Column-Action Methods in Image Reconstruction

Per Christian Hansen

joint work with
Tommy Elfving
Touraj Nikazad

DTU Compute
Department of Applied Mathematics and Computer Science

Overview of Talk

Part 1: the classical row-action method = ART

- The advantage of algebraic formulations
- Advantages of the optimization view of ART

Part 2: the column-action method

- Motivtion
- Derivation
- Block version
- Convergence results

Part 3: saving computational work

- Loping and flagging – update only when necessary
- A few examples

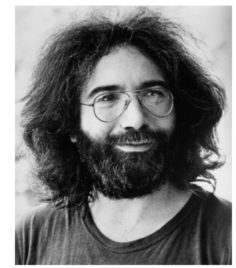
T. Elfving, P. C. Hansen, and T. Nikazad, *Convergence analysis for column-action methods in image reconstruction*, Numerical Algorithms, to appear.

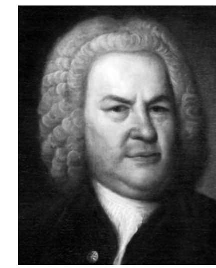
ART = Algebraic Reconstruction Technique
= A Classical Algorithm

Perspective:

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.





Our motivation: solve linear systems of equations $\boxed{A\,x = b}$ derived from discretization of an underlying tomography problem.

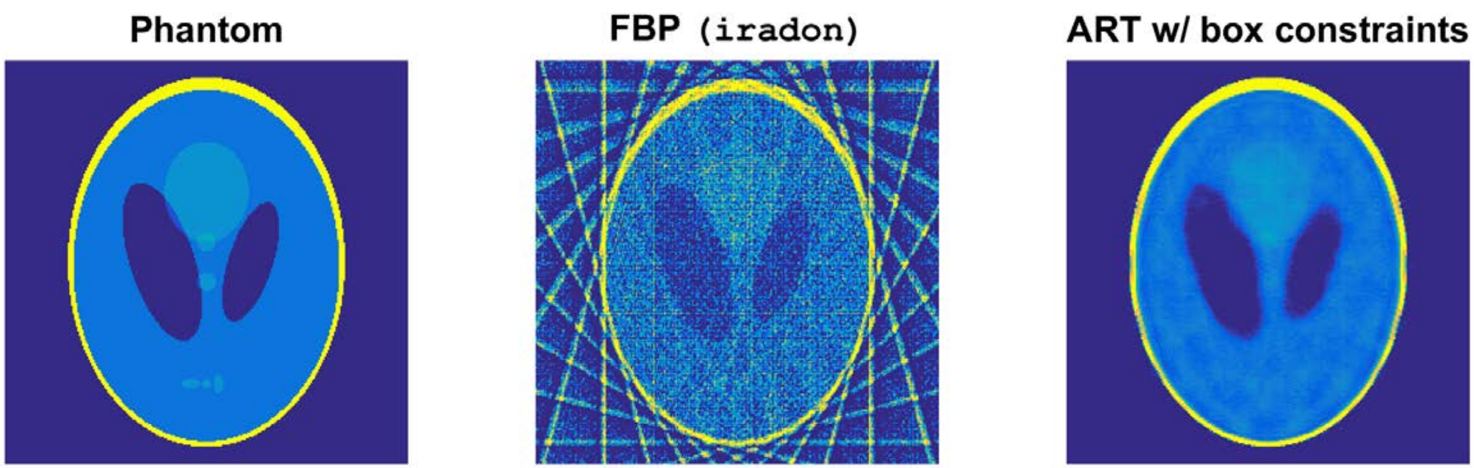
In this talk we do not pay attention to the discretization method.

ART is a simple iterative method for solving $A\,x = b$ where each iteration updates x via sweeps over the rows a_i^T of the matrix $A \in \mathbb{R}^{m \times n}$.

Filtered Back Projection (FBP) versus ART

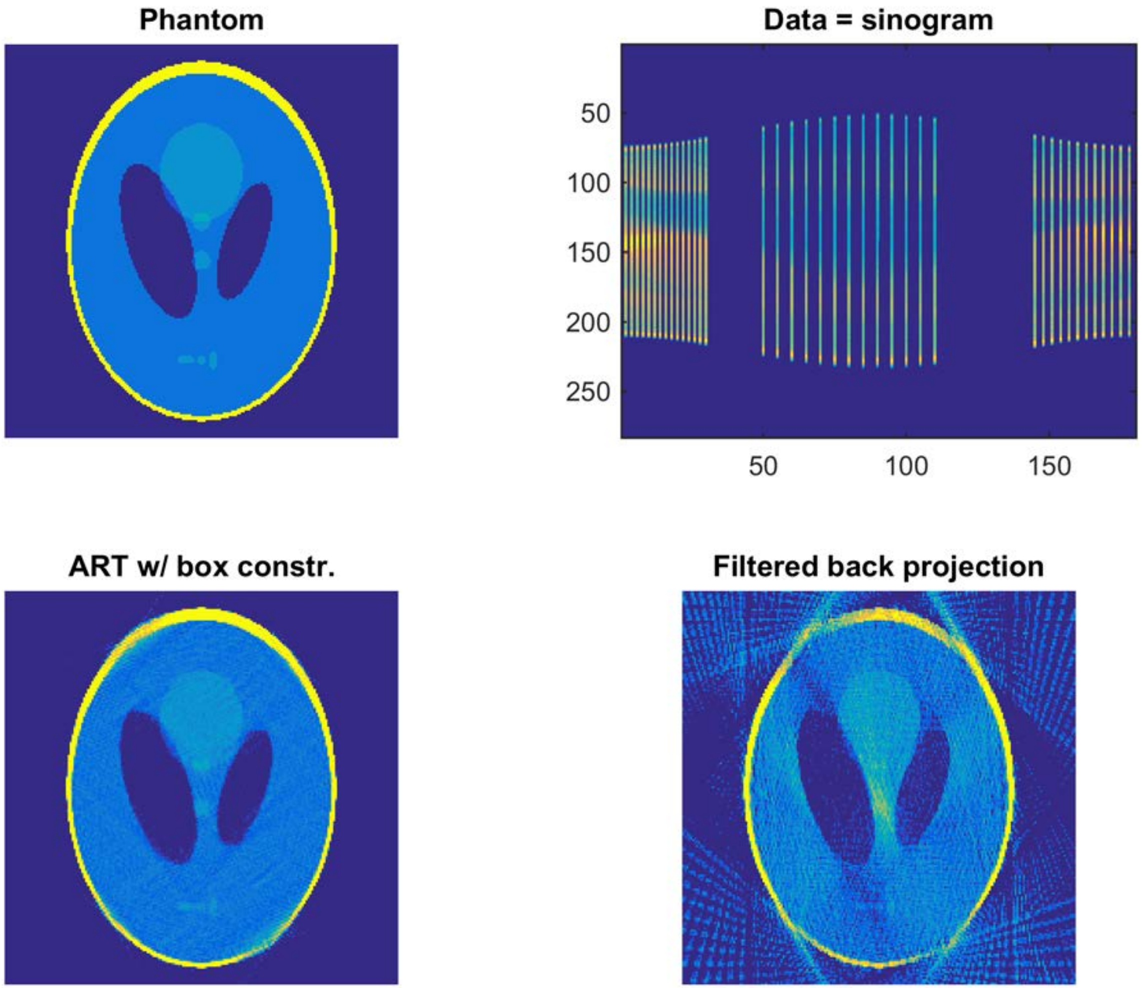
- FBP: low memory, works really well with many data.
- But *artifacts* appear with limited data, or nonuniform distribution of projection angles or ray.
- Difficult to incorporate constraints (e.g., nonnegativity) in FBP
- ART and other algebraic methods are more flexible and adaptive.

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



FBP versus ART – Limited Data

Irregularly spaced angles / "missing" angles also cause difficulties for FBP



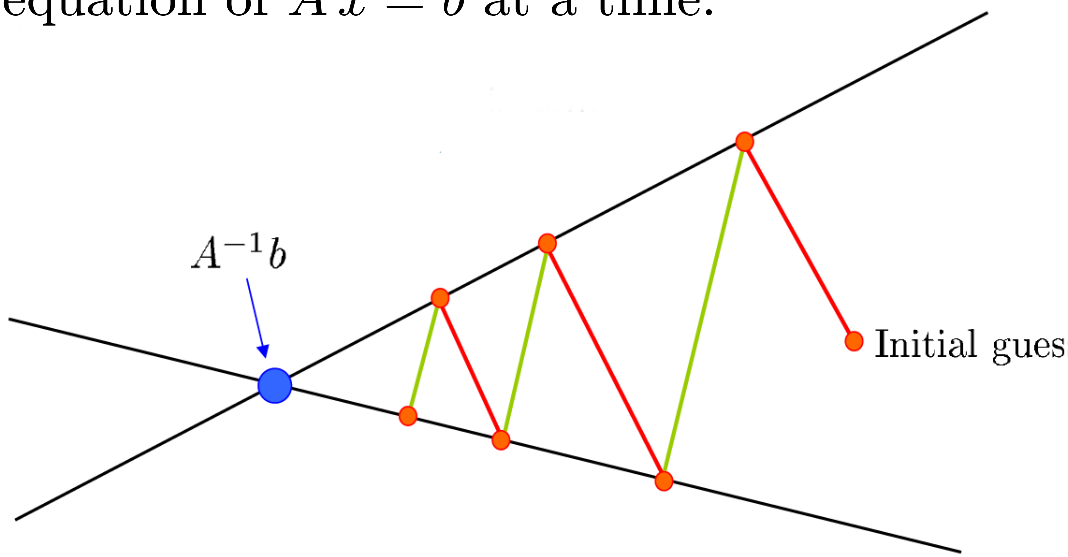
ART History

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 in-troduced a

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



$$x \quad \ldots \quad n \ .$$

Herman, L… …eters $\omega_k < 2$:

Today ART includes both ω_k and a projection $\mathcal{P}_\mathcal{C}$ on a convex set:

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

The Optimization Viewpoint

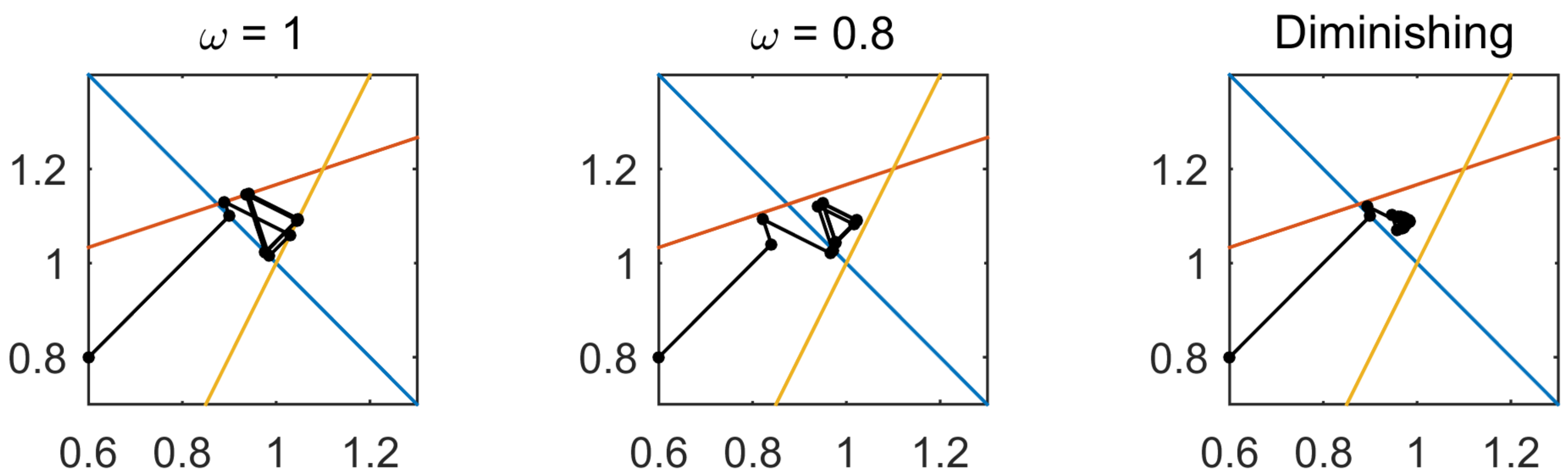
ART is usually considered as a solver for $A\,x = b$; but it is often more convenient to consider it as an **optimization method**.

- 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 ω, or
  - a parameter ω_k that changes with the iterations.
- In each updating step we can incorporate a *projection* $\mathcal{P}_\mathcal{C}$ on a suitably chosen convex set $\mathcal{C}$ that reflects prior knowledge.
- We can view it as a projected incremental gradient optimization method, which opens for further extensions and careful convergence analysis.

Iteration-Dependent Relax. Parameter

For inconsistent systems, basic ART with a fixed relaxation parameter ω gives 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.



There is also a *column version* of ART which always converges to the standard least squares solution.

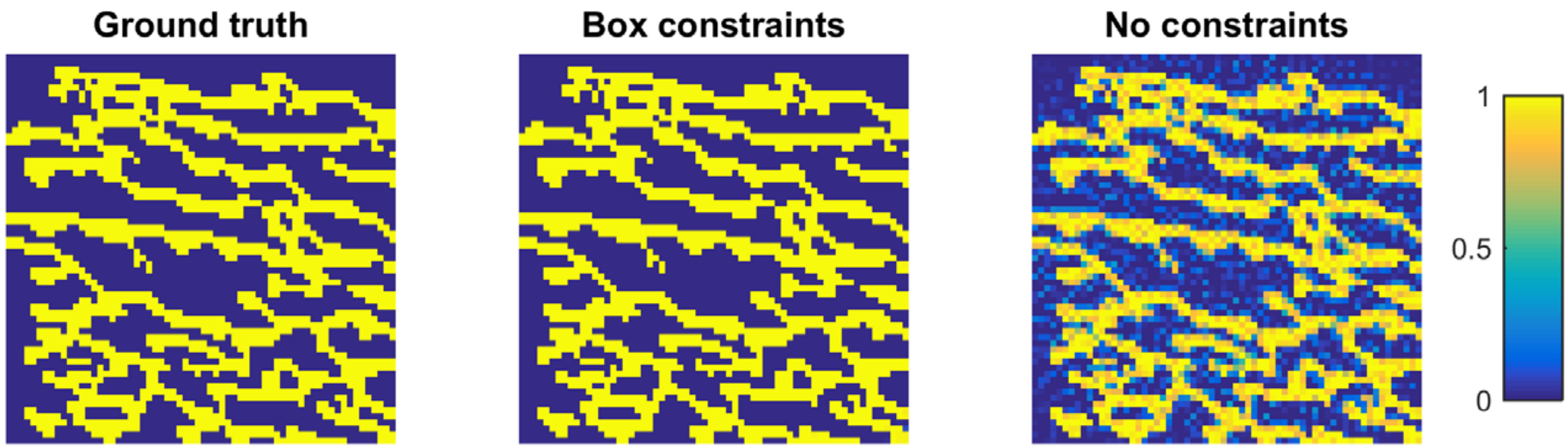
Simple Constraints

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 $\mathcal{C} = [0,1]^n$ corresponds to

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



= Projected Incremental Gradient Method

Consider the constrained weighted least squares problem

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

with $M = \mathrm{diag}(\|a_i\|_2^2)$, and then write the objective function as

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

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

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

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

Software for ART

- **SNARK09**: C++ package from NYU; 2D reconstructions.
  `www.dig.cs.gc.cuny.edu/software/snark09`

- **ASTRA**: MATLAB package with GPU accelleration and interfact to Python from Univ. of Antwerp + CWI, Amsterdam; 2D and 3D reconstructions.
  `sourceforge.net/p/astra-toolbox/wiki/Home`

- **Image reconstruction toolbox**: MATLAB package from Univ. of Michigan; 2D reconstructions.
  `web.eecs.umich.edu/~fessler/code`

- **AIR Tools**: MATLAB package from DTU; 2D reconstructions.
  `www.compute.dtu.dk/ pcha/AIRtools`

- **Xmipp**: C++ package from the Spanish National Biotechnology Centre; 3D electron microscopy.
  `xmipp.cnb.csic.es/twiki/bin/view/Xmipp/WebHome`

And Now: A Column-Action Method

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

It has the advantage that it always – even with a fixed relaxation parameter – converges to a least squares solution; if $m \geq n$ it converges to the (minimum-norm) least squares solution (see paper for proof).

Moreover, in some applications the column-action strategy may also have an advantage from an implementation point of view.

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 $x^{(k)} + \alpha_k \, e_j$ with $j = k \pmod n$, and the goal is to find the step length α_k that gives maximum reduction in the objective function:

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

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 and a projection):

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

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

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

Block Algorithm

Partition A into q block columns and partition x accordingly,

$$A = \left(A_1 \ A_2 \ \cdots \ A_q \right), \quad x = \begin{pmatrix} x_1 \\ \vdots \\ x_q \end{pmatrix},$$

and let $M_i \in \mathbb{R}^{n_i \times n_i}$, $i = 1, 2, \ldots, q$ be a set of given spd matrices.

Initialization: $x^0 \in \mathbb{R}^n$ is arbitrary; $r^{0,1} = b - A\,x^0$.
For $k = 0, 1, 2, \ldots$ (cycles or outer iterations)
  For $i = 1, 2, \ldots, q$ (inner iterations)
    $x_i^{k+1} = x_i^k + \omega_i M_i A_i^T r^{k,i}$
    $r^{k,i+1} = r^{k,i} - A_i (x_i^{k+1} - x_i^k)$
  End
  $r^{k+1,1} = r^{k,q+1}$
End

An Important Special Case of the Algorithm

Let a_i^j denote the jth column of block A_i and define the matrices

$$M_i = \frac{1}{n_i} \left(\mathrm{diag}\big(A_i^T A_i\big)\right)^{-1}$$

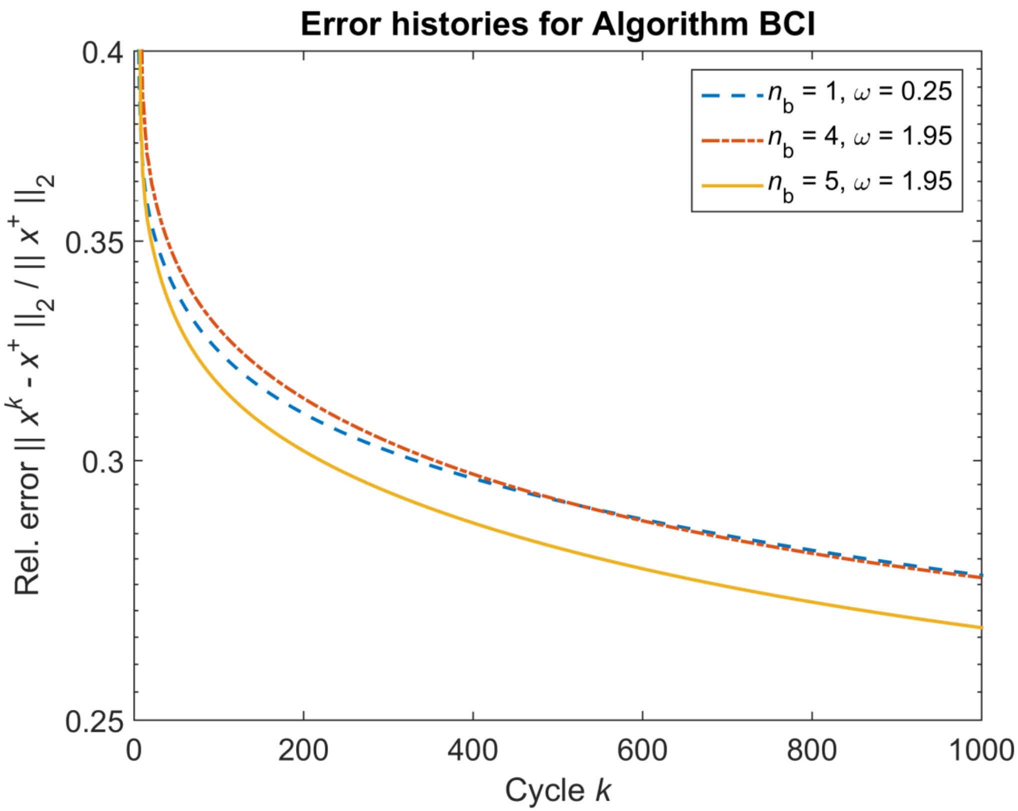
The condition for convergence is

$$\rho\big(A_i M_i A_i^T\big) = \big\|A_i M_i A_i^T\big\|_2 \square 1 \quad \Rightarrow \quad \omega_i \in (0, 2).$$

The upper bound 2 is only a sufficient condition and it may lead to slow rate of convergence.

A Numerical Example

Test problem: parallel-beam CT; no noise in the data.
Image is 50×50 Shepp-Logan phantom, detector has 71 pixels, and projection angles are $5°, 10°, \ldots, 180°$; thus A is 2556×2500.
All blocks have the same size $n_i = n_\mathrm{b}$ and $\omega_i = \omega$.



Loping in the Block Column Method

Haltmeier (2009) introduced a *loping* strategy for ART, which omits the updating step associated with block i if $|b_i - a_i^T x^{i-1}|$ is small.

We introduce a similar strategy where we don't update the solution block x_i^k if $d_i^k = \omega_i M_i A_i^T r^{k,i}$ has a small norm. This will save computational work for blocks that are not updated.

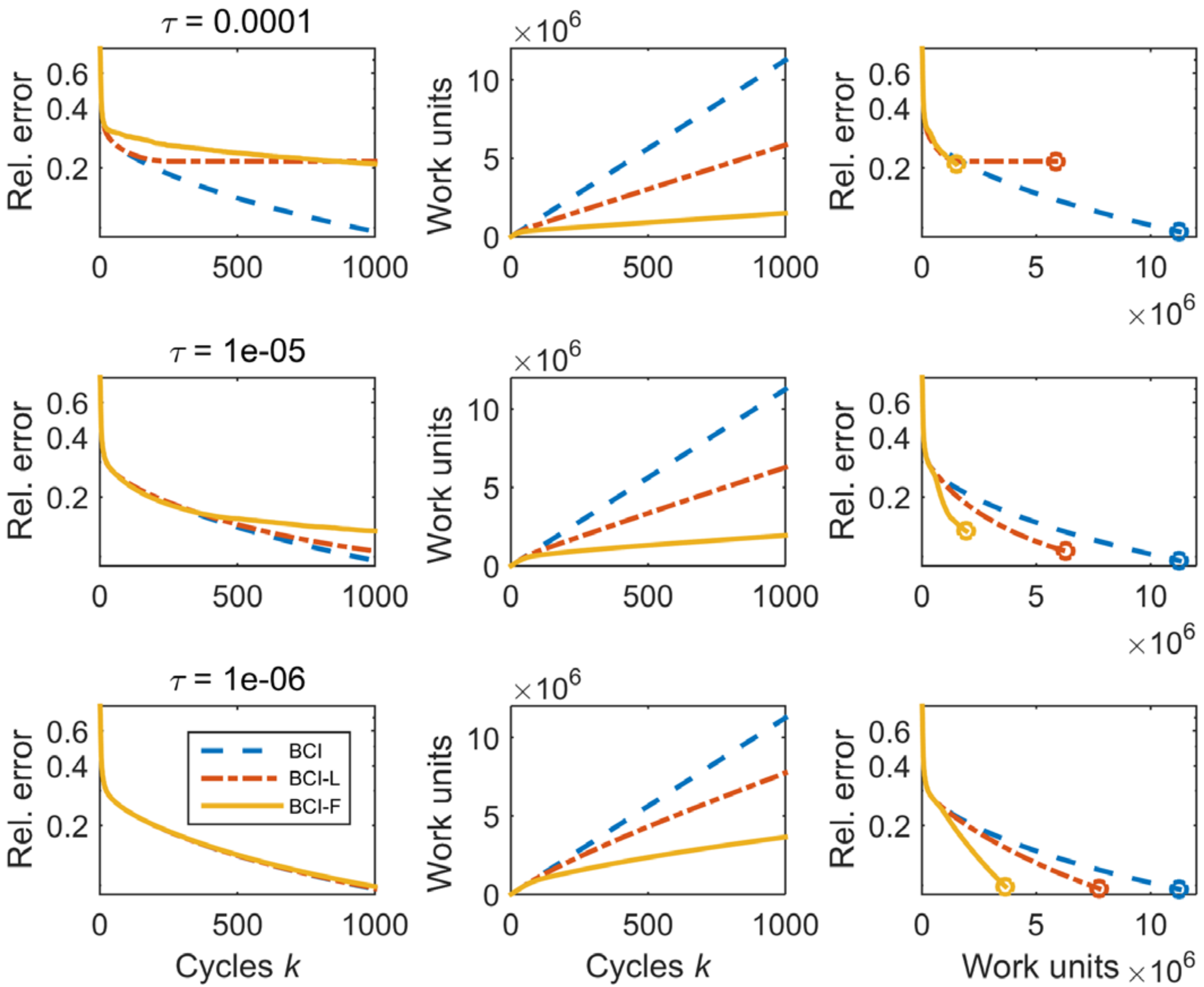
For $k = 1, 2, 3, \dots$ (cycles or outer iterations)
  For $i = 1, 2, \dots, q$ (inner iterations)
    $d_i^k = \omega_i M_i A_i^T r^{k,i}$
    If $\|d_i^k\|_2 > \tau$
      $x_i^{k+1} = x_i^k + d_i^k$
      $r^{k,i+1} = r^{k,i} - A_i(x_i^{k+1} - x_i^k)$
    End
  End
  $r^{k+1,1} = r^{k,q+1}$
End

Flagging in the Block Column Method

The situation $\|d_i^k\|_2 < \tau$ occurs when x_i has (almost) converged. Hence, we **flag** the ith block and don't update it in the next N_{flag} cycles – without computing $\|d_i^k\|_2$ thus saving more work.

For $k = 1, 2, 3, \ldots$ (cycles or outer iterations)
    For $i = 1, 2, \ldots, q$ (inner iterations)
        If block-i is not flagged
            $d_i^k = \omega_i M_i A_i^T r^{k,i}$
            If $\|d_i^k\|_2 > \tau$
                $x_i^{k+1} = x_i^k + d_i^k \qquad r^{k,i+1} = r^{k,i} - A_i(x_i^{k+1} - x_i^k)$
            Else
                Flag block-i
            End
        Else
            If block-i has been flagged for N_{flag} outer iterations
                Unflag block-i
            End
        End
    End
    $r^{k+1,1} = r^{k,q+1}$
End

Numerical Results – Loping and Flagging



BCI = blok column iteration

BCI-L = ditto with *loping*

BCI-F = ditto with *flagging*

Conclusions

- Block column-action methods are interesting alternatives to the row-action methods.
- Convergence to a least-squares solution is always guaranteed.
- Flagging can be used to save computational work, with only a minor effect on the convergence rate.
- Next step: efficient implementation!