Algebraic Iterative Methods for Computed Tomography

Per Christian Hansen

DTU Compute
Department of Applied Mathematics and Computer Science
Technical University of Denmark
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°, 30°, ..., 180°:

  ![Phantom](image1.png)  ![FBP (iradon)](image2.png)  ![ART w/ box constraints](image3.png)

Algebraic Reconstruction Technique, box constraints (pixel values ∈ [0,1]).
Another Motivating Example: Missing Data

Irregularly spaced angles & “missing” angles also cause difficulties for FBP:

Phantom

Data = sinogram

ART w/ box constr.

Filtered back projection
Setting up the Algebraic 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):

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

Assume that $\chi(s)$ is a constant $x_j$ in pixel $j$. This leads to:

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

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

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

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

If we collect all $m$ equations then we arrive at a system of linear equations

$$Ax = b$$

with a very 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:

\[ \begin{pmatrix} 3.1 \\ 3.2 \\ 4.1 \end{pmatrix} \begin{pmatrix} 1 \\ 1/\sqrt{2} \end{pmatrix} x = \begin{pmatrix} 3.1 \\ 3.2 \\ 4.1 \end{pmatrix} \]

\[ x = 3.025 \]

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 → 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 $k \in \mathbb{R}$:

\[
\begin{pmatrix}
1 & 2 \\
3 & 4 \\
\end{pmatrix}
= 
\begin{pmatrix}
-1 & 1 \\
1 & -1 \\
\end{pmatrix} + k \times
\begin{pmatrix}
1 \\
-1 \\
\end{pmatrix}
\]

(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** \( \kappa = \| A \|_2 \cdot \| A^{-1} \|_2 \) (not discussed today).
- 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 \( A x = b \):

Math: \( x = A^{-1} b \), \hspace{1cm} \text{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 (rarely used in tomography)**
- CGLS, LSQR, GMRES, ... 
- These methods are also based on matrix multiplications.
Matrix Notation and Interpretation

Notation:

\[
A = \begin{pmatrix}
| & | & | \\
c_1 & c_2 & \cdots & c_n \\
| & | & | \\
\end{pmatrix} = \begin{pmatrix}
| & | & | \\
r_1 & & & \\
| & | & | \\
\vdots & & & \\
r_m & & & \\
\end{pmatrix},
\]

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:

\[
b = \begin{pmatrix}
b_1 \\
b_2 \\
\vdots \\
b_m \\
\end{pmatrix} = A x = x_1 c_1 + x_2 c_2 + \cdots + x_n c_n = \begin{pmatrix}
r_1 \cdot x \\
r_2 \cdot x \\
\vdots \\
r_m \cdot x \\
\end{pmatrix}.
\]

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

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

$$\mathbf{b} = 0.6 \mathbf{c}_{206} + 1.0 \mathbf{c}_{468} + 0.8 \mathbf{c}_{618} + 0.4 \mathbf{c}_{793}. $$
Geometric Interpretation of $A x = b$

$$r_1 \cdot x = a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1$$

$$r_2 \cdot x = a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2$$

$$\vdots$$

$$r_m \cdot x = a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = b_m.$$

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

$\begin{align*}
R^2 & \quad a_{i1}x_1 + a_{i2}x_2 = b_i \\
R^3 & \quad a_{i1}x_1 + a_{i2}x_2 + a_{i3}x_3 = b_i
\end{align*}$
Geometric Interpretation of the Solution

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

Example with $m = n = 2$:

\[
\begin{align*}
    a_{11}x_1 + a_{12}x_2 &= b_1 \\
    a_{21}x_1 + a_{22}x_2 &= b_2
\end{align*}
\]
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 $\mathbf{r}_i \cdot \mathbf{x} = b_i$ 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 $r_i \cdot x = b_i$ is given by:

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

In words, we scale the row vector $r_i$ by $(b_i - r_i \cdot z)/\|r_i\|_2^2$ and add it to $z$. 
Kaczmarz’s Method

We thus obtain the following algebraic formulation:

**Basic Kaczmarz algorithm**

\[
\begin{align*}
\mathbf{x}^{(0)} &= \text{initial vector} \\
\text{for } k = 0, 1, 2, \ldots \quad &\quad i = k \, (\text{mod } m) \\
\mathbf{x}^{(k+1)} &= P_i(\mathbf{x}^{(k)}) = \mathbf{x}^{(k)} + \frac{b_i - \mathbf{r}_i \cdot \mathbf{x}^{(k)}}{\|\mathbf{r}_i\|_2^2} \mathbf{r}_i \\
\text{end}
\end{align*}
\]

Each time we have performed \( m \) iterations of this algorithm, we have performed one sweep over the rows of \( \mathbf{A} \). 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 \( \|\mathbf{r}_i\|_2 \).
Convergence Issues

The convergence of Kaczmarz’s method is quite obvious from the graph on slide 17 – 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.
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 $\mathcal{E}(\cdot)$ of the error norm satisfies:

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

where $\bar{x} = A^{-1}b$ and $\kappa = \|A\|_2 \|A^{-1}\|_2$. This is 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 $\bar{x} = A^{-1}b$ 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? $\rightarrow$ 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:

$$\mathbf{x}^{(k+1)} = \frac{1}{m} \sum_{i=1}^{m} P_i(\mathbf{x}^{(k)}) = \frac{1}{m} \sum_{i=1}^{m} \left( \mathbf{x}^{(k)} + \frac{b_i - \mathbf{r}_i \cdot \mathbf{x}^{(k)}}{\|\mathbf{r}_i\|^2_2} \mathbf{r}_i \right)$$

$$= \mathbf{x}^{(k)} + \frac{1}{m} \sum_{i=1}^{m} \frac{b_i - \mathbf{r}_i \cdot \mathbf{x}^{(k)}}{\|\mathbf{r}_i\|^2_2} \mathbf{r}_i.$$
Matrix formulation of Cimmino’s Method

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

\[
\begin{align*}
\mathbf{x}^{(k+1)} &= \mathbf{x}^{(k)} + \frac{1}{m} \sum_{i=1}^{m} \frac{b_i - \mathbf{r}_i \cdot \mathbf{x}^{(k)}}{\|\mathbf{r}_i\|_2^2} \mathbf{r}_i \\
&= \mathbf{x}^{(k)} + \frac{1}{m} \left( \begin{array}{c}
\frac{\mathbf{r}_1}{\|\mathbf{r}_1\|_2^2} \\
\vdots \\
\frac{\mathbf{r}_m}{\|\mathbf{r}_m\|_2^2}
\end{array} \right) \left( \begin{array}{c}
b_1 - \mathbf{r}_1 \cdot \mathbf{x}^{(k)} \\
\vdots \\
b_m - \mathbf{r}_m \cdot \mathbf{x}^{(k)}
\end{array} \right) \\
&= \mathbf{x}^{(k)} + \frac{1}{m} \left( \begin{array}{c}
\mathbf{r}_1^T \\
\vdots \\
\mathbf{r}_m
\end{array} \right) \left( \begin{array}{c}
\|\mathbf{r}_1\|_2^{-2} \\
\vdots \\
\|\mathbf{r}_m\|_2^{-2}
\end{array} \right) \left( \begin{array}{c}
b - \left( \begin{array}{c}
\mathbf{r}_1 \\
\vdots \\
\mathbf{r}_m
\end{array} \right) \mathbf{x}^{(k)}
\end{array} \right) \\
&= \mathbf{x}^{(k)} + \mathbf{A}^T \mathbf{M}^{-1} (\mathbf{b} - \mathbf{A} \mathbf{x}^{(k)}),
\end{align*}
\]

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

We thus obtain the following formulation:

**Basic Cimmino algorithm**

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

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

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

end

Note that one iteration here involves all the rows of \( \mathbf{A} \), 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.
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:

$$\min_x \|Ax - b\|_1, \quad \|Ax - b\|_1 = \sum_{i=1}^{m} |r_i \cdot x - b_i|.$$ 

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 $x^o_{LS}$; 1-norm solutions: $x_1$ and $x^o_1$:

$$x_{LS} = \begin{pmatrix} 1.0041 \\ 2.0051 \\ 2.9989 \end{pmatrix}, \quad x^o_{LS} = \begin{pmatrix} 1.0811 \\ 2.0151 \\ 2.9943 \end{pmatrix}, \quad x_1 = \begin{pmatrix} 0.9932 \\ 2.0087 \\ 2.9986 \end{pmatrix}, \quad x^o_1 = \begin{pmatrix} 0.9932 \\ 2.0088 \\ 2.9986 \end{pmatrix}.$$
The Least Squares Problem Revisited

The objective function and its gradient

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

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

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

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:

\[ \mathcal{F}_M(x) = \frac{1}{2} \| M^{-1/2}(b - A x) \|^2_2, \quad \nabla \mathcal{F}_M(x) = -A^T M^{-1}(b - A x). \]
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

$$0 \leq x_i \leq 1, \quad i = 1, 2, \ldots, n.$$
The Projected Algorithms

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

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

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

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

\[
\begin{align*}
x^{(0)} &= \text{initial vector} \\
\text{for } k = 0, 1, 2, \ldots \\
\quad i &= k \pmod{m} \\
 \quad x^{(k+1)} &= P_C \left( x^{(k)} + \omega_k \frac{b_i - r_i \cdot x}{\| r_i \|^2} r_i \right) \\
\end{align*}
\]
Iteration-Dependent Relaxation Parameter $\omega_k$

The basic Kaczmarz algorithm gives a cyclic and non-convergent behavior. Consider the example from slide 22 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 non-convergent 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 $x_{LS,M}$. 
A Few References


