Accelerated Parallel and Distributed Iterative Coordinate Descent (ICD) for X-ray CT

Madison G. McGaffin Jeffrey A. Fessler
Dept. of Electrical Engineering and Computer Science, University of Michigan

***Abstract*—This paper describes a new distributed algorithm for accelerating model-based image reconstruction in X-ray CT using iterated coordinate descent (ICD). The key novel component is a majorizer whose Hessian involves a block-diagonal matrix with triangular blocks (BDTriB). The resulting majorize-minimize algorithm combines aspects of ICD and the distributed block-separable surrogates (DBSS) algorithm for CT reconstruction [1]. Unlike traditional ICD, the proposed algorithm is also amenable to acceleration using Nesterov's momentum [2] and the optimized gradient method (OGM) [3]. A simple preliminary experiment indicates potential for significant acceleration over traditional ICD and promising performance for distributed computing.**

I. Introduction

Model-based image reconstruction (MBIR) in X-ray CT may improve image quality over direct reconstruction methods like filtered backprojection, but long reconstruction times impede widespread clinical use. Accelerating model-based reconstructions involves improving the mathematical structure of the numerical optimization algorithms [3], [4] and exploiting both modern hardware [5], [6] and distributed computing [1], [7]. This paper describes a parallelizable version of ICD.

Consider the following penalized weighted least-squares (PWLS) image reconstruction problem [8]:

$$\widehat{\boldsymbol{x}} = \underset{\boldsymbol{x} \succeq \boldsymbol{0}}{\operatorname{argmin}} \; \Psi(\boldsymbol{x}), \quad \Psi(\boldsymbol{x}) = \mathsf{L}(\boldsymbol{A}\boldsymbol{x}) + \mathsf{R}(\boldsymbol{C}\boldsymbol{x}), \qquad (1)$$

with CT system matrix $\boldsymbol{A} \in \mathbb{R}^{M \times N}$, finite differences matrix $\boldsymbol{C} \in \mathbb{R}^{K \times N}$, and data-fit and regularizer terms L and R:

$$\mathsf{L}(\boldsymbol{p}) = \sum_{i=1}^{M} \frac{w_i}{2}(p_i - y_i)^2, \qquad \mathsf{R}(\boldsymbol{d}) = \sum_{k=1}^{K} \beta_k \psi(d_k), \qquad (2)$$

with nonnegative statistical weights $\{w_i\}$ and regularization parameters $\{\beta_k\}$, where $\{y_i\}$ denotes the measured sinogram (log) data. We assume the convex potential function ψ is smooth with bounded curvature.

Suppose that we have B compute nodes that communicate via some interconnect; *e.g.*, multiple GPUs or processors on a single computer or multiple computers connected by a network. Data communication over this interconnect typically is slower than communication and computation within each node. Accordingly, we solve (1) by finding a majorizer consisting of a sum of B components that we minimize in parallel and then communicate the results to update the image $\boldsymbol{x}$.

Supported in part by NIH grant U01 EB018753 and Intel equipment donations. Email: {mcgaffin | fessler}@umich.edu .

II. Methods

The proposed method combines the distributed block-separable surrogate (DBSS) [1] in a majorize-minimize (MM) framework [9] using a form of ICD for the inner minimizations and momentum for acceleration [2], [3].

A. Distributed block-separable surrogates

We start with a majorizer based on the distributed block-separable surrogate (DBSS) [1]. We partition the image $\boldsymbol{x}$ into B blocks, $\{\boldsymbol{x}_b\}$, *i.e.*, $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_B)$, typically by axial slabs. The corresponding components of the CT system matrix $\boldsymbol{A}$ and the finite differencing matrix $\boldsymbol{C}$ are $\{\boldsymbol{A}_b\}$ and $\{\boldsymbol{C}_b\}$, respectively, where, *e.g.*, $\boldsymbol{A} = [\boldsymbol{A}_1, \ldots, \boldsymbol{A}_B]$ and $\boldsymbol{A}_b$ is the submatrix of $\boldsymbol{A}$ having columns correspond to the pixels in $\boldsymbol{x}_b$. During each outer iteration, each computational node updates one of these sub-images, after which the sub-images are communicated between blocks.

In each outer iteration n, we form a block separable surrogate $\Phi^{(n)}$ as follows:

$$\Phi^{(n)}(\boldsymbol{x}) \triangleq \frac{1}{2}\left\|\boldsymbol{x} - \boldsymbol{x}^{(n)}\right\|_{\boldsymbol{M}}^2 + \left(\boldsymbol{x} - \boldsymbol{x}^{(n)}\right)' \nabla \Psi\left(\boldsymbol{x}^{(n)}\right) + \Psi\left(\boldsymbol{x}^{(n)}\right), \qquad (3)$$

where the iteration-invariant, block-diagonal, $N \times N$ matrix $\boldsymbol{M}$ majorizes the Hessian of the cost function, *i.e.*, $\boldsymbol{M} \succeq \nabla^2 \Psi$. This is a tangent majorizer [9], *i.e.*, it satisfies

$$\Phi^{(n)}\left(\boldsymbol{x}^{(n)}\right) = \Psi\left(\boldsymbol{x}^{(n)}\right), \quad \Phi^{(n)}(\boldsymbol{x}) \geq \Psi(\boldsymbol{x}) \qquad \forall \boldsymbol{x} \succeq \boldsymbol{0}. \qquad (4)$$

Therefore, any $\boldsymbol{x}^{(n+1)}$ that descends the surrogate $\Phi^{(n)}$ will also descend the original cost function Ψ. For the MBIR problem (1), this property ensures convergence to $\widehat{\boldsymbol{x}}$ [9].

The DBSS [1] has the block-diagonal Hessian $\boldsymbol{M}_{\text{DBSS}} \triangleq \operatorname{diag}\{\boldsymbol{M}_{\text{DBSS},b}\}$, where

$$\boldsymbol{M}_{\text{DBSS},b} \triangleq \boldsymbol{A}_b' \boldsymbol{\Lambda}_b \boldsymbol{W} \boldsymbol{A}_b + \boldsymbol{C}_b' \boldsymbol{K}_b \boldsymbol{B} \boldsymbol{C}_b,$$
$$\boldsymbol{W} = \operatorname*{diag}_i\{w_i\}, \quad \boldsymbol{B} = \operatorname*{diag}_k\left\{\beta_k \cdot \max_d \psi''(d)\right\}, \qquad (5)$$

where $\boldsymbol{\Lambda}_b$ and $\boldsymbol{K}_b$ are determined by the partition of $\boldsymbol{x}$:

$$\boldsymbol{\Lambda}_b \triangleq \operatorname*{diag}_i\left\{\frac{[\boldsymbol{A}\boldsymbol{1}]_i}{[\boldsymbol{A}_b\boldsymbol{1}]_i}\right\}, \qquad \boldsymbol{K}_b \triangleq \operatorname*{diag}_k\left\{\frac{[|\boldsymbol{C}|\boldsymbol{1}]_k}{[|\boldsymbol{C}_b|\boldsymbol{1}]_k}\right\}, \qquad (6)$$

where $|\boldsymbol{C}|$ denotes the element-wise absolute value of $\boldsymbol{C}$. This majorizer is block-separable, *i.e.*, it decomposes into B independent functions over different groups of pixels $\{\boldsymbol{x}_b\}$:

$$\Phi^{(n)}_{\text{DBSS}}(\boldsymbol{x}) = \sum_{b=1}^{B} \Phi^{(n)}_{\text{DBSS},b}(\boldsymbol{x}_b), \tag{7}$$

The DBSS algorithm descends each block surrogate $\Phi^{(n)}_{\text{DBSS},b}$ using ordered subsets with momentum acceleration (OS-MOM) [1], [3].

B. Distributed iterated coordinate descent

The DBSS algorithm does not *exactly* minimize each block-separable surrogate $\Phi^{(n)}_{\text{DBSS},b}$ because that would require too many inner iterations of OS-MOM to be practical. Although merely descending each surrogate is sufficient to ensure convergence, mere descent precludes using momentum-based techniques to accelerate the outer iterations.

To design a majorizer for which ICD can perform *exact* minimization, we propose to further majorize the block-separable matrix $\boldsymbol{M}_{\text{DBSS}}$ with a block-diagonal matrix having (lower) triangular blocks (BDTriB): $\boldsymbol{D} + \boldsymbol{T}$. We invert each of the triangular blocks of the BDTriB matrix *exactly* simply using back-substitution, *i.e.*, one sweep of ICD. By a proof similar to the one in [3], this design allows us to accelerate the outer iterations using momentum.

For derivation (but not implementation), define the $N \times N$ block diagonal matrix $\boldsymbol{T} = \text{diag}\{\boldsymbol{T}_b\}$ where each block $\boldsymbol{T}_b$ is lower-triangular and defined as follows:

$$[\boldsymbol{T}_b]_{ij} = \begin{cases} [\boldsymbol{A}_b'\boldsymbol{W}\boldsymbol{A}_b + \boldsymbol{C}_b'\boldsymbol{B}\boldsymbol{C}_b]_{ij}, & i \geq j \\ 0, & \text{else.} \end{cases} \tag{8}$$

This definition retains the block structure of $\boldsymbol{M}_{\text{DBSS}}$. We choose the diagonal matrix $\boldsymbol{D}$ such that for all $\boldsymbol{z} \in \mathbb{R}^N$, the following majorization condition holds:

$$\boldsymbol{z}'(\boldsymbol{D}+\boldsymbol{T})\boldsymbol{z} = \boldsymbol{z}'\left(\boldsymbol{D} + \frac{1}{2}(\boldsymbol{T}+\boldsymbol{T}')\right)\boldsymbol{z} \geq \boldsymbol{z}'\boldsymbol{M}_{\text{DBSS}}\boldsymbol{z}. \tag{9}$$

In particular, in each block we design diagonal $\boldsymbol{D}_b$ such that:

$$\boldsymbol{z}_b'\left(\boldsymbol{D}_b + \frac{1}{2}(\boldsymbol{T}_b+\boldsymbol{T}_b')\right)\boldsymbol{z}_b \geq \boldsymbol{z}_b'\boldsymbol{M}_{\text{DBSS},b}\boldsymbol{z}_b. \tag{10}$$

Expanding the definition of $\boldsymbol{T}$ (8), we design $\boldsymbol{D}_b$ such that:

$$\begin{aligned} \boldsymbol{D}_b &\succeq \boldsymbol{M}_{\text{DBSS},b} - \frac{1}{2}(\boldsymbol{T}_b+\boldsymbol{T}_b') \\ &= \boldsymbol{A}_b'\boldsymbol{\Lambda}_b\boldsymbol{W}\boldsymbol{A}_b + \boldsymbol{C}_b'\boldsymbol{K}_b\boldsymbol{B}\boldsymbol{C}_b \qquad (11) \\ &\quad - \frac{1}{2}(\boldsymbol{A}_b'\boldsymbol{W}\boldsymbol{A}_b + \boldsymbol{C}_b'\boldsymbol{B}\boldsymbol{C}_b + \boldsymbol{G}_b) \\ &= \boldsymbol{A}_b'\boldsymbol{W}\left(\boldsymbol{\Lambda}_b - \frac{1}{2}\boldsymbol{I}\right)\boldsymbol{A}_b + \boldsymbol{C}_b'\boldsymbol{B}\left(\boldsymbol{K}_b - \frac{1}{2}\boldsymbol{I}\right)\boldsymbol{C}_b - \frac{1}{2}\boldsymbol{G}_b, \qquad (12) \end{aligned}$$

where $\boldsymbol{G}_b$ contains the diagonal of $\boldsymbol{A}_b'\boldsymbol{W}\boldsymbol{A}_b + \boldsymbol{C}_b'\boldsymbol{B}\boldsymbol{C}_b$. The entries of $\boldsymbol{\Lambda}_b$ and $\boldsymbol{K}_b$ (6) are greater than or equal to unity if nonzero. We majorize the nondiagonal terms in (12) with the following "SQS-like" majorizer [10]:

$$\begin{aligned} \boldsymbol{D}_{\text{SQS},b} \triangleq \text{diag}\Bigg\{ &\left(\boldsymbol{A}_b'\boldsymbol{W}\left(\boldsymbol{\Lambda}_b - \frac{1}{2}\boldsymbol{I}\right)\boldsymbol{A}_b \right. \\ &\left. + |\boldsymbol{C}_b|'\boldsymbol{B}\left(\boldsymbol{K}_b - \frac{1}{2}\boldsymbol{I}\right)|\boldsymbol{C}_b|\right)\boldsymbol{1}\Bigg\}. \end{aligned} \tag{13}$$

Thus, our final diagonal component for the bth block is

$$\boldsymbol{D}_b \triangleq \boldsymbol{D}_{\text{SQS},b} - \frac{1}{2}\boldsymbol{G}_b, \tag{14}$$

which one can verify is nonnegative. Computing this majorizer requires no more time than computing the diagonal majorizer for the SQS-MOM inner step of the DBSS algorithm. In other applications, *e.g.*, phase-contrast CT, the gram matrix $\boldsymbol{A}'\boldsymbol{\Lambda}_b\boldsymbol{A}$ may have negative entries. In these cases, the SQS majorizer (13) may be very loose or difficult to compute, and one can use another technique to find a diagonal majorizer, *e.g.*, the memory-efficient algorithm in [11].

C. Minimizing the new surrogate

The new surrogate for the bth block is

$$\Phi^{(n)}_b(\boldsymbol{x}_b) = \frac{1}{2}\left\|\boldsymbol{x}_b - \boldsymbol{x}_b^{(n)}\right\|^2_{\boldsymbol{D}_b+\boldsymbol{T}_b} + \boldsymbol{x}_b'\nabla_{\boldsymbol{x}_b}\Psi\left(\boldsymbol{x}^{(n)}\right). \tag{15}$$

The matrix $\boldsymbol{D}_b + \boldsymbol{T}_b$ couples the entries of $\boldsymbol{x}_b$ together, but because it is lower triangular, we minimize $\Phi^{(n)}_b$ *exactly* using back substitution with a nonnegativity constraint. That is, we loop though each pixel x_j of $\boldsymbol{x}_b$ in a predetermined order and for each pixel solve the 1D minimization problem

$$\begin{aligned} x_j^{(n+1)} &= \underset{x_j \geq 0}{\text{argmin}} \ \frac{\omega_j}{2}\left(x_j - x_j^{(n)}\right)^2 + x_j g_j, \\ &= \max\left(x_j^{(n)} - \frac{1}{\omega_j}g_j, 0\right), \end{aligned} \tag{16}$$

where

$$\begin{aligned} \omega_j &\triangleq [\boldsymbol{D}]_{jj} + \boldsymbol{a}_j'\boldsymbol{W}\boldsymbol{a}_j + \boldsymbol{c}_j'\boldsymbol{B}\boldsymbol{c}_j, \\ g_j &\triangleq \left[\nabla \mathsf{R}\left(\boldsymbol{x}^{(n)}\right)\right]_j + \boldsymbol{a}_j'\boldsymbol{W}\left(\boldsymbol{r}^{(n)} + \boldsymbol{r}_b\right) \\ &\quad + \boldsymbol{c}_j'\boldsymbol{B}\boldsymbol{C}_b\left(\boldsymbol{x}_b^{(n+)} - \boldsymbol{x}_b^{(n)}\right), \\ \boldsymbol{r}^{(n)} &\triangleq \boldsymbol{A}\boldsymbol{x}^{(n)} - \boldsymbol{y}, \quad \boldsymbol{r}_b \triangleq \boldsymbol{A}_b\left(\boldsymbol{x}_b^{(n+)} - \boldsymbol{x}_b^{(n)}\right), \end{aligned} \tag{17}$$

and $\boldsymbol{a}_j$ and $\boldsymbol{c}_j$ denote the jth columns of $\boldsymbol{A}$ and $\boldsymbol{C}$, respectively. The vector $\boldsymbol{x}_b^{(n+)}$ contains the state of $\boldsymbol{x}_b$ after the all the pixels before the jth pixel have been updated. Updating the jth pixel involves:

- computing the column vectors $\boldsymbol{a}_j$ and $\boldsymbol{c}_j$;
- computing ω_j and g_j, which involves finite differences (for the second term of g_j) and inner products;
- solving the one-pixel update problem (16);
- and finally updating the residual buffer

$$\boldsymbol{r}_b \leftarrow \boldsymbol{r}_b + \boldsymbol{a}_j\left(x_j^{(n+1)} - x_j^{(n)}\right). \tag{18}$$

These are the same steps as ICD [12] applied to (7) with the minor addition of the "relaxation" $[\boldsymbol{D}]_{jj}$ in ω_j.

TABLE I
MOMENTUM-ACCELERATED DISTRIBUTED ICD ALGORITHMS

1) Distribute $\left\{\boldsymbol{x}_b^{(0)}\right\}$ to the B computational nodes, initialize $\boldsymbol{z}^{(0)} = \boldsymbol{x}^{(0)}$. Compute $\boldsymbol{r}^{(0)} = \boldsymbol{A}\boldsymbol{x}^{(0)} - \boldsymbol{y}$ and $\{\boldsymbol{D}_b\}$ (14). Set $t^{(0)} = 1$.
2) Loop outer iteration $n = 1, \ldots, N_{\text{iter}}$:
   a) $t^{(n+1)} = \frac{1}{2}\left(1 + \sqrt{1 + 4\left(t^{(n)}\right)^2}\right)$
   b) In parallel for $b = 1, \ldots, B$:
      i) Minimize block surrogate (15) using ICD to compute $\boldsymbol{x}_b^{(n+1)}$.
      ii) Momentum update:

No momentum : $\boldsymbol{z}_b^{(n+1)} = \boldsymbol{x}_b^{(n+1)}$,

FGM : $\boldsymbol{z}_b^{(n+1)} = \boldsymbol{x}_b^{(n+1)} + \frac{t^{(n)}-1}{t^{(n+1)}}\left(\boldsymbol{x}_b^{(n+1)} - \boldsymbol{x}_b^{(n)}\right)$

OGM : $\boldsymbol{z}_b^{(n+1)} = \boldsymbol{x}_b^{(n+1)} + \frac{t^{(n)}-1}{t^{(n+1)}}\left(\boldsymbol{x}_b^{(n+1)} - \boldsymbol{x}_b^{(n)}\right) + \frac{t^{(n)}}{t^{(n+1)}}\left(\boldsymbol{x}_b^{(n+1)} - \boldsymbol{z}_b^{(n)}\right)$

   c) Broadcast node residuals $\{\boldsymbol{r}_b\}$ to compute $\boldsymbol{r}^{(n+1)}$ and edge slices of $\boldsymbol{x}_b^{(n+1)}$.
3) Output: $\boldsymbol{z}^{(N_{\text{iter}})}$.

Between outer iterations, we synchronize the regularizer gradient $\nabla \mathsf{R}\left(\boldsymbol{x}^{(n)}\right)$ and residual $\boldsymbol{r}^{(n)} = \boldsymbol{A}\boldsymbol{x}^{(n)} - \boldsymbol{y}$ between computational nodes. The latter needs no additional projections or backprojections because

$$\boldsymbol{r}^{(n+1)} - \boldsymbol{r}^{(n)} = \sum_{b=1}^{B} \boldsymbol{r}_b, \tag{19}$$

so each node needs only to communicate residual updates $\{\boldsymbol{r}_b\}$ and edge voxels to compute $\nabla_{\boldsymbol{x}_b} \mathsf{R}\left(\boldsymbol{x}^{(n)}\right)$.

D. Momentum-based acceleration

In each outer iteration, the proposed algorithm forms and exactly minimizes a block-separable quadratic surrogate for the original cost function. This places the proposed algorithm in the same category as iterative shrinkage and thresholding (ISTA) and momentum-accelerated SQS algorithms. Thus, the distributed ICD algorithm can be improved with momentum-based acceleration *e.g.*, Nesterov's fast gradient methods (FGM) [2] or the optimized gradient method (OGM) [13]. Table I summarizes these algorithms.

Traditional ICD methods are not in general amenable to momentum-based acceleration [14]. For example, simply running ICD on the block-separable surrogate $\Phi_{\text{DBSS}}^{(n)}$ in (7) is equivalent to the proposed algorithm with each $\boldsymbol{D}_b = \boldsymbol{0}$. Although this basic approach descends the surrogate $\Phi^{(n)}$ and converges, it is incompatible with momentum-based acceleration; applying momentum-based acceleration to this basic combination of DBSS with ICD may cause divergence. The additional under-relaxation provided by $\boldsymbol{D}_b$ (14) allows us to use ICD for exact minimization of a quadratic majorizer, making the algorithm compatible with momentum-based acceleration. This property is the key contribution of this work.

III. PRELIMINARY EXPERIMENT

To illustrate the concept, we simplified the CT reconstruction problem (1) to a one-dimensional 512-"pixel" problem. We formed the system matrix $\boldsymbol{A}$ using a Toeplitz matrix with point spread function $\frac{1}{r^{1/2}}$ so the Gram matrix $\boldsymbol{A}'\boldsymbol{A}$ has response proportional to $\frac{1}{r}$ [15]. The data-fit weights were uniform $w_i = 1$, and regularizer weights β_k were also uniform. We used a quadratic regularizer potential function $\psi(d) = \frac{1}{2}d^2$. This is an extreme simplification of the CT reconstruction problem for a preliminary investigation.

A. Single-node relaxation and acceleration

We compared 6 methods for $B = 1$: conventional **ICD** and ICD with the proposed relaxation (**RICD**), with and without FGM [2] (**(R)ICD+FGM**), or OGM [13] (**(R)ICD+OGM**). In this "one-node" setting, $\boldsymbol{\Lambda} = \boldsymbol{W} = \boldsymbol{I}$ in (6) and (1).

Fig. 2(a) plots the cost function per iteration for all 6 algorithms. Clearly, applying momentum to basic ICD without additional relaxation is infeasible.

The additional relaxation from the diagonal matrices $\{\boldsymbol{D}_b\}$ (14) slows the convergence of RICD compared to ICD. However, because every loop through all the pixels in RICD corresponds to minimizing a quadratic surrogate, we can apply momentum. With momentum acceleration, RICD converges faster than regular ICD. Furthermore RICD+FGM is provably convergent.

B. Distributed accelerated ICD

We also implemented the proposed distributed ICD algorithms on a simulated network of 2, 4 and 8 nodes. As the number of nodes B in the network increases, the block-separable majorizer becomes looser; Fig. 1(d) shows this effect in the increasing entries of the diagonal majorizer $\boldsymbol{D}$.

Fig. 2(b) shows the value of the cost function *vs.* iteration for normal ICD and RICD+OGM for 1, 2, 4, and 8 nodes. The ICD algorithms use the block-separable surrogate [1], and RICD-OGM uses the proposed BDTriB majorizer. As B increases, the larger majorizer values $\{\boldsymbol{D}_b\}$ slow convergence on a per-iteration basis. However the increase of the majorizer values as the number of nodes doubles is less than a factor of 2, so there is opportunity to accelerate the algorithm with distributed computing provided the communication overhead is not too high.

We simulated the time behavior of the distributed algorithms by

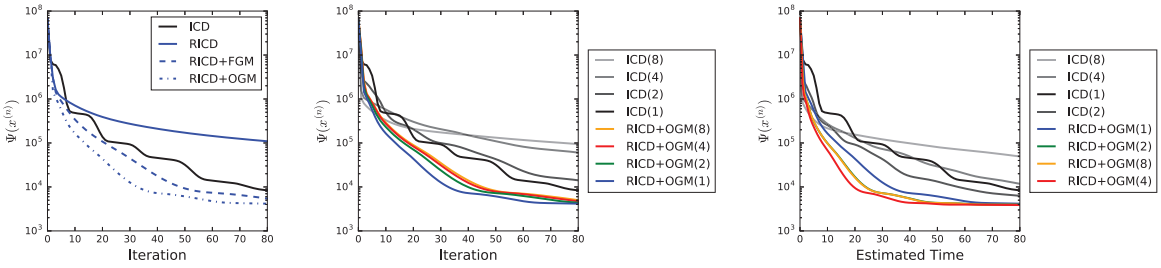
$$\Delta t_{\text{iter}} = \begin{cases} 1, & B = 1 \\ \alpha_{\text{overhead}} B + \frac{1}{B}, & B > 1. \end{cases} \tag{20}$$

Each node beyond the first ($B > 1$) adds some overhead due to communication and synchronization, α_{overhead}, but reduces how long it takes to compute the parallelizable workload, $\frac{1}{B}$. In this experiment, we assumed $\alpha_{\text{overhead}} = 0.05$. This is a pessimistic estimate; in other experiments with non-ICD algorithms, we found $\alpha_{\text{overhead}} \approx 0.01$ for multiple GPUs connected to the same computer or $\alpha_{\text{overhead}} \approx 0.03$ for computers connected by Ethernet.

Fig. 2(c) plots the value of the cost function *vs.* estimate time for the distributed (R)ICD algorithms. The distributed ICD algorithms reach peak performance at only two nodes, and is still slower than RICD on one node. The RICD algorithm

(a) Full $N \times N$ Hessian of Ψ (b) DBSS Hessian, $B = 4$ (c) $\boldsymbol{T}$ for Proposed BDTriB (d) Diagonal of Relaxation $\boldsymbol{D}$

Fig. 1. Full Hessian of Ψ, Hessian of block-separable majorizer for $B = 4$, and $\boldsymbol{T}$ and $\boldsymbol{D}$ for BDTriB majorizer for the simplified reconstruction problem in Section III.



(a) Accelerated ICD (b) Cost Ψ *vs.* Iteration for various B (c) Cost Ψ *vs.* Estimated Time

Fig. 2. Cost function curves for the experiment in Section III.

can exploit further parallelism, and convges most quickly on four nodes; the overhead from the eight-node configuration slows overall convergence. Although this preliminary experiment suggests only modest parallelization is useful, our selection for α_{overhead} is pessimistic and the time-characteristics of a real distributed system are difficult to estimate *a priori*.

IV. Summary

We explored accelerating ICD using momentum and distributed computing. With additional relaxation based on matrix majorization, we can combine ICD [3] with the momentum methods that have been effective in accelerating ordered-subsets methods. A proof-of-concept experiment suggests that this approach can accelerate ICD. The proposed method also shows promising accelerations via distributed computing, by combining momentum-based acceleration with block-separable surrogates [1]. The voxels on the boundary between blocks might converge slower than others; this effect could be mitigated by dithering the block boundaries [1].

The sequential nature of ICD algorithms seems not well matched to computing hardware that becomes increasingly parallel. Nevertheless, the general techniques for accelerating and distributing ICD described here are also relevant to accelerating dual coordinate ascent algorithms for CT, *e.g.*, [5], that are amenable to implementation on modern hardware.

References

[1] D. Kim and J. A. Fessler. Distributed block-separable ordered subsets for helical X-ray CT image reconstruction. In *Proc. Intl. Mtg. on Fully 3D Image Recon. in Rad. and Nuc. Med*, pp. 138–41, 2015.

[2] Y. Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. *Soviet Math. Dokl.*, 27(2):372–76, 1983.

[3] D. Kim, S. Ramani, and J. A. Fessler. Combining ordered subsets and momentum for accelerated X-ray CT image reconstruction. *IEEE Trans. Med. Imag.*, 34(1):167–78, January 2015.

[4] H. Nien and J. A. Fessler. Fast X-ray CT image reconstruction using a linearized augmented Lagrangian method with ordered subsets. *IEEE Trans. Med. Imag.*, 34(2):388–99, February 2015.

[5] M. McGaffin and J. A. Fessler. Alternating dual updates algorithm for X-ray CT reconstruction on the GPU. *IEEE Trans. Computational Imaging*, 1(3):186–99, September 2015.

[6] R. Sampson, M. G. McGaffin, T. F. Wenisch, and J. A. Fessler. Investigating multi-threaded SIMD for helical CT reconstruction on a CPU. In *Proc. 4th Intl. Mtg. on image formation in X-ray CT*, 2016.

[7] J. M. Rosen, J. Wu, T. F. Wenisch, and J. A. Fessler. Iterative helical CT reconstruction in the cloud for ten dollars in five minutes. In *Proc. Intl. Mtg. on Fully 3D Image Recon. in Rad. and Nuc. Med*, pp. 241–4, 2013.

[8] J-B. Thibault, K. Sauer, C. Bouman, and J. Hsieh. A three-dimensional statistical approach to improved image quality for multi-slice helical CT. *Med. Phys.*, 34(11):4526–44, November 2007.

[9] M. W. Jacobson and J. A. Fessler. An expanded theoretical treatment of iteration-dependent majorize-minimize algorithms. *IEEE Trans. Im. Proc.*, 16(10):2411–22, October 2007.

[10] H. Erdoğan and J. A. Fessler. Ordered subsets algorithms for transmission tomography. *Phys. Med. Biol.*, 44(11):2835–51, November 1999.

[11] M. G. McGaffin and J. A. Fessler. Algorithmic design of majorizers for large-scale inverse problems, 2015. arxiv 1508.02958.

[12] Z. Yu, J-B. Thibault, C. A. Bouman, K. D. Sauer, and J. Hsieh. Fast model-based X-ray CT reconstruction using spatially non-homogeneous ICD optimization. *IEEE Trans. Im. Proc.*, 20(1):161–75, January 2011.

[13] D. Kim and J. A. Fessler. Optimized first-order methods for smooth convex minimization. *Mathematical Programming*, 2016. To appear.

[14] A. Chambolle and T. Pock. A remark on accelerated block coordinate descent for computing the proximity operators of a sum of convex functions. *SMAI J. of Computational Mathematics*, 1:29–54, 2015.

[15] N. H. Clinthorne, T. S. Pan, P. C. Chiao, W. L. Rogers, and J. A. Stamos. Preconditioning methods for improved convergence rates in iterative reconstructions. *IEEE Trans. Med. Imag.*, 12(1):78–83, March 1993.