Algebraic Iterative Methods for Computed Tomography

Per Christian Hansen

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

1. A bit of motivation.
2. The algebraic formulation; matrix notation and interpretation.
3. Kaczmarz’s method (also known as ART) – fully sequential.
6. The optimization viewpoint.

Points to take home today:

- Linear algebra provides a 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 years of use → 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:
FBP Versus Algebraic Methods

- Limited data, or nonuniform distribution of projection angles or rays → 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 Line Projection Model

The damping of the $i$th X-ray through the object is a line integral of the attenuation coefficient $\chi(s)$ along the ray (from Lambert-Beer's law):

$$ b_i = \int_{\text{ray}_i} f(x_1, x_2) \, d\ell, \quad i = 1, 2, \ldots, m. $$

Assume that $f(x_1, x_2)$ is a constant $f_j$ in pixel $j$. This leads to:

$$ b_i = \sum_{j \sim \text{ray}_i} a_{ij} f_j, \quad a_{ij} = \text{length of ray}_i \text{ in pixel } j, $$

where the sum is over those pixels $j$ that are intersected by ray$_i$.

If we define $a_{ij} = 0$ for those pixels not intersected by ray$_i$, then we have a simple sum

$$ b_i = \sum_{j=1}^{n} a_{ij} f_j, \quad n = \text{number of pixels}. $$
A Big and Sparse System

If we collect all \( m \) equations then we arrive at a system of linear equations

\[
A x = b, \quad x = (f_1, f_2, \ldots, f_n)^T
\]

with a very \textbf{sparse system matrix} \( A \). Example: \( 5 \times 5 \) pixels and 9 rays:

A really big advantage is that we only set up equations for the data that we actually have. In case of missing data, e.g., for certain projection angles or certain rays in a projection, we just omit those from the linear system.
The System Matrix is Very Sparse

Another example: $256 \times 256$ pixels and 180 projections with 362 rays each. The system matrix $A$ 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%.
The Simplest Algebraic Problem

One unknown, no noise:

\[1 \cdot x = 3\]

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

\[
\begin{pmatrix}
1 & 1 \\
1 & \sqrt{2}
\end{pmatrix}
\begin{pmatrix}
x \\
x
\end{pmatrix}
=
\begin{pmatrix}
3.1 \\
3.2
\end{pmatrix}
\Rightarrow
x = 3.025
\]

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 $2 \times 2$ image . . .
A “Sudoku” Problem

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

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

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

$$
\begin{array}{c}
\begin{pmatrix}
1 & 2 \\
3 & 4
\end{pmatrix}

+ k \times 
\begin{pmatrix}
-1 & 1 \\
1 & -1
\end{pmatrix}
\end{array}
$$

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

- The *sensitivity* of the solution to the data errors is characterized by the *condition number* \( \text{cond}(A) = \|A\|_2 \cdot \|A^{-1}\|_2 \) (not discussed here).
- The *uniqueness* of the solution is characterized by the *rank* of \( A \), the number of linearly independent row or columns.
In principle, all we need to do in the algebraic formulation is to solve the large sparse linear system \( Ax = b \):

Math: \( x = A^{-1}b \),  
MATLAB: \( x = A\backslash b \).

How hard can that be?

Actually, this can be a formidable task if we try to 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 $A$ at a time.
- Fast convergence.

Fully Simultaneous Methods

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

Krylov subspace methods (not covered in this course)

- CGLS, LSQR, GMRES, ... 
- These methods are also based on matrix multiplications.
Matrix Notation and Interpretation

Notation:

\[
A = \begin{pmatrix}
| & & & |
\end{pmatrix} = \begin{pmatrix}
\begin{array}{c}
\cdots \\
-\cdots \\
-\cdots \\
\vdots
\end{array}
\end{pmatrix},
\]

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

\[
b = \begin{pmatrix}
\begin{array}{c}
\cdots \\
r_1^T \\
r_2^T \\
\vdots \\
r_m^T
\end{array}
\end{pmatrix}
\]

\[
x_1 c_1 + x_2 c_2 + \cdots + x_n c_n = \begin{pmatrix}
r_1^T x \\
r_2^T x \\
\vdots \\
r_m^T x
\end{pmatrix}.
\]

The \(i\)th row of \(A\) maps \(x\) to detector element \(i\) (through the \(i\)th ray) via the inner product:

\[
b_i = r_i^T x = \sum_{j=1}^{n} a_{ij} x_j, \quad i = 1, 2, \ldots, m.
\]
A $32 \times 32$ image has four nonzero pixels with intensities 1, 0.8, 0.6, 0.4. In the vector $x$ these four pixels correspond to entries 468, 618, 206, 793. Hence the sinogram, represented as a vector $b$, takes the form

$$b = 0.6 \, c_{206} + 1.0 \, c_{468} + 0.8 \, c_{618} + 0.4 \, c_{793}.$$
Forward and Back Again

Forward projection. Consider ray $i$:

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

$$r_i = [a_{i1} \ a_{i2} \ 0 \ a_{i4} \ 0 \ 0 \ a_{i7} \ 0 \ 0]$$

$$b_i = r_i^T x = a_{i1}x_1 + a_{i2}x_2 + a_{i4}x_4 + a_{i7}x_7$$

The forward projection $b = Ax$ maps all image pixels $x_j$ along the ray to the detector data $b_i$.

Back projection. Another name for multiplication with $A^T$:

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

Each data $b_i$ is “distributed back” along the $i$th ray to the pixels, with “weights” = $a_{ij}$:

$$b_i r_i = [b_i a_{i1} \ b_i a_{i2} \ 0 \ b_i a_{i4} \ 0 \ 0 \ b_i a_{i7} \ 0 \ 0]^T$$
Geometric Interpretation of \( A \mathbf{x} = \mathbf{b} \)

\[
\begin{align*}
\mathbf{r}_1^T \mathbf{x} & = a_{11} x_1 + a_{12} x_2 + \cdots + a_{1n} x_n = b_1 \\
\mathbf{r}_2^T \mathbf{x} & = a_{21} x_1 + a_{22} x_2 + \cdots + a_{2n} x_n = b_2 \\
& \vdots \\
\mathbf{r}_m^T \mathbf{x} & = a_{m1} x_1 + a_{m2} x_2 + \cdots + a_{mn} x_n = b_m.
\end{align*}
\]

Each equation \( \mathbf{r}_i^T \mathbf{x} = b_i \) defines an affine hyperplane in \( \mathbb{R}^n \):
Geometric Interpretation of the Solution

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

Example with $m = n = 2$:

\begin{align*}
\mathbf{a}_1 \mathbf{x}_1 + \mathbf{a}_2 \mathbf{x}_2 &= \mathbf{b}_1 \\
\mathbf{a}_2 \mathbf{x}_1 + \mathbf{a}_2 \mathbf{x}_2 &= \mathbf{b}_2
\end{align*}
Kaczmarz’s Method = Algebraic Reconstruction Technique

A simple 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 $\mathbf{x}$ on one of the hyperplanes $r_i^T \mathbf{x} = b_i$ 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

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

$$P_i(z) = z + \frac{b_i - r_i^T z}{\|r_i\|_2^2} r_i,$$

$$\|r_i\|_2^2 = r_i^T r_i.$$

In words, we scale the row vector $r_i$ by $(b_i - r_i^T z)/\|r_i\|_2^2$ and add it to $z$. 
Kaczmarz’s Method

We thus obtain the following algebraic formulation:

Basic Kaczmarz algorithm

\[ x^{(0)} = \text{initial vector} \]

for \( k = 0, 1, 2, \ldots \)

\[ i = k \mod m \]

\[ x^{(k+1)} = P_i(x^{(k)}) = x^{(k)} + \frac{b_i - r_i^T x^{(k)}}{\|r_i\|_2^2} r_i \]

end

Each time we have performed \( m \) iterations of this algorithm, we have performed one sweep over the rows of \( A \). 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 \( \|r_i\|_2 \).
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 $A$ influences the convergence rate:

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

\[
\begin{pmatrix}
1.0 & 1.0 \\
1.0 & 1.1 \\
1.0 & 3.0 \\
1.0 & 3.7 \\
\end{pmatrix} \quad x = \begin{pmatrix}
2.0 \\
2.1 \\
4.0 \\
4.7 \\
\end{pmatrix}
\]
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 $A$ is invertible and that all rows of $A$ are scaled to unit 2-norm. Then the expected value $E(\cdot)$ of the error norm satisfies:

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

where $\bar{x} = A^{-1}b$ and $\kappa = \text{cond}(A) = \|A\|_2 \|A^{-1}\|_2$. Linear convergence.

When $\kappa$ 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 $\tilde{x} = A^{-1}b$ that satisfies $Ax = b$, i.e., all the affine hyperplanes associated with the rows of $A$ intersect in a single point.

What happens when this is not true? → cyclic convergence:

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:

\[
x^{(k+1)} = \frac{1}{m} \sum_{i=1}^{m} P_i(x^{(k)}) = \frac{1}{m} \sum_{i=1}^{m} \left( x^{(k)} + \frac{b_i - r_i^T x^{(k)}}{\|r_i\|^2} r_i \right)
\]

\[
= x^{(k)} + \frac{1}{m} \sum_{i=1}^{m} \frac{b_i - r_i^T x^{(k)}}{\|r_i\|^2} r_i.
\]
Matrix Formulation of Cimmino’s Method

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

\[
x^{(k+1)} = x^{(k)} + \frac{1}{m} \sum_{i=1}^{m} \frac{b_i - r_i^T x^{(k)}}{\|r_i\|^2_2} r_i
\]

\[
= x^{(k)} + \frac{1}{m} \begin{pmatrix} \frac{r_1}{\|r_1\|^2_2} & \cdots & \frac{r_m}{\|r_m\|^2_2} \end{pmatrix} \begin{pmatrix} b_1 - r_1^T x^{(k)} \\ \vdots \\ b_m - r_m^T x^{(k)} \end{pmatrix}
\]

\[
= x^{(k)} + \frac{1}{m} \begin{pmatrix} r_1^T \|r_1\|^{-2} \\ \vdots \\ r_m^T \|r_m\|^{-2} \end{pmatrix} \begin{pmatrix} \|r_1\|^{-2} \\ \vdots \\ \|r_m\|^{-2} \end{pmatrix} \begin{pmatrix} b - \begin{pmatrix} r_1 \\ \vdots \\ r_m \end{pmatrix} x^{(k)} \end{pmatrix}
\]

\[
= x^{(k)} + A^T M (b - A x^{(k)}),
\]

where we introduced the diagonal matrix \( M = \text{diag}(1/(m\|r_i\|^2_2)) \).
Cimmino’s Method

We thus obtain the following formulation, with $M = \text{diag}(1/(m\|r_i\|_2^2))$:

Basic Cimmino algorithm

\[
x^{(0)} = \text{initial vector}
\]

for $k = 0, 1, 2, \ldots$

\[
x^{(k+1)} = x^{(k)} + A^T M (b - Ax^{(k)})
\]

end

Note that one iteration here involves all the rows of $A$, 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 \( x^{(0)} = 0 \) and let \( I \) denote the \( n \times n \) identity matrix; then:

\[
x^{(k+1)} = \sum_{j=0}^{k} (I - A^T M A)^j A^T M b
\]

\[
= \left(I - (I - A^T M A)^{k+1}\right) (A^T M A)^{-1} A^T M b.
\]

If \( A \) is invertible then

\[
(A^T M A)^{-1} A^T M b = A^{-1} M^{-1} A^{-T} A^T M b = A^{-1} b = \bar{x}.
\]

Moreover, the largest eigenvalue of the symmetric matrix \( I - A^T M A \) is strictly smaller than one, and therefore

\[
\left(I - (I - A^T M A)^{k+1}\right) \to I \quad \text{for} \quad k \to \infty.
\]

Hence the iterates \( x^{(k)} \) converge to the solution \( \bar{x} = A^{-1} b \).
To simplify the result, assume that $A$ is invertible and that the rows of $A$ are scaled such that $\|A\|_2^2 = m$. Then

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

where $\bar{x} = A^{-1}b$, $\kappa = \text{cond}(A) = \|A\|_2 \|A^{-1}\|_2$, and again we have linear convergence.

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

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

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

It is also very common that $A$ does not have full rank.

*We need to set the stage for treating such matrices.*

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

The **range** $\text{Range}(A)$ is the linear subspace spanned by the columns of $A$:

$$\text{Range}(A) \equiv \{ u \in \mathbb{R}^m \mid u = \alpha_1 c_1 + \alpha_2 c_2 + \cdots + \alpha_n c_n, \text{ arbitrary } \alpha_j \}.$$

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

$$\text{Null}(A) \equiv \{ v \in \mathbb{R}^n \mid A v = 0 \}.$$

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

Consider the $3 \times 3$ matrix

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

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 $\text{Range}(A)$ and null space $\text{Null}(A)$ consist of all vectors of the forms

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

respectively, for arbitrary $\alpha_1$, $\alpha_2$, and $\alpha_3$. 
Consider two linear systems with the matrix $A$ from the previous example:

\[
\begin{pmatrix}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9
\end{pmatrix}
\begin{pmatrix}
x_1 \\
x_2 \\
x_3
\end{pmatrix}
=
\begin{pmatrix}
14 \\
20 \\
50
\end{pmatrix}
\quad \text{and} \quad
\begin{pmatrix}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9
\end{pmatrix}
\begin{pmatrix}
x_1 \\
x_2 \\
x_3
\end{pmatrix}
=
\begin{pmatrix}
6 \\
15 \\
24
\end{pmatrix}.
\]

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

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

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

satisfies this equation. The arbitrary component is in the null space $\text{Null}(A)$. 
Null Space Artifacts in Tomography I

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

Left: three different vectors in the null space $\text{Null}(A)$.
Right: the exact ("ground truth") image $\bar{x}$ 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.
The mage has $29 \times 29$ pixels and we use projection angles in \([50^\circ, 130^\circ]\). $A$ is $841 \times 841$ and rank deficient; the dimension of $\text{Null}(A)$ 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 \( \text{Null}(A) \) 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 $x$ such that $Ax = b$, i.e., such that $b$ is a linear combination of the columns $c_i$ of $A$. This is equivalent to the requirement $b \in \text{Range}(A)$.

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

This is actually the normal situation in problems with measurement noise.
### Overview of Systems: $m \leq n$

<table>
<thead>
<tr>
<th>$m &lt; n$</th>
<th>Full rank</th>
<th>Rank deficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>Underdetermined</td>
<td>$r = m$</td>
<td>$r &lt; m$</td>
</tr>
<tr>
<td>$= \begin{bmatrix}</td>
<td>\end{bmatrix}$</td>
<td>Always consistent.</td>
</tr>
<tr>
<td>$= \begin{bmatrix}</td>
<td>\end{bmatrix}$</td>
<td>Always infinitely many solutions.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$m = n$</th>
<th>$r = m = n$</th>
<th>$r &lt; m = n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Square</td>
<td>Always consistent.</td>
<td>Can be inconsistent.</td>
</tr>
<tr>
<td>$= \begin{bmatrix}</td>
<td>\end{bmatrix}$</td>
<td>Always a unique solution.</td>
</tr>
</tbody>
</table>

The system is *inconsistent* when $\mathbf{b} \notin \text{Range}(\mathbf{A})$.

There is a unique solution only if $r = m = n$. 
Overview of Systems: $m > n$

<table>
<thead>
<tr>
<th>$m &gt; n$</th>
<th>Full rank</th>
<th>Rank deficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overdetermined</td>
<td>$r = n$</td>
<td>$r &lt; n$</td>
</tr>
<tr>
<td></td>
<td>Can be inconsistent.</td>
<td>Can be inconsistent.</td>
</tr>
<tr>
<td></td>
<td>No solution or a unique solution.</td>
<td>No solution or infinitely many solutions</td>
</tr>
</tbody>
</table>

The system is *inconsistent* when $b \notin \text{Range}(A)$.

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 \( \mathbf{b} = \mathbf{A} \bar{x} + \mathbf{e} \) and \( \mathbf{e} \) is zero-mean Gaussian noise. The best linear unbiased estimate of \( \bar{x} \) is the solution to the least squares problem:

\[
x_{\text{LS}} = \arg \min_{\mathbf{x}} \frac{1}{2} \| \mathbf{b} - \mathbf{A} \mathbf{x} \|_2^2,
\]

and \( x_{\text{LS}} \) is unique when \( r = n \). Geometrically, this corresponds to finding \( x_{\text{LS}} \) such that \( \mathbf{A} x_{\text{LS}} \) is orthogonal to the residual vector \( \mathbf{b} - \mathbf{A} x_{\text{LS}} \).
Computing the Least Squares Solution

The requirement that $A x_{LS} \perp (b - A x_{LS})$ leads to:

$$(A x_{LS})^T (b - A x_{LS}) = 0 \iff x_{LS}^T (A^T b - A^T A x_{LS}) = 0$$

which means that $x_{LS}$ is the solution to the normal equations:

$$A^T A x = A^T b \quad \Rightarrow \quad x_{LS} = (A^T A)^{-1} A^T b.$$

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

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

---

Per Christian Hansen
Algebraic Iterative Methods
The Minimum-Norm Least Squares Solution

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

$$x_{\text{LS}}^0 = \underset{x}{\text{arg min}} \|x\|_2 \quad \text{subject to} \quad A^T A x = A^T b.$$ 

Example. Consider again the problem

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

$x_{\text{LS}}$ is not unique, and all least squares solutions have the form

$$x_{\text{LS}} = \begin{pmatrix} 3 \\ 2 \\ 1 \end{pmatrix} + \alpha \begin{pmatrix} 1 \\ -2 \\ 1 \end{pmatrix}, \quad \alpha \text{ arbitrary}.$$ 

The minimum-norm least squares solution $x_{\text{LS}}^0$ is obtained by setting $\alpha = 0$. 
Weighted Least Squares Solutions and Cimmino

Recall our definition of the diagonal matrix \( M = \text{diag}(1/(m\|r_i\|_2^2)) \).

We also define the *weighted least squares problem*

\[
\min_{x} \frac{1}{2} \| M^{1/2}(Ax - b) \|_2^2 \quad \Leftrightarrow \quad (A^T M A) x = A^T M b
\]

and the corresponding solution \( x_{\text{LS},M} = (A^T M A)^{-1} A^T M b \).

Similarly we define the minimum-norm weighted least squares solution

\[
x_{\text{LS},M}^0 = \arg \min_{x} \|x\|_2 \quad \text{subject to} \quad A^T M^{-1} A x = A^T M^{-1} b.
\]

The full picture of Cimmino’s method:

- \( r = n = m \): convergence to \( A^{-1} b \).
- \( r = n < m \) and \( b \in \text{Range}(A) \): convergence to \( x_{\text{LS}} \).
- \( r = n < m \) and \( b \not\in \text{Range}(A) \): convergence to \( x_{\text{LS},M} \).
- \( r < \min(m,n) \): convergence to \( x_{\text{LS},M}^0 \).
The Optimization Viewpoint ($\rightarrow$ Module 3)

Karczmarz, Cimmino and similar algebraic iterative methods are 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 $\omega$, or
  - a parameter $\omega_k$ that changes with the iterations.
- We can also, in each updating step, incorporate a projection $P_C$ on a suitably chosen convex set $C$ that reflects prior knowledge, such as
  - the positive orthant $\mathbb{R}^+_n \rightarrow$ nonnegative solutions,
  - the $n$-dimensional box $[0, 1]^n \rightarrow$ 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:

\[ \min_x \| A x - b \|_1, \quad \| A x - b \|_1 = \sum_{i=1}^m |r_i^T x - b_i|. \]

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

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

Least squares solutions: \( x_{LS} \) and \( x_{LS}^o \); 1-norm solutions: \( x_1 \) and \( x_1^o \):

\[
x_{LS} = \begin{pmatrix} 1.0041 \\ 2.0051 \\ 2.9989 \end{pmatrix}, \quad x_{LS}^o = \begin{pmatrix} 1.0811 \\ 2.0151 \\ 2.9943 \end{pmatrix}, \quad x_1 = \begin{pmatrix} 0.9932 \\ 2.0087 \\ 2.9986 \end{pmatrix}, \quad x_1^o = \begin{pmatrix} 0.9932 \\ 2.0088 \\ 2.9986 \end{pmatrix}.
\]
The Least Squares Problem Revisited

The objective function and its gradient

\[ F(x) = \frac{1}{2} \| b - Ax \|_2^2, \quad \nabla F(x) = -AT(b - Ax). \]

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

\[ x^{(k+1)} = x^{(k)} - \omega_k \nabla F(x) = x^{(k)} + \omega_k A^T(b - Ax), \]

where \( \omega_k \) is a step-length parameter that can depend on the iteration.

Also known as Landweber’s method – corresponds to \( M = I \) in Cimmino.

Cimmino’s method corresponds to the weighted problem:

\[ F_M(x) = \frac{1}{2} \| M^{1/2}(b - Ax) \|_2^2, \quad \nabla F_M(x) = -A^TM(b - Ax). \]
Incorporating Simple Constraints

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

Assume that we can write the constraint as \( x \in C \), where \( C \) is a convex set; this includes two very common special cases:

**Non-negativity constraints.** The set \( C = \mathbb{R}_+^n \) corresponds to

\[
x_i \geq 0, \quad i = 1, 2, \ldots, n.
\]

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

\[
0 \leq x_i \leq 1, \quad i = 1, 2, \ldots, n.
\]
Given a set $C$, the orthogonal projection $P_C(x)$ of an arbitrary vector $x \in \mathbb{R}^n$ on $C$ is the unique vector that satisfies: $P_C(x) \perp (x - P_C(x))$.

If $C = \mathbb{R}_+^n$ (non-negativity constraints) then, in MATLAB, we compute the corresponding projection of $x$ as $\max(x, 0)$. 
The Projected Algorithms

Both algorithms below solve \( \min \| M^{1/2} (b - A x) \|_2 \) s.t. \( x \in C \).

Projected gradient algorithm \((\omega_k < 2/\|A^T M A\|_2)\)

\[ x^{(0)} = \text{initial vector} \]
for \( k = 0, 1, 2, \ldots \)

\[ x^{(k+1)} = P_C (x^{(k)} + \omega_k A^T M (b - A x^{(k)})) \]
end

Projected incremental gradient (Kaczmarz) algorithm \((\omega_k < 2)\)

\[ x^{(0)} = \text{initial vector} \]
for \( k = 0, 1, 2, \ldots \)

\[ i = k \, (\text{mod} \, m) \]

\[ x^{(k+1)} = P_C \left( x^{(k)} + \frac{b_i - r_i^T x}{\| r_i \|_2^2} r_i \right) \]
end
Iteration-Dependent Relaxation Parameter $\omega_k$

The basic Kaczmarz algorithm gives a cyclic behavior.

Consider the example from slide 24 with:

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

The rightmost plot is a “zoom” of the middle plot.

- With a fixed $\omega_k < 1$ we still have a cyclic behavior.
- With the diminishing relaxation parameter $\omega_k = \frac{1}{\sqrt{k}} \to 0$ as $k \to \infty$ the iterates converge to the weighted least squares solution $x_{LS,M}$. 
Overview of Convergence (see also slides 37–38)

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

- 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.

<table>
<thead>
<tr>
<th>$r &lt; \min(m, n)$</th>
<th>$r = \min(m, n)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>[rank def.]</td>
<td>[full rank]</td>
</tr>
<tr>
<td>$b \in \text{Range}(A)$</td>
<td>$b \notin \text{Range}(A)$</td>
</tr>
<tr>
<td>$m &lt; n$</td>
<td>$m = n$</td>
</tr>
<tr>
<td>$x_{LS}^0 = x_{LS,M}^0$</td>
<td>Kac: cyclic</td>
</tr>
<tr>
<td>$x_{LS}^0 = x_{LS,M}^0$</td>
<td>K–d: $x_{LS,M}^0$</td>
</tr>
</tbody>
</table>

Consistent systems, $b \in \text{Range}(A)$, do not “feel” the weight $M$. When $b \notin \text{Range}(A)$ then we have $x_{LS,M}^0 \neq x_{LS}^0$ and $x_{LS,M} \neq x_{LS}$. 
Many algorithms proposed in the literature (Landweber, CAV, DROP, SART, SIRT, ...) are special cases of the following general formulation.

**General projected gradient algorithm** ($\omega_k < 2$)

\[
\begin{align*}
    x^{(0)} &= \text{initial vector} \\
    \text{for } k = 0, 1, 2, \ldots \\
    x^{(k+1)} &= P_C(x^{(k)} + \omega_k D_1 A^T M_1 (b - A x^{(k)}))
\end{align*}
\]

end

Of particular interest is the method **SIRT** in which:

\[
\begin{align*}
    D_1 &= \text{diag}(1/\|c_j\|_1), \quad \|c_j\|_1 = \sum_{i=1}^{m} a_{ij}, \\
    M_1 &= \text{diag}(1/\|r_i\|_1), \quad \|r_i\|_1 = \sum_{j=1}^{n} a_{ij}
\end{align*}
\]

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