Axial block coordinate descent (ABCD) algorithm for X-ray CT image reconstruction

Jeffrey A. Fessler and Donghwan Kim
EECS Department
University of Michigan

Fully 3D Image Reconstruction Conference
July 13, 2011

http://www.eecs.umich.edu/~fessler
Full disclosure

- Research support from GE Healthcare
- Research support to GE Global Research
- Work supported in part by NIH grant R01-HL-098686
- Research support from Intel
Goal:
Faster iterative (fully statistical) 3D CT reconstruction

Thin-slice FBP  ASIR  Statistical
Cost function

Penalized weighted least-squares (PWLS):

\[ \hat{x} = \arg \min_x \Psi(x), \quad \Psi(x) = \sum_{i=1}^{M} \frac{w_i}{2} (y_i - [Ax]_i)^2 + R(x) \]

- unknown 3D image \( x = (x_1, \ldots, x_N) \) with \( N \) voxels
- \( y = (y_1, \ldots, y_M) \) CT (log) projection data with \( M \) rays
- \( w_i \) statistical weighting for \( i \)th ray, \( i = 1, \ldots, M \)
- \( A: M \times N \) system matrix
- \( R(x) \): edge-preserving regularizer
- forward projector: \( [Ax]_i = \sum_{j=1}^{N} a_{ij} x_j \).

The principles generalize readily to other statistical models.
Traditional iterative minimization algorithms

- **Iterative coordinate descent (ICD)**
  - Sauer & Bouman, 1993; Thibault et al., 2007
  - + few iterations
  - - challenging to parallelize because sequential

- **Preconditioned conjugate gradient (PCG)**
  - + simultaneous update of all voxels using all views
  - - more iterations
  - - challenging to precondition effectively for 3D WLS
  - - challenging to precondition effectively for nonquadratic $R(x)$
  - Fessler & Booth, 1999

- **Ordered-subsets (OS) based on separable quadratic surrogates (SQS)**
  - + update all pixels simultaneously using some views
  - - regularizer gradient $\nabla R(x)$ for every block of views
  - - does not converge, worsening for large number of subsets
  - - requires many more iterations to converge than ICD
  - Deman et al., 2005

Update each voxel **sequentially** or update all voxels **simultaneously**?
Block coordinate descent / Grouped coordinate descent

- Update a block of voxels simultaneously.
- Loop over all blocks.

Long history in general optimization
Bertsekas, 1999, *Nonlinear programming*

Global convergence for strictly convex cost functions

Long history in general statistical estimation problems
Hathaway and Bezdek, 1991; Jensen, 1991

Applications to tomographic image reconstruction
Sauer *et al*., 1995; Fessler *et al*., 1995; Fessler *et al*., 1997; Benson *et al*., 2010

Choice of order important for fastest possible convergence
Yu *et al*., 2011
2D grouped coordinate descent

Fessler et al., 1997

- Spatially separated grouped of pixels (in 2D)
- Pixels within group updated simultaneously using optimization transfer
- Moderately strong coupling of pixels within slice
  \[\Rightarrow\text{undesirably high surrogate curvatures}\]
  \[\Rightarrow\text{modest acceleration compared to all-voxel SPS}\]
3D (transaxial) block coordinate descent

Benson et al., 2010

- Blocks of $k \times k$ neighboring pixels – strongly coupled
- Solved simultaneously by inverting a dense $k^2 \times k^2$ matrix
- Loop over $z$ before proceeding to next transaxial block
3D axial block coordinate descent (ABCD)

Proposed approach:
- update a block of all $N_z$ voxels along an axial line simultaneously
- loop over all $x,y$ locations sequentially
  (possibly inhomogeneously, cf. Yu et al., T-IP, 2011)
Axial block coordinate descent (ABCD) outline

for $k = 1, \ldots, K$:

$$x_k^{(n+1)} = \arg\min_{x_k \in \mathbb{R}^{N_z}} \Psi(x_1^{(n+1)}, \ldots, x_{k-1}^{(n+1)}, x_k, x_{k+1}^{(n)}, \ldots, x_K^{(n)}) .$$

end

If the regularizer is quadratic, then the ABCD update is simply:

$$x_k^{(n+1)} = x_k^{(n)} - \left[H_k^{(n)}\right]^{-1} \nabla x_k \Psi(x_1^{(n+1)}, \ldots, x_{k-1}^{(n+1)}, x_k, x_{k+1}^{(n)}, \ldots, x_K^{(n)}) \bigg|_{x_k=x_k^{(n)}} .$$

Requires inverting the $N_z \times N_z$ Hessian matrix

$$H_k^{(n)} = \nabla^2 x_k \Psi(x_1^{(n+1)}, \ldots, x_{k-1}^{(n+1)}, x_k, x_{k+1}^{(n)}, \ldots, x_K^{(n)}) \bigg|_{x_k=x_k^{(n)}}$$

$$= A_k' W A_k + \nabla^2 x_k R(x)$$

where $A_k$ is the $M \times N_z$ submatrix of $A$ with the columns that correspond to the voxels in the block being updated.

(A = [A_1, A_2, \ldots, A_K]

(For edge-preserving case we use a quadratic surrogate for the regularizer.)
Axial block coordinate descent (ABCD) properties

- \(N_z\)-times more parallelism opportunities than ICD (e.g., \(N_z = 64\) for axial study; \(N_z = 700\) for helical scan)
- Weak coupling among voxels axially \(\Rightarrow\) reasonably fast convergence
- \(N_z \times N_z\) Hessian matrix is banded; typically tri-diagonal or penta-diagonal. Invertible in \(O(N_z)\) operations, not \(O(N_z^2)\)
- Particularly well suited to separable footprint (SF) projector Long et al., 2010.
  Assumes alignment of rotation axis with detector axis (no C-arms?)
- Converges much faster than conventional optimization transfer methods based on separable quadratic surrogates [5, 16].
Typically the axial footprints of 2-3 voxels overlap on any given detector cell. Amount of overlap depends on magnification factor. The $N_z \times N_z$ Hessian matrix is banded; typically penta-diagonal. (In contrast, for transaxial blocks the Hessian is dense.)
Example for axial scan with $N_z = 64$ slices. In contrast, for any transaxial block the Hessian is dense.
3D edge-preserving regularizer couples each voxel to 26 nearest neighbors:

\[ R = \sum_{x,y,z} \sum_{j,k,l \in \{-1,1\}} \psi(f[x+j,y+k,z+l] - f[x,y,z]). \]
3D regularizer couples each voxel in an axial block to two adjacent voxels. (One in the slice above, one in the slice below.)

\[ N_z \times N_z \] Hessian of the regularizer for each axial block is \textbf{tri-diagonal}.

Inverting \( N_z \times N_z \) pentadiagonal + tri-diagonal matrix is easy. Easily fits in cache.

Alternatives

- Use separable quadratic surrogate (diagonal Hessian) for the axial block. Less work per iteration but probably more iterations.
- Use quasi-separable surrogate with tri-diagonal Hessian. Compromise between work per iteration and convergence rate?
Algorithm comparison

- ICD: “blocks” with just one voxel
- ABCD-BAND: axial blocks with banded Hessian
- ABCD-SQS: axial blocks with separable quadratic surrogate (small diagonal Hessian)
- SQS: entire 3D image is one “block” with separable quadratic surrogate (large diagonal Hessian)

Expected wall time per iteration for well-parallelized implementations:

SQS < ABCD-SQS < ABCD-BAND < ICD
Matlab simulation example

Reconstructed images after 15 iterations for a small 3D problem.
Convergence rate comparison

Cost function $\Psi(x^{(n)})$ versus iteration $n$ for four algorithms.
Summary

- ICD: small number of iterations but hard to parallelize
- ABCD: small number of iterations but more amenable to parallelization
- SQS: most amenable to parallelization but slowest convergence rate
Bibliography


