ART Performance

The <u>a</u>lgebraic <u>r</u>econstruction <u>t</u>echnique for tomography

Per Christian Hansen

joint work with, among others,

Tommy Elfving

Touraj Nikazad

Hans Henrik B. Sørensen

DTU Compute

Department of Applied Mathematics and Computer Science

X-Ray Tomography

X-ray tomography is the science of *seeing inside objects*.

1. X-rays are sent through an object from many different angles.
2. The response of the object to the signal is measured (projections).
3. Use the data + a mathematical model to compute an image of the object's interior.

The underlying model is

$$I = I_0 \, e^{-\int_{\text{ray}} \xi(s,t) \, d\ell} \qquad \ell = \text{length along ray}$$

where ξ = attenuat. coef., I_0 = source intensity, and I = measured ditto.

This leads to the linear relation

$$\text{“data”} = \log(I_0/I) = \int_{\text{ray}} \xi(s,t) \, d\ell.$$

ART = Algebraic Reconstruction Technique

Webster: “art” = the conscious use of skill and creative imagination.

In relation to tomography

1. A way of doing things: handling the tomographic reconstruction problem by discretization of the model, to obtain a large system of linear equations.
2. An algorithm: a classical iterative algorithm for solving a large system of linear equations; very succesfully used in computed tomography.

Reconstruction methods based on *analytical formulations*:

- Filtered back projection (PBP).
- Fast implementation based on FFT.
- Very good results provided we have a lot of data.

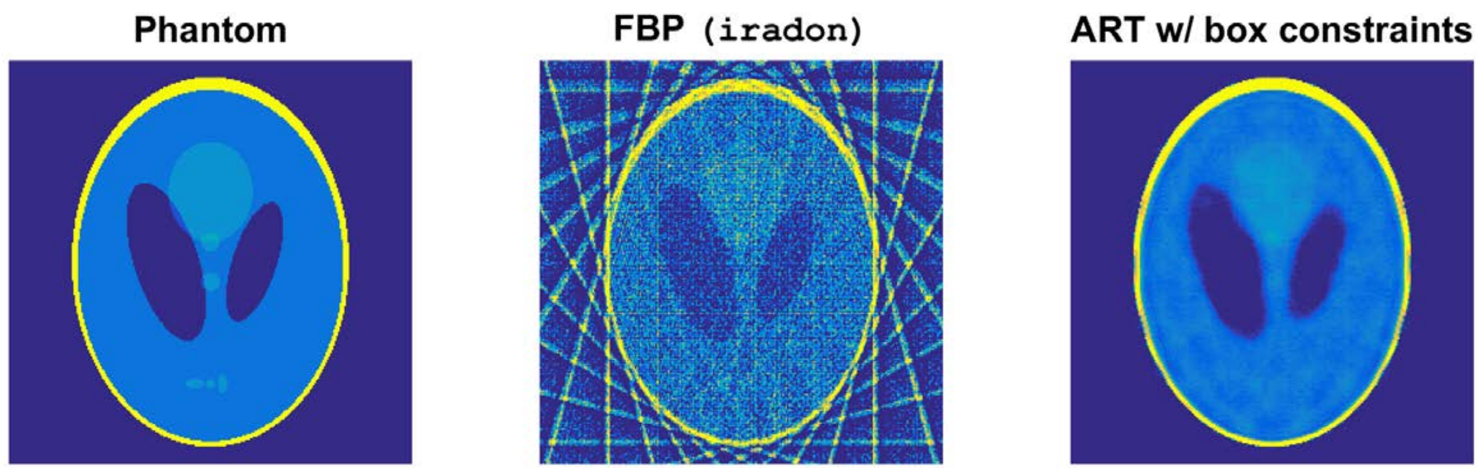
The *algebraic formulations* provide an important alternative:

- Better handling of limited data and sparse data.
- Easy incorporation of simple constraints, such as nonnegativity and box.
- A general framework for handling priors such as sparsity & total variation.

Filtered Back Projection (FBP) versus ART

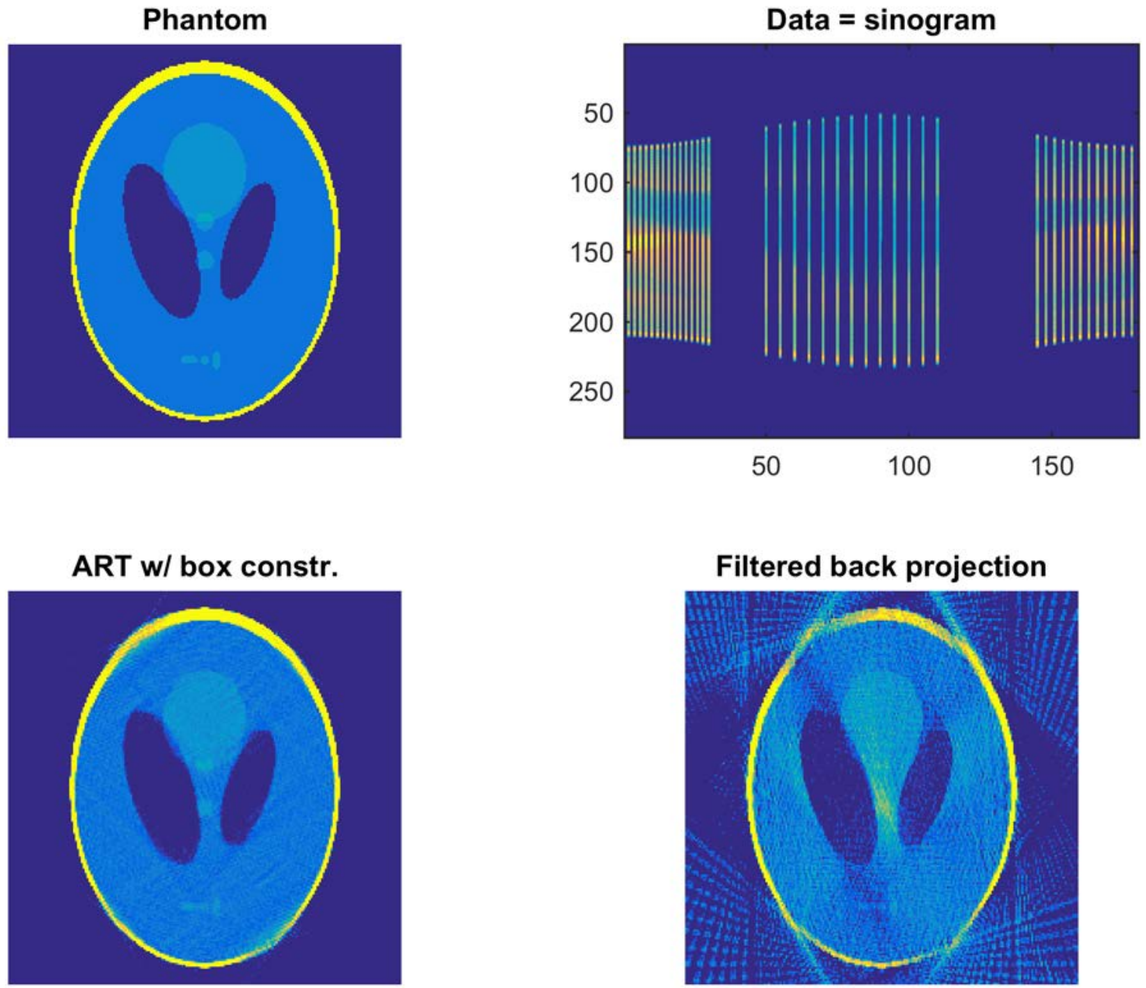
- FBP: low memory, works really well with many data.
- But *artifacts* appear with limited data, or nonuniform distribution of projection angles or ray.
- Difficult to incorporate constraints (e.g., nonnegativity) in FBP
- ART and other algebraic methods are more flexible and adaptive.

Example with 3% noise and projection angles $15^\circ, 30^\circ, \ldots, 180^\circ$.



FBP versus ART – A Second Example

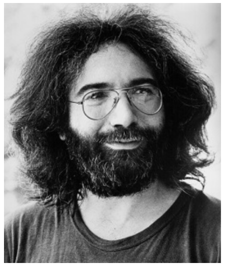
Irregularly spaced angles / “missing” angles also cause difficulties for FBP

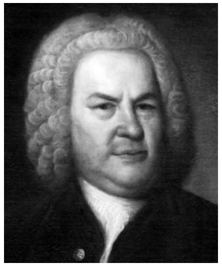


ART Academy

Listen to Grateful Dead (1965–1995) $\rightarrow$ old fashioned.
Listen to Mozart (1756–91) or Bach (1685–28) $\rightarrow$ the classics!

Talk about total variation (1992) $\rightarrow$ old stuff.
Talk about ART (1937) $\rightarrow$ classical algorithm.





ART is a rich source for research problems!

- Constraints and convergence.
- Performance $\rightarrow$ block algorithms.
- Column version of ART.

This talk

- Choice of relaxation parameter.
- Stopping rules.
- Acceleration techniques.
- Variations and extensions ART, e.g., for Poisson noise.
- Implementation aspect for high-performance computing.

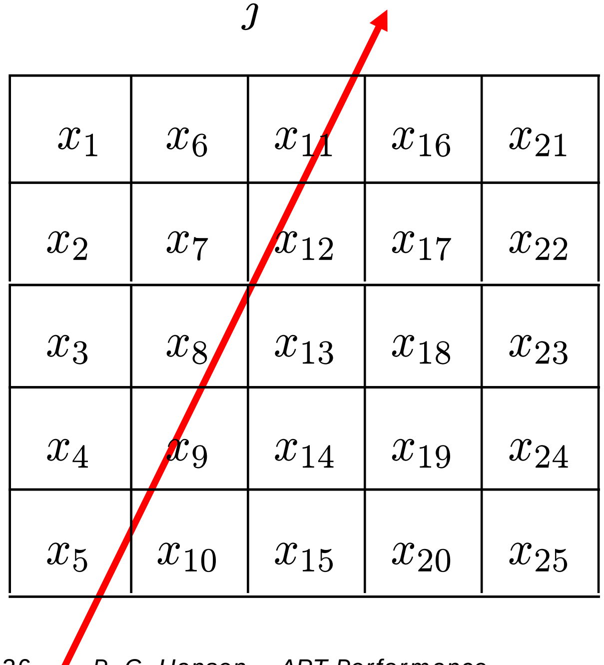
Setting Up the Algebraic Model

The data b_i associated with the ith X-ray through the domain:

$$b_i = \int_{\mathrm{ray}_i} \xi(s,t)\, d\ell, \qquad \xi = \text{attenuation coef.}$$

Assume ξ is a constant x_j in pixel j. This leads to:

$$b_i = \sum_j a_{ij}\, x_j, \qquad a_{ij} = \begin{cases} \text{length of ray } i \text{ in pixel } j \\ 0 \text{ if ray } i \text{ does not intersect pixel } j. \end{cases}$$



For the ith ray shown in red:

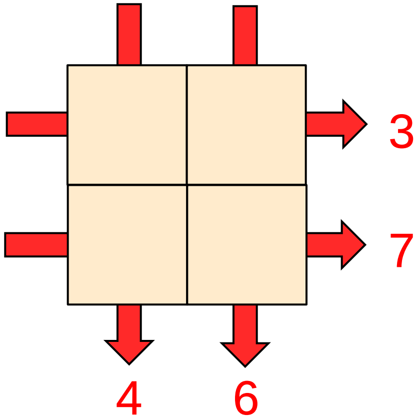
$$\begin{aligned} b_i \;&=\; a_{i,5}\, x_5 + a_{i8}\, x_8 + a_{i9}\, x_9 + \cdots \\ &\quad\; a_{i,10}\, x_{10} + a_{i,11}\, x_{11} + a_{i,12}\, x_{12} \end{aligned}$$

The corresponding row of A:

$$A(i,:) = (0\,0\,0\,0 \times 0\,0 \times \times \times \times \times 0\,0\,0 \cdots 0)$$

The matrix is **sparse** – it has lots of zeros!

Analogy: the “Sudoku” Problem – 数独



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

<u>Infinitely many solutions ($c \in \mathbb{R}$):</u>

$$\begin{array}{|c|c|}\hline & \\ \hline & \\ \hline \end{array} = \begin{array}{|c|c|}\hline 1 & 2 \\ \hline 3 & 4 \\ \hline \end{array} + c \times \begin{array}{|c|c|}\hline -1 & 1 \\ \hline 1 & -1 \\ \hline \end{array}$$

0	3
4	3

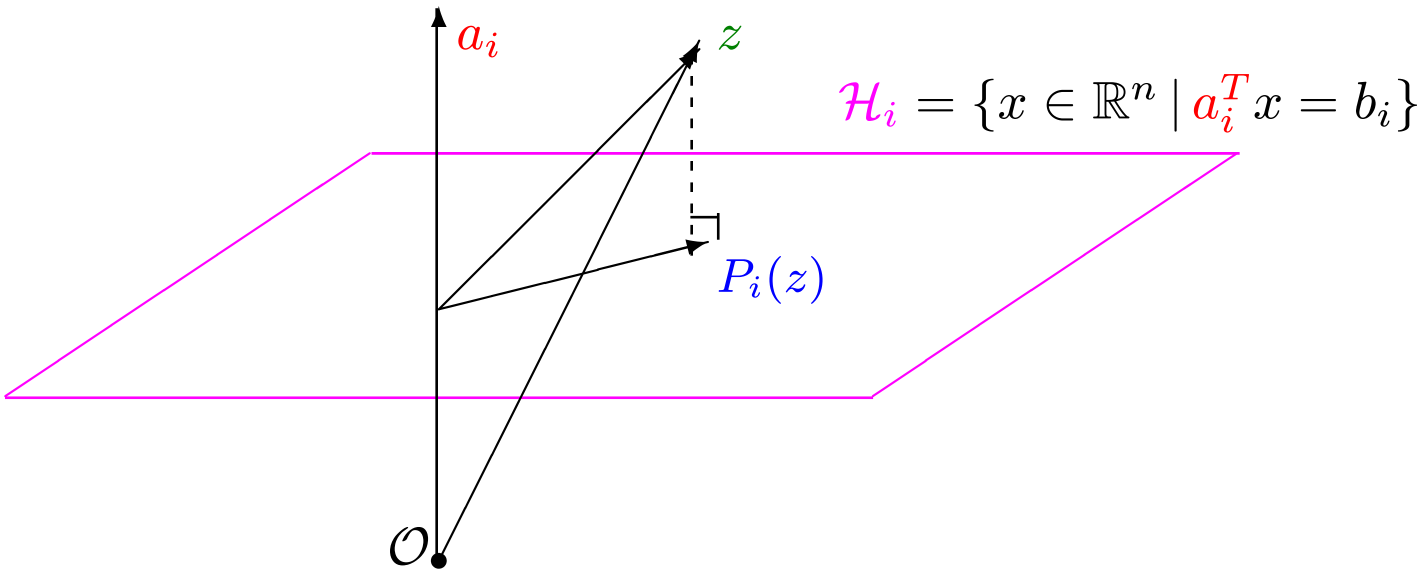
1	2
3	4

2	1
2	5

Prior: solution is integer and non-negative

3	0
1	6

Orthogonal Projection of Affine Hyperplane



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

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

In words, we scale the row vector a_i by $(b_i - a_i^T z)/\|a_i\|_2^2$ and add it to z.

ART History

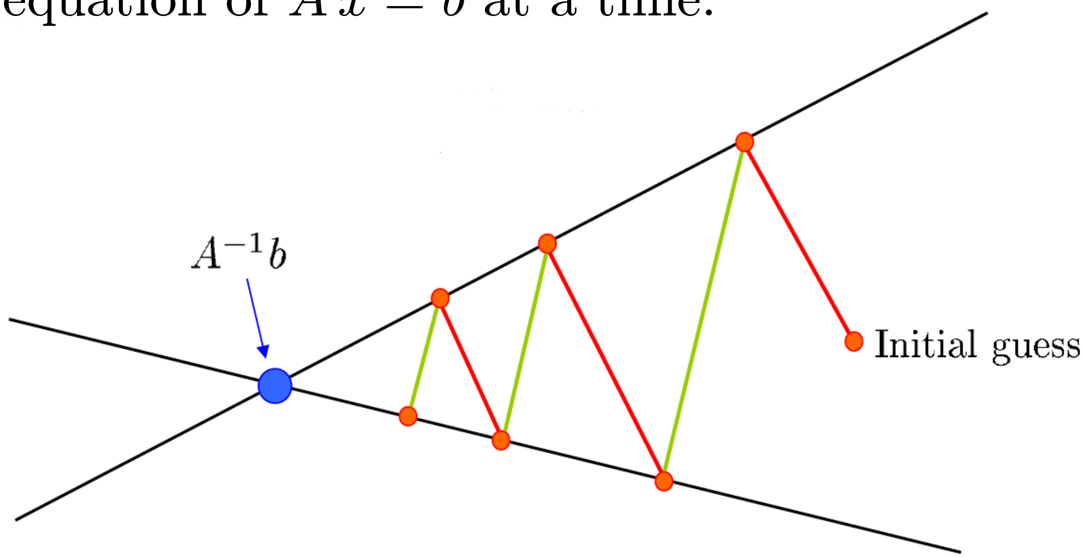
Kaczmarz (1937): orthogonally project x on the hyperplane defined by the ith row a_i^T and the corresponding element b_i of the right-hand side:

$$x \leftarrow \mathcal{P}_i(x) = x + \frac{b_i - a_i^T x}{\|a_i\|_2^2}\, a_i \ , \qquad i = 1, 2, \ldots, m \ .$$

Gordon, Bender, Herman (1970): coined the term "ART" and also introduced a

Satisfy one equation of $A\,x = b$ at a time:

x ... n .

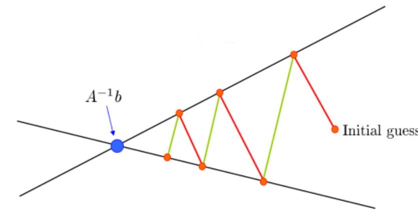


Herman, L... ...eters $\omega_k < 2$:

Today ART ... on a convex set:

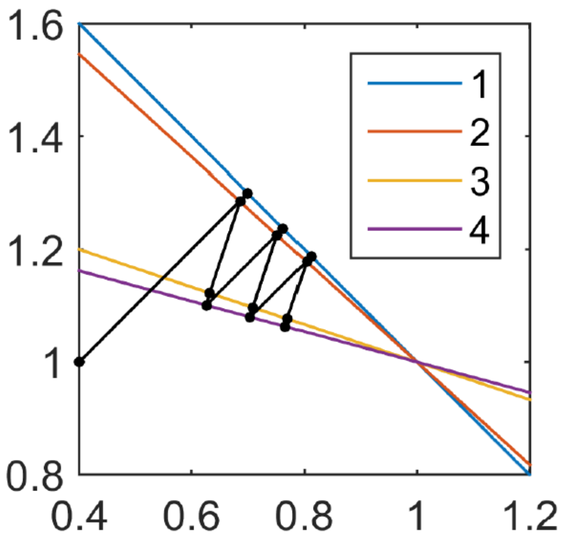
$$x \leftarrow \mathcal{P}_{\mathcal{C}}\left(x + \omega_k \frac{b_i - a_i^T x}{\|a_i\|_2^2}\, a_i \right) \ , \qquad i = 1, 2, \ldots, m \ .$$

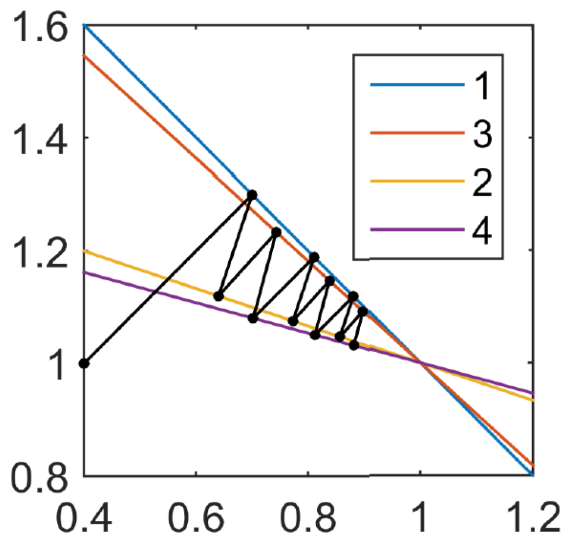
Convergence Issues



If the system $A\,x = b$ is consistent then ART converges to $\bar{x} = A^{\dagger}b$.

Difficulty: *ordering* of the rows of A influences the convergence rate:





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

The ordering 1–3–2–4 er preferable and almost twice as fast.

Convergence of ART

Assume that we *select the rows randomly*, 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:

Strohmer & Vershynin, 2009

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

where $\bar{x} = A^{-1}b$ and $\kappa = \|A\|_2\,\|A^{-1}\|_2$. **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* over all the rows of A, the reduction factor is $1 - 1/\kappa^2$.

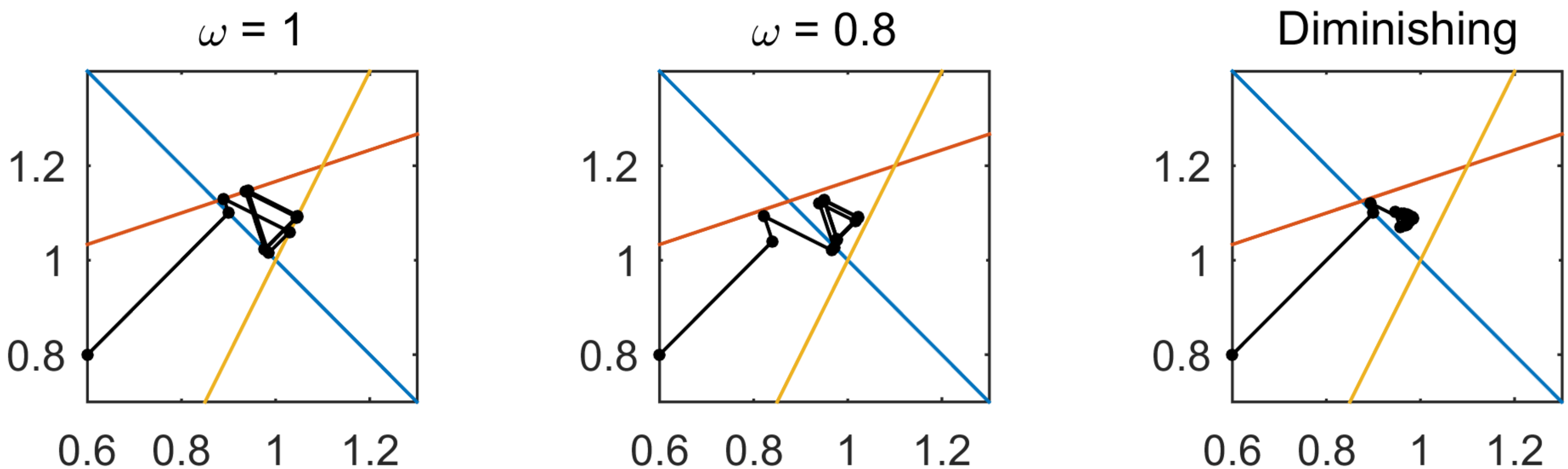
Note: there are often orderings for which the convergence is faster!

Iteration-Dependent Relax. Parameter

For inconsistent systems, ART with a fixed relaxation parameter ω has cyclic and non-convergent behavior.

With the *diminishing relaxation parameter* $\omega_k = 1/\sqrt{k} \to 0$ as $k \to \infty$ the iterates converge to a weighted least squares solution:

$$\bar{x}_M = \arg\min_x \|D^{-1}(b - A\,x)\|_2 \ , \qquad D = \mathrm{diag}(\|a_i\|_2) \ .$$



There is also a *column version* of ART which always converges to the standard least squares solution $\to$ end of this talk.

ART: Projected Incremental Gradient Method

Consider the constrained weighted least squares problem

$$\min_x \, {}^1\!/\!{}_2 \|D^{-1}(b - A\,x)\|_2^2 \qquad \text{subject to} \qquad x \in \mathcal{C}$$

with $D = \mathrm{diag}(\|a_i\|_2)$, and then write the objective function as

$$ {}^1\!/\!{}_2 \|D^{-1}(b - A\,x)\|_2^2 = \sum_{i=1}^{n} f_i(x)$$

$$f_i(x) = {}^1\!/\!{}_2 \frac{(b_i - a_i^T x)^2}{\|a_i\|_2^2} \qquad \Rightarrow \qquad \nabla f_i(x) = -\frac{b_i - a_i^T x}{\|a_i\|_2^2}$$

Incremental gradient methods use only the gradient of one singe term $f_i(x)$ in each iteration, leading to the ART update:

$$x \leftarrow \mathcal{P}_{\mathcal{C}}\left(x + \omega_k \, \frac{b_i - a_i^T x}{\|a_i\|_2^2} \, a_i \right) , \qquad i = 1, 2, \ldots, m \, ,$$

where $\mathcal{P}_{\mathcal{C}}$ = projection on convex set $\mathcal{C}$ (e.g., nonneg. or box constr.).

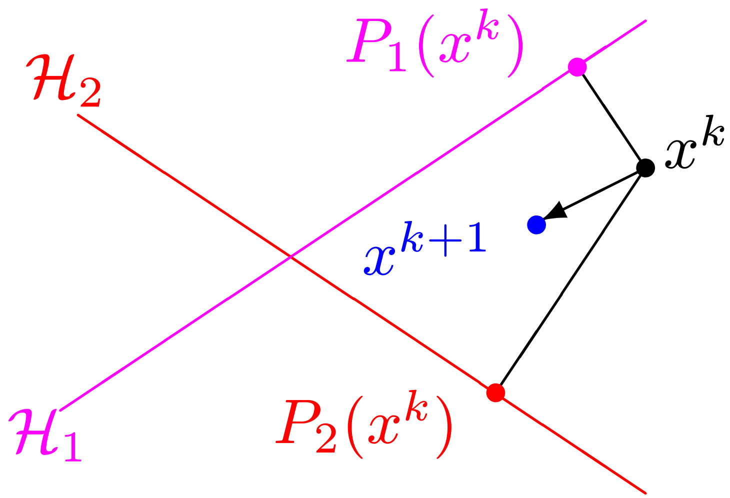
From Sequantial to Simultaneous Updates

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

$$x^{k+1} = \frac{1}{m}\sum_{i=1}^{m} P_i\big(x^k\big) = \frac{1}{m}\sum_{i=1}^{m}\Big(x^k + \frac{b_i - a_i^T x^k}{\|a_i\|_2^2}\, a_i\Big)$$

$$= x^k + \frac{1}{m}\sum_{i=1}^{m}\frac{b_i - a_i^T x^k}{\|a_i\|_2^2}\, a_i = x^k + {}^1\!/\!_m A^T D^{-2}(b - A\,x^k)$$

$$D = \mathrm{diag}(\|a_i\|_2)$$



Cimmino's Method

We obtain the following formulation:

Cimmino's algorithm

$x^{(0)}$ = initial vector
for $k = 0, 1, 2, \ldots$
$$x^{k+1} = x^k + A^T M \big(b - A\, x^{(k)} \big) \quad , \qquad M = {}^1\!/_m D^{-2}$$
end

Note that one iteration here involves all the rows of A, while one iteration in ART involves a single row.

Therefore, the computational work in one Cimmino iteration is equivalent to m iterations (a sweep over all the rows) in ART.

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

Convergence of Cimmino's Method

Assume that A is invertible and that the rows of A are scaled such that $\|A\|_2^2 = m$. Then. with $\bar{x} = A^{-1}b$

Nesterov, 2004

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

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

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

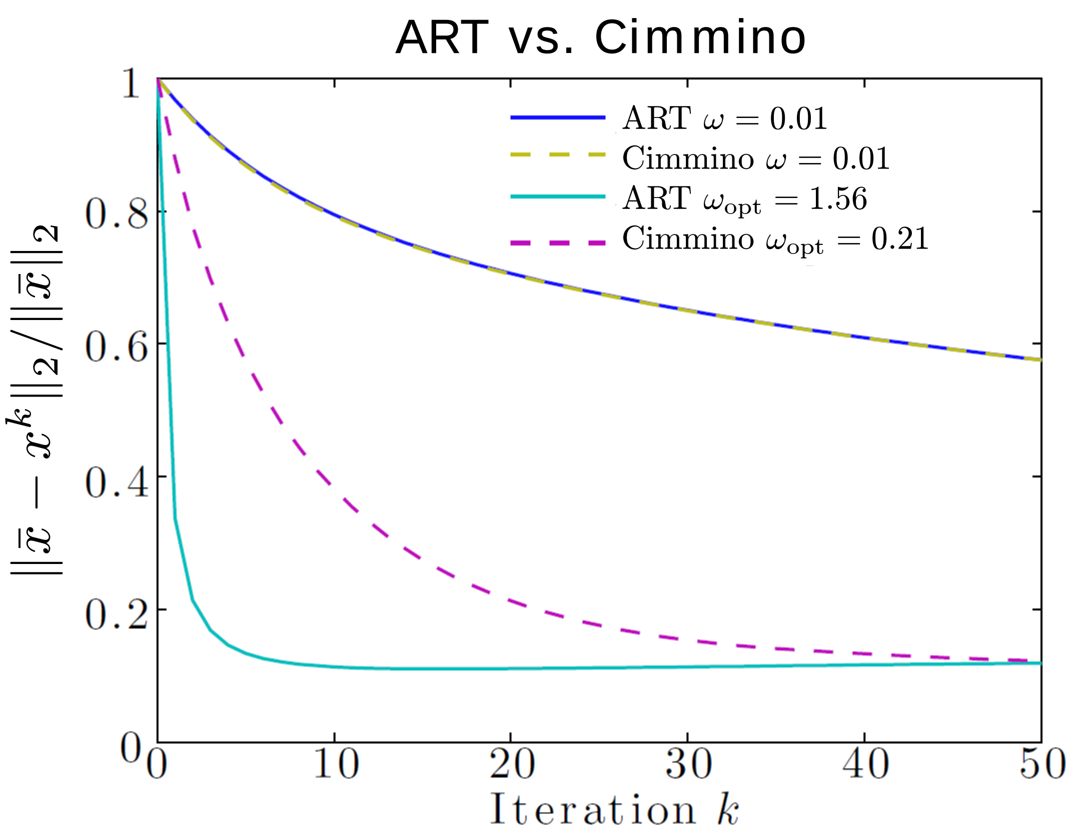
$$\|\bar{x} - x^k\|_2^2 \lesssim (1 - 2/\kappa^2)^k \, \|\bar{x} - x^0\|_2^2,$$

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

This is $\approx$ the same factor as in one sweep through the rows of A in ART.

Performance Issues

In these numerical experiments we compute and store A explicitly!



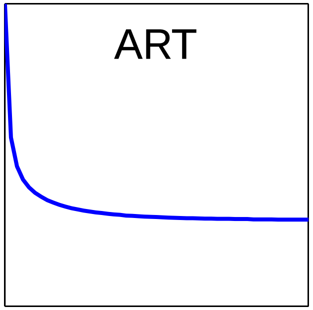
$\omega_{\rm opt}$ gives fastest convergence.

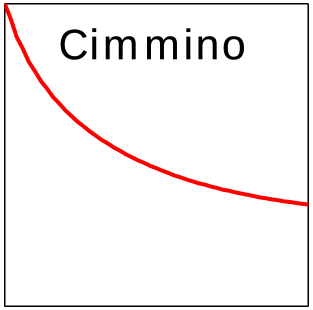
Cimmino:
slow convergence.

ART can converge a lot *faster* than SIRT.

Sørensen & Hansen, 2014

Computing Times





Intel Xeon E5620
2.40 GHz (4 cores)

Four cores are better suited for **block** matrix-vector operations.

	ART	Cimmino 1 core	Cimmino 4 cores
$m \times n$	t/iter	t/iter	t/iter
$13 \cdot 128^2 \times 64^3$	0.08 s	0.08 s	0.04 s
$13 \cdot 256^2 \times 128^3$	0.93 s	1.02 s	0.41 s
$13 \cdot 512^2 \times 256^3$	10.8 s	14.7 s	4.12 s

ART has more reduction of the error per iteration.

Cimmino can better take advantage of *multi-core architecture*.

How to achieve the "best of both worlds?" $\rightarrow$ Block methods!

Block Methods (Ordered Subset Methods)

$$A = \begin{pmatrix} A_1 \\ A_2 \\ \vdots \\ A_p \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_p \end{pmatrix}, \qquad A_\ell \in \mathbb{R}^{m_\ell \times n}, \quad \ell = 1, \ldots, p,$$

In each iteration we can:

- Treat *all blocks* sequentially or simultaneously (i.e., in parallel).
- Treat *each block* by an iterative method or by a direct computation.

We obtain several methods:

- ~~Sequential processing + ART on each block → classical ART~~
- Sequential processing + SIRT on each block
- Sequential processing + pseudoinverse of A_ℓ
- Parallel processing + ART on each block
- ~~Parallel processing + SIRT on each block → classical SIRT~~
- Parallel processing + pseudoinverse of A_ℓ

Block-Sequential Methods

SART: Andersen, Kak (1984)
Block-Iteration: Censor (1988)

Initialization: choose an arbitrary $x^0 \in \mathbb{R}^n$
Iteration: for $k = 0, 1, 2, \ldots$

$$z \leftarrow x^k$$
$$z \leftarrow P\big(z + \omega\, A_\ell^T\, M_\ell\, (b_\ell - A_\ell\, z) \big), \quad \ell = 1, 2, \ldots, p$$
$$x^{k+1} \leftarrow z$$

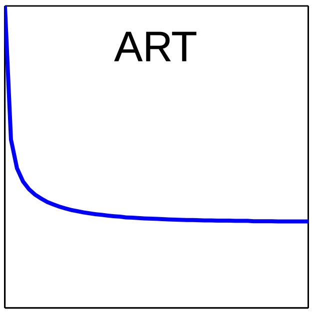
The convergence depends on the number of blocks *p*:

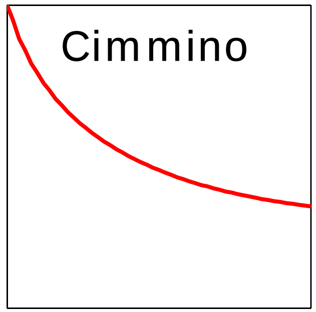
- If *p* = 1, we recover Cimmino
- If *p* = *m*, we recover ART

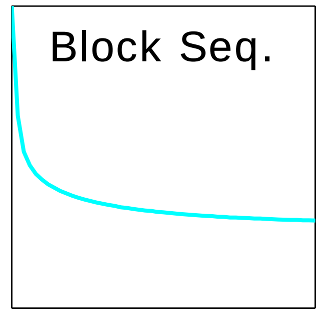
Parallelism within each block of m/p rows

Variant by **Elfving** (1980): $M_\ell = (A_\ell A_\ell^T)^\dagger \;\Rightarrow\; A_\ell^T M_\ell = A_\ell^\dagger$

Block Sequential Performance







Intel Xeon E5620
2.40 GHz (4 cores)

	ART	Cimmino	Block Seq.
$m \times n$	t/iter	t/iter	t/iter
$13 \cdot 128^2 \times 64^3$	0.08 s	0.04 s	0.05 s
$13 \cdot 256^2 \times 128^3$	0.93 s	0.41 s	0.48 s
$13 \cdot 512^2 \times 256^3$	10.8 s	4.12 s	4.36 s

- The ”building blocks” are Cimminoiterations, suited for multicore.
- The error reduction per iteration is close to that of ART.

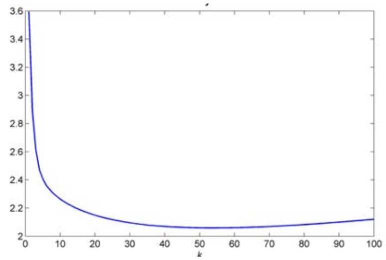
Semi-Convergence

During the first iterations, the iterates x^k capture the “important” information in the noisy right-hand side b.

- In this phase, the iterates x^k approach the exact solution $\bar{x}$.

At later stages, the iterates starts to capture undesired noise components.

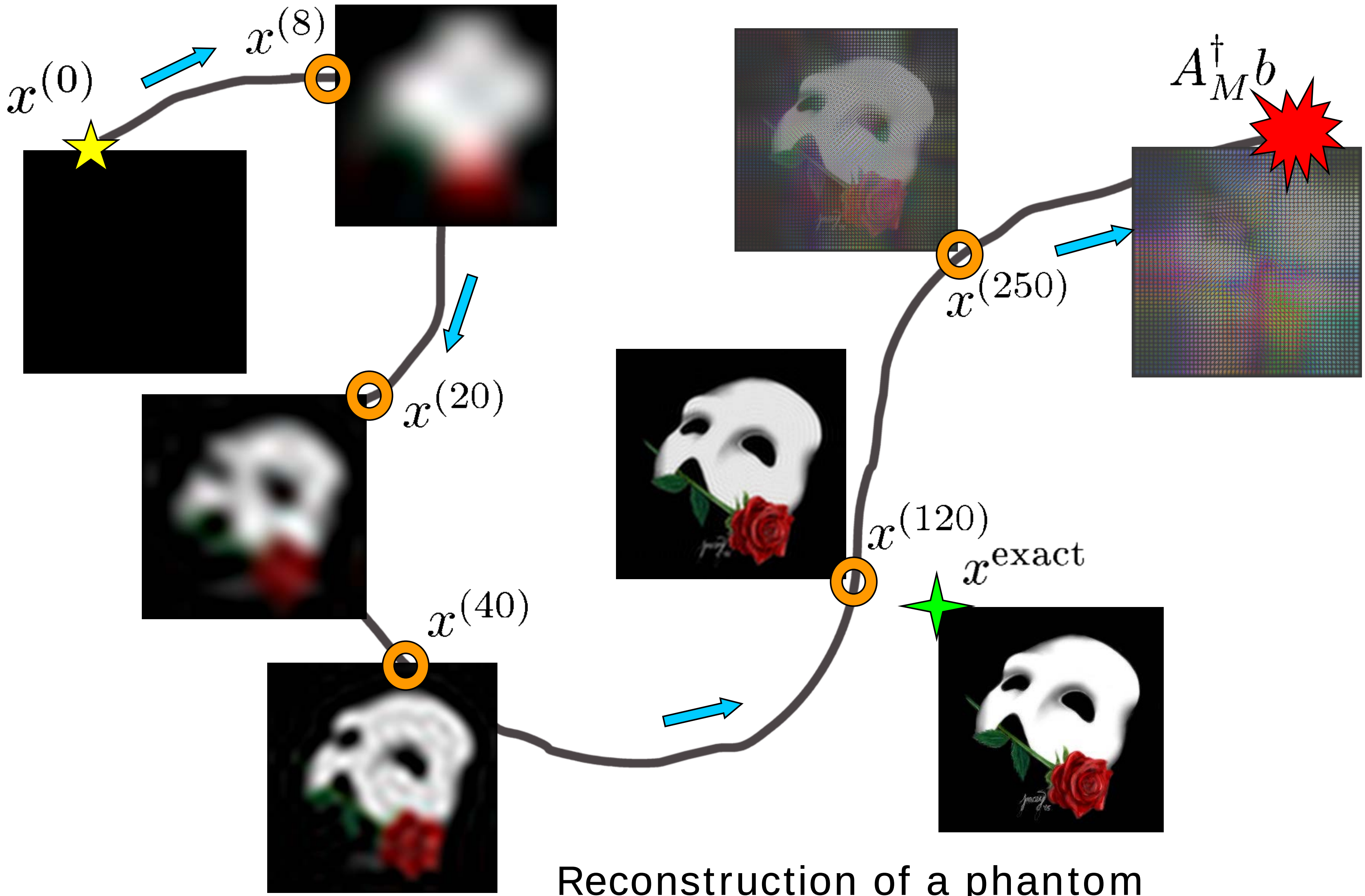
- Now the iterates x^k diverge from the exact solution and they approach the undesired solution $A^{-1}b$.



This behavior is called *semi-convergence*.

- F. Natterer, *The Mathematics of Computerized Tomography* (1986)
- A. van der Sluis & H. van der Vorst, *SIRT- and CG-type methods for the iterative solution of sparse linear least-squares problems* (1990)
- M. Bertero & P. Boccacci, *Inverse Problems in Imaging* (1998)
- M. Kilmer & G. W. Stewart, *Iterative Regularization And Minres* (1999)
- H. W. Engl, M. Hanke & A. Neubauer, *Regularization of Inverse Problems* (2000)

Illustration of Semi-Convergence



Reconstruction of a phantom

Analysis of Semi-Convergence

Let $\bar{x}$ = solution to noise-free problem, and let x^k and $\bar{x}^k$ denote the iterates when applying ART to b and $\bar{b} = A\,\bar{x}$:

$$\|\bar{x} - x^k\|_2 \;\square\; \|\bar{x} - \bar{x}^k\|_2 + \|\bar{x}_k - x^k\|_2 \;.$$

Iteration error

Noise error

Convergence theory for ART for noise-free data is well established and ensures that the iteration error $\bar{x} - \bar{x}^k$ goes to zero. See the convegence results in the previous slides.

Our concern here is the noise error $e_{\mathrm{N}}^k = \bar{x}^k - x^k$. We wish to establish that it increases, and how fast.

Analysis of Semi-Convergence – Cimmino

Consider the Cimmino's methods with the SVD:

$$M^{1/2} A = U\,\Sigma\,V^T = \textstyle\sum_{i=1}^n u_i\,\sigma_i\,v_i^T\,.$$

Then x^k is a filtered SVD solution:

$$x^k = \textstyle\sum_{i=1}^n \varphi_i^{[k]}\,\frac{u_i^T (M^{\frac{1}{2}} b)}{\sigma_i}\,v_i\,, \qquad \varphi_i^{[k]} = 1 - \left(1 - \omega\,\sigma_i^2\right)^k.$$

Recall that we solve *noisy* systems $A\,x = b$ with $b = A\,\bar{x} + e$.

The ith component of the error, in the SVD basis, is

$$v_i^T(\bar{x} - x^k) = \left(1 - \varphi_i^{[k]}\right) v_i^T\,\bar{x} - \varphi_i^{[k]}\,\frac{u_i^T (M^{\frac{1}{2}} e)}{\sigma_i}.$$

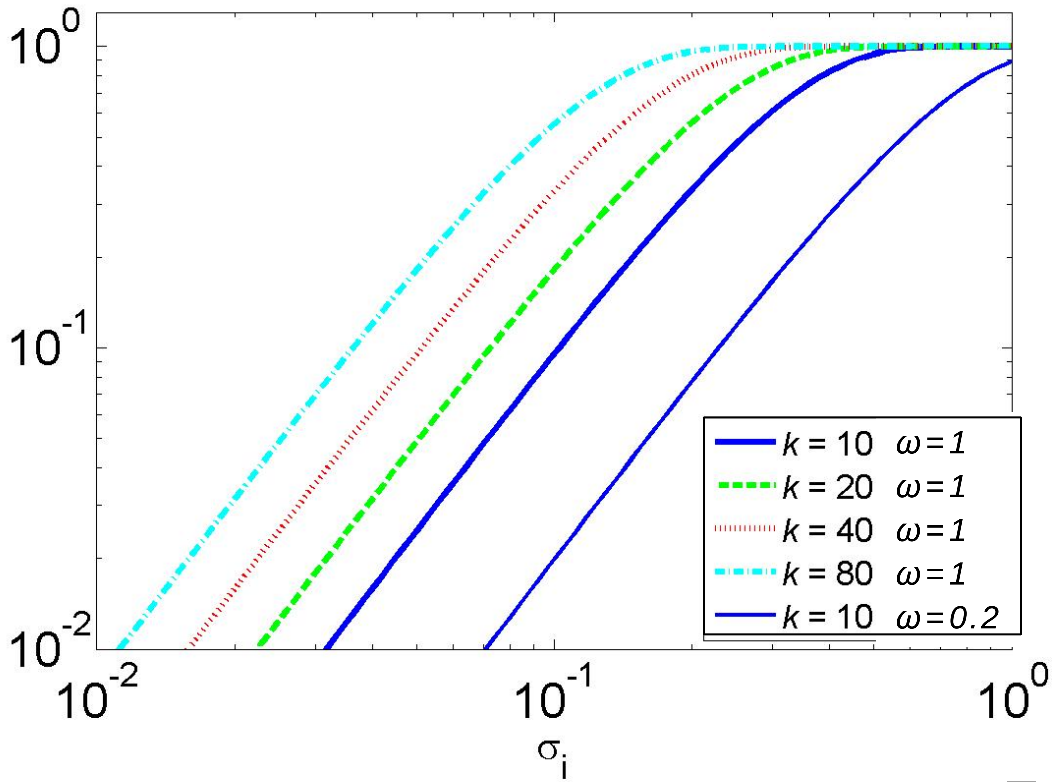
Iteration error

Noise error

Van der Sluis & Van der Vorst, 1990

The Behavior of the Filter Factors

Filter factors $\varphi_i^{[k]} = 1 - \left(1 - \omega\,\sigma_i^2\right)^k$



The filter factors *dampen* the "inverted noise" $u_i^T(M^{\frac{1}{2}}e)/\sigma_i$.

$$\omega\,\sigma_i^2 \ll 1 \Rightarrow \varphi_i^{[k]} \approx k\,\omega\,\sigma_i^2 \Rightarrow k \text{ and } \omega \text{ play the same role.}$$

Noise Error – Projected Cimmino

The iteration and noise error in *projected* Cimmino are bounded by

$$\|\bar{x} - \bar{x}^k\|_2 \quad \square \quad (1 - \omega\,\sigma_n^2)\,\|\bar{x} - x^0\|_2$$

$$\|\bar{x}^k - x^k\|_2 \quad \square \quad \frac{\sigma_1}{\sigma_n}\,\frac{1-(1-\omega\sigma_n^2)^k}{\sigma_n}\,\|M^{1/2}\delta b\|_2 .$$

Elfving, H, Nikazad, 2012

As long as $\omega\sigma_n^2 \ll 1$ we have

$$\|\bar{x}^k - x^k\|_2 \approx k\,\omega\,\sigma_1 \|M^{1/2}\delta b\|_2 .$$

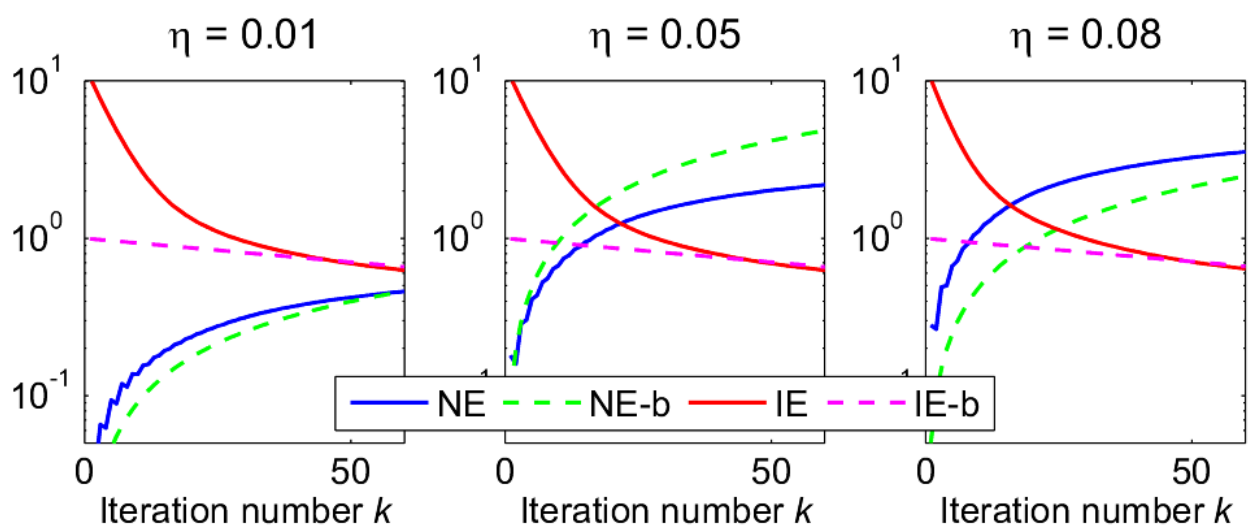
NE: actual noise error

NE-b: our bound

IE: actual iteration error

IE-b: our bound without the factor $\|\bar{x} - x^0\|_2$

We *track* the errors well.



Noise Error – ART

Successive Over-Relaxation

ART is equivalent to applying SOR to $A\,A^T y = b$, $x = A^T y$. Splitting:

$$A A^T = L + D + L^T, \qquad \widehat{M} = (D + \omega L)^{-1},$$

where L is strictly lower triangular and $D = \mathrm{diag}(\|a_i\|_2^2)$. Then:

$$\boxed{x^{k+1} = x^k + \omega A^T \widehat{M}\,(b - A\,x^k)\ .}$$

We introduce: $e = b - \bar{b} = \text{noise in data}$, $\quad Q = I - \omega A^T \widehat{M}\,A.$

Then simple manipulations show that the noise error is given by

$$e_{\mathrm{N}}^k = x^k - \bar{x}^k = Q\ e_{k-1}^{\mathrm{N}} + \omega A^T \widehat{M}\,e = \omega \sum_{j=1}^{k-1} Q^j A^T \widehat{M}\,e\ .$$

After some work (see the paper) we obtain the bound

$$\boxed{\|e_{\mathrm{N}}^k\|_2 \approx k\,\omega\,\|A^T \widehat{M}\,e\|_2.}$$

Elfving, H, Nikazad, 2014

Noise Error Analysis – A Tighter Bound

Further analysis (see the paper) shows that the noise error in ART is bounded above as:

$$\begin{aligned} \|e_{\mathrm{N}}^k\|_2 \quad &\square \quad \frac{1-(1-\omega\sigma_{\min}^2)^k}{\sigma_{\min}} \, \frac{\|A^T \widehat{M}\, e\|_2}{\sigma_{\min}} + \mathcal{O}(\sigma_{\min}^2), \\ \sigma_{\min} \quad &= \quad \text{smallest singular value of } A. \end{aligned}$$

As long as $\omega\sigma_{\min}^2 < 1$ we have

$$\frac{1-(1-\omega\sigma_{\min}^2)^k}{\sigma_{\min}} \;\square\; \sqrt{k}\,\sqrt{\omega}$$

and thus

$$\boxed{\|e_{\mathrm{N}}^k\|_2 \;\square\; \sqrt{k}\, \frac{\sqrt{\omega}\,\|A^T \widehat{M} e\|_2}{\sigma_{\min}} + \mathcal{O}(\sigma_{\min}^2).}$$

Elfving, H, Nikazad, 2012

This also holds for *projected* ART provided that A and $\mathcal{P}_{\mathcal{C}}$ satisfy

$$y \in \mathcal{R}(A^T) \Rightarrow \mathcal{P}_{\mathcal{C}}\, y \in \mathcal{R}(A^T).$$

Column Iterations

This algorithm operates on the columns a_j of A, instead of the rows.

"Rows are red and columns are blue, ..."

This method always converges to a least squares solution, and it may also have an advantage from an implementation point of view.

- A. de la Garza, *An iterative method for solving systems of linear equations*, Oak Ridge, Report K-731, 1951.
- D. W. Watt, *Column-relaxed algebraic reconstruction algorithm for tomography with noisy data*, Appl. Opt. 33, 4420–4427, 1994.

The column-action method takes its basis in the simple coordinate descent optimization algorithm, in which each step is performed cyclically in the direction of the unit vectors

$$e_j = (\,\underbrace{0\ 0\ \cdots\ 0}_{j-1}\ 1\ \underbrace{0\ 0\ \cdots\ 0}_{n-j-1}\,), \qquad j = 1, 2, \ldots, n.$$

Derivation

The least-squares objective function is

$$f(x) = {}^1\!/\!{}_2\,\|A\,x - b\|_2^2.$$

At iteration k we consider the update

$$x^k + \alpha_k\, e_j, \qquad j = k \pmod n.$$

Step length α_k that gives maximum reduction in objective function:

$$\begin{aligned}
\alpha_k &= \mathrm{argmin}_\alpha\, {}^1\!/\!{}_2 \|A\,(x^k + \alpha\, e_j) - b\|_2^2 \\
&= \mathrm{argmin}_\alpha\, {}^1\!/\!{}_2 \|\alpha\,(A\,e_j) - (b - A\,x^k)\|_2^2 \\
&= \mathrm{argmin}_\alpha\, {}^1\!/\!{}_2 \|a_j\,\alpha - (b - A\,x^k)\|_2^2.
\end{aligned}$$

The minimizer is

$$\alpha_k = (a_j)^\dagger (b - A\,x^k) = \frac{a_j^T (b - A\,x^k)}{\|a_j\|_2^2}\ .$$

Formulation of the Algorithm

Hence we obtain the following overall algorithm (where again we have introduced a relaxation parameter ω_k and a projection $\mathcal{P}_\mathcal{C}$):

$$
\begin{aligned}
& x^0 = \text{initial vector} \\
& \text{for } k = 0, 1, 2, \ldots \\
& \qquad j = k \pmod{n} \\
& \qquad x^{k+1} = \mathcal{P}_\mathcal{C}\left(x^k + \omega_k \, \frac{a_j^T (b - A\, x^k)}{\|a_j\|_2^2} \, e_j \right) \quad . \\
& \text{end}
\end{aligned}
$$

Note that the operation in the inner loop simply overwrites the jth element of the iteration vector with an updated value:

$$
x_j \leftarrow \mathcal{P}_\mathcal{C}\left(x_j + \omega_k \, \frac{a_j^T (b - A\, x^k)}{\|a_j\|_2^2} \right) .
$$

Loping in the Column-Action Method

We can introduce a "loping" strategy where we don't update the solution element x_j^k if $d_j^k = \omega a_j^T r^{k,j} / \|a_j\|_2^2$ is small. This will save computational work for blocks that are not updated.

For $k = 1, 2, 3, \ldots$ (cycles or outer iterations)

For $j = 1, 2, \ldots, n$ (inner iterations)

$d_j^k = \omega a_j^T r^{k,j} / \|a_j\|_2^2$

If $\|d_j^k\|_2 > \tau$

$x_j^{k+1} \leftarrow x_j^k + d_j^k$

$r^k \leftarrow r^k - a_j (x_j^{k+1} - x_j^k)$

End

End

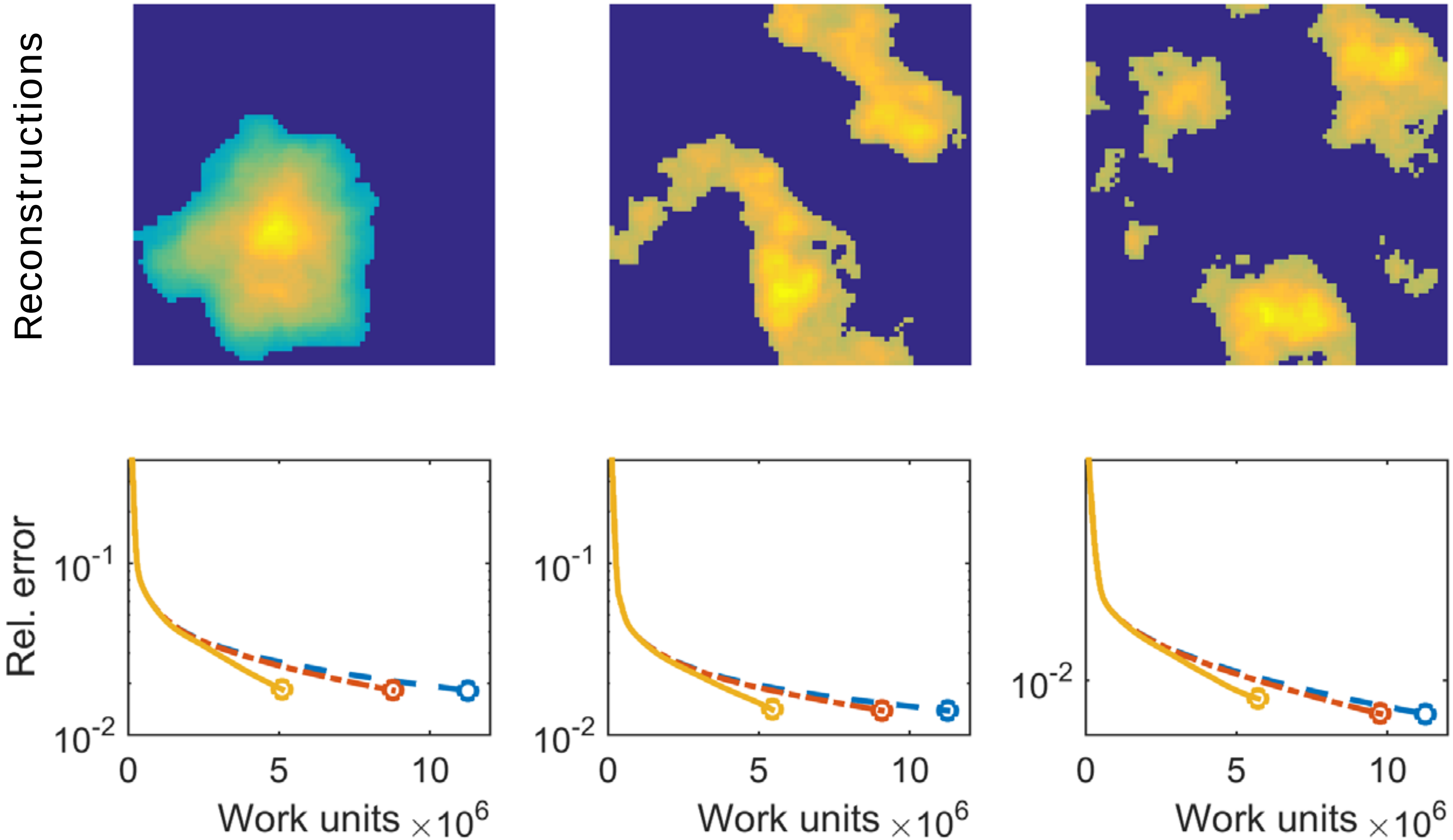
$r^{k+1} \leftarrow r^k$

End

Elfving, H, Nikazad, 2016

Numerical Results

Test image: `phantomgallery('ppower',75)` from AIR Tools with large regions of zeros and nonzeros; A is 19080×5625.



Conclusions

- Algebraic methods are fascinating algorithms with important applications in computed tomography.
- Their convergence properties are well understood.
- Block-sequential methods: fast performance because they combine good intrinsic convergence with good utilization of hardware.
- Semi-convergence provides the necessary filtering effect.
- Semi-convergence is quite well understood.
- Column-action methods allow us to reduce computational work by skipping unnecessary updates.