Distributed Computation of Linear Inverse Problems with Application to Computed Tomography Reconstruction

Yushan Gao, Thomas Blumensath,

Abstract—The inversion of linear systems is a fundamental step in many inverse problems. Computational challenges exist when trying to invert large linear systems, where limited computing resources mean that only part of the system can be kept in computer memory at any one time. We are here motivated by tomographic inversion problems that often lead to linear inverse problems. In state of the art x-ray systems, even a standard scan can produce 4 million individual measurements and the reconstruction of x-ray attenuation profiles typically requires the estimation of a million attenuation coefficients. To deal with the large data sets encountered in real applications and to utilise modern graphics processing unit (GPU) based computing architectures, combinations of iterative reconstruction algorithms and parallel computing schemes are increasingly applied. Although both row and column action methods have been proposed to utilise parallel computing architectures, individual computations in current methods need to know either the entire set of observations or the entire set of estimated x-ray absorptions, which can be prohibitive in many realistic big data applications. We present a fully parallelizable computed tomography (CT) image reconstruction algorithm that works with arbitrary partial subsets of the data and the reconstructed volume. We further develop a non-homogeneously randomised selection criteria which guarantees that sub-matrices of the system matrix are selected more frequently if they are dense, thus maximising information flow through the algorithm. A grouped version of the algorithm is also proposed to further improve convergence speed and performance. Algorithm performance is verified experimentally.

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

I. INTRODUCTION

In transmission computed tomography (CT), standard scan trajectories, such as rotation based or helical trajectories, allow the use of efficient analytical reconstruction techniques such as the filtered backprojection algorithm (FBP) [1], [2] and the Feldkamp Davis Kress (FDK) [3], [4] method. However, in low signal to noise settings, if scan angles are under-sampled or if nonstandard trajectories are used, then less efficient, iterative reconstruction methods can provide significantly better reconstructions [5]–[8]. These methods model the x-ray system as a linear system of equations:

\[
y = Ax + e,\]

(1)

where \(y\), \(A\), \(x\) and \(e\) are projection data, system matrix, reconstructed image vector and measurement noise respectively. However, the relatively lower computational efficiency limits their use, especially in many x-ray tomography problems, where \(y\) and \(x\) can have millions of entries each [9] and where \(A\), even though it is a sparse matrix, can have billions of entries.

For realistic dataset sizes, these methods thus require significant computational resources, especially as the matrix \(A\) is seldom kept in memory but is instead re-computed on the fly, which can be done relatively efficiently using the latest Graphical Processor Unit (GPU) based parallel computing platforms. However, as GPUs have limited internal memory, this typically requires the data \(y\) and/or reconstruction volume \(x\) to be broken into smaller subsets on which individual computations are performed. The development of efficient algorithms that only work on subsets of the data at any one time is thus becoming of increasing interest [10], [11].

Currently, most of these methods can be divided into two categories: row action methods, which operate on subsets of the observations \(y\) at a time and column action methods, which operate on subsets of the voxels \(x\) at a time [12]–[14].

Row action methods: A classical method is the Kaczmarz algorithm (also known as the Algebraic Reconstruction Technique (ART)) which together with its block based variants are widely used in tomographic image reconstruction [7], [15]–[17]. Increased convergence rates are obtained with block Kaczmarz methods, which are also known as the Simultaneous Algebraic Reconstruction Technique (SART) [11], [18]. By simultaneously selecting several rows of projection data, SART can also be converted into efficient parallel strategies [19]. Apart from the Kaczmarz method, another classical row action method is the simultaneous iterative reconstruction technique (SIRT). The SIRT method updates each reconstructed image voxel by combining all projection values whose corresponding x-rays passing through this voxel. In this way, the convergence rate significantly increases but so does the computation load [20], [21]. To lower the computational load within one iteration, the block form of SIRT is widely used in distributed computing systems [11], [22], [23], yielding results of superior quality at the cost of increasing the reconstruction time.

Since in CT systems the system matrix is often large and sparse, component averaging (CAV) and its block form (BICAV) methods have been developed to utilise the sparsity property [24]–[26]. Furthermore, BICAV uses an additional coefficient matrix based on the number of non-zero elements.
in each column within a row block of the system matrix, thus significantly increases the convergence rate compared to the original CAV method.

**Column action methods:** Column action methods are also called iterative coordinate descent method (ICD). They reduces the N-dimensional optimization problem into a one-dimensional problem, which is shown to have a faster convergence rate in the margins of the reconstructed image [27]. A non-homogeneous (NH-ICD) update strategy is proposed in [28], [29] to increase the convergence rate. The strategy first generates a pixel/voxel selection criterion and then based on this criterion voxels which are furthest from convergence are selected to be updated. To increase the scalability and parallelism, a block form ICD (B-ICD) is proposed in [30]. The volume object is sliced along with the helical rotation axis and within one slice several neighbouring voxels are grouped together as a block. After all blocks within one slice are updated, the slice index steps along with the axial direction to the next slice and repeats the same iteration. Experiments show that a bigger block size is of help to increase the convergence rate, but at the cost of heavier computation amount. To reduce the computation complexity brought by B-ICD, the ABCD algorithm, which is a derivative of B-ICD method, combines the pixel line feature [28], [29] and the block form [30] to update the pixel line simultaneously, leading to a faster convergence rate. Inspired by NH-ICD, the NH idea is also applied to the ABCD method by more frequently updating the axial blocks which change by the largest amount during an iteration [31]. However, this block method is only suitable for standard CT scanners with circular or helical scanning trajectories and may not be appropriate for arbitrary scanning trajectories.

To optimize systems of linear equation \( y = Ax \), [32] presents a concise summary of both row action and column action algorithm, which are show in Algo.1 and Algo.2.

**Algorithm 1** Generic row action iteration

**Initialization:** \( x^0 \in \mathbb{R}^n \) is arbitrary. Both matrix \( A \) and vector \( x \) are divided into \( p \) row blocks. \( A_i \) and \( y_i \) are corresponding \( i^{th} \) row blocks. \( x^k \) is the estimate or \( x \) in the \( k^{th} \) iteration. \( \omega_i \) and \( M_i \) are relaxation parameters and coefficient matrices respectively.

**for** \( k = 0, 1, 2, \ldots \) (epochs or outer iterations) **do**

**for** \( i = 1, 2, \ldots, p \) (inner iterations) **do**

\[
\begin{align*}
z^i &= z^{i-1} + \omega_i A_i^T M_i (y_i - A_i z^{i-1}) \\
x^{k+1} &= z^p
\end{align*}
\]

**end for**

**end for**

These two generic algorithms can be applied in parallel by parallelising the inner loop and summing or averaging the updates of \( z^i \) of \( r^{i,j} \) respectively. Each algorithm has their own advantages and drawbacks. For row action method, each sub-iteration does not require the calculation or storage of the entire matrix \( A \) in advance but only needs to calculate a row block \( A_i \) at a time. This can be an advantage in large 3D reconstruction problems where the storage of the whole matrix is infeasible. However, if the algorithm is applied in a parallel form, then each processor (or node) needs to store the whole reconstructed image vector \( x \) as each update is performed on the entire image, which can be computationally challenging in large data reconstructions, especially in terms of the required forward projection \( A_i x \) and back projection \( A_i^T r \) [24]. On the other hand, using column action algorithms in a parallel computing scheme does not require each processor to store the whole reconstructed image but only to store a small part of it. However, they instead require access to all of \( y \) or \( r \), which again can be prohibitive in large scale situations.

Thus, row action and column action methods require access to the entire vectors \( y \) (or \( r \)) or \( x \) in each sub-iteration. There are few exceptions to this. One exception is the work in [33], where a row action method is discussed in which each node only requires parts of the reconstructed image vector \( x \). By splitting the image volume along planes perpendicular to the rotation axis, this variant of SIRT calculates the area of overlap on the detector between two adjacent sub-volumes. Only the data within the overlapping areas on the detector are exchanged between computation nodes. However, this method only works for circular scan trajectories and its scalability is limited due to the requirement that the overlap area should be small to minimise data communication overheads.

Our goal here is to develop an algorithm working with generic x-ray tomographic scanning trajectories and with large, realistic data sizes. In this paper, we thus develop a novel algorithm called coordinate-reduced steepest gradient descent (CSGD) that combines aspects of row action and column action methods. Unlike traditional algorithms, our parallelizable algorithm operates on arbitrary subsets of both \( x \) and \( y \) at any one time. There are thus no longer any restrictions on scan trajectories or on the way in which the volume is decomposed. The algorithm is thus applicable to arbitrary scan trajectories and is scalable so that it can be run on a range of computing platforms, including low memory GPU clusters and high performance CPU based clusters. In our algorithm, the reconstruction volume is divided into several sub-volumes, which can be updated separately at different computing nodes. This update is based on a subset of the observations, as well as the reconstructed volume, so that only a small sub-matrix of the system matrix \( A \) is used in each step.

The rest of this paper is organised as follows. The first
section gives an overview over our new block iterative method and the second section describes the proposed algorithm in more details. Some simulation results are illustrated in section III and conclusions and further discussions are presented in the last section.

II. DESCRIPTION OF CSGD

A. Notation

In this paper we assume a linear, i.e. monochromatic x-ray model [34]–[36]:

\[ y = Ax + e. \]  

(2)

For now, we use a simple least square cost function to solve this inverse problem:

\[ f(x) = (y - Ax)^T (y - Ax), \]  

(3)

where \( A \) is the system matrix derived using, for example, Siddon’s method [37]. The element \( A_{ij} \) is the intersection length of the \( i \)th x-ray beam with the \( j \)th reconstruction voxel. In this paper, \( I \) will be an index set that indexes rows in \( A \) (or the subset of x-ray measurements \( y \) ) \( I = i_1, i_2, \ldots \). Similarly, \( J \) will be a set of column indexes of \( A \) (or the index set of a subset of voxels \( x \) ) \( J = j_1, j_2, \ldots \). Thus the matrix \( A_{I,J} \) will be a sub-matrix of \( A \) with row indexes \( I \) and column indexes \( J \). Thus we can divide the linear system into several blocks:

\[
\begin{bmatrix}
  y_{I1} \\
  \vdots \\
  y_{Im}
\end{bmatrix} \approx
\begin{bmatrix}
  A_{I11} & A_{I1J_2} & \cdots & A_{I1J_m} \\
  \vdots & \vdots & \cdots & \vdots \\
  A_{Im1} & A_{ImJ_2} & \cdots & A_{ImJ_m}
\end{bmatrix}
\begin{bmatrix}
  x_{J1} \\
  \vdots \\
  x_{Jm}
\end{bmatrix} = \begin{bmatrix}
  A_{I1} \\
  \vdots \\
  A_{Im}
\end{bmatrix} x.
\]  

(4)

Note that the index sets can be arbitrary partitions of the columns and rows and do not necessarily have to be consecutive. For the convenience of the latter discussion, we also define \( r = y - Ax \) and \( r_I \) as the subset of \( r \) defined as \( y_I - A_Ix \).

B. Derivation of the algorithm

The key idea in CSGD is to minimize the partial residual \( r_I \) with partial coordinate \( x_I \). In one iteration, after selecting row and column index sets \( I \) and \( J \), the object function becomes:

\[ f(x) = r_I^T r_I = (y_I - A_Ix)^T (y_I - A_Ix). \]  

(5)

The steepest descend direction \( g \) of \( f(x) \) is then:

\[ g = -\nabla f(x) = A_I^T (y_I - A_I x) = A_I^T r_I. \]  

Assume that only those voxels whose indices are in the set \( J \) are updated, then the new descend direction becomes:

\[ g = \begin{bmatrix}
  g_J \\
  0
\end{bmatrix} = \begin{bmatrix}
  (A_J^T)^T r_J \\
  0
\end{bmatrix} \]  

(7)

and the update on the selected voxels becomes:

\[ \begin{align*}
x^{n+1}_J &= x^*_J + \mu g_J, \\
x^{n+1}_j &= x^{n+1}_j,
\end{align*} \]  

(8)

where \( \mu \) is the gradient step length and \( J \) is the complement to the set \( J \).

Ideally, we would like to compute the step length \( \mu \) such that \( f(x^{n+1}) \) is minimised:

\[ \nabla f(x^{n+1})^T g = 0, \]  

(10)

where \( \nabla f(x^{n+1}) = (A_J)^T (A_J x^{n+1} - y_J) \). Using the fact that \( A_J g = A_J^T g_J \), the steepest step length is thus calculated as:

\[ \mu = \frac{g_J^T (A_J^T)^T r_J}{g_J^T (A_J^T)^T A_J^T g_J} = \frac{g_J^T g_J}{g_J^T (A_J^T)^T A_J^T g_J}. \]  

(11)

With this step length, each iteration only requires the information of \( A_J^T \), which significantly reduces the calculation amount when the matrix is generated on the fly with Siddon’s method. Although the calculation of the step size \( \mu \) still requires access to the error \( r_J = y_J - A_J x \), the calculation of \( r_J \) can be replaced by an update process [38], [39]. Since the update within one iteration only changes \( x_J \), we have:

\[ A_J x^{n+1} = A_J x^n - A_J^T x_J + A_J^T x_{J, n+1}^*, \]  

(12)

where \( x_J^* \) is the result of the \( n \)th iteration. So the update on \( r_J \) can be written as:

\[ r_J^{n+1} = r_J^n + A_J^T (x_J^n - x_J^{n+1}) = r_J^n - \mu A_J^T g_J. \]  

(13)

We can see that this update only requires computation and storage of \( A_J^T x_J^{n+1}. \)

An important issue is that the step size derived in (11) minimizes \( ||r|| \) instead of \( ||r|| \) and that the step length is always positive. However, we are not interested in the reduction of \( r_J \) but in the reduction of \( r \). Our choice of \( \mu \) can thus potentially be too large to reduce \( r \). Furthermore, it is not guaranteed that our update direction \( g \) is always a descend direction. To stabilise our algorithm, we thus introduce an additional relaxation parameter \( \beta \) into the calculation of \( \mu \). This helps us to avoid overshooting the minimum if \( g \) is a descend direction whilst in cases in which \( g \) is not a descend direction, the increase in \( r \) remains small. The pseudo-code of the basic computation blocks is shown in Algo.3.

Algorithm 3 The algorithm for a basic iteration

| Initialization: select system matrix’s row index \( I \) and column index \( J \) |
| \[ g_J = (A_J^T)^T r_J \] |
| \[ \mu = \beta \frac{g_J^T g_J}{(g_J^T g_J)^2} \] |
| \[ x_J = x_J + \mu g_J \] |
| \[ z_J = A_J^T x_J \] |

It is straightforward to see that these computations can be computed in parallel over \( J \) since parameters in the update on \( x_J \) are independent of each other. Further analyses and improvements show that the parallel computation can be even performed over both subsets of \( J \) and \( I \).

The main trick in parallelizing the above code is to estimate the error \( r \). Ideally, after the parallel computations have updated subsections of the volume \( x \), we would need to compute a new error vector \( r \), which is required in the computation of subsequent gradient directions. However, this would require the computation of \( r_J^T + A_J^T (x_J^* - x_J^{n+1}) \). We instead use
a scheme that approximates $r$. This is done by calculating vectors $z_{I,n+1} = A_I^jx_{I,n+1}$.

Algo.4 shows a fully parallel computing scheme over both row and column blocks. Different sub-matrix block of matrix $A$ and sub-volume slice $x_J$ are assigned to calculating nodes simultaneously and the master node communicate with all computing nodes by averaging over the individual updates of subsets $x_J$. It is worth noting at this point that these algorithms do not require us to use all subsets $I$ and $J$ in each iteration, but also work if we randomly choose new subsets of these sets in each iteration.

Algorithm 4 CSGD method

Initialization: Partition row and column indices into sets $\{I_i\}$ ($i \in [1,m]$) and $\{J_j\}$ ($j \in [1,n]$), $x^0 = 0, z^0_I = 0$ for all blocks $I_i, J_j$ and $r = y$.

for epoch=1,2,... do
    $\hat{x} = 0$
    for $k = 1,2,...n$ do
        select $J_k$ as index $J$
        for $i = 1,2,...m$ do
            select $I_i$ as index $I$
            $g_{ij} = (A^T_{I_i}r_J)_{ij}$
            $\mu = \beta (g_{ij})^T(A^T_{I_i}A_{I_i})^{-1}g_{ij}$
            $\hat{x}_{ij} = x_{ij} + g_{ij} + \mu g_{ij}$
            $z_{I_i,j} = A^T_{I_i}(\hat{x}_{ij} + \mu g_{ij})$
        end for
        $r = y - \sum_{j} z_{J}$
    end for
    for all blocks $J$ that have been updated, $x_J = \hat{x}_J/(\text{number of times block } J \text{ has been updated})$.
end for

C. The importance sampling strategy

Based on the proposed algorithm, it is straightforward to develop a random sampling strategy that goes through all projection views and all detector sub-areas arbitrarily. Looking at Algo.4, we could randomly select subsets of $\{I_i\}$ and $\{J_j\}$, with each set being chosen with equal probability. Considering the sparsity of $A$ and taking inspiration from the randomized Kaczmarz method in [40], we instead develop an importance sampling strategy. Sets in $\{I_i\}$ and $\{J_j\}$ are selected with a probability that is proportional to the sparsity of the sub-matrix $A^T_{I_i}$. To estimate this sparsity without the need to construct the full matrix $A$, we instead compute the overlap between the detector area and the projection of the voxels labelled by $J$. Using the importance sampling strategy for $I$ and iterating over partial $J$, Algo.4 become Algo.5. It should be noted that although in Algo.4 and 5 the update on $r$ is performed as a whole, in effect, only elements which have been chosen for the latest iteration have been changed. As a result, we can also take the union of all selected $I$s and call this set $I_t$. The update of $r$ then becomes $r_{It} = y_{It} - \sum z_{I,t}$.

D. Domain decompositions in tomography

The above algorithms have been designed so that each computation is carried out on a generic subset $I$ (i.e. a subset of the observations) and a generic subset $J$ (i.e. a subset of the voxels) at any one time. For tomographic reconstruction, the question thus arises how to partition the observations and the reconstruction volume. Whilst generic partitions are possible, given the need to compute $P(I,J)$, it makes sense to partition the reconstruction volume and detector areas into blocks. We use a 3D cone beam CT geometry to demonstrate and similar arguments can be made for a parallel beam setup. The reconstruction volume and one pair of source/detector locations are shown in Fig.1a. For simplicity, the detector plane is always parallel to the line connecting point source and the geometry center of the detector plane.

We label each location of the point source $S$ and detector location with parameter $\theta$, whose trajectories do not have to be circular or helical. For each source/detector location $\theta$, we partition the detector into blocks. We also partition the reconstruction volume into rectangular cuboids, as shown in Fig.1b. For such a partition, given any one block of the reconstruction volume $J$, for each source location $\theta$, there might only be a part of the detector area that is involved in the update in CSGD. Thus, for a given $J$, the sub-matrices $A^T_{I_J}$ can have different levels of sparsity for different $I$. An illustration of this is shown in Fig.2, which shows that in

Algorithm 5 CSGD method with importance sampling

Initialization: Partition row and column indices into sets $\{I_i\}$ ($i \in [1,m]$) and $\{J_j\}$ ($j \in [1,n]$), $x^0 = 0, z^0_I = 0$ for all row blocks $I_i$ and $J_j$, $r = y$. $\gamma$ is the percentage of selected volume blocks in the total volume blocks. $\alpha$ is the percentage of selected row blocks in the total row blocks. Relaxation parameter $\beta$ is defined as $\beta = P_S / P_T$, where $P_S$ is the projection area of the selected sub-volume on the selected sub-area on current iteration and $P_T$ is the total projection area for the selected volume object on the whole detector plane under the whole scanning trajectory. For each pair of indices from $\{I_i\}$ and $\{J_j\}$ compute the probability $P(I,J) = P_S / P_T$ between the projection of the volume $x_J$ and the subset of all detector pixels indexed by $I$.

for epoch=1,2,... do
    $\hat{x} = 0$
    for $k = 1,2,...\gamma m$ do
        select a column block $J$ from $\{J_j\}$ randomly
        for $l = 1,2,...\gamma n$ do
            select a row block $I$ from $\{I_i\}$ without replacement with probability $P(I,J)$
            $g_{ij} = (A^T_{I_i}r_J)_{ij}$
            $\mu = \beta (g_{ij})^T(A^T_{I_i}A_{I_i})^{-1}g_{ij}$
            $\hat{x}_{ij} = x_{ij} + g_{ij} + \mu g_{ij}$
            $z_{I_i,j} = A^T_{I_i}(\hat{x}_{ij} + \mu g_{ij})$
        end for
        $r = y - \sum_{j} z_{J}$
    end for
    for all blocks $J$ that have been updated, $x_J = \hat{x}_J/(\text{number of times block } J \text{ has been updated})$.
end for
this fixed view, sub-area 4 on the detector does not receive rays passing through the selected sub-volume and so there is no need to select this area to update \( x_J \). What is more, the bulk of the volume block projection mainly lies in sub-areas 2 and 3 of the detector so that the corresponding \( A_{I_{\text{sub}}2} \) and \( A_{I_{\text{sub}}3} \) are much denser than \( A_{I_{\text{sub}1}} \). This suggests that the algorithm should select sub-area 2 and 3 more frequently than sub-area 1 and this is here achieved with our importance sampling strategy.

E. Group CSGD

The advantage of having dense sub-matrices leads to the following group version of our algorithms. The idea here is to dynamically build larger, dense sub-matrices \( A_j^I \) out of a large selection of smaller sub-matrices. In the previous CSGD method, one sub-block of the image combines only one sub-detector area for one projection view. In GCSGD, the sub-block combines a group of sub-areas for several projection views. It is straightforward that the GCSGD method uses more row information than the CSGD method. Let us demonstrate the idea by modifying Algo.5 into Algo.6.

F. Computational complexity

There are several important aspects when comparing computational efficiency of the methods. The methods are designed to allow parallel computation. We envisage this to be performed in a distributed network of computing nodes\(^1\). Most of these nodes will be used to perform the parallel computations. They produce two outputs,

1. \( \hat{x}_J^{(i)} = x_J^{(i)} + \mu g_J \)
2. \( z_J^I = A_j^I x_J + \mu g_J \)

These are then either sent to larger, but slow storage or directly to other nodes, where they are eventually used to compute

1. \( x_J = \text{mean}_I(\hat{x}_J^{(i)}) \)
2. \( r = y - \sum_j z_J^I \) or \( r_{II} = y_{II} - \sum_j z_{II}^I \),

which can be performed efficiently using message passing interface reduction methods.

The three main points that affect performance of the method are:

1. Computational complexity in terms of multiply add operations.
2. Data transfer requirements between data storage and a processing unit as well as between different processing units.
3. Data storage requirements, both in terms of fast access RAM and in slower access (e.g. disk based) data storage.

Each of these costs are dominated by different aspects:

1. Computational complexity is dominated by the computation of matrix vector products involving \( A_j^I \) and its

---

\(^1\)A serial version running on a single computing node where each computation is done independently, but one after the other, is also possible and this is how many of the simulations reported here were computed.
transpose, especially as \( A \) is not generally stored but might have to be re-computed every time it is needed. The computational complexity is thus \( O(|I| \times |J|) \), though computations performed on highly parallel architectures, such as modern GPUs, are able to perform millions of these computations in parallel.

2) Data transfer requirements are dominated by the need for each of the parallel computing nodes to need \( r_I \) and \( x_J \) as input and \( \hat{x}_J(i) \) and \( z_J^t \) as output. Note that the size of the required input and output vectors are the same, the data transfer requirement is thus \( O(|I| + |J|) \).

3) Central data storage requirements are dominated by the need to store the original data and the current estimate of \( x \). We also need to compute and store averages over \( \hat{x}_J(i) \) and \( z_J^t \). These computations can be performed efficiently using parallel data reduction techniques. Our approach would mean that each node would thus require \( O(|I| + |J|) \) local memory.

III. SIMULATIONS

Before introducing the simulation results, we first introduce three important parameters:

- \( \alpha \) is the percentage of selected row blocks (detector areas) for one column block (volume block).
- \( \beta \) is the relaxation parameter tuning the update on \( x_J \). It is expressed as \( \beta = b \times P_S / P_F \), as shown in Algo.5. In the following simulations we mainly change the value of \( b \) to tune the relaxation parameter \( \beta \).
- \( \gamma \) is the percentage of the selected column blocks (volume blocks) during one iteration. In our simulations, \( \gamma \) is usually set as 1, i.e. all column blocks are selected during one iteration.

In simulations, we used two evaluation criteria to evaluate the reconstruction quality: Signal to Noise ratio (SNR): \( 20 \times \log_{10} \| x_J^{true} \| / \| x_J^{true} - x_J^{est} \| \) and observation gap: \( 20 \times \log_{10} \| y \| / \| y - Ax_J^{est} \| \), where \( y \) are the projection data, \( x_J^{true} \) and \( x_J^{est} \) are the true phantom image vector and the reconstructed image vector respectively.

We explored the performance of Algo.6 on a range of tomographic reconstruction problems. We started with a simulated 2D phantom with \( 64 \times 64 \) pixels. Pixel sizes were normalised to be 1. The point source adopted a circular trajectory with radius of 115 and the rotation centre was located at the centre of the object. The linear detector had 187 pixels whose spacing was 1 and the detector was always perpendicular to the line connecting the point source and the geometric centre of the linear detector. The detector centre also followed a circular trajectory with the same radius as the source. The number of projections was 360 with the angular intervals being \( 1^\circ \). The object was partitioned into 4 parts (2 parts in both vertical and horizontal directions) and the detector area was partitioned into 2 parts by default. Scan geometry including the partition method as well as the original phantom are shown in Fig.3. We used the parallel computing toolbox (version 6.8) in Matlab R2016a to perform CSGD as described in Algo.5 and GCSGD in Algo.6. In our simulation, the term \( \text{epoch} \) refers to the outer iteration. The number of sub-matrices \( A_J^t \) that the algorithm used per epoch was proportional to the parameter \( \alpha \), where \( \alpha = 1 \) means that we used all sub-matrices of \( A \), whilst \( \alpha = 0.5 \) means that only half of the sub-matrices were used.

As the computational load is dominated by matrix vector products, when we compared the difference in convergence rate for methods using different \( \alpha \) (for example, \( \alpha = 1 \) and \( \alpha = 0.5 \)), we took account of the reduction in computational effort when using smaller \( \alpha \) by scaling the epoch count by multiplying it by \( \alpha \), a measure we call \( \text{effective epoch} \).

As mentioned before, CSGD and GCSGD methods can be applied in both using importance sampling and random sampling strategies. In our simulations, unless it is particularly mentioned, the sampling strategy always adopts the importance one.

We run CSGD and GCSGD algorithms for over one thousand iterations and show their average convergence and stability. However, in realistic applications we are typically only interested in the initial iterations. For example, the convergence performance within 50 effective epochs. This is because in real applications, due to the large data sets that have to be processed and due to the influence of noise and model mismatch, performing more iterations is typically not feasible and is often not helpful in obtaining better reconstruction results.

A. Influence of \( \beta \)

We started with an evaluation of the optimal choice of \( \beta \), which is crucial for the performance of the method. A simulation was conducted to show the difference in convergence rate when changing \( b \). Results are shown in Fig.4.

It can be seen that from \( b = 70 \) to \( b = 160 \), larger \( \beta \) increases the convergence rate. However, further increasing the value of \( b \) to 190 leads to a divergence of the algorithm, which suggests that there is a range for \( \beta \) that is guaranteed to converge. However, currently the justification for our choice of \( \beta \) remains empirical and more analysis of the algorithm is required to fully understand the optimal parameter.

B. Choice of \( \alpha \)

When we set \( \alpha = 1 \), the importance sampling strategy cannot show any advantage since we always use all data. To
Fig. 4. Different $\beta$ lead to different convergence rates when reconstructing the 2D image. $\alpha$ and group size are both set as 1. All 4 volume sub-blocks are selected in each epoch.

demonstrate the difference, we set the group size to 1, $b = 100$ and set $\alpha$ to 0.8, 0.5 and 0.2 respectively. Fig.5 shows that setting $\alpha$ smaller than 1 increases the convergence rate of the CSGD. Furthermore, different values of $\alpha$ always lead to the same precision.

Considering that the smaller $\alpha$ is, the more actual epochs are required to achieve the same number of effective epochs and that MATLAB is not efficient in performing loops, in following simulations, we chose a moderate parameter $\alpha = 0.5$ to test other qualities of CSGD and GCSGD. For example, we test the influence of group size of the GCSGD. Simulation results are shown in Fig.6.

Fig. 5. When $b = 100$, groups size is 1 and all 4 volume blocks are selected for each epoch, setting $\alpha$ smaller than 1 is guaranteed to achieve high precision level and is of help to increase the convergence rate.

It can be seen that the GCSGD with importance sampling strategy all converges to a high precision level regard less of the group size. Generally speaking, setting the group size as large as possible helps to increase the initial convergence rate. This is reasonable since more row information is used for each epoch when the group size is enlarged, making the iteration more likely to move towards the global downhill direction. Furthermore, we empirically observed that a larger group size requires smaller parameter $b$ to maintain convergence. This is because when the group size increases, the ratio $P_S/P_T$ also increases and thus a smaller $b$ is required, otherwise it is easy for the step to overshoot, which makes the algorithm fail to converge. In practice, however, for parallel computing architectures, we suggest that the group size should be designed depending on the storage ability of each computing node. To show differences when using different group sizes and different $\alpha$s, some reconstruction results are presented in Fig.7.

C. Different partitions

The above simulations show the effectiveness of the CSGD and GCSGD method with importance sampling. In following simulations, we explored the influence of different partition methods. For example, the reconstructed image was still divided into 4 square blocks and the detector area was partitioned into 2, 4, 8 and 16 sets respectively. We compared the difference between group size being 1 and 100 to further verify that a larger group helps to increase the convergence rate. The simulation results are shown in Fig.8.

When the group size is only one, we can see that dividing the detector into 8 and 16 areas leads to rather slow convergence rates. It suggests that when the group size is small, the detector cannot be divided into more sub-areas. For a group
size of 100, however, the algorithm allows for a division of the detector into more areas and the final precision level is not affected. In the initial iterations, we can see that different detector partitions have nearly the same convergence rate in the GCSGD method. When the precision level is high, the detector partitions have nearly the same convergence rate when using GCSGD, especially when the group size is large. It is straightforward to see that the calculation amount of $A_f$ for a selected sub-volume and sub-detector area decreases when the number of partitions on the detector increases. As a result, when addressing the partition strategy for large scale tomography problems, a more flexible partition method is allowed for large group size situation. If the big group sizes poses a heavy storage burden for work nodes, a finer partition method would be of help to reduce the computation amount for $A_f$. In other words, although dividing the detector area into more sub-areas cannot provide a higher precision level, it helps to reduce the computation load when only a few iterations are performed.

Another variable is the number of volume blocks. To investigate this, we partitioned the 2D image into 4,16,64,256 areas respectively (divide each dimension into 2,4,8,16 parts evenly). The detector area was always partitioned into 2 sub-areas, as shown in Fig.3. The group size was set to 100 and the corresponding $b$ was 2,1,0.5,0.25 respectively to ensure convergence. The simulation results are shown in Fig.9. It can be seen that all partition methods show similar final precision level, and in the initial iterations, the differences between different volume block numbers are also negligible, demonstrating the good scalability of this algorithm. Since a finer partition allows more computation nodes to be used and the computation load of each node is reduced, the similar convergence in the initial iterations is useful when addressing large scale data where only a few iterations are performed.

We also explored the situation when not all of $J$ were selected. The group size was set to 100 and the number of detector sub-areas was set to 2 by default. We set $\gamma$ as different values when the image were partitioned into 16 and 64 sub-areas, simulation results are shown in Fig.10.

It can be seen that GCSGD method converges for every $\gamma$, and their final precision level is similar. Also, as long as the $\gamma$ is not too small ($\gamma = 0.2$), then the initial convergence...
rates for different $\gamma$ are almost identical. This suggests that randomly leaving a small portion of the image blocks not to be updated does not influence the initial convergence rate and final precision level. This property will be useful when we perform the algorithm in parallel computing architecture where the algorithm maybe run asynchronously.

D. Discussions on importance sampling strategy

In this part we compared the importance sampling and random sampling strategy based on the previous simulation geometry. Since the convergence property has been verified in previous simulations, in this part, we were only interested in the initial convergence rate and thus we only perform the algorithm (with importance sampling or random sampling strategy) for 50 effective epochs but repeat it 10 times for the same data and then calculate the average of SNR and observation gap.

The simulations not only show the drawbacks and advantages of the importance sampling strategy, but also explain why the flat area appears when the precision level is high and group size is large, and why using importance sampling can lead to images artefacts.

We first verified that the importance sampling strategy indeed helps to increase the convergence rate compared with random sampling. However, it only happens when the group size is small. In simulation, we applied two sampling strategies to sample the sub-detector areas and calculate the averaged SNR for each iteration. Two dataset $SNR_{Imp}$ and $SNR_{Ran}$ were obtained, where the former one was the SNR through importance sampling and the latter one was obtained through random sampling. The difference $SNR_{Imp} - SNR_{Ran}$ is shown in the Fig.11. The reason why importance sampling cannot outperform random one when group size is 100 is due to the non-uniform update on $z_{I2}^j$. For each volume slice, the selection criteria for choosing row blocks (or sub-detector areas from different projection views) is based on the projection area, which reflect the sub-matrix’s density. The difference between projection areas can sometimes be huge. For example, as shown in Fig.12, for volume slice $J_1$, the projection area on the sub-detector 2 is far larger than that on the sub-detector 1. That means when updating $J_1$ and selecting sub-detector areas, or row blocks, it is more likely that sub-detector 2 is chosen. This property makes the update of $z_{I2}^{J_1}$ be $A_{I2}^{J_1} x_{J1}$ very frequent and the update on $z_{I1}^{J_1}$ be $A_{I1}^{J_1} x_{J1}$ rather infrequent. For residue terms $r_{I1}$ and $r_{I2}$, the contribution from $z_{I2}^{J_1}$ and $z_{I2}^{J_1}$ is different, especially in the initial iterations. Since the density of $A_{I2}^{J_1}$ is higher than $A_{I1}^{J_1}$, this means the influence of $z_{I2}^{J_1}$ on $r_{I2}$ is more significant than $z_{I1}^{J_1}$ on $r_{I1}$. At the
initial iterations, the frequent updates of $z_{I_1}$ make $r_{I_1}$ decrease efficiently whereas the infrequent updated of $z_{I_2}$ has little influence on the update of $r_{I_1}$. (It should be remembered that at the same time $r_{I_1}$ is also updated by computations with the other volume slices.) This property helps the importance sampling strategy to obtain a fast initial convergence rate but when the precision level is already high, the update on $r_{I_1}$ then needs the contribution from $z_{I_2}$ and unless this term is updated frequently, then $r_{I_1}$ cannot be further reduced. This explains why at the initial stage the importance sampling strategy is faster than random sampling and when the iteration goes on, there are some flat areas in Fig.8, Fig.9, Fig.10. The reason for the artefacts in the reconstructed images is similar: the shaded area in Fig.12 is rarely used to update $J_1$, thus allowing the reconstruction error of the inner margin of $J_1$, (the same holds in the other volume slices) a little higher than the other parts. The reason why the importance sampling strategy works and the flat areas disappear when the group size is small (for example, group size is set as 1) is because that in this situation, the total convergence rate is slow and when the precision level reaches a high level, then the update on $z_{I_1}$ is already frequent enough to allow an effective change on $r_{I_1}$.

We proposed some methods to overcome the disadvantages of importance sampling. One method would be to allow the algorithm to make a choice before sampling the row blocks, deciding whether to use importance or random sampling in the current iteration. This choice can be done with a specific probability. Another method would define a threshold in the probability distribution, enforcing it to be above the threshold. However, simulation results show that these two methods are not very effective in terms of improving the performance of importance sampling strategy. The third method, we call a mixed sampling strategy, is to gradually change the probability distribution along with the iterations. By reducing the gap between different probability values gradually, we experimentally showed that this method can maintain the fast initial convergence due to the importance sampling and eliminate the artefacts in the reconstructed images and the flat area in the convergence line. Specifically, in Fig.12, we defined the projection area of $J_1$ on sub-detector area 1 as $P_1$ and that on sub-detector area 2 as $P_2$. In the previous importance sampling strategy, the probability for choosing sub-detector area 1 is $P_1/P_T$, where $P_T$ is the total projection area of sub-volume $J_1$. Similarly, the probability for choosing sub-detector 2 is $P_2/P_T$. The mixed sampling method gradually fills the gap between $P_1/P_T$ and $P_2/P_T$. The probabilities to choose sub-detector 1 and 2 are set as $[P_1 + \theta(P_{\text{max}} - P_1)]/P_T$ and $[P_2 + \theta(P_{\text{max}} - P_2)]/P_T$ respectively, where $P_{\text{max}}$ is the largest value of $P_1$ and $P_2$. The parameter $\theta$ is used to reduce the difference between two possibilities by gradually increasing it from 0 to 1. We make a comparison with importance sampling, random sampling and the mixed sampling method when the group size is 100 and $\alpha = 0.5$. The convergence rate is shown in Fig.13.

We also present the reconstructed image after 20 effective epochs for group size being 1, 5, 100 respectively. The simulation condition is the same as Fig.7. For simplicity, we only simulated the $\alpha = 0.5$ situations. The simulation results are shown in Fig.14.

The mixed sampling method can also eliminate the flat area in the convergence trend. For example, we here present the simulation results of different detector partition methods, as shown in Fig.15. Compared with Fig.8, the convergence trend is more smooth and it requires less iterations to achieve the...
Fig. 13. Comparisons for different sampling strategies. The θ in mixed sampling method starts from 0 and the increment step length is 1/40, it means that after 40 epochs the probability distributions on different sub-detector areas becomes the same within one projection. The mixed sampling method maintains the same convergence rate at the initial stage as importance sampling does, and this increment trend is maintained along with the iteration going on.

Fig. 14. Simulation results after 20 effective epochs when use mixed sampling strategy. (a) group size=1, α = 0.5, b = 100; (b) group size=5, α = 0.5, b = 25; (c) group size=100, α = 0.5, b = 2. Compared with Fig.7, the reconstructed images for group size 5 and 100 are free from the influence of artefacts while keeping the similar SNRs.

Fig. 15. Using the mix sampling strategy eliminate the flat area, thus shorten the iteration numbers needed to achieve the precision limit.

Fig. 16. Basic 3D simulation settings. (a) shows that the location of the point source is \( x = r \sin a \cos b, y = r \sin a \sin b, z = r \cos a \), where \( a \) and \( b \) are randomly changed. The connection line of point source and the centre of the detector plane is always perpendicular to the detector plane. (b) is a slice of the 3D volume.

precision limit.

E. 3D simulation

To demonstrate the performance in a 3D setting, we defined a 3D phantom of 128*128*128 voxels describing a 32*32*32 volume. The CCD detector plane was a square with side length 101 whose detector spacing was 0.5. To demonstrate how our method can be used for nonstandard trajectories, we chose a randomised scanning path with 720 projection views, i.e. the point source location was randomly changed with a fixed radius \( r \) of 66 but with a random rotation. The rotation center was placed at the centre of the 3D object. The detector was always perpendicular to the line connecting the geometric centre of the detector and the point source, and the vertical distance between point source and detector plane was 132. The illustration of the scanning geometry and a slice of the 3D volume is shown in Fig.16. This simulation was performed on a computer with 48 Intel Xeon CPU cores and 256GB RAMs.

In this simulation, we did not divide the CCD area into sub-areas but simply treated each projection as a whole. We divided the volume into 2 sections for each dimension, thus we had 8 blocks in total. We set the group size to 90. \( b \) in \( \beta \) is set to 1, \( \alpha = 0.5 \) and \( \gamma = 1 \). Reconstruction results after different epochs are shown in Fig.17.

IV. CONCLUSION

Here we propose a parallel algorithm CSGD that is suitable for linear inverse problems \( y = Ax + b \) and enables additional flexibility in the way we can partition data sets in algebraic CT image reconstruction. As far as we know, CSGD is the first algorithm that combines row action methods and column action methods. Unlike other algorithms that demand the slice strategy of the reconstructed volume to be perpendicular to the rotation axial of the object to reduce data sharing between
different computing nodes and a circular or helical scanning strategy, CSGD does not pose any restrictions on how to partition the reconstruction volume. It is thus applicable to generic scan trajectories. Furthermore, the CSGD method can be implemented using a server parameter parallel computing architecture. It thus has the potential to solve very large image reconstruction problems by using several computing nodes. The GCSGD method better utilises row information of the system matrix $A$ and thus has faster convergence rates than CSGD. We have furthermore developed an importance sampling strategy. By non-uniformly selecting row blocks of $A$ based on the projection area of the associated sub-volume, the sub-matrices of $A$ with relatively higher density get chosen more often than those with higher sparsity, which increases the initial convergence rates. A mixed sampling strategy can be used to overcome the artefacts brought by importance sampling while keeping the fast convergence rate. We here present 2D simulation results which demonstrate the convergence properties of the proposed algorithms. 3D reconstruction results further verify the effectiveness of GCSGD in term of presenting visually acceptable reconstruction results.

V. OPEN AREAS

Simulation results have shown that when the group size is fixed, there is a range for parameter $b$ (or $\beta$) that guarantee the convergence of GCSGD. However, currently the choice on $b$ is based on experience and trials, thus we want to develop a more systematic method, or an adaptive self-correction scheme for the determination of parameter $b$ or $\beta$. Besides, in more realistic tomographic imaging settings, with noisy observations and an inaccurate system matrix, simply using current method is often difficult to obtain high quality reconstructions. As a result, incorporating regularisation terms into GCSGD is another important direction for our future work. We are also interested in applying the proposed algorithm in reality dataset under a distributed network and explore the properties and differences of synchronous and asynchronous communication strategies. In our paper we have tested the convergence property when $\gamma < 1$, which can be viewed as a form of asynchronous communication in a parallel computing architecture. More systematic investigations of this will be included in future work.

ACKNOWLEDGMENT

The authors would like to thank Professor Paul White who gave us positive feedbacks on the project. We also thank Dr Jordan Cheer who suggested us to improve our description on the simulation results part.

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. Radiol., 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] 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.
[10] J. Hsieh, B. Nett, and Z. Yu, “Recent advances in CT image reconstruction,” Eur. J. Radiol., vol. 1, no. 1, pp. 39–51, Mar. 2013.
[11] J. Bilbao-Castro, J. Carazo, J. Fernández, and I. García, “Performance of parallel 3D iterative reconstruction algorithms,” WSEAS Trans. on Bio. and Biomed., vol. 1, no. 1, pp. 112–119, Jan. 2004.
[12] Y. Censor, “Row-action methods for huge and sparse systems and their applications,” SIAM Rev., vol. 23, no. 4, pp. 444–466, Jun. 1981.
[13] D. W. Watt, “Column-relaxed algebraic reconstruction algorithm for tomography with noisy data,” Appl. Opt., vol. 33, no. 20, pp. 4420–4427, Sep. 1994.
[14] T. Elfving, “Block-iterative methods for consistent and inconsistent linear equations,” Numer. Math., vol. 35, no. 1, pp. 1–12, Mar. 1980.
[15] J. Kole and F. Beekman, “Parallel statistical image reconstruction for cone-beam x-ray CT on a shared memory computation platform,” Phys. Med. Biol., vol. 50, no. 6, pp. 1265–1272, Mar. 2005.
[16] 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.
[17] Y. Censor, P. P. Eggermont, and D. Gordon, “Strong underrelaxation in kaczmarz’s method for inconsistent systems,” Numer. Math., vol. 41, no. 1, pp. 83–92, Feb. 1983.
[18] A. H. Andersen and A. C. Kak, “Simultaneous algebraic reconstruction technique (SART): a superior implementation of the ART algorithm,” *Ultra. Imag.*, vol. 6, no. 1, pp. 81–94, Jan. 1984.

[19] Y. Censor, “Parallel application of block-iterative methods in medical imaging and radiation therapy,” *Math. Program.*, vol. 42, no. 1, pp. 307–325, Apr. 1988.

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

[21] N. D. Tang, N. De Ruiter, J. Mohr, and A. P. Butler, “Using algebraic reconstruction in computed tomography,” in *27th Conf. IVCNZ, Dunedin, New Zealand*, pp. 216–221, ACM, Nov. 2012.

[22] T. M. Benson and J. Gregor, “Framework for iterative cone-beam micro-CT reconstruction,” *IEEE Trans. Nucl. Sci.*, vol. 52, no. 5, pp. 1335–1340, Oct. 2005.

[23] H. M. Hudson and R. S. Larkin, “Accelerated image reconstruction using ordered subsets of projection data,” *IEEE Trans. Med. Imag.*, vol. 13, no. 4, pp. 601–609, Dec. 1994.

[24] 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.

[25] Y. Censor and T. Elfving, “Block-iterative algorithms with diagonally scaled oblique projections for the linear feasibility problem,” *SIAM. J. Matrix Anal. and Appl.*, vol. 24, no. 1, pp. 40–58, Jul. 2002.

[26] 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.

[27] C. A. Bouman and K. Sauer, “A unified approach to statistical tomography using coordinate descent optimization,” *IEEE Trans. Image Process.*, vol. 5, no. 3, pp. 480–492, Mar. 1996.

[28] Z. Yu, J. B. Thibault, C. A. Bouman, K. D. Sauer, and J. Hsieh, “Non-homogeneous updates for the iterative coordinate descent algorithm,” in *Electronic Imaging 2007, San Jose, CA, United States*, pp. 64981B1–64981B12, International Society for Optics and Photonics, Feb. 2007.

[29] 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.

[30] 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.

[31] 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.

[32] T. Elfving, P. C. Hansen, and T. Nikazad, “Convergence analysis for column-action methods in image reconstruction,” *Numer. Algorithms*, vol. 74, no. 3, pp. 905–924, Mar. 2017.

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

[34] M. Soleimani and T. Pengpen, “Introduction: a brief overview of iterative algorithms in x-ray computed tomography,” May. 2015.

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

[36] 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.

[37] 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.

[38] Z. Shen, H. Qian, C. Zhang, and T. Zhou, “Accelerated variance reduced block coordinate descent,” arXiv preprint arXiv:1611.04149, Nov. 2016.

[39] Z. Qu and P. Richtárik, “Coordinate descent with arbitrary sampling I: Algorithms and complexity,” *Optim. Method. Softw.*, vol. 31, no. 5, pp. 829–857, May. 2016.

[40] T. Strohmer and R. Vershynin, “A randomized kaczmarz algorithm with exponential convergence,” *J. Fourier Anal. Appl.*, vol. 15, no. 2, pp. 262–278, Apr. 2009.