# Distributed Randomized Kaczmarz and Applications to Seismic Imaging in Sensor Network

Goutham Kamath Paritosh Ramanan Wen-Zhan Song
Department of Computer Science
Georgia State University
{gkamath1,pramanan1}@student.gsu.edu, wsong@gsu.edu

***Abstract*—Many real-world wireless sensor network applications such as environmental monitoring, structural health monitoring, and smart grid can be formulated as a least-squares problem. In distributed Cyber-Physical System (CPS), each sensor node observes partial phenomena due to spatial and temporal restriction and is able to form only partial rows of least-squares. Traditionally, these partial measurements were gathered at a centralized location. However, with the increase in sensors and their measurements, aggregation is becoming challenging and infeasible. In this paper, we propose distributed randomized kaczmarz that performs in-network computation to solve least-squares over the network by avoiding costly communication. As a case study, we present a volcano monitoring application on a distributed CORE emulator and use real data from Mt. St. Helens to evaluate our proposed method.**

***Keywords*—Distributed Computing, Gossip Methods, Least-squares, Iterative Methods, Mesh Network, Randomized Kaczmarz**

## I. Introduction

Advancement in wireless sensor network (WSN) technology has enabled us to deploy large number of small, low cost sensors that can sense, actuate and communicate information in various fields such as environmental monitoring [22], structural health monitoring [17] and smart grids [19]. These real-world applications involve parameter estimation and can be formulated as a least-squares problem. In distributed Cyber-Physical System (CPS), each sensor node observes partial phenomena due to spatial and temporal restriction and is able to form only partial rows of least-squares Eq. (1). Traditionally, these partial measurements were gathered at a centralized location, however, with the increase in sensors and their measurements, aggregation is becoming challenging and in some cases infeasible. For instance in volcano monitoring, a dense network of stations are used to record seismic vibrations and obtain high-resolution images of volcano conduit [15]. However, data recorded by these stations are being manually gathered due to its high fidelity and bandwidth limitations [28]. In other CPS such as smart grid, sharing of sensor measurements are restricted due to privacy and security concerns. These constraints demand distributed algorithms that can run on a loosely coupled system such as WSN.

Sensors attached to a CPS are now capable of sampling high volume, large dimensional data in real time. Tiny but powerful computational units (e.g. Beaglebone Black, Arduino) can now perform few pre-processing steps such as cleaning of raw data or discretization of continuous events. For instance, in volcano monitoring system, a seismic vibration sensor (geophone) sample in a range of 16-24 bit at 50-200 Hz generating a high volume of raw data [28]. Pre-processing steps such as detecting an earthquake, finding its origin time and location can be performed in-situ [20]. Using these pre-processing steps, discretization of raw data leads us to the formation of a partial least squares problem at each node [26]. The dimension of such problems is typically in the range of hundreds of thousands of equations and unknowns. Interestingly, in case of volcano monitoring with the increase in the earthquakes, the partial least squares problem increases its row size, making the system bigger.

Mathematically, these problems can be modeled as a linear least squares problem and in this paper, we consider solving,

$$x_{LS} = \arg\min_x \frac{1}{2}\|Ax - b\|_2^2 \tag{1}$$

for $x \in \mathbb{R}^n$ in a distributed way, where $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$. In particular, we assume that $A$ and $b$ are distributed row-wise over the $P$ nodes of a network, i.e $A = \{A_1, \cdots, A_P\}$ and $b = \{b_1, \cdots, b_P\}$, where $A_i \in \mathbb{R}^{m_i\times n}, b_i \in \mathbb{R}^{m_i}$ form a subsystem at the $i^{th}$ node. These subsystems are assumed to be huge and therefore, we are interested in solving a large scale problem over a loosely connected, decentralized network (WSN), where each node holds part of the input data. For such large sparse system of equations, iterative methods, especially randomized methods are becoming popular due to its simplicity and performance.

Recently, methods like stochastic gradient descent (SGD), randomized coordinate descent (RCD) have received renewed attention for confronting very large scale problems, especially in the context of machine learning (ML) [3]. These methods are based on the computation of partial gradients involving random sampling of a subset of the entire system and have been proven to be better for noisy data due to random sampling. Randomized Kaczmarz (RK) is a type of SGD which has been recently popular for its exponential convergence rate with appropriate choice of rows for projection [30]. RK processes single row at a time and requires only $\mathcal{O}(n)$ storage. For a extremely large sparse system of linear equations, RK is even more efficient than the conjugate gradient method [30].

In this paper, we first propose a parallel randomized kaczmarz (Par-RK) approach that uses fusion center to merge intermediate partial least square solution $x$. We extend this algorithm by eliminating fusion center to make it distributed randomized kaczmarz (D-RK) which limits the communication only to the immediate neighbor. In this paper, we use the term

*parallel* for algorithms developed primarily using a central coordinator or a fusion center, whereas *distributed* implies algorithms in which communication in each node is limited to its immediate neighbor without using any multi-hop communication. In this paper, we provide theoretical convergence of these algorithms and apply this algorithm to a real world problem of volcano monitoring. We evaluate its performance using synthetic and real data traces implemented on the CORE network emulator [1].

The rest of the paper is organized as follows. Section II presents related work on iterative methods to solve least squares on sensor networks. Section III describes the proposed algorithms and its convergence result. In Section IV, we present seismic tomography on sensor network as an application case study and evaluate the performance of our proposed methods. In Section V we carry our experiments using real seismic data traces from Mt. St. Helens, WA, USA. Discussion and future work is presented in Section VI. We conclude the paper in Section VII.

## II. Related Work

Parallel least squares solvers for sensor network are restricted to only those that collect only partial solution $x$ from all the nodes and perform low complexity calculation (merging) and broadcasts them. Methods that involve transferring large data set from all the nodes are not discussed here as it is not possible in few decentralized application involving big data. The distributed multi-splitting is one of the popular methods that requires fusion center and was originally proposed by Renaut [25]. This method partitions the system into columns instead of rows, letting each processor apply a well-known fixed point iteration methods such as Jacobi, Gauss-Seidel and successive over-relaxation to the normal equation. Since it is a column splitting method, we will not be considering it in this paper.

In order to handle large scale data, many block parallel iterative methods suitable for parallel computing have been developed recently [7]. Among them Component Averaging (CAV) [6] and Diagonally-Relaxed Orthogonal Projection (DROP) [5] methods can be used on sensor networks [15] using fusion center. CAV [6] is a Cimmino-type method that projects current iterates simultaneously onto all the systems' hyperplanes. In this method, they adopt diagonal weighting of the linear equation based on its sparsity rather than fixed weights used in Cimmino, exhibiting faster numerical convergence. CAV retains the desired convergence properties of the Cimmino's method, in the sense that it converges in the inconsistent case. Par-RK, which uses fusion center, differs from these methods as it uses randomized kaczmarz to accelerate the convergence [30].

Truly distributed least squares solver proposed by Zhou et al. [33] is robust against node failures. The algorithm is designed for a single variable case, and higher dimension is not considered. Recently, distributed gradient descent algorithms are being developed to solve Eq. (1) [12]. These methods are designed to be truly distributed and avoid data fusion center or long-distance communication, offering better load balance to the network. These methods use a diminishing step size to ensure convergence and sometimes are very sensitive to the choice of step-size. In this paper, we compare D-RK, with popular distributed gradient descent methods such as EXTRA [27] and DGD [32].

Authors in [24] have studied a distributed least squares solver based on distributed QR factorization. This least square solver first computes the local solution using distributed QR, which in turn uses gossip-based distributed modified Gram-Schmidt method described in [29]. This method uses distributed matrix-vector multiplication that can be expensive for a large sparse matrix. The existing push-sum methods to solve least squares problem are designed using the direct approach as opposed to iterative methods. To our best knowledge of literature, our work is the first attempt to use push sum based methods with randomized kaczmarz to solve least squares problem in wireless sensor network.

## III. Algorithm Design

In this section, we use $\mathcal{G}(\mathcal{V},\mathcal{E})$ to denote undirected connected graph with node (sensor) set $\mathcal{V} = \{1, \cdots, P\}$ and edge set $\mathcal{E}$, where each edge $\{i,j\} \in \mathcal{E}$ is unordered pair of distinct node. Node $i$ carries out communication only with its neighbors $\mathcal{N}_i = (j|\{i,j\}) \in \mathcal{E}$. Let $x \in \mathbb{R}^n$ be column vector and $x^k_{(i)}$ be the partial solution obtained at node $i$ after $k^{th}$ iteration for every $i \in \mathcal{V}$. Also, let $x_j$ denote the $j^{th}$ component of $x$. From Eq. (1) let $F(x) = \frac{1}{2}\|Ax-b\|_2^2$ and according to the gradient descent method the optimal solution $x_*$ is obtained by traversing towards $-\nabla F(x^k)$ at every iteration $k = \{1, 2, \cdots\}$ by certain step size $\rho_k$ starting from initial value $x_0$. In other words, $(k+1)^{th}$ iteration is given by, $x^{k+1} = x^k - \rho_k \nabla F(x_k)$.

Decentralized version of this problem recently developed by [12], [27], aims to minimize each nodes objective objective function $F_i(x)$ independent of other nodes. At every $k^{th}$ iteration $\nabla F_i(x^k)$ requires the computation of $A_i^T(A_i x^k - b_i)$. Matrix multiplication becomes an issue for a large data set especially on a sensor with a very limited memory footprint. To avoid this, we use only partial gradients that involve sparse vector multiplication such as stochastic gradient descent (SGD). Recently, methods like SGD are becoming popular in ML communities, and it involves random sampling to compute the gradient of a subsystem instead of an entire system. This method has been proven to be better for noisy data due to random sampling.

Randomized Kacmarz (RK) is a type of SGD commonly used to compute Eq. (1). This method starts with an arbitrary initial vector $x^0$ and at every iteration $k$, it randomly selects a row $i(k) \in \{1, \cdots .m\}$ of the linear system (with probability of choosing row $i$ is $\frac{\|a_i\|_2^2}{\|A\|_F^2}$, where $\|\|_F$ denotes the frobenius norm). Next, it performs an orthogonal projection of the current estimate vector onto the hyperplane $a_{i_{(j)}}^T x_i = b_{i_{(j)}}$ as shown in Algorithm 1.

**Algorithm 1** Randomized Kaczmarz Algorithm

1: **for** $k \leftarrow 0$ until convergence or max iteration **do**
2: Pick $i_{(k)} \in \{1, \cdots .m\}$ with probability $p_i = \frac{\|a_i\|_2^2}{\|A\|_F^2}$
3: $x^{(k+1)} = x^{(k)} + \rho_{i_{(k)}} \frac{b_{i_{(k)}} - \langle a_{i_{(k)}}, x^{(k)} \rangle}{\|a_{i_{(k)}}\|^2} a_{i_{(k)}}$
4: **end**

Recently, Strohmer and Vershynin [30] proved the following exponential bound on the expected rate of convergence of RK given by, $\mathbb{E}\|x_k - x\|_2^2 \leq (1 - \frac{1}{R})^k \|x_0 - x\|_2^2$ where $R = \|A^{-1}\|^2 \|A\|_F^2$, $x_0$ is an arbitrary initial value, while $\mathbb{E}$ denotes the expectation (over the choice of rows). Needell [23] studied the convergence of RK for the inconsistent system. RK and it's variants are inherently sequential and assume the availability of entire matrix $A$ and vector $b$ at a central location. This method cannot be directly applied to a loosely coupled system such as WSN and for this reason we first develop a parallel version of RK using fusion center.

Now, for $\mathcal{V} = \{1, \cdots, P\}$ number of nodes deployed for monitoring, each node receives some partial row information of $A, b$ i.e,

$$Ax = b \tag{2}$$

where,

$$\mathbb{A} = \begin{pmatrix} A_1 \\ A_2 \\ \vdots \\ A_P \end{pmatrix}; = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_P \end{pmatrix}; A_i \in \mathbb{R}^{m_i \times n}; b_i \in \mathbb{R}^{m_i}$$

where, $x \in \mathbb{R}^n$ is the unknown vector. We assume that, $\{A_i, b_i\}$ forms subsystem in each sensor node. This subsystem has non-zero columns and such a column if exists can be removed as a preliminary step as it corresponds to coefficient of fictitious variable, whose values can be arbitrary. Also, $\|a_j\|^2 > 0$ i.e $j^{th}$ row $1 \leq j \leq m_i$ of $A_i$ is non-zero.

From the above formulation, each node has only a partial information of the whole equation. Therefore, Par-RK starts with some arbitrary initial guess and performs three main steps until convergence. Step 1: Perform RK at each node $i \in \mathcal{V}$ in parallel for fixed $k_i > 0$ iterations. Step 2: Shared component of unknown variable $x_{(i)}$ between different nodes are merged by taking a component-wise average at a fusion center. Step 3: The merged vector is used as a new estimate for next iteration. This approach can be formalized as follows.

For each $1 \leq j \leq n$, denoted by $I_j$ the index set of blocks which contain an equation with a non-zero coefficient of $x_j$ ($I_j = \{1 \leq q \leq P | x_j\}$ has a non-zero coefficient) in some equation $A_t, b_t$. Let $s_j = |I_j|$ and denotes $s = \sum_{j=1}^{n} s_j$. Next, we provide the definition for merge operation.

*Definition 1:* Let $A = \{A_1, A_2, \cdots, A_P\}$ and $b = \{b_1, b_2, \cdots, b_P\}$ be partitioned into $P$ blocks. The component average relative to $\{A, b\}$ is mapping $CA_{A,b} : (\mathbb{R}^n)^P \to (\mathbb{R}^n)$, defined as follows: let $\{x_1, \cdots, x_P\} \in \mathbb{R}^n$. Then $CA_{A,b}(x^1, \cdots, x^P)$ is the point in $\mathbb{R}^n$ whose $j^{th}$ component is given by $CA_{A,b}(x_1, \cdots, x_P)_j = \frac{1}{S_j} \sum_{\ell=1}^{P} x_j^\ell$, where $x_j^\ell$ is the $j^{th}$ component of $x^\ell$, for $1 \leq \ell \leq P$.

Given $\{A_i, b_i\}$, relaxation parameter $\rho_i$ for $i \in \{1, \cdots, P\}$ and from the Definition (1) Par-RK Algorithm 2 is as follows. Step 6 of the Algorithm 2 is a merging operation and is carried out in a fusion center. After every local iteration at each node, partial solution $y^\ell$ from all the nodes are sent to a fusion center using data aggregation scheme. At a fusion center, these partial solutions are merged according to the Definition (1).

**Algorithm 2** Par-RK Algorithm

1: **set** $x^0 \in \mathbb{R}^n$ to an arbitrary value.
2: **for** $k \leftarrow 0$ until convergence or max iteration **do**
3: **for** each $1 \leq \ell \leq P$ in parallel **do**
4: $y^\ell$ =RK$(A^\ell, b^\ell, x^k, \rho_\ell)$
5: **end**
6: $x^{(k+1)} = CA_{A,b}(y^1, \cdots, y^P)$
7: **end**

The merged solution is again disseminated to the network, and all the nodes use this as the initial value to carry out further iterations.

### A. Convergence Analysis

To prove the convergence of Par-RK we first transform the system given in Eq. (2) into system of equations in some superspace $\mathbb{R}^s$ of $\mathbb{R}^n$. Let $B_t = \{A_t, b_t\}$, be a tuple containing known terms of subsystem and $x_t \in \mathbb{R}^n$ be the partial solution of subsystem $t$ for $1 \leq t \leq P$. We know that, $x_t$ can share some common variable with $x_{t'}$ if $A_t$ and $A_{t'}$ has common non-zero column. Now without loss of generality, for $1 \leq r \leq n$ the components $\{x_1, \cdots, x_r\}$ be exactly share with two or more nodes i.e $s_1, \cdots, s_r \geq 2$, while $s_{r+1}, \cdots, s_n = 1$. From this we have $n - r$ components of $x$ not shared by any blocks and can be computed without needing any data exchange.

Now, from the Definition (1) we have an expansion mapping $E : \mathbb{R}^n \to \mathbb{R}^s$:

$$\begin{aligned} &E(x_1, \cdots, x_n) = \\ &(y_{1,1}, \cdots, y_{1,s_1}, \cdots, y_{r,1}, \cdots, y_{r,s_r}, y_{r+1}, \cdots, y_n), \end{aligned} \tag{3}$$

where $y_{j,1} = \cdots = y_{j,s_j} = x_j$ for $1 \leq j \leq r$ and $y_j = x_j$ for $r < j \leq n$.

Similarly, we can transform the equation of the subsystem $B_t$ from $\mathbb{R}^n$ to $\mathbb{R}^s$ which we will denote it as $B'_t = \{A'_t, b'_t\}$. The new transformed equation in $B'_t$ do not share any common variable with any other blocks. Let, $B' = \cup_{t=1}^{P} B'_t$ represent all the subsystem stacked together. Now, parallel execution RK on each node $B_t$ for $1 \leq t \leq P$ is equivalent to performing RK on $B'$. Next, we will show that averaging the shared variable of the system $B$, is equivalent to certain row projections. From this it follows that Par-RK is just RK in $\mathbb{R}^s$.

*Lemma 1:* Let $1 \leq m \leq P$, $y^0 = (y_1^0, \cdots, y_P^0) \in (\mathbb{R})^P$ and let $y^1 = (y_1^1, \cdots, y_P^1) \in (\mathbb{R})^P$ be defined as follows: $y_i^1 = (y_1^0 + \cdots + y_m^0)/m$ for $1 \leq i \leq m$, and $y_i^1 = y_i^0$ for $m < i \leq P$. Then $y^1$ can be obtained from $y^0$ by performing a sequence of $(m-1)$ orthogonal projections on hyperplanes of $\mathbb{R}^n$ as in KACZ.

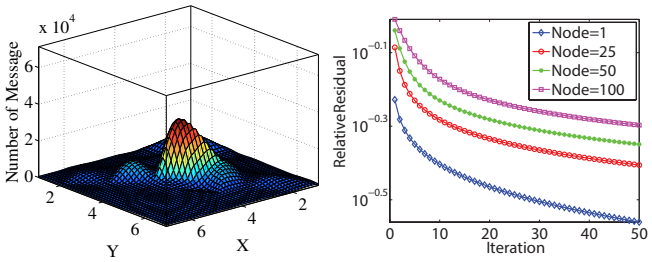
*Proof:* (refer Appendix A) ∎

We have shown that averaging is equivalent to row projection in some superspace $\mathbb{R}^s$. Let, $\tilde{B}'$ be the auxiliary averaging equation and we form $B''$ by adding $B'$ with $\tilde{B}'$, with increased row proportional to number of shared variables. Now, if the Eq. (2) is consistent, then set of transformed equations $B'$ and average equations $\tilde{B}'$ are also consistent. Strohmer and Vershynin [30] showed that RK converges on a consistent system even if the projections are not performed cyclically; all that is required is that each equation should

be used infinitely often. This allows us to perform RK in Algorithm 2 for any positive number of iteration in each block. We refer to work of [23] in case of the inconsistent system. This proves the convergence of Par-RK.

### B. Communication Cost and Performance

The communication cost of Par-RK is of the order $2kPn$ where $k, P, n$ represents iteration, number of nodes and size of $\vec{x}$ respectively. Although the communication cost is in terms of iteration, Par-RK involves aggregation and requires a fusion center to perform merging. These fusion center based method can be effective in applications that require coordinator node and has relatively smaller data size. The data collection and dissemination on a large scale network for each iteration can become challenging especially in a harsh environment. This method also requires synchronization between all the nodes for aggregation and is vulnerable to failure especially at the fusion center (Fig. 1(a)). From the evaluation, we saw that the performance of Par-RK decreases with increase in the number of nodes (Fig. 1(b)). The multihop communication increases the communication overhead, packet loss, and slower convergence. These practical limitations motivated us to develop another class of algorithm to solve Eq. (2) that exchanged information only with the neighbors decreasing the communication overhead (Fig 2). In the next sub-section, we will provide some overview of gossip method that will setup the background for the distributed randomized kaczmarz (D-RK).




(a) Communication Cost Par-RK (b) Convergence Performance

Fig. 1. Parallel RK Communication and Performance

### C. Distributed Randomized Kaczmarz

Gossip methods are emerging as a new communication paradigm for large-scale distributed systems [16]. Some of the features that makes gossip methods attractive are: i) absence of central entity or coordinator node ii) high fault tolerance and robustness iii) *self healing* or error recovery mechanism [31] iv) efficient message exchange due to only neighbor communication v) provision for asynchronous communication. These interesting characteristics make them suitable for WSN to carry out decentralized computation [29].

There are several variants of gossip algorithms designed specifically for tasks such as i) disseminate information [8], ii) compute sum/average [16] and iii) reach consensus [2]. The design of these algorithms vary slightly based on the communication pattern and also with the type of information exchanged at each iteration. Fore.g., in push-sum [16], only one node wakes up at a time and exchanges information with another neighboring node whereas, in broadcast gossip [2] information is sent to all its neighboring node.

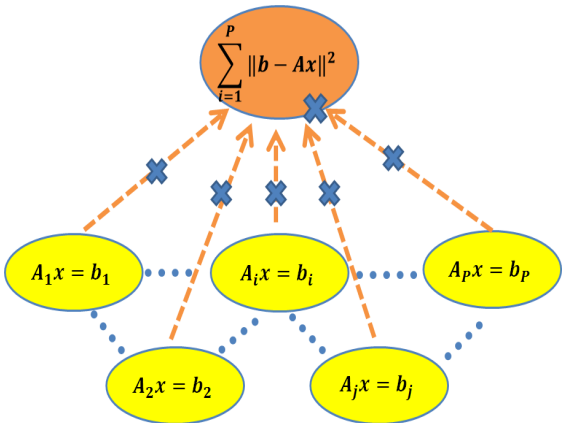



Fig. 2. Gossip based Push Sum method where information is exchanged between neighbor without fusion center.

To design truly distributed method, we avoid the fusion center to compute average and replace it with decentralized methods such as push-sum [16]. In push-sum, when node $i$ activates at $t^{th}$ time slot, the following set of events occur: i) Node $i$ sends its current state value $x_i^t$ to neighboring node $j$. ii) Node $j$ receives $x_i^t$ and updates it in following way: $x_j^{t+1} = \frac{x_j^t}{2} + \frac{x_i^t}{2}$. iii) Node $j$ sends $x_j^{t+1}$ to $i$, where it updates $x_i^{t+1} = x_j^{t+1}$. iv) Remaining nodes update their value as: $x_\ell^{t+1} = x_\ell^t, \forall \ell \in \{\mathcal{V} - \{i, j\}\}$.

Now, if we denote $x^t \in \mathbb{R}^P$ a vector whose each component represents state of each node in network, then for every clock tick $t$ we have, $x^{t+1} = W^t x^t$, where, $W^t$ is a random matrix given by,

$$W_{ij}^t = \begin{cases} \frac{1}{P} & \{i,j\} \in \mathcal{E} \\ 1 - \frac{|\mathcal{N}_i|}{P} & i = j \\ 0 & \text{otherwise} \end{cases}$$

From the above weights, $W^t$ exhibits following property: $W^t 1 = 1$ and $1^T W^t = 1^T$. Therefore, for every $t$, the iteration preserves the sums while vector of averages must be fixed point of iteration [4]. Next, we use the above gossip model and extend it to component-wise vector sum which will be used for computation of decentralized average.

Let $x_{(i)}^k$ denote intermediate solution of $A_i, b_i$ at $i^{th}$ node after $k^{th}$ iteration. Also, let $x_{(i)j}^k$ denote $j^{th}$ component of $x_{(i)}^k$. We define $X_j^t = (x_{(1)j}^k, \cdots, x_{(P)j}^k)^T$, containing the $j^{th}$ component of all the nodes. From the above gossip model we update the $j^{th}$ component by $X_j^{t+1} = W^t X_j^t$. Similarly, we can extend this to all the component $j \in \{1, \cdots, n\}$. We denote this gossip scheme as push-vector.

Using the above definition of push-vector we propose the distributed randomized kaczmarz (D-RK) to solve linear equation over a decentralized system such as WSN. Algorithm 3 describes D-RK that combines component-wise gossip average with RK to solve a linear system of equations. In step 1 - 6 algorithm performs a certain iteration of RK simultaneously on all the nodes using its initial vector $x^k$. Step 7 - 10 of

**Algorithm 3** D-RK Algorithm

**set** $x_\ell^0 \in \mathbb{R}^n$ to an arbitrary value $\forall \ell \in \mathcal{V}$.
**for** $k \leftarrow 0$ until convergence or max iteration **do**
  **for** each $1 \leq l \leq P$ in parallel **do**
    $y_\ell = \text{RK}(A^\ell, b^\ell, x_\ell^k, \rho_\ell)$
    $\tilde{y}_{(\ell)}^0 \leftarrow \quad y_\ell$
  **end**
  **for** $t \leftarrow 0$ until convergence or max iteration **do**
    Node $i \in \mathcal{V}$ contacts $j \in \mathcal{N}_i$ and updates
    $\tilde{y}_{(j)}^{t+1} = \tilde{y}_{(i)}^{t+1} = \frac{\tilde{y}_{(j)}^t}{2} + \frac{\tilde{y}_{(i)}^t}{2}$
  **end**
  $x_{(\ell)}^{(k+1)} = \tilde{y}_{(\ell)}^t \quad 1 \leq l \leq P$
**end**

the algorithm describes push-vector. This algorithm is truly distributed and does not involve any fusion center. Push-vector continues until the relative update of the average is below a certain threshold.

*Lemma 2:* If $Q_i = \{x \in \mathbb{R}^n | A_i x = b_i\}$, $Q = \cap\{Q_i | i \in V\}$ and Eq. 1 has a solution $x^* \in Q$, then any sequence generated by Algorithm 3 converges to a fixed point $x^* \in Q$ for $\rho = 1$.

*Proof:* Since D-RK has similar structure as Par-RK except for averaging scheme, to prove the convergence, it is enough to show that push-vector is equivalent to row projection of RK. From above definition the update of $z^{th}$ component of $x_{(i)}^t$ and $x_{(j)}^t$ is given by

$$x_{(i)z}^{t+1} = x_{(j)z}^{t+1} = \frac{x_{(i)z}^t + x_{(j)z}^t}{2} \tag{4}$$

Let us assume $y^t = \{0, \cdots, x_{(i)z}^t, x_{(j)z}^t, \cdots, 0\}$ be a vector consisting of $x_{(i)z}^t$ and $x_{(j)z}^t$ at $i^{th}$ and $j^{th}$ position respectively. Also, assume a plane whose $i^{th}$ and $j^{th}$ components are related by $-m_i + m_j = 0$. Therefore, the coefficient $a = \{0, \cdots, -1, 1, \cdots, 0\}$, $\|a\|^2 = 2$ and $b = 0$. Now, from RK Algorithm 1 we have

$$\begin{aligned} y^{t+1} &= y^t + \rho \frac{(b - \langle a, y^t \rangle)\, a}{\|a\|^2} \\ &= y^t - \rho \frac{\langle a, y^t \rangle a}{2} \\ &= (0, \cdots, x_{(i)z}^t, x_{(j)z}^t, \cdots, 0) - \frac{\rho}{2}(-x_{(i)z}^t + x_{(j)z}^t) \\ &\quad (0, \cdots, -1, 1, \cdots, 0) \\ &\quad \text{for } \rho = 1 \text{ we have,} \\ &= (0, \cdots, \frac{x_{(i)}^t + x_{(j)z}^t}{2}, \frac{x_{(i)z}^t + x_{(j)z}^t}{2}, \cdots, 0) \end{aligned}$$

This is equivalent to expression in Eq. (4). Similarly, we can extend this argument for every component $z \in \{1, \cdots, n\}$ of the vector. Hence, push-vector updates are equivalent to row projection of random kaczmarz for $\rho = 1$. ■

Push-vector in D-RK performs gossip only with one neighboring node at any time slot $t$. In a wireless sensor network, each node has an advantage of inherently broadcasting the messages to its neighbor within a certain radius. Now at the cost of one transmission, a node can gossip with all of its neighbors, andthishasbeen studied under broadcast gossip [2].The broadcast gossip, however, converges to a consensus rather than an average as the weights used there does not preserve the sum (i.e., $1^T W \neq 1^T$). This will affect the convergence analysis of D-RK and will be studied in future.

## IV. Case Study - Decentralized Seismic Tomography

In this section, we use volcano monitoring as a case study to evaluate the performance of our proposed algorithms. Volcano monitoring is one such application where distributed least squares is mandatory for real-time high-resolution imaging that can help scientists to analyze and predict the occurrence of volcanoes in real-time [26]. Here, we first provide an overview of the seismic imaging problem along with the experimental setup on CORE. Later, we evaluate the performance of our proposed method with state of the art algorithms. Finally, we also show the applicability of our methods towards real volcano monitoring using trace data from Mt. St. Helens, WA.

### A. Overview

The current seismic imaging system uses sensors (geophones) deployed on the volcano to acquire travel time of the acoustic wave, known as P-waves, generated by underground seismic activities such as earthquakes. The received travel times are then used to derive the internal velocity structure with which one can monitor the volcano edifice. The principle of seismic tomography is illustrated in the Fig. 3. The travel time tomography involves three main steps: (i) sensor nodes (green triangles on the surface) measure seismic vibration after occurrence of earthquake and estimate and calculate the arrival time of the p-wave [20] to form $b_i$, ii) this is later used to estimate origin time and location of earthquake given prior estimate of geological structure, (iii) at each node $i$, ray tracing is performed (blue rays) from earthquake location to sensor node to form $A_i$, and (iv) using these ray information, a 3D tomography reconstruction of the velocity structure $x$ of the volcano is obtained by solving the linear system $A_i x = b_i$ in a decentralized way.

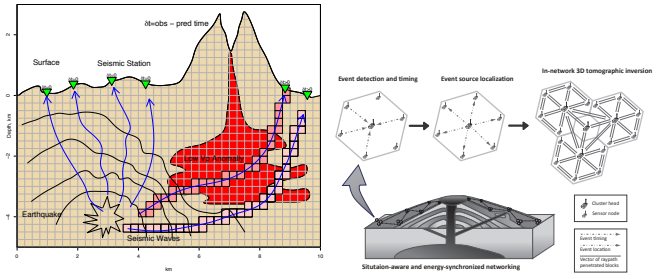



Fig. 3. Illustration of process of seismic tomography and decentralized network topology. (a) Principle of travel-time seismic tomography. (b) Real-time In-situ Seismic Imaging network.

Typically, to test tomography inversion algorithm a synthetic model is used. This serves two purpose: a) the real data set such as from Mt. St Helens do not have an exact ground truth b) simulations using synthetic model enables us to investigate individually various phenomena which cannot be separated physically. For example, p-wave data always contain noise due to measurement and scattering, but simulation can indicate the specific effect separately. Therefore, we first test

our algorithm using a synthetic model and later with real data trace from Mt. St. Helens. To evaluate the Par-RK and D-RK, a data generator is implemented to generate a magma area (Fig. 4(a)) and earthquake events assuming the tomography model is a cube of dimension $10 \times 10 \times 10$ km. Then we set a predefined magma area as the ground truth as shown in Figure 4(a). The velocities of seismic waves inside and outside the magma area are $V$ and $0.9V$ where V is 4.5km/s which is a typical P-wave velocity.

### B. System Setup

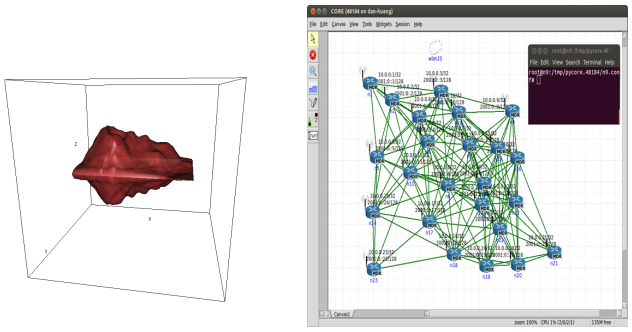
To evaluate the network performance of the proposed algorithm we setup a simulation framework on CORE [2] which supports the implementation of custom protocols for WSN [1]. For an efficient communication on WSN in a harsh, unpredictable environment, we developed Bundle Layer Protocol, which adopts the properties of Disruption-Toleration Network (DTN) technique to maintain efficient and reliable end-to-end connectivity [26]. In our design, the data is buffered into a bundle and then transferred hop by hop in a store-and-forward manner until it arrives at the destination. Our implementation of the bundle layer does not make any changes to underlying network services and TCP for one-hop reliable bundle transfer and uses the routing table to indicate the next hop. We use BATMAN (Better Approach to Mobile Ad-hoc Networking) [13] routing protocol which minimizes the network overhead by maintaining only next hop neighbor entry to forward messages from a source to the destination.

The two proposed algorithm have different communication paradigm, i.e., Par-RK uses a fusion center whereas D-RK gossips only with its neighbors. We implement Par-RK using *Aggregation Tree* [21] while gossip using [16]. Both gossip and aggregation tree protocol runs on top of bundle layer that ensures reliability while performing summation over the network. We refer to [14] for the implementation details, and we skip them in this paper.

A network of 100 nodes (Fig. 4(b)) are setup in the CORE to monitor the magma area. We set the seismic image resolution to be $32 \times 32 \times 32$ where each block is of the size $0.315\ km^3$. The data generator then generates 500 earthquake events with random location and time and calculates ray travel time from event location to all sensor nodes. A white Gaussian noise is added to the travel time to simulate the event location estimation and ray tracing errors. Each node can calculate the predicted travel time based on the initial model in different resolution.

In the implementation, the Par-RK and D-RK is performed for 10 iterations locally to solve the equations on each node. We use the relative update ($\phi$) of the estimation between the two sweeps (one sweep means that all partial solutions are averaged to calculate next iterate) as the stopping criteria. If the relative update ($\phi$) is less than a tolerance value, the algorithm terminates. Performance of the algorithms are compared using their relative update ($\phi$), relative residual ($\chi$) and relative error ($\delta$) given by: $\chi = \|Ax^k - B\|/\|B\|$, $\phi = \|x^{(k+1)} - x^{(k)}\|/\|x^{(k)}\|$, and $\delta = \|x^{(k)} - x^{truth}\|/\|x^{truth}\|$

[2]http://cs.itd.nrl.navy.mil/work/core/



(a) 3D Synthetic Phantom (b) Station setup in CORE

Fig. 4. (a) 3D synthetic magma model (b) Snapshot of the CORE GUI which displays the nodes along with communication link.

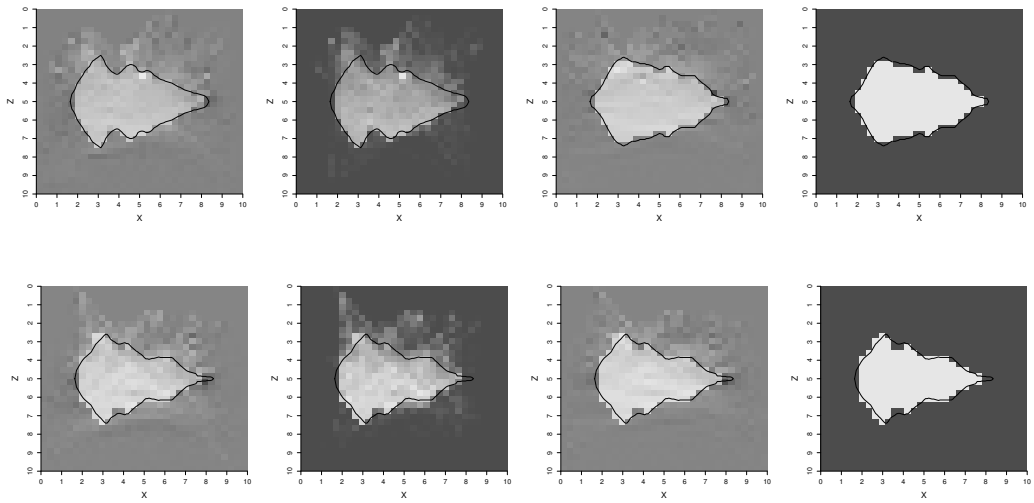
### C. Correctness and Accuracy

In this set of experiments, we intend to demonstrate the correctness of our algorithm through visualization. We stopped the algorithm when its relative update $\phi \leq 0.001$. Fig. 5 shows the result slice by slice along the X and Y axes and Fig. 5(d) we have the ground truth. Each row of the figure shows the same tomography slice on some layer along with X or Y axes (the total layers of each figure is equal to the resolution dimension of the result). The black polygon gives the cross section outline of the surface of magma area represented in Fig. 4(a). From this experiment, we can see that both Par-RK and D-RK were able to generate tomography image almost similar to centralized RK. From the detailed examination of the Fig. 5 we can say that, Par-RK produces fewer artifacts near the boundary whereas, D-RK has sharper differences. We believe this is due to the fact that Par-RK calculates true average unlike D-RK. This evaluation suggests that both Par-RK and D-RK can be a good candidate for distributed tomographic inversion.

Next, we compare the performance of the proposed algorithm in terms of the relative update. We compare Par-RK with algorithms like Cimmino, CAV, and DROP, which are all algorithms used to solve the system of linear equations using fusion center. On the other hand, we compare D-RK with decentralized algorithms such as EXTRA [27] and DGD [32] and results are shown in Fig. 6. From Fig. 6(a) we can infer that Par-RK performs better than other parallel methods in terms of convergence. This is due to the faster convergence rate of RK method compared to other methods. In case of D-RK (Fig. 6(b)) we see that it is faster than DGD, however, slower than EXTRA. In EXTRA, an optimal step size is calculated to accelerate the convergence. It should be noted that, Par-RK has faster convergence compared to D-RK and is because of the faster mixing of the partial solutions at the expense of fusion center and multi-hop aggregation scheme.

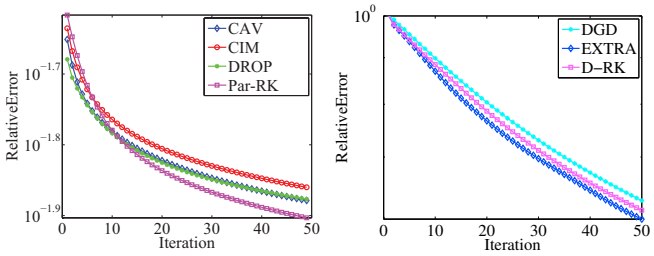
### D. Communication and Robustness

In this section, we compare the communication cost of the centralized algorithm with proposed distributed algorithms in terms of number of messages exchanged to reach the solution. Here for centralized and parallel case SINK(M) and SINK(C) refers to sink (fusion center) node placed in middle and at the corner respectively. From Fig. 7(a) we can see that communication cost in a centralized setup is high near the SINK as all the ray information is transferred over the network



(a) Par-RK (b) D-RK (c) RK (d) Ground Truth

Fig. 5. 2D vertical slice of 3D Tomography using Synthetic Data. Top row represents slice 16 of 32 and bottom row represents slice 18 of 32 of different algorithms.

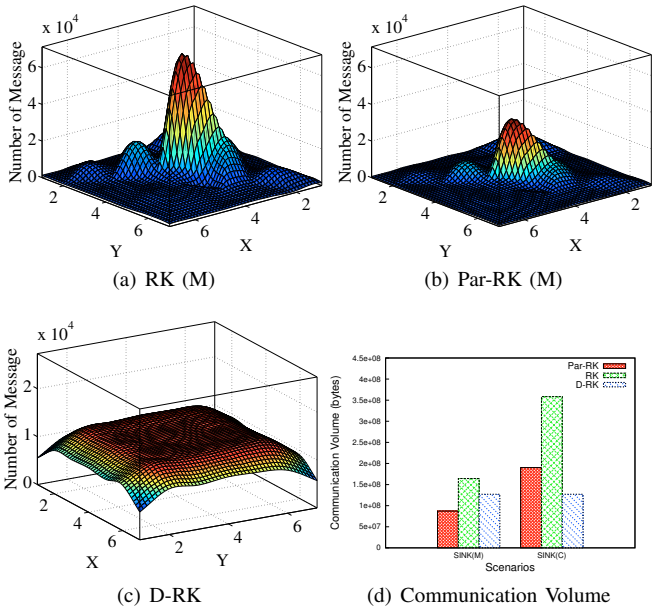



(a) (b)

Fig. 6. Comparison of (a) Par-RK and (b) D-RK with different parallel and distributed algorithms.

to sink before the computation. In this case, the volume of data is proportional to the number of earthquakes and also number of stations. Fig. 7(b) shows the communication pattern for Par-RK and from this we can see that the communication cost is lesser compared centralized scheme (RK). This is mainly because communication cost in Par-RK depends on number of iteration and typically with the semi-convergent property of iterative methods [9] the number of iteration is much less compared to the number of earthquake events.

In Fig. 7(c) we present the communication pattern of D-RK, which is flat compared to Par-RK and RK. Neighbor gossip helps us to balance the load in the network while avoiding other overheads such as routing, etc. It should be noted that due to the slower convergence of D-RK compared to Par-RK, a larger volume of packets will be exchanged in the entire process. This is verified from Fig. 7(d) where we compare volume (bytes) transferred in all the three settings. Fig. 7(d) shows that in case of SINK(M) volume of bytes transferred Par-RK is lower than D-RK, which is due to the slower convergence as mentioned earlier. However, placing SINK(C) at the corner increases the communication cost of the Par-RK, and this is due to packet loss caused by increased congestion. It should be noted that D-RK has no effect on the placement of SINK node as it communicates only with the neighbors..



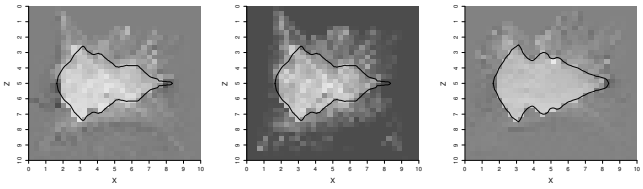

(a) RK (M) (b) Par-RK (M)

(c) D-RK (d) Communication Volume

Fig. 7. Communication Cost

In the next set of experiments, loss tolerance and robustness of proposed algorithms are evaluated. The algorithm runs with the same configuration for a packet loss ratio of 20% in the emulator. Fig. 8 gives part of the 2D slice rendered along Y axes with packet loss. We can see that in Par-RK and D-RK with 20% packet loss there is no significance difference in terms of the magma area outline when compared to the results with no packet loss Fig. 8(c). Since the computation is distributed, and all the nodes are involved in slowness calculation, the proposed algorithm is tolerant to a severe packet loss. A closer look at the result tells that packet loss in Par-RK has a slightly larger effect compared to D-RK. This

can be seen in the visualization result where D-RK can get a sharper image than Par-RK. This result can be attributed to the truly distributed communication as opposed to fusion center in Par-RK. Loss of packet near fusion center has a profound effect on the tomography result, whereas D-RK can tolerate such single point failures.



(a) Par-RK 20% Loss (b) D-RK 20% Loss (c) No Loss

Fig. 8. Robustness of distributed algorithms in terms of packet loss.

## V. Imaging of Mt. St. Helens, Washington, USA

Mount St. Helens (MSH), WA, USA situated in Pacific northwest cascade region erupted on May 18, 1980 and was one of the deadliest volcano in the history of United States, killing 57 people and destroying several homes. MSH is one of the most widely studied volcanoes in the world mainly due its recent volcanic activity and also due its unusual sideways eruption, which surprised many geophysicists and questioned many the existing theory. Due to its location close to human habitat there is an increasing effort to understand the dynamics of this volcano and to obtain high-resolution imagery.

### A. Distributed Algorithms for Real Data

In this section, we assume that all the raw data has been pre-processed to form a subsystem at each node. Real data measurements are often prone to severe measurement noise making the linear system often inconsistent, i.e., $Ax = b + r$. Because of this solutions obtain from simple RK does not describe actual tomography [18]. To obtain satisfactory results, we must add additional constraints such as regularity or smoothness to RK that suppresses unwanted noise. This process is termed as regularization which is required to avoid strong, undesired influence of small singular values dominating the solutions. The solution $x_\lambda$ is defined as the solution to the regularized problem

$$x_\lambda = \arg\min_x \frac{1}{2}\|Ax - b\|_2^2 + \lambda^2\|x\|_2^2 \tag{5}$$

Here, the *regularization parameter* $\lambda$ is a positive number that controls the weight between $\|Ax - b\|_2^2$ (goodness fit measurement) and $\|x\|_2^2$ (regularity measurement). The larger the value of $\lambda$, the more weight is given to the minimization of the solution norm. On the other hand, small $\lambda$ means more weight is given to fit the noisy data. This type of regularization is commonly known as Tikhonov regularization [9].

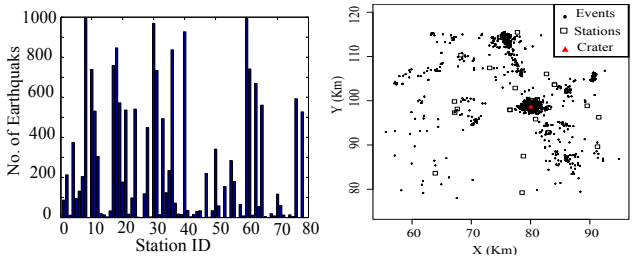
To solve Eq. 5, Bayesian version of RK was proposed by G.T Herman [10], [11]. Since the system is inconsistent, we consider, $Ax + r = b$ where, $r$ is chosen such that given any $x$, $r = b - Ax$. With this assumption, the system becomes well-posed. Therefore, we solve for $x$ and $r$ simultaneously,i.e., $[\lambda I \quad A][r \quad s]^T = b$. This modification makes the system consistent and the solution to this is also the solution to the regularized Eq. (5). Now, by applying RK to modified equation we obtain Bayesian RK (BaRK) given in Algorithm 4, where, $a_i$ is the $i^{th}$ row of A ($a_i^T$ is its transpose), $\hat{e}_i$ is a unit vector with $i^{th}$ element set to one, $\rho_k$ relaxation parameter and $\lambda$ is the regularization parameter. BaRK can be treated as RK with regularization which is necessary for the systems that are ill-posed and have high noise. In the experiment with real data, we replace RK in our proposed distributed algorithms with BaRK. The convergence property of the modified algorithm remains the same and we skip the details in this paper.

**Algorithm 4** Bayesian RK (BaRK)

1: Initialize: $r^0 \leftarrow 0; x^0 \leftarrow 0$
2: **for** $k \leftarrow 0$ until convergence or maximum number of iteration **do**
3: Pick $i \in \{1, \cdots .m\}$ with probability $p_i = \frac{\|a_i\|_2^2}{\|A\|_F^2}$
4: $d^{(k)} = \rho^{(k)} \frac{b_i - \lambda(r_i^{(k)}) - (a_i^T \cdot x^{(k)})}{\lambda^2 + \|a_i\|^2}$
5: $x^{(k+1)} = x^{(k)} + d^{(k)} a_i$
6: $r^{(k+1)} = r^{(k)} + \lambda d^{(k)} \hat{e}_i$
7: **end**

### B. Experiment Setting

In this section, we study and present a P-wave velocity model for Mount St. Helens (MSH) using our proposed distributed algorithm. The results are verified by comparing it with the centralized algorithm BaRK [18]. We used the data set from [18] which has data from 78 stations spread over $160 \times 200$ kms in area. The depth analyzed in this paper is up to 24 km. A total of 1141 earthquake event data are used which gave around 18161 rays. Each station records different sets of earthquake events due to its location and accuracy. Due to this, the number of rays traced at each station varied from $0 - 1141$. Fig. 9(a) shows the distribution of rays traced across each station. For the distributed tomography, ray count at each station is equivalent to number of rows of linear system at that station and nodes with ray count zero does not involve in the computation.



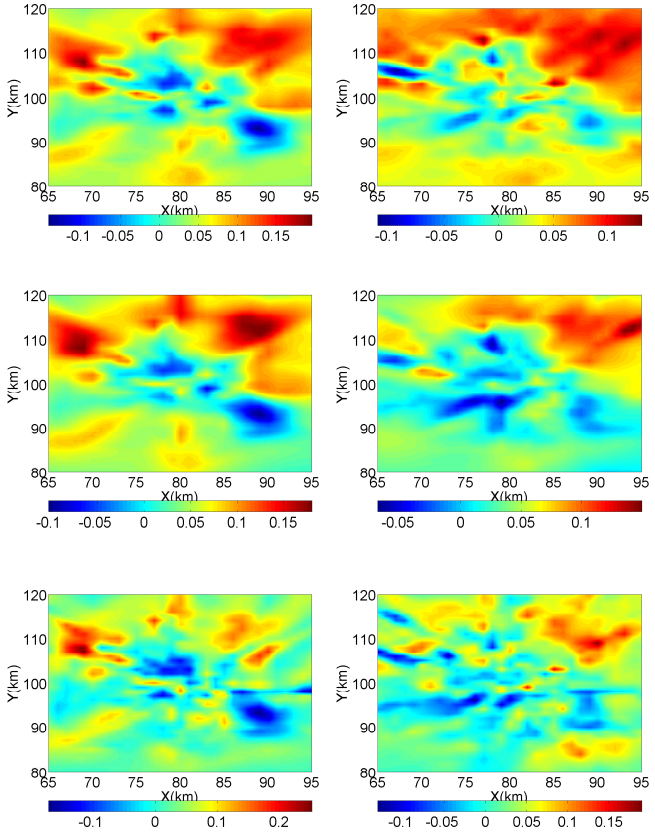

(a) Node wise event distribution (b) Events from Mt. St Helens, Washington

Fig. 9. Non-uniform distribution of rays and events at Mt St Helens. (a) Number of event from Mt. St. Helens detected by different nodes. (b). Black dots and squares denote the location of the earthquake and station location respectively and the red triangle denotes the location of the crater.

In this paper, the column size (i.e resolution dimension) is around $160 \times 200 \times 24 = 768000$ and with such high dimension, methods involving full gradients such as [32] can be challenging. Geophysicists calculate relative seismic velocity variations ($Vrel_P$) to monitor the volcano edifice.

This velocity can be represented as a 3D map of slices of earth, depicting the regions of higher or lower seismic wave velocity. The region where seismic waves move more slowly, is interpreted to be due to the presence of warm, partially melted rock; that is the crustal magma storage region. The final solution $x$ obtained from distributed algorithms represents $Vrel_P$ which is shown in Fig. 10.

Notice that unlike synthetic data used in previous section, there is no ground truth for the velocity of MSH. Hence we focus on the comparison of the proposed methods with a centralized processing scheme, which can be seen as a benchmark that fully utilize the data available. Interpretation of this data requires in-depth knowledge of geophysics and is out of the scope of this paper. Final result generated by three algorithms are provided in Fig. 10. From the result we can see that Par-RK (top row) and D-RK (middle row) can generate tomography similar to centralized BaRK (bottom row) for two different depths. The low velocity region (blue) which represents partially melted rock or magma can be identified more clearly by distributed algorithms as opposed to high velocity region (red). This depends on the selection of regularization and smoothing constraints that decides weights on the values of the solutions. This kind of tomography can provide scientists an initial reference in real-time which could be used for further research and also for hazard mitigation.




(a) Depth 2.9 km (b) Depth 4.9 km

Fig. 10. Relative velocity perturbation map of MSH at different depths. Top row shows $Vrel_p$ using Par-BaRK, while middle row show results of D-BaRK and bottom row using centralized BaRK.

## VI. Discussion and Future Work

The proposed distributed algorithms require synchronization between computation and communication. This can be very challenging in large-scale systems, where synchronization requires extra overhead. To avoid this, we are exploring distributed asynchronous methods that need not wait for others to finish computation or communication. We have designed and implemented both hardware and algorithms for volcano monitoring, and we have deployed 20 stations on Llaima volcano in Chile for the first phase of testing. Currently, nodes are running simple algorithms for recording seismic waves and pre-processing of the data. In future, we intend to implement the proposed distributed algorithms on such systems for real-time volcano monitoring.

## VII. Conclusion

In this paper, we presented two distributed algorithms to compute least-squares solution in a loosely coupled system such as a WSN. We first presented Par-RK, that uses fusion center to merge the intermediate result. Later, we eliminated fusion center using gossip method to make it truly distributed. These algorithms are designed specifically for a large sparse system of linear equations. Performance evaluation of both these algorithms for seismic application showed that they are robust in terms of both computation and communication. Further, we also tested our algorithms using real data traces from Mt. St. Helens and the results obtained from proposed methods was close to that of a centralized approach. The robustness and loss tolerance feature of Par-RK and D-RK are very attractive which makes it suitable for sensor network applications.

## Acknowledgment

We like to thank Dr. Greg Waite and Dr. Jonathan Lees for providing the data set of Mt. St Helens and help us visualize the velocity model. We thank Dr. Xiaojing Ye and Liang Zhao for introducing us to distributed gradient descent methods. Our research is partially supported by NSF-CNS-1066391, NSF-CNS-0914371, NSF-CPS-1135814 and NSF-CDI-1125165.

## References

[1] J. Ahrenholz, T. Goff, and B. Adamson. Integration of the CORE and EMANE Network Emulators. In *MILITARY COMMUNICATIONS CONFERENCE, 2011 - MILCOM 2011*, pages 1870–1875, 2011.

[2] T. C. Aysal, M. E. Yildiz, A. D. Sarwate, and A. Scaglione. Broadcast Gossip Algorithms for Consensus. *Signal Processing, IEEE Transactions on*, 57(7):2748–2761, July 2009.

[3] L. Bottou. Large-Scale Machine Learning with Stochastic Gradient Descent. In Y. Lechevallier and G. Saporta, editors, *Proceedings of COMPSTAT'2010*, pages 177–186. Physica-Verlag HD, 2010.

[4] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Randomized gossip algorithms. *IEEE Transactions on Information Theory*, 52(6):2508–2530, June 2006.

[5] Y. Censor, T. Elfving, and G. T. Herman. On diagonally-relaxed orthogonal projection methods. *SIAM J. Sci. Comput*, 30(1):473–504, June 2007.

[6] Y. Censor, D. Gordon, and R. Gordon. Component averaging: An efficient iterative parallel algorithm for large and sparse unstructured problems. *Parallel Computing*, 27(6):777–808, 2001.

[7] J. M. Elble, N. V. Sahinidis, and P. Vouzis. GPU computing with Kaczmarz's and other iterative algorithms for linear systems. *Parallel Computing*, 36:215–231, June 2010.

[8] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker. An Empirical Study of Epidemic Algorithms in Large Scale Multihop Wireless Networks. 2002.

[9] P. C. Hansen. *Discrete Inverse Problems*. Society for Industrial and Applied Mathematics, Jan. 2010.

[10] G. T. Herman. *Reconstruction from Projections: The Fundamentals of Computerized Tomography*. Academic Press, 1980.

[11] G. T. Herman, H. Hurwitz, A. Lent, and H.-P. Lung. On the Bayesian Approach to Image Reconstruction. *Information and Control*, 42:60–71, 1979.

[12] D. Jakovetic, J. Xavier, and J. M. F. Moura. Fast Distributed Gradient Methods. *arXiv preprint arXiv:1112.2972v4*, Apr. 2014.

[13] D. Johnson, N. Ntlatlapa, and C. Aichele. A simple pragmatic approach to mesh routing using BATMAN. In *2nd IFIP International Symposium on Wireless Communications and Information Technology in Developing Countries*, 2008.

[14] Y. Y. Jun and M. Rabbat. Performance comparison of randomized gossip, broadcast gossip and collection tree protocol for distributed averaging. In *Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), 2013 IEEE 5th International Workshop on*, pages 93–96. IEEE, Dec. 2013.

[15] G. Kamath, L. Shi, and W.-Z. Song. Component-Average based Distributed Seismic Tomography in Sensor Networks. In *IEEE DCOSS*, 2013.

[16] D. Kempe, A. Dobra, and J. Gehrke. Gossip-Based Computation of Aggregate Information. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '03, pages 482–491, 2003.

[17] S. Kim, S. Pakzad, D. Culler, J. Demmel, G. Fenves, S. Glaser, and M. Turon. Wireless sensor networks for structural health monitoring. In *Proc. 4th ACM conference on Embedded networked sensor systems (SenSys)*, Nov. 2006.

[18] J. M. Lees. The magma system of Mount St. Helens: non-linear high-resolution P-wave tomography. *Journal of Volcanology and Geothermal Research*, 53:103–116, 1992.

[19] H. B. Lim, Y. M. Teo, P. Mukherjee, V. T. Lam, W. F. Wong, and S. See. Sensor Grid: Integration of Wireless Sensor Networks and the Grid. In *Local Computer Networks*, Nov. 2005.

[20] G. Liu, R. Tan, R. Zhou, G. Xing, W. Song, and J. Lees. Volcanic Earthquake Timing using Wireless Sensor Networks. pages 91–102, 2013.

[21] S. Madden, M. J. Franklin, J. Hellerstein, and W. Hong. TAG: a Tiny Aggregation Service for Ad-Hoc Sensor Networks. In *ACM Symposium on Operating System Design and Implementation*, Dec. 2002.

[22] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson. Wireless Sensor Networks for Habitat Monitoring. In *The First ACM International Workshop on Wireless Sensor Networks and Applications*, 2002.

[23] D. Needell. Randomized Kaczmarz solver for noisy linear systems. *BIT*, 50(2):395–403, 2010.

[24] K. Prikopa, H. Straková, and W. Gansterer. Analysis and Comparison of Truly Distributed Solvers for Linear Least Squares Problems on Wireless Sensor Networks. In F. Silva, I. Dutra, and V. Santos Costa, editors, *Euro-Par 2014 Parallel Processing*, volume 8632 of *Lecture Notes in Computer Science*, pages 403–414. Springer International Publishing, 2014.

[25] R. A. Renaut. A Parallel Multisplitting Solution of the Least Squares Problem. *Numerical Linear Algebra with Applications*, 5(1):11–31, 1998.

[26] L. Shi, W.-Z. Song, M. Xu, Q. Xiao, J. M. Lee, and G. Xing. Imaging Seismic Tomography in Sensor Network. In *IEEE SECON*, 2013.

[27] W. Shi, Q. Ling, G. Wu, and W. Yin. EXTRA: An Exact First-Order Algorithm for Decentralized Consensus Optimization. *arXiv preprint arXiv:1404.6264*, Nov. 2014.

[28] W.-Z. Song, R. Huang, M. Xu, A. Ma, B. Shirazi, and R. Lahusen. Air-dropped Sensor Network for Real-time High-fidelity Volcano Monitoring. In *The 7th Annual International Conference on Mobile Systems, Applications and Services (MobiSys)*, June 2009.

[29] H. Straková, W. N. Gansterer, and T. Zemen. Distributed QR factorization based on randomized algorithms. In *PPAM'11 Proceedings of the 9th international conference on Parallel Processing and Applied Mathematics - Volume Part I*, pages 235–244, 2012.

[30] T. Strohmer and R. Vershynin. A randomized Kaczmarz algorithm with exponential convergence. *J. Fourier Anal. Appl*, 15:262–278, 2009.

[31] R. Van Renesse, K. P. Birman, and W. Vogels. Astrolabe: A Robust and Scalable Technology For Distributed . . . In *ACM TRANSACTIONS ON COMPUTER SYSTEMS*, 2003.

[32] K. Yuan, Q. Ling, and W. Yin. On the Convergence of Decentralized Gradient Descent. *arXiv preprint arXiv:1310.7063v2*, Feb. 2014.

[33] Q. Zhou, S. Kar, L. Huie, H. V. Poor, and S. Cui. Robust Distributed Least-Squares Estimation in Sensor Networks with Node Failures. In *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, pages 1–6. IEEE, Dec. 2011.


## APPENDIX A
## PROOF OF LEMMA 1

*Proof:* The proof is by induction on m. For m = 1, there is nothing to prove. For m = 2, project $y^0$ onto the plane defined by the equation $-y_1 + y_2 = 0$. The vector of coefficient of a are $\{-1, 1, 0, \cdots, 0\}$ and $\|a\|^2 = 2$. The projection $\tilde{y} = \{\tilde{y}_1, \tilde{y}_2, \cdots\}$ is

$$\begin{aligned}\tilde{y} &= y^0 - \frac{1}{2}\langle y^0, a\rangle a \\ &= (y_1^0, y_2^0, \cdots) - \frac{1}{2}(-y_1^0 + y_2^0)(-1, 1, 0, \cdots, 0) \\ &= \left(\frac{1}{2}(y_1^0 + y_2^0), \frac{1}{2}(y_1^0 + y_2^0), y_3^0, y_4^0, \cdots\right)\end{aligned} \tag{6}$$

In other words, for m = 2 nodes we obtain $\tilde{y}_1 = \tilde{y}_2 = \frac{(y_1^0 + y_2^0)}{2}$ by performing one orthogonal projection on a suitable hyperplane. We assume that the statement is true for m, and we will prove it for m+1. Let, $y^0 = \{y_1^0, \cdots, y_n^0\}$ and we project $y^0$ onto the hyperlane defined by the equation $-y_1 - y_2 \cdots - y_m + m y_{m+1} = 0$. Now, $a = (-1, \cdots, -1, m, 0, \cdots, 0)$ and $\|a\|^2 = m + m^2 = m(m+1)$. The projection is the point $y' = \{y_1', \cdots, y_n'\|$ defined by $y' = y^0 - (\langle y^0, a\rangle a)/(m(m+1))$. Substituting $y^0$ and $a$, we have

$$\begin{aligned}y' = (y_1^0, \cdots, y_n^0) - \frac{-y_1 - y_2 \cdots - y_m + m y_{m+1}}{m(m+1)} \\ (-1, \cdots, -1, m, 0, \cdots, 0)\end{aligned}$$

For each $1 \le i \le m$, we have

$$y_i' = y_i^0 - \frac{1}{m(m+1)}\left(\sum_{j=1}^{m} y_j^0 - m y_{m+1}^0\right),$$

and the (m+1)st coefficient is

$$\begin{aligned}y_{m+1}' &= y_{m+1}^0 + \frac{(y_1^0 + \cdots + y_m^0 - m y_{m+1}^0}{m(m+1)} m \\ &= \frac{1}{m+1}(y_1^0 + \cdots + y_m^0) + \left(1 - \frac{m}{m+1}\right) y_{m+1}^0 \\ &= \frac{1}{m+1}(y_1^0 + \cdots + y_m^0) + \frac{m+1-m}{m+1} y_{m+1}^0 \\ &= \frac{1}{m+1}(y_1^0 + \cdots + y_m^0 + y_{m+1}^0)\end{aligned} \tag{7}$$

This proves the induction hypothesis and the lemma ■