Block stochastic gradient descent for large-scale tomographic reconstruction in a parallel network

Yushan Gao, Ander Biguri, and Thomas Blumensath

Abstract—Iterative algorithms have many advantages for linear tomographic image reconstruction when compared to back-projection based methods. However, iterative methods tend to have significantly higher computational complexity. To overcome this, parallel processing schemes that can utilise several computing nodes are desirable. Popular methods here are row action methods, which update the entire image simultaneously and column action methods, which require access to all measurements at each node. In large scale tomographic reconstruction with limited storage capacity of each node, data communication overheads between nodes becomes a significant performance limiting factor. To reduce this overhead, we proposed a row action method BSGD. The method is based on the stochastic gradient descent method but it does not update the entire image at each iteration, which reduces between node communication. To further increase convergence speeds, an importance sampling strategy is proposed. We compare BSGD to other existing stochastic methods and show its effectiveness and efficiency. Other properties of BSGD are also explored, including its ability to incorporate total variation (TV) regularization and automatic parameter tuning.

Index Terms—CT image reconstruction, parallel computing, gradient descent, coordinate descent, linear inverse problems.

1 Introduction

In transmission X-ray computed tomography (CT), when using non-standard scan trajectories or when operating with high noise levels, traditional analytical reconstruction techniques such as the filtered backprojection algorithm (FBP) [1], [2] and the Feldkamp Davis Kress (FDK) [3], [4] method are no longer applicable. In these circumstances, less efficient, iterative reconstruction methods can provide significantly better reconstructions [5], [6], [7], [8]. These methods model the x-ray system as a linear system:

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{e}, \tag{1}$$

where $\mathbf{y} = [y_1, \cdots, y_r]^T, \mathbf{x} = [x_1, \cdots, x_c]^T$ and $\mathbf{e} = [e_1, \cdots, e_r]^T$ are x-ray projection data, the unknown vectorised image and measurement noise respectively. The system matrix $\mathbf{A} \in \mathbb{R}^{r*c}$ has non-negative elements, which can be computed using Siddon's method [9]. Image reconstruction can then be cast as an optimisation problem [10], [11], [12]:

$$\min_{\mathbf{x}} f(\mathbf{x}) = \min_{\mathbf{x}} \frac{1}{2}\|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2, \tag{2}$$

In many applications, such as industrial CT scanning, the system matrix $\mathbf{A}$ can be enormous [13]. Iterative methods apply matrices $\mathbf{A}$ and $\mathbf{A}^T$ to compute "forward projection"(FP) and "back projection"(BP) respectively, to iteratively find an approximate solution to minimize Eq.2. Note that in realistic applications, due to its size, the matrix $\mathbf{A}$ is never stored [14], FP and BP are instead computed 'on the fly' using Graphical Processor Units (GPUs).

For our discussion, we classify iterative methods into column action methods and row action methods. Column action methods include iterative coordinate descent (ICD) [15], [16] and axial block coordinate descent (ABCD) [17], [18]. They divide $\mathbf{x}$ into several blocks and update individual blocks in each iteration using the most recent estimates of all other blocks. Row action methods include Kaczmarz methods(ART) [19], simultaneous iterative reconstruction technique(SIRT) [20], and component averaging (CAV) [21] and their ordered set variations [22], [23]. Unlike column action methods, row action methods divide the projection data $\mathbf{y}$ into several blocks and update all of $\mathbf{x}$ simultaneously using one or several blocks of $\mathbf{y}$. Despite the superior reconstructions achievable with iterative methods in many applications, the high computational complexity remains a significant bottleneck limiting their application in realistic settings. To overcome these issues, parallization is desirable. For example, the ICD algorithm can be run on multiple CPUs [24] or on several graphics processing units(GPUs) [25], [26]. Parallization of row action methods is straightforward: each node receives a copy of $\mathbf{x}$ and different blocks of $\mathbf{y}$. Each node independently updates $\mathbf{x}$ and message passing between nodes computes weighted sums of partial results [27], [28]. Compared with column action methods, row action methods is more amenable to parallel processing in a multiple-node network because that different nodes do not have to update the same elements in the error vector $\mathbf{y} - \mathbf{A}\mathbf{x}$ [29].

A range of row and column action methods have been specifically designed for tomographic reconstruction. Recently, advances in machine learning have also led to significant advances in stochastic optimization and many of these ideas are also applicable to tomographic reconstruction. Most methods here are row action methods. These include stochastic gradient descent [30], stochastic variance reduced gradient(SVRG) [31], incremental aggregated gra-

- *Y. Gao, A. Biguri and T. Blumensath are with the Faculty of Engineering and Environment, University of Southampton, Southampton, UK, SO17 1BJ.*
E-mail: y.gao, a.biguri, Thomas.Blumensath@soton.ac.uk

Manuscript received March 27, 2019.

dient (IAG) [32] and stochastic average gradient (SAG) [33]. These stochastic algorithms have often been parallelized to operate on large data sets [34], [35], [36], [37].

2 The BSGD algorithm

In this paper, we develop a parallel row action algorithm specifically for large scale tomographic reconstruction. Whilst previous work in parallel tomographic reconstruction has concentrated on standard tomography, where an object is rotated around a single axis, we are here particularly interested in a setting that allows more general trajectories such as those found in laminographic scanning [38]. With "large scale" we here mean that both $\mathbf{y}$ and $\mathbf{x}$ are too large to be stored within one computing node. We thus divide $\mathbf{y}$, $\mathbf{A}$ and $\mathbf{x}$ into several blocks and design algorithms in which each node only has partial access to both $\mathbf{y}$ and $\mathbf{x}$ at each iteration. Let $\mathbf{A}$ be divided into M row blocks and N column blocks (possibly after row and column permutation). Let $\{I_i\}_{i=1}^M$ be a set of row indices and $\{J_j\}_{j=1}^N$ a set of column indices. $\mathbf{A}_I^J$ thus is a sub-matrix indexed by $I \in \{I_i\}_{i=1}^M$ and $J \in \{J_j\}_{j=1}^N$. The forward X-ray projection process can then be approximated as:

$$\begin{bmatrix} \mathbf{y}_{I_1} \\ \vdots \\ \mathbf{y}_{I_M} \end{bmatrix} \approx \begin{bmatrix} \mathbf{A}_{I_1}^{J_1} & \cdots & \mathbf{A}_{I_1}^{J_N} \\ \vdots & \vdots & \vdots \\ \mathbf{A}_{I_M}^{J_1} & \cdots & \mathbf{A}_{I_M}^{J_N} \end{bmatrix} \begin{bmatrix} \mathbf{x}_{J_1} \\ \vdots \\ \mathbf{x}_{J_N} \end{bmatrix} \equiv \begin{bmatrix} \mathbf{A}_{I_1} \\ \vdots \\ \mathbf{A}_{I_M} \end{bmatrix} \mathbf{x}. \quad (3)$$

With this partitioning, both column and row action methods can be inefficient in terms of communication between nodes since they all require full access to either all of $\mathbf{y}$ or all of $\mathbf{x}$. For example, in row action methods, the most time consuming operations are the FP and BP [39]. These projections are parallelizable [40]. For the FP $\mathbf{A}_I\mathbf{x} \equiv \sum_{j=1}^N \mathbf{A}_I^{J_j}\mathbf{x}_{J_j}$, each parallel node calculates a forward projection $\mathbf{A}_I^{J_j}\mathbf{x}_{J_j}$. The summation over j is then calculated at a master node or using an ALLREDUCE procedure [41]. A similar parallel scheme is also applicable to the BP. If the number of computing nodes is smaller than the number of column or row blocks N or M, then the parallel calculations require several communications between data storage and computation nodes, which can be time consuming [7], [27].

To reduce the communication overhead and the algorithm's dependency on access to all of $\mathbf{y}$ or $\mathbf{x}$, we previously proposed a parallel algorithm called Coordinate-Reduced Steepest Gradient Descent (CSGD) [42] to solve the block linear model Eq.3. However, our previous method only converged to a weighted least squares solution. Here we propose an improved algorithm, which we call 'Block Stochastic Gradient Descent" (BSGD). We empirically show that BSGD converges closer to the least squares solution than CSGD. The new method is similar to SAG and accumulates previously calculated direction to obtain the current update direction, but it differs from SAG in that we also incorporate a coordinate descent strategy. In the origin SAG algorithm, the gradient summands must be calculated by accessing all of $\mathbf{x}$ while in BSGD only part of $\mathbf{x}$ is required and updated. Furthermore, we exploit the sparsity of the system matrix $\mathbf{A}$ found in CT imaging and proposed an "importance sampling" strategy for BSGD (BSGD-IM). An automatic parameter tuning strategy is also adopted to tune the step length. Simulation results show that the convergence speed of BSGD-IM is faster than other row action methods such as SAG and SVRG, making BSGD an ideal candidate for distributed tomographic reconstruction.

We derive BSGD for large scale CT reconstruction where a parallel multiple-GPU network is available. The network uses a master-servant architecture and the servants/nodes(i.e. GPUs) in the network have limited access to both projection data $\mathbf{y}$ and reconstructed volume $\mathbf{x}$. To facilitate latter discussions, we define $\mathbf{r} = \mathbf{y} - \mathbf{A}\mathbf{x}$ and let $\mathbf{r}_I$ be the subset of $\mathbf{r}$ representing $\mathbf{y}_I - \mathbf{A}_I\mathbf{x}$.

To motivate our approach, let us consider iterative mini-batch stochastic gradient descent [43] with a decreasing step length μ. At the k^{th} iteration, this method computes:

$$\begin{aligned} \mathbf{r}_I &= \mathbf{y}_I - \mathbf{A}_I\mathbf{x} \\ \mathbf{g} &= 2\mathbf{A}_I^T\mathbf{r}_I \\ \mathbf{x}^{k+1} &= \mathbf{x}^k + \mu\mathbf{g}. \end{aligned} \quad (4)$$

In our setting where a master or storage node assigns data to servant nodes for processing, if local memory at a servant node is restricted, then each node can only process a partial block $\mathbf{x}_J$ and calculate $\mathbf{A}_I^J\mathbf{x}_J$. To compute $\mathbf{A}_I\mathbf{x}$, we would thus need to repeatedly sent different blocks to the servant nodes before the result is combined in the master node to update $\mathbf{r}_I$. This process is shown in Fig.1. Computation of

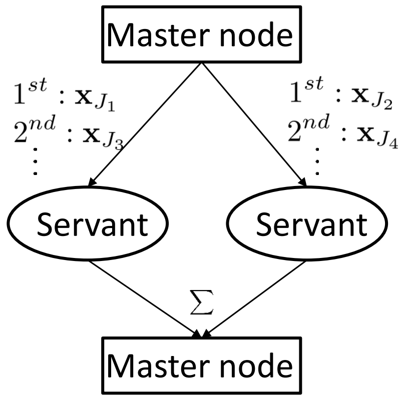

Master node
1^{st} : $\mathbf{x}_{J_1}$
2^{nd} : $\mathbf{x}_{J_3}$
1^{st} : $\mathbf{x}_{J_2}$
2^{nd} : $\mathbf{x}_{J_4}$
Servant
Servant
Σ
Master node

Fig. 1. In a master-servant network, in each iteration, each servant node receives one block $\mathbf{x}_J$ to calculate $\mathbf{A}_I^J\mathbf{x}_J$. The results are accumulated in the master node to update the corresponding residual $\mathbf{r}_I$. If the number of the servant nodes is less than the number of column blocks N, then we require repeated communication between servants and the master nodes.

$\mathbf{A}_I^T\mathbf{r}_I$ would need to follow a similar strategy. Instead of updating $\mathbf{x}$ only once the exact residual $\mathbf{r}$ has been computed, our innovation is to compute a stochastic approximation of the residual by only processing a subset of $\mathbf{x}$ in each iteration and using previously computed estimates of $\mathbf{A}_I^J\mathbf{x}_J$ for the blocks not used in this step. The hope is that the increase in uncertainty in the gradient estimate is compensated for by a reduction in computation and communication cost. BSGD thus does not compute all $\mathbf{A}_I^T\mathbf{r}_I$ and $\mathbf{A}_I\mathbf{x}_I$ in each iteration. For a fixed number of servant nodes, let α and γ be the fraction of row and column blocks that can be used in parallel computations at any one time. During each iteration, we thus propose to only use $\alpha\gamma MN$ sub-matrices ($\mathbf{A}_I^J$)

together with the corresponding data ($\mathbf{y}_I$) and volume($\mathbf{x}_J$) sub-vectors to compute updates. To gradually reduce the error between the stochastic gradient and the true gradient, BSGD adopts a gradient aggregation strategy that is similar to that of SAG. As we show below for our CT reconstruction problem, this accumulation strategy enables the algorithm to converge with a constant step length μ.

The BSGD algorithm is shown in Algo.1. When $\gamma = 1$, the method becomes SAG. In this paper we mainly focus on α and $\gamma < 1$, which allows us to use a reduced number of servant nodes.

Algorithm 1 BSGD

```
1: Initial: $\mathbf{g} = \mathbf{0}, \{\hat{\mathbf{g}}^i\}_{i=1}^M = \{\mathbf{0}\}, \{\mathbf{z}^j\}_{j=1}^N = \{\mathbf{0}\}, \mathbf{r} = \mathbf{y}$,
   $\mu =$ const. $\mathbf{x}_{est} = \mathbf{0}$.
2: for epoch =1,2,..., Max Iteration do
3:     randomly select $\alpha M$ row blocks from $\{I_i\}_{i=1}^M$ and $\gamma N$
       column blocks from $\{J_j\}_{j=1}^N$
4:     for the selected $I_i$ and $J_j$ in parallel do
5:         $\mathbf{z}_{I_i}^j = \mathbf{A}_{I_i}^{J_j} \mathbf{x}_{est\,J_j}$
6:     end for
7:     $\mathbf{r} = \mathbf{y} - \sum_{j=1}^N \mathbf{z}^j$
8:     for the selected $I_i$ and $J_j$ in parallel do
9:         $\hat{\mathbf{g}}_{J_j}^i = 2(\mathbf{A}_{I_i}^{J_j})^T \mathbf{r}_{I_i}$
10:    end for
11:    $\mathbf{g} = \sum_{i=1}^M \hat{\mathbf{g}}^i$
12:    for the selected $J_j$ in parallel do
13:        $\mathbf{x}_{est\,J_j} = \mathbf{x}_{est\,J_j} + \mu \mathbf{g}_{J_j}$
14:    end for
15: end for
```

2.1 Improving BSGD performance

There are two tricks to improve BSGD performance. The first trick is the use of an importance sampling strategy which we call "BSGD-IM". In tomographic reconstruction, the matrix $\mathbf{A}$ is often sparse with different blocks $\mathbf{A}_I^J$ often varying widely in their sparsity. This sparsity can be enhanced further for 3D tomographic problems if we partition $\mathbf{y}$ such that each partition is made up of a selection of sub-blocks, where each sub-block corresponds to a partition of the X-ray detector at one projection angle as shown in Fig. 2. Inspired

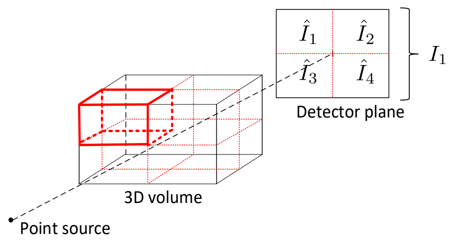


Fig. 2. Example cone beam CT setting. The 3D volume is divided into 8 sub volumes. In the cone beam scanning geometry, the projection of one sub-block is mainly concentrated in a small area on the detector. If the detector is also divided into 4 sub-areas, then the currently selected sub-volume (bold red frame) is mainly projected onto the top-left and top-right sub-detector area, i.e. $\hat{I}_1$ and $\hat{I}_2$ area.

by this property, BSGD-IM breaks each single projection into several sub-projections (4 sub-projections in Fig.2) and only samples 1 sub-projections for the selected sub-volume $\mathbf{x}_J$. This sampling is not done uniformly but is based on a selection criteria that uses the relative sparsity of each matrix $\{\mathbf{A}_{\hat{I}_i}^J\}$, i.e. the denser a sub-matrix is, the higher the probability that it is selected. As we do not have access to matrix $\mathbf{A}$, to estimate the sparsity pattern of $\mathbf{A}$, BSGD-IM computes the fraction of a volume block's projection area on each sub-detector. When the row block I_i contains several projection angles, the importance sampling strategy is repeatedly applied to each projection angle contained in the row block. The advantage of BSGD-IM is that it provides each computation node more projection angles within one row block than BSGD when a nodes' storage capacity is limited and thus reduces the row block number M as well as the total storage requirement. For example, if we assume that one GPU can only process a single projection in the original BSGD, then BSGD-IM with the partition of Fig.2 can use four projection angles by choosing one detector sub-areas for each projection angle.

BSGD-IM is shown in Algo.2. Whilst importance sampling speeds up initial convergence, it also introduces a bias is the stochastic gradient due to the inhomogeneous sampling. To overcome this, it is suggested that after initial fast convergence, the last few iterations should be run without importance sampling.

Algorithm 2 BSGD-IM

```
1: Initial: $\mathbf{g} = \mathbf{0}, \{\hat{\mathbf{g}}^i\}_{i=1}^M = \{\mathbf{0}\}, \{\mathbf{z}^j\}_{j=1}^N = \{\mathbf{0}\}, \mathbf{r} = \mathbf{y}$,
   $\mu =$ const. $\mathbf{x}_{est} = \mathbf{0}$.
2: for epoch =1,2,..., Max Iteration do
3:     randomly select $\alpha M$ row blocks from $\{I_i\}_{i=1}^M$ and $\gamma N$
       column blocks from $\{J_j\}_{j=1}^N$.
4:     for the selected $I_i$ and $J_j$ in parallel do
5:         For each column block $\mathbf{x}_{J_j}$, importance sample
           one sub-detector area from each single projection
           view. All indexes represented by those selected sub-
           detector areas form the row index set $\hat{I}_t$.
6:         $\mathbf{z}_{\hat{I}_t}^j = \mathbf{A}_{\hat{I}_t}^{J_j} \mathbf{x}_{est\,J_j}$
7:     end for
8:     $\mathbf{r} = \mathbf{y} - \sum_{j=1}^N \mathbf{z}^j$
9:     for the selected $I_i$ and $J_j$ in parallel do
10:        $\hat{\mathbf{g}}_{J_j}^i = 2(\mathbf{A}_{\hat{I}_t}^{J_j})^T \mathbf{r}_{\hat{I}_t}$
11:    end for
12:    $\mathbf{g} = \sum_{i=1}^M \hat{\mathbf{g}}^i$
13:    for the selected $J_j$ in parallel do
14:        $\mathbf{x}_{est\,J_j} = \mathbf{x}_{est\,J_j} + \mu \mathbf{g}_{J_j}$
15:    end for
16: end for
```

The second trick is to use automatic parameter tuning. Broadly speaking, up to a limit, increasing μ increases convergence speed. However, in practice, it is difficult to determine the upper limit. As a result, in realistic large scale tomographic reconstruction, instead of using a fixed step-length μ, we developed an automatic parameter tuning approach. Parameter tuning is not a new concept in machine learning and optimization. For example, the hypergradient descent [44] or the Barzilai-Borwein (BB) method [45] can be used for SGD or SVRG. However these methods are not directly applicable to BSGD, as they require updates to all of $\mathbf{x}$ in each iteration. Furthermore, BSGD uses dummy variables $\mathbf{z}$ to stores information about previous $\mathbf{x}$. Due to this, the stochastic gradient $\mathbf{g}$ of BSGD is much noisier than the estimate obtained by traditional stochastic gradient methods. We here proposed an automatic parameter tuning methods that is different from the BB method or hyper-gradient descent method. It only exploits the parameters generated during the iteration process: the residual $\mathbf{r}$ and iteration direction $\mathbf{g}$, as shown in Algo.3.

Algorithm 3 Automatic μ tunning strategy

```
1: $\epsilon$ and $\delta$ are positive constants.
2: At each $k^{th}$iteration where $mod(k, M) == 0$, sum up all
   $\mathbf{g}$ in the past $M$ epochs as an effective update direction
   (EUD) $\overline{\mathbf{g}}^{k/M}$.
3: calculate the inner product between two consecutive $\overline{\mathbf{g}}$
   as $\theta^{k/M} = \frac{(\overline{\mathbf{g}}^{k/M})^T \overline{\mathbf{g}}^{k/M-1}}{\|\overline{\mathbf{g}}^{k/M}\| \|\overline{\mathbf{g}}^{k/M-1}\|}$.
4: if mod($k$,$M$)==0&$k > M$ then
5:     if $\|\mathbf{r}\|^k < \|\mathbf{r}\|^{k-M} < \|\mathbf{r}\|^{k-2M}$ then
6:         $\mu = (1+\epsilon) * \mu$
7:     end if
8:     if $\|\mathbf{r}\|^k > \|\mathbf{r}\|^{k-M} > \|\mathbf{r}\|^{k-2M}$ then
9:         if $| \theta^{k/M} - \theta^{k/M-1} |> t_1$ or $\theta^{k/M} < t_2$ then
10:            $\mu = (1-\delta)\mu$
11:        end if
12:    end if
13: end if
```

This automatic parameter tuning is applied after M iterations. It tests whether $\mathbf{r}$ decreased during the past $2M$ epochs, in which case μ is increased by $1+\epsilon$. To reduce μ, using a similar condition on $\mathbf{r}$ alone (i.e. line 8 in Algo.3, named as "criteria 1") was not found to be sufficient to ensure convergence. We thus use an additional criteria (line 9 in Algo.3, named as "criteria 2"). The criteria 2 is motivated by the general parameter tuning methods that computes inner-products between adjacent gradients and determines to increase or decrease μ according to the positivity of the inner-product [44], [46]. We thus compare the inner-products of two gradients. To do this, we accumulate several stochastic gradients during a period of several iterations (M epochs in Algo.3) to compute an effective update direction (EUD) $\overline{\mathbf{g}}$ to reduce the stochastic error variance. We have observed that when BSGD converges with a properly chosen fixed μ, then the change of two adjacent EUDs do not vary significantly. On the contrary, these two directions vary significantly when BSGD suffers from oscillatory behaviour or an increase in the norm of $\mathbf{r}$. This is due to our method using some old values $\mathbf{z}$ in the calculation of each update. If the change in $\mathbf{x}$ is not too large, then these old values for $\mathbf{z}$ are good approximations to the current values. As a result, the two EUDs, should also be similar to each other. If we assume that during M epochs it is likely that all of $\mathbf{x}$ (and thus all of the $\mathbf{z}$'s) have been updated, then two EUDs can be computed from the previous $2M$ epochs. The inner product of the two normalized EUDs should always be close to 1. If not, it means that the step length is too big and the iteration is likely to diverge.

For both increasing and decreasing μ part, the frequency of parameter changing is M epochs rather than 1 epoch. One reason is that the high stochastic noise effect in the gradient update can be reduced after a period of epochs. Another reason is that the calculation of $\|\mathbf{r}\|$ can be time consuming when the size of $\mathbf{r}$ is large, reducing the frequency of computation on $\|\mathbf{r}\|$ is beneficial to save the reconstruction time. We have experimentally validated that setting the test frequency to M leads to a good compromise between increased computational demand and improved overall convergence speed.

2.2 Incorporating TV regularization into BSGD

When we have few projections, a TV regularization term is often used to increase the reconstruction quality. Reconstruction with a TV regularization term often minimizes a quadratic objective function plus a non-smooth TV-regularization term:

$$\underbrace{(\mathbf{y}-\mathbf{A}\mathbf{x})^T(\mathbf{y}-\mathbf{A}\mathbf{x})}_{f(\mathbf{x})} + \underbrace{2\lambda \text{TV}(\mathbf{x})}_{g(\mathbf{x})}, \tag{5}$$

where λ is a relaxation parameter and TV($\mathbf{x}$) is the total variation (TV) of $\mathbf{x}$. For 2D images, the total variation penalty can be defined as:

$$\text{TV}(\mathbf{x}) = \sum_{c,d} \sqrt{(x_{c,d}-x_{c-1,d})^2 + (x_{c,d}-x_{c,d-1})^2}, \tag{6}$$

where $x_{c,d}$ is the intensity of image pixel in row c and column d.

Traditional methods, including ISTA [47] and FISTA [48], minimize the TV regularized objective function with two steps: Each iteration starts with using the gradient of $f(\mathbf{x})$ to reduce the data fidelity, i.e. to reduce $f(\mathbf{x})$, followed by a TV-based de-noising procedure. We empirically show that BSGD is able to replace the gradient descent (GD) step. Since BSGD updates only some components of $\mathbf{x}$ with partial projection data in each iteration, the TV-based de-noising procedure is only performed after a period of time, enabling the computation load of BSGD is scalable to that in ISTA or FISTA. The algorithm is shown in Algo.4.

3 Results

We start the evaluation of our method using a 2D scanning setup (sections 3.1 to 3.4) to explore convergence properties of BSGD. In the final section (3.5), we then look at a more representative 3D cone beam setting.

3.1 BSGD convergence

We here use the scanning geometry as shown in Fig.3. In our first experiments we set K to 16, OP and OD is 50, the detector has 30 elements and the angular interval is 10° so that $\mathbf{A} \in \mathbb{R}^{1080*256}$. This model is used from section 3.1 to section 3.3.

We first examine if BSGD (without additional regularisation) converges to the least square solution of the linear model. We here set $M = 4$ and $N = 2$. α and γ are initially set to 1. We add Gaussian noise to the data so that the Signal to Noise ratio (SNR) of $\mathbf{y}$ is 17.5 dB. The distance to the least square solution (DS) is defined as

$$DS = \|\mathbf{x}_{rec} - \mathbf{x}_{lsq}\|, \tag{7}$$

Algorithm 4 BSGD-TV

```
1: Initial: $\mathbf{g} = \mathbf{0}, \{\hat{\mathbf{g}}^i\}_{i=1}^M = \{\mathbf{0}\}, \{\mathbf{z}^j\}_{j=1}^N = \{\mathbf{0}\}, \mathbf{r} = \mathbf{y}$, $\mu =$ const. $\mathbf{x}_{est} = \mathbf{0}$.
2: for epoch =1,2,..., Max Iteration do
3:     randomly select $\alpha M$ row blocks from $\{I_i\}_{i=1}^M$ and $\gamma N$ column blocks from $\{J_j\}_{j=1}^N$
4:     for the selected $I_i$ and $J_j$ in parallel do
5:         $\mathbf{z}_{I_i}^j = \mathbf{A}_{I_i}^{J_j} \mathbf{x}_{est\,J_j}$
6:     end for
7:     $\mathbf{r} = \mathbf{y} - \sum_{j=1}^N \mathbf{z}^j$
8:     for the selected $I_i$ and $J_j$ in parallel do
9:         $\hat{\mathbf{g}}_{J_j}^i = 2(\mathbf{A}_{I_i}^{J_j})^T \mathbf{r}_{I_i}$
10:    end for
11:    $\mathbf{g} = \sum_{i=1}^M \hat{\mathbf{g}}^i$
12:    for the selected $J_j$ in parallel do
13:        $\mathbf{x}_{est\,J_j} = \mathbf{x}_{est\,J_j} + \mu \mathbf{g}_{J_j}$
14:    end for
15:    if $mod(k, \frac{1}{\alpha\gamma}) == 0$ then
16:        $\mathbf{x}_{est} = \arg\min_{\mathbf{t}} \|\mathbf{t} - \mathbf{x}_{est}\|^2 + 2\mu\lambda \mathrm{TV}(\mathbf{t})$
17:    end if
18: end for
```

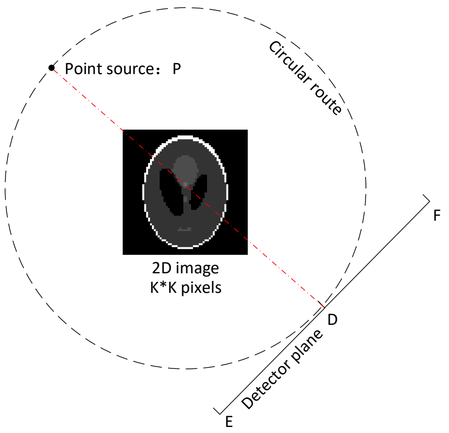


Fig. 3. A standard 2D scanning geometry with a Shepp-Logan phantom, where P is the x-ray source, O is the centre of the object and the rotation centre. D is the centre of the detector. Source and detector rotate around the centre and take measurements at different angles. The linear detector is evenly divided into to sub-areas DE and DF, which will be used in importance sampling discussed later. In this paper, unless particularly mentioned, the size of the image pixels (or voxels in 3D) and the detector pixel size are both 1.

where $\mathbf{x}_{rec}$ is the reconstructed image vector and $\mathbf{x}_{lsq}$ is the least square solution obtained here using the LSQR method.

We compared BSGD with other mature methods including SIRT and CAV as well as with our previous algorithm CSGD. The results are shown in Fig.4, where we see the linear convergence of our method to the least squares solution. All parameters in the methods were well tuned to ensure the fastest convergence rate. We see that BSGD not only approaches the least square solution, it also shows a faster convergence rate compared to the other methods.

The initial simulations used $\alpha = \gamma = 1$. We thus next study the more realistic setting where α and γ are smaller than 1.The results, shown in Fig.5, show that even in this scenario, BSGD still approaches the least square solution. We here divided $\mathbf{A}$ into 64 blocks, using different partitions

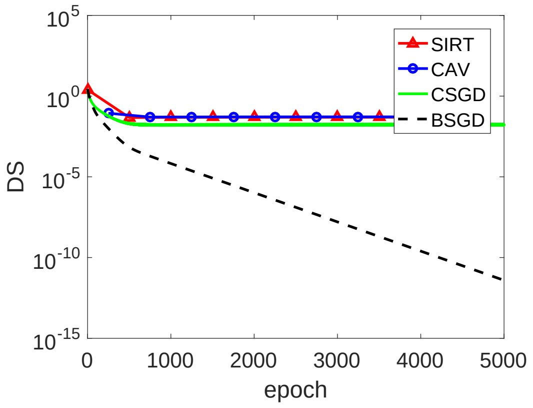


Fig. 4. In contrast to BSGD, SIRT, CAV and CSGD do not achieve the least square solution.

that varied in the numbers of row and column blocks (2 row blocks and 31 columns blocks, 4 row blocks and 16 column blocks and 8 row blocks and 8 column blocks). We set α and γ to $\frac{1}{M}$ and $\frac{1}{N}$ respectively. To measure convergence speed and communication costs between master node and computation node in a realistic parallel network, we plot the DS as a function of the number of multiplications of a vector by $\mathbf{A}_I^J$(or $(\mathbf{A}_I^J)^T$), which is proportional to the number of forward/backwards projections as well as the corresponding communication time. We see from Fig.5 that BSGD performs better in terms of reaching the least squares solution. We also see differences in the convergence speed depending on the way in which we partition the matrix.

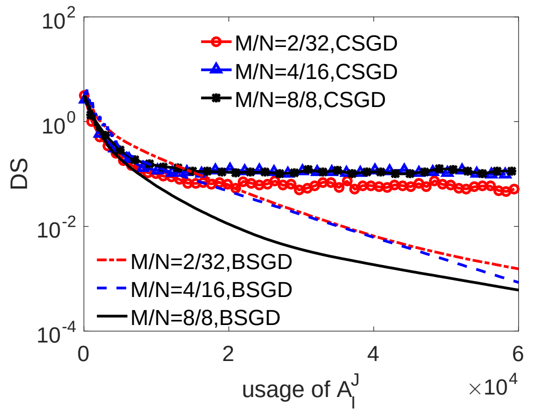
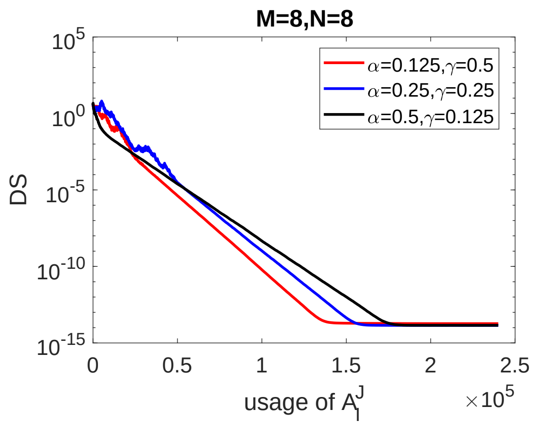


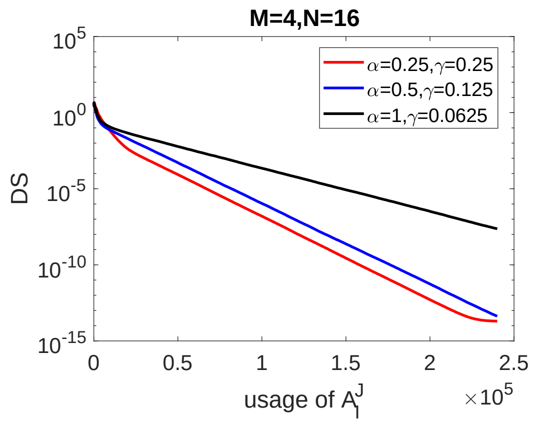
Fig. 5. BSGD shows better converge compared to CSGD in terms of achieving the least squares solution. Different ways to partition the rows and columns lead to different convergence speeds.

3.2 Setting α, γ, M and N

We nexrt study the influence of α and γ for a fixed partition of $\mathbf{A}$. In Fig.5 α and γ were set to an extreme value where only one node is used. In realistic applications, several nodes might be available. Assume we have 4 nodes so that $MN\alpha\gamma = 1$. Simulations, shown in Fig.6, show that reducing γ slows down convergence.



(a)



(b)

Fig. 6. When M and N are fixed, reducing γ slows down the convergence speed.

Based on these and similar results, we suggest to use the following selection criteria for α,γ

$$\begin{cases} \gamma = \min\{1, \frac{NodeNum}{N}\}, \\ \alpha = \frac{NodeNum}{MN\gamma}, \end{cases} \tag{8}$$

where the $NodeNum$ is the number of separate computation nodes.

According to Eq.8, we divide $\mathbf{A}$ into different numbers of row and column blocks and compare convergence speed. The results shown in Fig.7 indicate that BSGD does not encourage the partition in the column direction. Whilst this indicates that we do not want to partition in the column direction, different partitions lead to different storage demand. This is shown in Fig.8, where we show the amount of storage that is required in the compute nodes and the master node.

Note that in the previous simulations, we kept the product MN fixed as in this case, computation speed per iteration remains constant. However, when storage demand is the limiting factor, then the $m+n$ become limited (m, n are rows and columns of $\mathbf{A}_I^J$). To explore this, we fix $m+n \leq 140$. The results, shown in Fig.9 for different values of M and N are now plotted against the number of times we compute multiplications by matrices $\mathbf{A}_I^J$, multiplied by mn to normalise computation time differences. We here assume that there are only two nodes in the parallel network and the

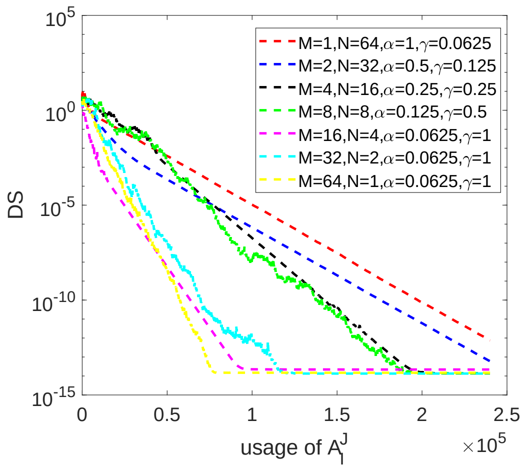


Fig. 7. BSGD does not encourage the partition in the column direction. When N increases, convergence speed decreases.

selection criteria for α and γ follows Eq.8. The convergence speed and storage demand in the master node is shown in Fig.9. Note that when $M = 135, N = 2$ (green line), BSGD becomes SAG. For large N, storage demand in the master node increases significantly. Luckily, convergence is the fastest for moderate values of N.

3.3 Automatic parameter tuning

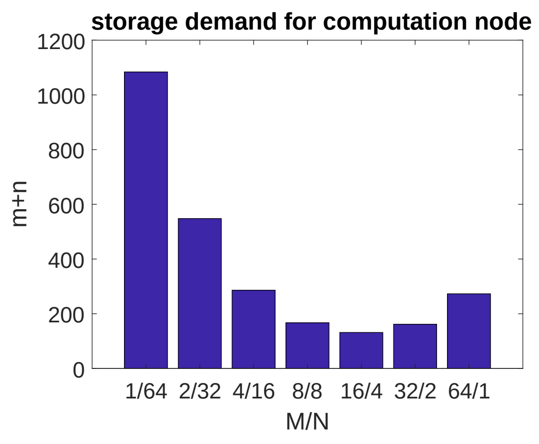
We first demonstrate that criteria 1 (line 8 in Algo 3) on its own is not sufficient for parameter tuning. If δ is small, μ is not effectively reduced, whilst for large δ, the μ tends to become too small and the iterations get 'stuck'. Simulation results to prove this are shown in Fig.10.

Automatic parameter tuning works if we combine criteria 1 with criteria 2 (line 9 in Algo 3). We here use $\delta = 0.4$. The compute nodes is still set as 2. The results, shown in Fig.11, demonstrate that automatic tuning (dashed lines) allows faster convergence than the original method (dashed line) and the method using criteria 1 only (solid line).

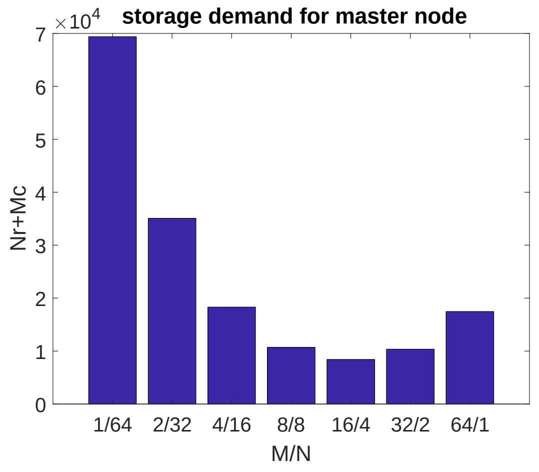
3.4 Incorporating the TV constraint

To explore Total Variation regularisation, in Fig.3, the K is increased to 64. OP and OD is 100 and the detector has 180 elements. The scanning angle increment is $1°$ and the point source only rotates through $180°$. In this section, the SNR of $\mathbf{x}$ is defined as $20\log_{10}\frac{\|\mathbf{x}_{true}\|}{\|\mathbf{x}_{dif}\|}$, where $\|\mathbf{x}_{dif}\|$ is the ℓ_2 norm of the difference between reconstructed image vector and the original vector $\mathbf{x}_{true}$. The $\lambda = 0.1$ and the projection $\mathbf{y}$ are influenced by Gaussian noise, with SNR of 28.8dB.

The change of SNR is shown in Fig.12. We here plot SNR against effective epochs, where an effective epoch is a normalized iteration count that corrects for the fact that the stochastic version of our algorithm only updates a subset of elements at each iteration.Comparing BSGD-TV against FISTA and ISTA, BSGD-TV shows a faster convergence speed, even though it only update blocks of $\mathbf{x}$ in each iteration and only applies the TV-based de-noising after a period of iterations. A visual comparison after a fixed number of effective epochs is shown in Fig. 13.

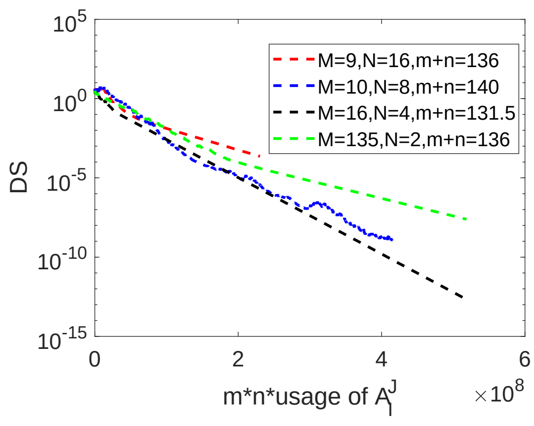


(a)

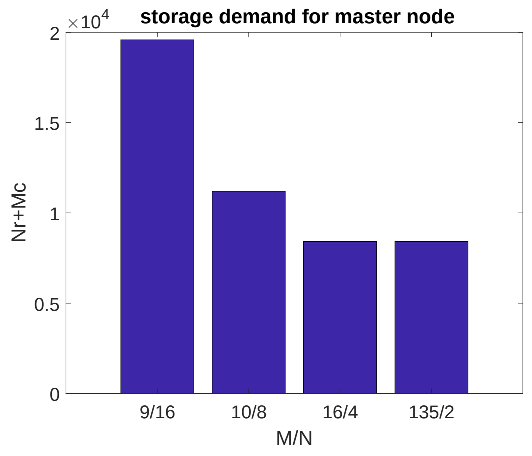


(b)

Fig. 8. The computational complexity of the computations in each node is proportional to mn, where m and n are the numbers of rows and columns in$\mathbf{A}_I^J$. Storage requirements in each compute node is however proportional to $m+n$, which is shown here in panel (a). For the master node, storage requirements are proportional to $Mr+Nc$, which is shown in panel (b).



(a)



(b)

Fig. 9. Convergence of BSGD over 3000 iterations (a) when we have 2 compute nodes. The best compromise between convergence speed and master node storage demand (b) is found for $M=16, N=4$.

Note that ADMM-TV [49] is another algorithm that can be distributed over several nodes, however, our previous work has demonstrated that ADMM-TV is significantly slower than TV constraint version of CSGD [42] and thus has not been included in the comparison here.

TABLE 1
Three different reconstruction scales

	case1	case2	case3
Detector	400*400	1000*1000	2000*2000
Volume	256*256*256	512*512*512	1024*1024*1024
Rows of $\mathbf{A}$	5.76×10^7	3.6×10^8	1.44×10^9
Columns of $\mathbf{A}$	1.68×10^7	1.34×10^8	1.07×10^9

3.5 Applying BSGD in 3D CT reconstruction

In this section, a workstation containing two NVIDIA GEFORCE GTX 1080Ti GPUs is adopted to demonstrate BSGD's performance on realistic data sizes. We used a 3D cone beam scanning geometry similar to the 2D simulation as defined in Fig.3. In simulation, $OP = 1536$ mm, $OD = 1000$ mm and the detector is a square panel with side length(EF) of 400 mm. The reconstruction volume is a cube with side length of 256 mm. The point source and the centre of the square detector are located at the middle slice of the 3D volume. They rotate around the volume horizontally for a full circle with angular increments of 1°. We test BSGD for increasing data sizes (see Table 1) and compare it to other methods. The simulations are performed in MATLAB R2016b together with a purpose built version of the TIGRE toolkit [50] that uses OpenMP to synchronize the two GPUs, performing two FPs or two BPs simultaneously. Simulation results show that BSGD with importance sampling and automatic parameter tuning can be applied in realistic settings and is faster than existing methods in a multi-GPU work station.

3.5.1 *Time required for FP, BP and other operations*

We start by looking at the time required to compute FB/BP and contrast this time to the other computational overheads of the method. For one GPU, we process two data blocks one after the other whilst for two GPUs two blocks are computed in parallel. We here divided the image into 8 cubic blocks ($N = 8$) and randomly partitioned the 360

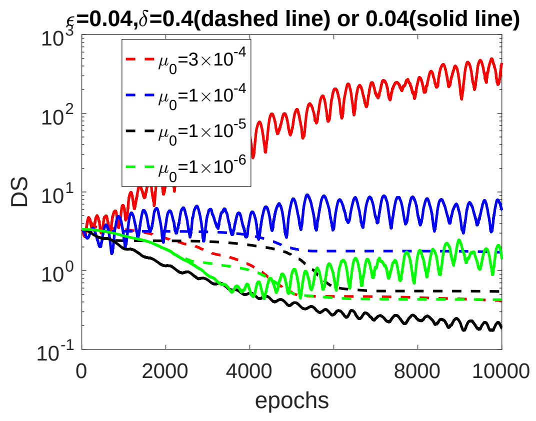
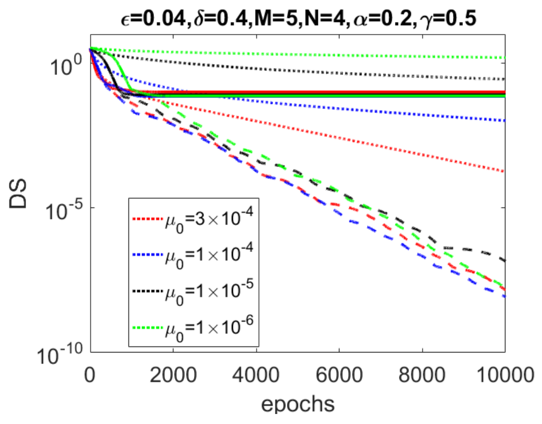
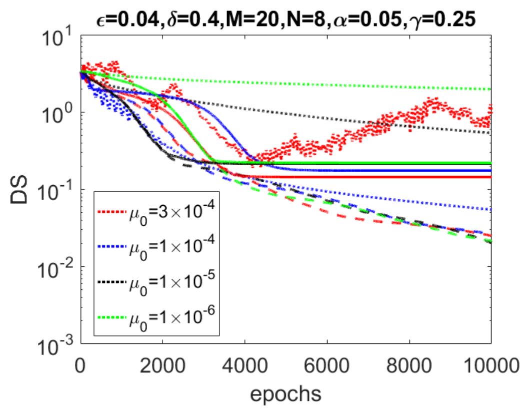


Fig. 10. μ_0 is the initial step length. Different color stands for different μ_0. Only using criteria 1 does not overcome issues with parameter selection. The dashed lines use a large δ leading to the algorithm getting stuck, whilst solid lines use a small δ leading to oscillation and divergence.



(a)



(b)

Fig. 11. The dotted line is the original BSGD method with constant step length μ_0. The solid line is the automatic parameter tuning method using criteria 1 and the dashed line is the automatic parameter tuning using criteria 1 and 2. Panels (a) and (b) show different scenarios in terms of M, N, α and γ.

projections into 5 groups ($M = 5$). As we here look at computation speed of individual FP and BPs, no noise was added to the projections.

The overall time spent on FP and BP per iteration are shown in Fig.14. The measurements here are averaged over 10 repeated runs.

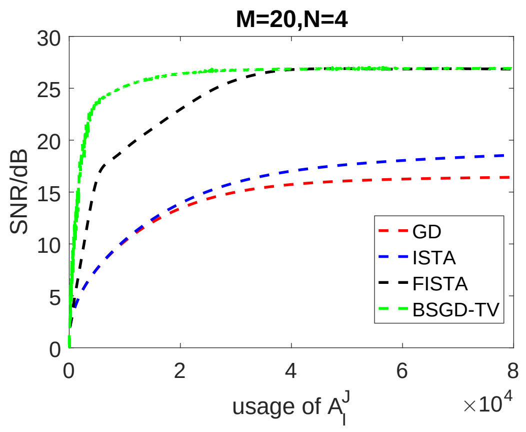
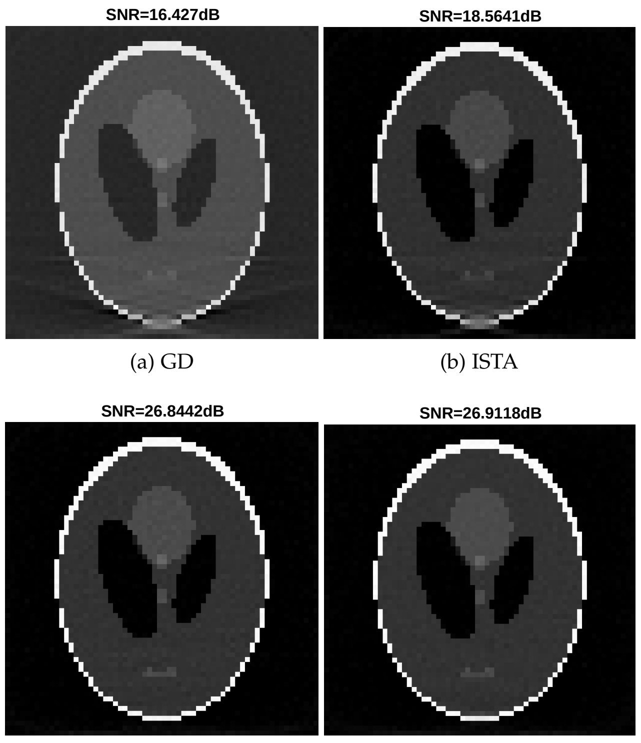


Fig. 12. Comparison between BSGD-TV, FISTA, ISTA and Gradient Descend (GD) (without TV constraint). The step length μ for ISTA, GD and BSGD-TV is 0.0004. BSGD-TV uses $M = 20, N = 4, \alpha = 0.05 and \gamma = 0.5$ with 2 nodes available in the network. λ in Eq.5 is 0.1. The low SNR of GD suggests the necessity of incorporating the TV norm here. It can be seen that the BSGD-TV converge faster than FISTA methods.

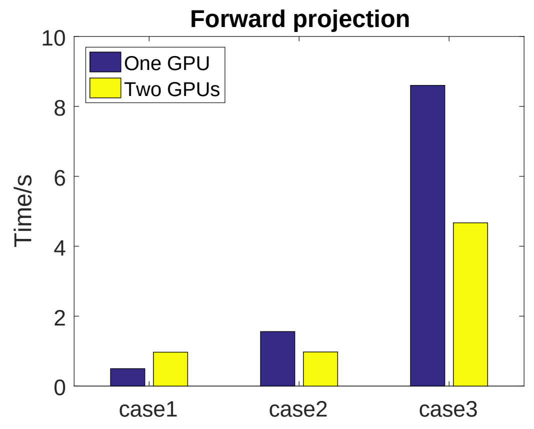


(a) GD (b) ISTA

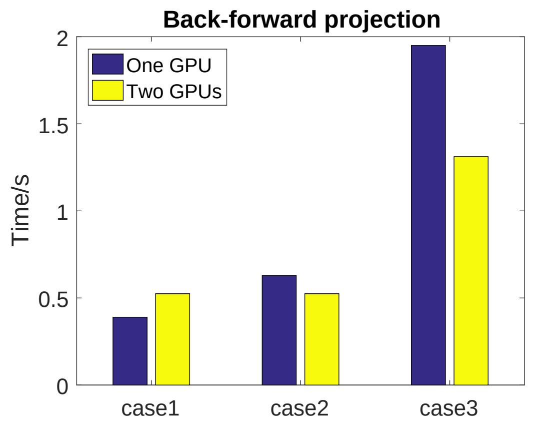
(c) FISTA (d) BSGD

Fig. 13. Reconstruction results after 500 effective epochs.

We also measured the proportion of time each iteration of BSGD spent on FP and BP during reconstruction (See Fig.15), which shows that for increasing problem sizes, FP and BP become increasingly smaller fractions of overall cost. As data transfer and other operations are similar in the 1 and 2 GPU settings, this further demonstrates that multi-GPU reconstruction is beneficial to reduce the time spent on FP/BP.

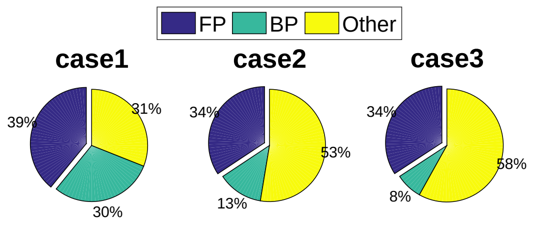


(a) FP

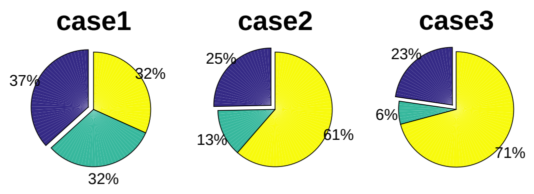


(b) BP

Fig. 14. When the problem size is small, using two GPUs does not provide acceleration as the overhead on communication and synchronization dominates. However, for realistic scales, using two GPUs nearly achieves the optimal doubling of computation speed with two GPUs for the FP and 2/3 acceleration of the BP.



(a) Using one GPU

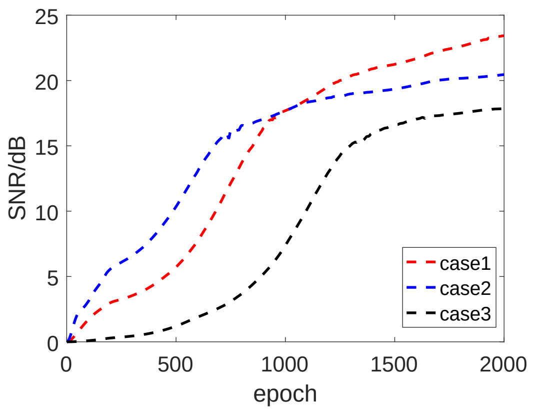


(b) Using two GPUs

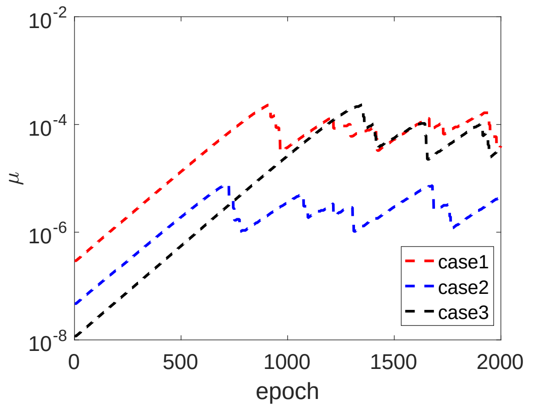
Fig. 15. The relative time spent on each stage during an entire reconstruction task. For increased problem sizes, the percentage of time spent on FP and BP decreases faster when using two GPUs.

3.5.2 *Quality of reconstruction*

We next look at the quality of reconstruction for the three scenarios. Since there are two GPUs are available, we thus set $M = 5, N = 8, \alpha = \frac{1}{5}$ and $\gamma = \frac{2}{8}$. We here measure quality in terms of SNR. The results are shown in Fig.16.



(a) SNR trend



(b) step length trend

Fig. 16. During 2000 epochs, BSGD provides an increasing SNR trend under different data-set scales. The step length can also be automatically adjusted into a range that enable the iteration results move towards to the true solution.

Reconstructed slices are shown in Fig.17.

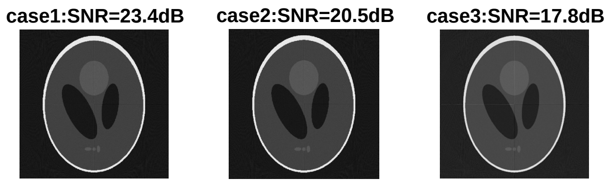


Fig. 17. Slices from the 3D reconstruction for different problem dimensions, showing the effectiveness of BSGD under different cases.

3.5.3 *Comparison with other methods*

In this section, we demonstrate the advantage of BSGD compared with other methods. A cubic skull skeleton provided by the TIGRE toolkit was used and reconstructed using different methods. The object to detector distance was 536 mm, source to object distance was 1000 mm and we again collected 360 equally spaced projections and reconstructed onto a 256 by 256 by 256 grid. The detector used 512×512

pixels. The side length of each voxel and detector pixel were 1 mm, so that $\mathbf{A} \in \mathbb{R}^{9.4\times10^7 * 1.7\times10^7}$. The projection data $\mathbf{y}$ is deteriorated by white Gaussian noise with SNR of 28.1 dB. In our simulations, we assume that each computation node (GPU) can only process $\frac{1}{8}$ of the volume and 18 projections. Since in the previous simulation we have demonstrated that BSGD can outperform CAV and SIRT, we here compared BSGD-IM with SAG, SVRG, GD, GD-BB [51], FISTA and ORBCDVD [52]. Except for BSGD-IM, the other methods divide $\mathbf{A}$ into $20 * 8$ sub-matrices (i.e. $M = 20, N = 8$ to enable each row blocks contain 18 projections and each column blocks contain $\frac{1}{8}^{th}$volume. $\alpha = \frac{1}{20}, \gamma = \frac{1}{4}$ is set according to Eq.8) and consecutively process the sub-matrices on each GPU one at a time. BSGD-IM, sampling $\frac{1}{4}^{th}$ projection from each projection angle, allows us to divide $\mathbf{A}$ into $5*8$ sub-matrices with α and γ set to 0.2 and 0.5. By this division, it is guaranteed that the computation amount for FP and BP of BSGD-IM are the same with the other methods. This is because that despite that the BSGD-IM process 72 projection angles each time, for each projection angle, only $\frac{1}{4}^{th}$ projection data are used for each projection angle. As a result, the actual projection data size is equivalent to $72 * \frac{1}{4} = 18$ full projection angles, which is the size for other methods.

In the large scale reconstruction case, calculating the least square solution itself can be time consuming, thus using the term DS is inapplicable in realistic case. To reflect the speed of the iteration result approaching to the least square solution, we thus plot $GAP = \|\mathbf{y} - \mathbf{A}\mathbf{x}_{est}\|$ as a function of the number of usages of A_I^J for each FP and BP. This is reasonable since the least square solution minimizes GAP and a faster downward trend suggests a faster reconstruction speed. Convergence results are shown in Fig.18. It can be seen that BSGD-IM is faster than the other methods. Reconstruction results are shown in Fig.19, where we show a subsection of a 2D slice after 2000 forward and backward projections.

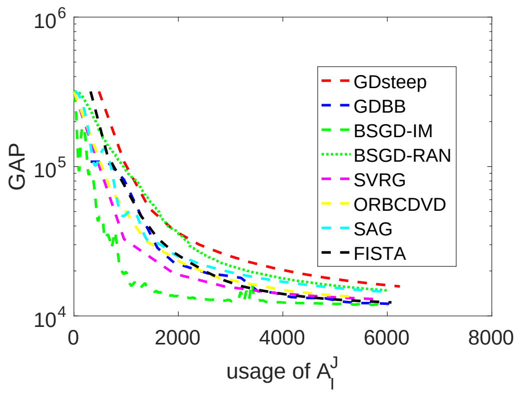
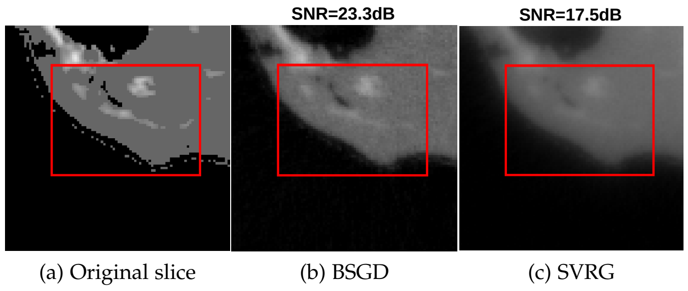


Fig. 18. BSGD-RAN is similar to BSGD-IM, but it uniformly selects the sub-projections at each projection angle while BSGD-IM selects sub-projections based on sub-matrix sparsity. Both BSGD methods use automatic parameter tuning. The parameters in the other methods were optimised to ensure optimal performance.



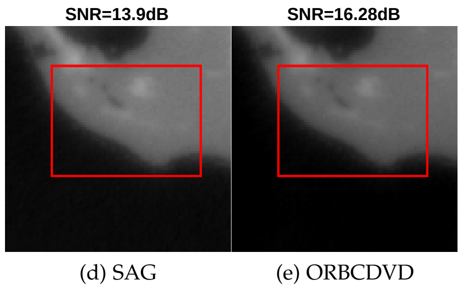


Fig. 19. Reconstruction results after 2000 projections. BSGD provides better reconstructions with a higher signal noise ratio(SNR).

4 Fixed Point Analysis

In this part we show that BSGD has a single fixed point at the least square solution of the optimisation problem. The system matrix $\mathbf{A} \in \mathbb{R}^{r*c}$, has M row blocks and N column blocks. We vectorise the sets $\{\mathbf{z}^j\}_{j=1}^N$ and $\{\mathbf{g}^i\}_{i=1}^M$ and put their elements into the vector $\overline{\mathbf{z}} \in \mathbb{R}^{Nr*1}$ and $\overline{\mathbf{g}} \in \mathbb{R}^{Mc*1}$. $\overline{\mathbf{A}} \in \mathbb{R}^{Nr*c}$ is a deformation of $\mathbf{A}$, defined as:

$$\overline{\mathbf{A}} = \begin{bmatrix} \mathbf{A}_{I_1}^{J_1} & \mathbf{0} & \dots & \mathbf{0} \\ \vdots & \vdots & & \vdots \\ \mathbf{A}_{I_M}^{J_1} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{I_1}^{J_2} & \dots & \mathbf{0} \\ \vdots & \vdots & & \vdots \\ \mathbf{0} & \mathbf{A}_{I_M}^{J_2} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & & \mathbf{A}_{I_1}^{J_N} \\ \vdots & \vdots & & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{A}_{I_M}^{J_N} \end{bmatrix} \quad (9)$$

$\overline{\mathbf{A}^T} \in \mathbb{R}^{Mc*r}$ is a similar deformation of $\mathbf{A}^T$. Let us also introduce the matrix $\overline{\mathbf{I}}_{Nr} = [\mathbf{I}_r, \mathbf{I}_r, \dots, \mathbf{I}_r] \in \mathbb{R}^{r*Nr}$, where we concatenate N identity matrices $\mathbf{I}_r$ each of size $r*r$. With this notation we obtain:

$$\begin{aligned} \overline{\mathbf{I}}_{Nr}\overline{\mathbf{A}} &= \mathbf{A}, \\ \overline{\mathbf{I}}_{Mc}\overline{\mathbf{A}^T} &= \mathbf{A}^T, \\ \sum_{j=1}^{N} \mathbf{z}^j &= \overline{\mathbf{I}}_{Nr}\overline{\mathbf{z}}, \\ \mathbf{g} &= \overline{\mathbf{I}}_{Mc}\overline{\mathbf{g}}, \end{aligned} \quad (10)$$

where the definition of $\mathbf{g}$ and $\{\mathbf{z}^j\}$ can be found in the algorithm description. To encode the random updates over subsets of index pairs I_i, J_j, we introduce the random matrices $\mathbf{R}_1 \in \mathbb{R}^{Nr*Nr}$, $\mathbf{R}_2 \in \mathbb{R}^{Mc*Mc}$ and $\mathbf{R}_3 \in \mathbb{R}^{c*c}$, which are diagonal matrices whose diagonal entries are either 0 or 1. With this notation, we can write the update of $\mathbf{z}$, $\mathbf{g}$ and $\mathbf{x}$ as:

$$\overline{\mathbf{z}}^{k+1} = \overline{\mathbf{z}}^{k} + \mathbf{R}_1 \left[\overline{\mathbf{A}}\mathbf{x}^k - \overline{\mathbf{z}}^k\right], \tag{11}$$

$$\overline{\mathbf{g}}^{k+1} = \overline{\mathbf{g}}^{k} + \mathbf{R}_2 \left[\overline{\mathbf{A}^T}\left(\mathbf{y} - \overline{\mathbf{I}}_{Nr}\overline{\mathbf{z}}^{k+1}\right) - \overline{\mathbf{g}}^k\right] \tag{12}$$

and

$$\mathbf{x}^{k+1} = \mathbf{x}^{k} + \mu\mathbf{R}_3\overline{\mathbf{I}}_{Mc}\overline{\mathbf{g}}^{k+1}, \tag{13}$$

where k is the epoch number. Inserting Eq.11 into Eq.12 and then Eq.12 into Eq.13, the recursion in Eq.14 is obtained:

$$\begin{bmatrix}\overline{\mathbf{z}}^{k+1}\\ \overline{\mathbf{g}}^{k+1}\\ \mathbf{x}^{k+1}\end{bmatrix} = \mathbf{M}\begin{bmatrix}\overline{\mathbf{z}}^{k}\\ \overline{\mathbf{g}}^{k}\\ \mathbf{x}^{k}\end{bmatrix} + \begin{bmatrix}\mathbf{0}\\ \mathbf{R}_2\overline{\mathbf{A}^T}\mathbf{y}\\ \mu\mathbf{R}_3\overline{\mathbf{I}}_{Mc}\mathbf{R}_2\overline{\mathbf{A}^T}\mathbf{y},\end{bmatrix}, \tag{14}$$

where $\mathbf{M}$ is

$$\begin{bmatrix}\mathbf{I}-\mathbf{R}_1 & \mathbf{0} & \mathbf{R}_1\overline{\mathbf{A}}\\ \mathbf{R}_2\overline{\mathbf{A}^T}\overline{\mathbf{I}}_{Nr}(\mathbf{R}_1-\mathbf{I}) & \mathbf{I}-\mathbf{R}_2 & -\mathbf{R}_2\overline{\mathbf{A}^T}\overline{\mathbf{I}}_{Nr}\mathbf{R}_1\overline{\mathbf{A}}\\ \mu\mathbf{R}_3\overline{\mathbf{I}}_{Mc}\mathbf{R}_2\overline{\mathbf{A}^T}\overline{\mathbf{I}}_{Nr}(\mathbf{R}_1-\mathbf{I}) & \mu\mathbf{R}_3\overline{\mathbf{I}}_{Mc}(\mathbf{I}-\mathbf{R}_2) & \mathbf{I}-\mu\mathbf{R}_3\overline{\mathbf{I}}_{Mc}\mathbf{R}_2\overline{\mathbf{A}^T}\overline{\mathbf{I}}_{Nr}\mathbf{R}_1\overline{\mathbf{A}}\end{bmatrix}$$

In the fixed point analysis, note that for any fixed point $\mathbf{x}^\star$, the random updates of $\mathbf{z}$ mean that we require that $\mathbf{z}^j_{I_i} = \mathbf{A}^{J_j}_{I_i}\mathbf{x}^\star_J$ for all i and j, so that the fixed $\mathbf{z}^\star$ must be of the form $\mathbf{z}^\star = \overline{\mathbf{A}}\mathbf{x}^\star$. So we need:

$$\begin{bmatrix}\overline{\mathbf{A}}\mathbf{x}^\star\\ \overline{\mathbf{g}}^\star\\ \mathbf{x}^\star\end{bmatrix} = \mathbf{M}\begin{bmatrix}\overline{\mathbf{A}}\mathbf{x}^\star\\ \overline{\mathbf{g}}^\star\\ \mathbf{x}^\star\end{bmatrix} + \begin{bmatrix}\mathbf{0}\\ \mathbf{R}_2\overline{\mathbf{A}^T}\mathbf{y}\\ \mu\mathbf{R}_3\overline{\mathbf{I}}_{Mc}\mathbf{R}_2\overline{\mathbf{A}^T}\mathbf{y}\end{bmatrix}. \tag{15}$$

Since the first line of Eq.15 is an identity, we only focus on the second and third line. The second line can be expressed as:

$$\mathbf{R}_2\left(\overline{\mathbf{A}^T}(\mathbf{y}-\mathbf{A}\mathbf{x}^*) - \overline{\mathbf{g}}^*\right) = \mathbf{0}. \tag{16}$$

As $\mathbf{R}_2$ is a random diagonal matrix, this implies that $\mathbf{g}^* = \overline{\mathbf{A}^T}(\mathbf{y}-\mathbf{A}\mathbf{x}^*)$. The third line, after the deformation, can be expressed as:

$$\mathbf{R}_3\overline{\mathbf{I}}_{Mc}\mathbf{R}_2\left(\overline{\mathbf{A}^T}\left(\mathbf{y}-\mathbf{A}\mathbf{x}^\star\right) - \overline{\mathbf{g}}^\star\right) + \mathbf{R}_3\overline{\mathbf{I}}_{Mc}\overline{\mathbf{g}}^\star = \mathbf{0} \tag{17}$$

which suggest that $\mathbf{g}^\star \equiv \overline{\mathbf{I}}_{Mc}\overline{\mathbf{g}}^\star \equiv 2\mathbf{A}^T(\mathbf{y}-\mathbf{A}\mathbf{x}^\star) = \mathbf{0}$. This proves that the fixed point is the least square solution. Our empirical results show convergence to the fixed point. A theoretical analysis and formal convergence proof is in preparation.

5 Conclusion

BSGD can be viewed as an improvement of our previous CSGD algorithm. It mainly focus on the case where a distributed network is adopted to reconstruct a large scale CT image/volume in parallel and the nodes in the network have limited access to both projection data and volume. When noise is Gaussian type, which is a common case in CT reconstruction area, iteration results obtained by BSGD approaches closer to the least square solution than other mature CT reconstruction algorithms such as SIRT, CAV and our previously CSGD method. Compared with the other optimization algorithm proposed in machine learning area, such as SVRG, ORBCDVD, the BSGD has higher computation efficiency. It also has the ability to address the sparse view CT reconstruction by combining itself with TV regularization. Simulations prove that both BSGD and BSGD-TV have the ability to be applied on a distributed network.

References

[1] Y. Sagara, A. K. Hara, and W. Pavlicek, "Abdominal CT: comparison of low-dose CT with adaptive statistical iterative reconstruction and routine-dose CT with filtered back projection in 53 patients," *Am. J. Roentgenol.*, vol. 195, no. 3, pp. 713–719, Sep. 2010.

[2] E. J. Hoffman, S. C. Huang and M. E. Phelps, "Quantitation in positron emission computed tomography: 1. Effect of object size," *J. Comput. Assist. Tomo.*, vol. 3, no. 3, pp. 299–308, Jun. 1979.

[3] T. Rodet, F. Noo, and M. Defrise, "The cone-beam algorithm of feldkamp, davis, and kress preserves oblique line integrals," *Med. Phys.*, vol. 31, no. 7, pp. 1972–1975, Jun. 2004.

[4] L. Feldkamp, L. Davis and J. Kress, "Practical cone-beam algorithm," *JOSA A*, vol. 1, no. 6, pp. 612–619, Jun. 1984.

[5] A. Gervaise, B. Osemont, and S. Lecocq, "CT image quality improvement using adaptive iterative dose reduction with wide-volume acquisition on 320-detector CT," *Euro. Radio.*, vol. 22, no. 2, pp. 295–301, Feb. 2012.

[6] G. Wang, H. Yu, and B. De Man, "An outlook on x-ray ct research and development," *Med. Phys.*, vol. 35, no. 3, pp. 1051–1064, Feb. 2008.

[7] J. Deng, H. Yu, and J. Ni, "Parallelism of iterative ct reconstruction based on local reconstruction algorithm," *J. Supercomput.*, vol. 48, no. 1, pp. 1–14, Apr. 2009.

[8] M. J. Willemink, P. A. Jong and T. Leiner, "Iterative reconstruction techniques for computed tomography Part 1: technical principles," *Eur. Radiol*, vol. 23, no. 6, pp. 1623–1631, Jun. 2013.

[9] F. Jacobs, E. Sundermann, B. De Sutter, and M. Christiaens, "A fast algorithm to calculate the exact radiological path through a pixel or voxel space," *J. CIT.*, vol. 6, no. 1, pp. 89–94, Mar. 1998.

[10] M. Soleimani and T. Pengpen, "Introduction: a brief overview of iterative algorithms in x-ray computed tomography," *Phil. Trans. Rol. Soc.* vol. 373, no. 2043, pp. 1–6, Jun. 2015.

[11] X. Guo, "Convergence studies on block iterative algorithms for image reconstruction," *Appl. Math. Comput.*, vol. 273, pp. 525–534, Jan. 2016.

[12] M. Beister, D. Kolditz, and W. A. Kalender, "Iterative reconstruction methods in x-ray CT," *Phys. Medica*, vol. 28, no. 2, pp. 94–108, Apr. 2012.

[13] J. Ni, X. Li, T. He, and G. Wang, "Review of parallel computing techniques for computed tomography image reconstruction," *Curr. Med. Imaging Rev.*, vol. 2, no. 4, pp. 405–414, Nov. 2006.

[14] W. Aarle, W. Palenstijn, J. Beenhouwer, T. Altantzis, S. Bals, K. Batenburg, J. Sijbers, "The ASTRA Toolbox: A platform for advanced algorithm development in electron tomography,"*Ultramicroscopy* vol. 157, pp. 35–47, May 2015.

[15] T. M. Benson, B. K. De Man, L. Fu, and J.-B. Thibault, "Block-based iterative coordinate descent," in *NSS/MIC,2010 IEEE*, pp. 2856–2859, IEEE, 2010.

[16] Z. Yu, J.-B. Thibault, C. A. Bouman, and K. D. Sauer, "Fast model-based x-ray CT reconstruction using spatially nonhomogeneous icd optimization," *IEEE Trans. Image Process*, vol. 20, no. 1, pp. 161–175, Jan. 2011.

[17] D. Kim and J. A. Fessler, "Parallelizable algorithms for x-ray CT image reconstruction with spatially non-uniform updates," *Proc. 2nd Intl. Mtg. on image formation in X-ray CT*, pp. 33–36, 2012.

[18] J. Fessler and D. Kim,"Axial block coordinate descent (ABCD) algorithm for X-ray CT image reconstruction,"*Proc. Fully Three-Dimensional Image Reconstruct. Radiol. Nucl. Med.*, Jul. 2011.

[19] T. Li, T. J. Kao, and Isaacson, "Adaptive kaczmarz method for image reconstruction in electrical impedance tomography," *Physiol. Meas.*, vol. 34, no. 6, pp. 595, May. 2013.

[20] J. Gregor and T. Benson, "Computational analysis and improvement of SIRT," *IEEE Trans. on Med. Ima.*, vol. 27, no. 7, pp. 918–924, Jun. 2008.

[21] Y. Censor, D. Gordon, and R. Gordon, "Component averaging: An efficient iterative parallel algorithm for large and sparse unstructured problems," *Parallel Comput.*, vol. 27, no. 5, pp. 777–808, May. 2001.

[22] Y. Censor, D. Gordon, and R. Gordon, "BICAV: A block-iterative parallel algorithm for sparse systems with pixel-related weighting," *IEEE Trans. Med. Imag.*, vol. 20, no. 10, pp. 1050–1060, Oct. 2001.

[23] F. Xu, W. Xu, M. Jones, B. Keszthelyi, J. Sedat, D. Agard, K. Mueller, "On the efficiency of iterative ordered subset reconstruction algorithms for acceleration on GPUs,"*Computer methods and programs in biomedicine* vol. 98, no. 3, pp. 261–270, 2010.

[24] X. Wang, A. Sabne,S. Kisner,A. Raghunathan, C. Bouman and S. Midkiff, "High performance model based image reconstruction,"*ACM SIGPLAN Notices* vol. 51, no. 8, Mar. 2016.

[25] A. Sabne, X. Wang, S. Kisner, C. Bouman, A. Raghunathan and S. Midkiff, "Model-based iterative CT image reconstruction on GPUs,"*ACM SIGPLAN Notices* vol. 52, no. 8, pp. 207–220, Feb. 2017.

[26] X. Wang, A. Sabne, P. Sakdhnagool, S.J. Kisner, C.A. Bouman and S.P.Midkiff, "Massively parallel 3D image reconstruction,"*Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* , 2017.

[27] J.R. Bilbao-Castro, J.M. Carazo, J.J. Fernandez and I. Garcia,"Performance of Parallel 3D Iterative Reconstruction Algorithms,"*WSEAS Transactions on Biology and Biomedicine* vol. 1, no. 1, pp. 112–119, 2004.

[28] L.A. Flores, V. Vidal, P. Mayo, F. Rodenas, G. Verdu, "Fast parallel algorithm for CT image reconstruction,"*ngineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE*, pp. 4374–4377, 2012.

[29] X. Rui, L. Fu, K. Choi and B.D. Man, "Evaluation of convergence speed of a modified Nesterov gradient method for CT reconstruction,"*Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC), 2012 IEEE* , pp. 3667–3670, 2012.

[30] R. Sebastian, "An overview of gradient descent optimization algorithms,"*arXiv preprint arXiv:1609.04747* 2016.

[31] R. Johnson and T.Zhang, "Accelerating Stochastic Gradient Descent using Predictive Variance Reduction,"*NIPS* pp. 315–323, Dec. 2013.

[32] D. Blatt, O.H. Alfred, H. Gauchman, "A convergent incremental gradient method with a constant step size,"*SIAM* vol. 18, no. 1, pp. 29–51, Feb. 2007.

[33] M. Schmidt, N.L. Roux, F. Bach, "Minimizing finite sums with the stochastic average gradient,"*Mathematical Programming* vol. 162, no. 1, pp. 93–112, May. 2017.

[34] F. Niu, B. Recht and C. Re,"Hogwild!: A lock-free approach to parallelizing stochastic gradient descent,"*Advances in neural information processing systems*, pp. 693–701, 2011.

[35] S. Zhao and W. Li, "Fast Asynchronous Parallel Stochastic Gradient Descent: A Lock-Free Approach with Convergence Guarantee,"*AAAI* , pp. 2379–2385, 2016.

[36] R. Leblond, F. Pedregosa and S.L. Julien, "ASAGA: asynchronous parallel SAGA,"*arXiv preprint arXiv:1606.04809* , 2016.

[37] R. Zhang, S. Zheng and J.T. Kwok, "Fast distributed asynchronous SGD with variance reduction,"*CoRR, abs/1508.01633*, 2015.

[38] C. Wood, N. O'Brien, A. Denysov and T. Blumensath "Computed laminography of CFRP using an X-ray cone beam and robotic sample manipulator systems," *IEEE Transactions on Nuclear Science*, 65(7), pp. 1384–1393.

[39] W. J. Palenstijn, J. Bédorf, and K. J. Batenburg, "A distributed SIRT implementation for the ASTRA toolbox," in *Proc. Fully Three-Dimensional Image Reconstruct. Radiol. Nucl. Med.*, pp. 166–169, Jun. 2015.

[40] J.M. Rose, J.Wu, J.A. Fessler, T.F.Wenisch, "Iterative helical CT reconstruction in the cloud for ten dollars in five minutes,"*Proc. Intl. Mtg. on Fully 3D Image Recon. in Rad. and Nuc. Med*, pp. 241–244, 2013.

[41] M.D. Jones, R. Yao, C.P. Bhole "Hybrid MPI-OpenMP programming for parallel OSEM PET reconstruction,"*IEEE Transactions on nuclear science* vol. 53, no. 5, pp. 2752–2758 Oct. 2006.

[42] Y. Gao, T. Blumensath "A Joint Row and Column Action Method for Cone-Beam Computed Tomography,"*IEEE Transactions on Computational Imaging* vol. 4, no. 4, pp. 599–608, 2018.

[43] L. Mu, T. Zhang, Y. Chen and S. Alexander, "Efficient mini-batch training for stochastic optimization,"*Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 661–670, 2014.

[44] A.G. Baydin, R. Cornish, D.M. Rubio, M. Schmidt, F. Wood, "Online learning rate adaptation with hypergradient descent,"*arXiv preprint arXiv:1703.04782*, 2017.

[45] C. Tan, S. Ma, Y. Dai and Y. Qian, "Barzilai-Borwein step size for stochastic gradient descent,"*Advances in Neural Information Processing Systems*, pp. 685–693, 2016.

[46] A. Plakhov and P. Cruz, "A stochastic approximation algorithm with step-size adaptation,"*Journal of Mathematical Sciences*, vol. 120, no. 1, pp. 964–973.

[47] Combettes, "Signal recovery by proximal forward-backward splitting,"*Multiscale Modeling & Simulation*, vol.—4,no.—4 pp.—1168–1200, 2005.

[48] A. Beck, M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems,"*SIAM J. on Ima. Sci.*, vol. 2 ,no. 1, pp. 183–202, Oct. 2008.

[49] S. Boyd, N. Parikh, E. Chu, B. P and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers,"*Foundations and Trends in Machine Learning* vol. 3, no. 1, pp. 1–122, Jul. 2011.

[50] A. Biguri, M. Dosanjh, S. Hancock and M. Soleimani, "TIGRE: a MATLAB-GPU toolbox for CBCT image reconstruction,"*IMA journal of numerical analysis* vol. 2, no. 5, pp. 055010, Jun. 2016.

[51] J. Barzilai, M.B. Jonathan, "Two-point step size gradient methods,"*IMA journal of numerical analysis* vol. 8, no. 1, pp. 141–148 Jan. 1988.

[52] H. Wang, A. Banerjee, "Randomized block coordinate descent for online and stochastic optimization,"*arXiv preprint arXiv:1407.0107* Jul. 2014.

Yushan Gao received the B.E. degree in optical engineering from Beijing Institute of Technology, Beijing, China, in 2014. He finished the master study in National University of Defense Technology, Changsha, China, in 2016. He is currently pursing the Ph.D. degree with the Faculty of Engineering and Environment, University of Southampton, Southampton, UK. His research interests include image reconstruction and parallel computation.

Ander Biguri received the MSc in Electronic Engineering at Mondragon Unibertsitatea, Spain, in 2013). He received the Ph.D. degree at the University of Bath, UK jointly with CERN in 2018. His research interests include CT iterative algorithms, GPU computing and motion compensation.

Thomas Blumensath received the B.Sc. (Hons.) degree in music technology from Derby University, Derby, U.K., in 2002 and the Ph.D. degree in electronic engineering from Queen Mary, University of London, London, U.K., in 2006. He is now an Associate Professor at the University of Southampton where he specialises in statistical and mathematical methods for signal and image processing with a focus on tomographic imaging problems.