Landweber's method* – corresponds to $\boldsymbol{M} = \boldsymbol{I}$ in Cimmino.

Cimmino's method corresponds to the weighted problem:

$$\mathcal{F}_{\boldsymbol{M}}(\boldsymbol{x}) = {}^1\!/_2 \, \|\boldsymbol{M}^{-1/2}(\boldsymbol{b} - \boldsymbol{A}\,\boldsymbol{x})\|_2^2, \quad \nabla\mathcal{F}_{\boldsymbol{M}}(\boldsymbol{x}) = -\boldsymbol{A}^T\boldsymbol{M}^{-1}(\boldsymbol{b} - \boldsymbol{A}\,\boldsymbol{x}).$$

Incorporating Simple Constraints

We can include constraints on the elements of the reconstructed image.

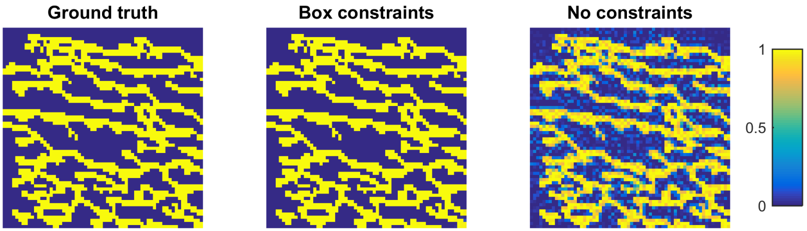
Assume that we can write the constraint as $\boldsymbol{x} \in \mathcal{C}$, where $\mathcal{C}$ is a convex set; this includes two very common special cases:

Non-negativity constraints. The set $\mathcal{C} = \mathbb{R}^n_+$ corresponds to

$$x_i \geq 0, \qquad i = 1, 2, \ldots, n.$$

Box constraints. The set $\mathcal{C} = [0,1]^n$ (n-dimensional box) corresponds to

$$0 \leq x_i \leq 1, \qquad i = 1, 2, \ldots, n.$$



The Projected Algorithms

Both algorithms below solve $\min_{\boldsymbol{x}\in\mathcal{C}} \|\boldsymbol{M}^{-1/2}(\boldsymbol{b}-\boldsymbol{A}\,\boldsymbol{x})\|_2$.

Projected gradient algorithm ($\lambda_k < 2/\|\boldsymbol{A}^T\boldsymbol{M}\boldsymbol{A}\|_2$)

```
$x^{(0)}$ = initial vector
for $k = 0, 1, 2, \ldots$
    $\boldsymbol{x}^{(k+1)} = P_{\mathcal{C}}\big(\boldsymbol{x}^{(k)} + \lambda_k\,\boldsymbol{A}^T\boldsymbol{M}^{-1}(\boldsymbol{b}-\boldsymbol{A}\,\boldsymbol{x}^{(k)})\big)$
end
```

Projected incremental gradient (Kaczmarz) algorithm ($\lambda_k < 2$)

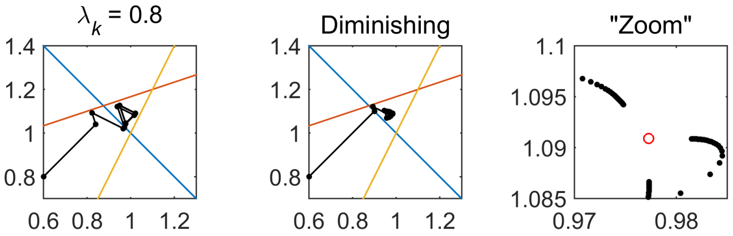
```
$x^{(0)}$ = initial vector
for $k = 0, 1, 2, \ldots$
    $i = k \pmod{m}$
    $\boldsymbol{x}^{(k+1)} = P_{\mathcal{C}}\left(\boldsymbol{x}^{(k)} + \lambda_k\,\frac{b_i - \boldsymbol{r}_i\cdot\boldsymbol{x}}{\|\boldsymbol{r}_i\|_2^2}\,\boldsymbol{r}_i\right)$
end
```

Iteration-Dependent Relaxation Parameter λ_k

The basic Kaczmarz algorithm gives a cyclic and non-convergent behavior.

Consider the example from slide 24 with:

$$\lambda_k = 0.8 \text{ (independent of } k) \qquad \text{and} \qquad \lambda_k = 1/\sqrt{k}, \quad k = 0, 1, 2, \ldots$$



The rightmost plot is a "zoom" of the middle plot.

- With a fixed $\lambda_k < 1$ we still have a cyclic non-convergent behavior.
- With the *diminishing relaxation parameter* $\lambda_k = 1/\sqrt{k} \to 0$ as $k \to \infty$ the iterates converge to the weighted least squares solution $\boldsymbol{x}_{\mathrm{LS},\boldsymbol{M}}$.

Overview of Convergence (see also slides 37–38)

What the unconstr. methods converge to, with starting vector $\boldsymbol{x}^{(0)} = \boldsymbol{0}$:

- Kac: Kaczmarz's method with a fixed relaxation parameter.
- K–d: Kaczmarz's method with a diminishing parameter.
- Cim: Cimmino's method with a fixed relaxation parameter.

	$r < \min(m,n)$		$r = \min(m,n)$	
	$\boldsymbol{b} \in \mathcal{R}(\boldsymbol{A})$	$\boldsymbol{b} \notin \mathcal{R}(\boldsymbol{A})$	$\boldsymbol{b} \in \mathcal{R}(\boldsymbol{A})$	$\boldsymbol{b} \notin \mathcal{R}(\boldsymbol{A})$
$m < n$	$\boldsymbol{x}^0_{\text{LS}} = \boldsymbol{x}^0_{\text{LS},\boldsymbol{M}}$			▨
$m = n$	$\boldsymbol{x}^0_{\text{LS}} = \boldsymbol{x}^0_{\text{LS},\boldsymbol{M}}$		$\boldsymbol{A}^{-1}\boldsymbol{b}$	▨
$m > n$	$\boldsymbol{x}^0_{\text{LS}} = \boldsymbol{x}^0_{\text{LS},\boldsymbol{M}}$	Kac: cyclic K–d: $\boldsymbol{x}^0_{\text{LS},\boldsymbol{M}}$ Cim: $\boldsymbol{x}^0_{\text{LS},\boldsymbol{M}}$	$\boldsymbol{x}_{\text{LS}} = \boldsymbol{x}_{\text{LS},\boldsymbol{M}}$	Kac: cyclic K–d: $\boldsymbol{x}_{\text{LS},\boldsymbol{M}}$ Cim: $\boldsymbol{x}_{\text{LS},\boldsymbol{M}}$

NB: for over-det. systems with $\boldsymbol{b} \notin \mathcal{R}(\boldsymbol{A})$, $\boldsymbol{x}^0_{\text{LS},\boldsymbol{M}} \neq \boldsymbol{x}^0_{\text{LS}}$ and $\boldsymbol{x}_{\text{LS},\boldsymbol{M}} \neq \boldsymbol{x}_{\text{LS}}$.

Last Slide: Other Projected Gradient Algorithms

Many algorithm proposed in the literature (Landweber, CAV, DROP, SART, SIRT, ...) are special cases of the following general formulation.

General projected gradient algorithm ($\lambda_k < 2$)

```
$x^{(0)}$ = initial vector
for $k = 0, 1, 2, \dots$
    $\mathbf{x}^{(k+1)} = P_{\mathcal{C}}\big(\mathbf{x}^{(k)} + \lambda_k\, \mathbf{D}^{-1} \mathbf{A}^T \mathbf{M}^{-1} (\mathbf{b} - \mathbf{A}\, \mathbf{x}^{(k)})\big)$
end
```

Of particular interest is the method SIRT in which:

$$\begin{aligned} \mathbf{D} &= \operatorname{diag}\big(\|\mathbf{c}_j\|_1\big), & \|\mathbf{c}_j\|_1 &= \textstyle\sum_{i=1}^m a_{ij}, \\ \mathbf{M} &= \operatorname{diag}\big(\|\mathbf{r}_i\|_1\big), & \|\mathbf{r}_i\|_1 &= \textstyle\sum_{j=1}^n a_{ij} \end{aligned}$$

SIRT is implemented in the ASTRA software, to be discussed tomorrow.