ULTRA-FAST ROBUST COMPRESSIVE SENSING BASED ON MEMRISTOR CROSSBARS

Sijia Liu[†] *Ao Ren*[‡] *Yanzhi Wang*[‡] *Pramod K. Varshney*[‡]

[†]Department of EECS, University of Michigan, Ann Arbor, MI 48109, USA
[‡]Department of EECS, Syracuse University, Syracuse, NY 13244, USA
[†]lsjxjtu@umich.edu [‡]{aren, ywang393, varshney}@syr.edu

***Abstract*—In this paper, we propose a new approach for robust compressive sensing (CS) using memristor crossbars that are constructed by recently invented memristor devices. The exciting features of a memristor crossbar, such as high density, low power and great scalability, make it a promising candidate to perform large-scale matrix operations. To apply memristor crossbars to solve a robust CS problem, the alternating directions method of multipliers (ADMM) is employed to split the original problem into subproblems that involve the solution of systems of linear equations. A system of linear equations can then be solved using memristor crossbars with astonishing $O(1)$ time complexity. We also study the impact of hardware variations on the memristor crossbar based CS solver from both theoretical and practical points of view. The resulting overall complexity is given by $O(n)$, which achieves $O(n^{2.5})$ speed-up compared to the state-of-the-art software approach. Numerical results are provided to illustrate the effectiveness of the proposed CS solver.**

***Keywords*—Memristor crossbars, sparsity, compressive sensing, alternating direction method of multipliers, signal processing hardware.**

I. INTRODUCTION

Compressive sensing (CS) has received a great deal of attention over the past decade, and has achieved tremendous success in a wide range of applications such as medical imaging, radar, spectrum sensing, and data gathering from large wireless sensor networks [1]. CS attempts to perform sampling and compression simultaneously, and under mild conditions, allows the recovery of a high dimensional sparse signal from low dimensional (noiseless or noisy) observations [2]. In [3], [4], it was established that exact recovery of sparse signals from *noiseless* observations is possible using simple linear programming (LP) which yields $O(n^3)$ complexity, where n is the dimension of the signal to be recovered. In order to improve the computational efficiency, subspace pursuit [5] and expander graph [6] based methods were proposed for exact recovery with complexity $O(n) \sim O(n^2)$. However, when the measurements are *noisy*, signal recovery from imperfect measurements (also known as robust CS [7]) becomes involved, and requires the solution of a special instance of a second-order cone program (SOCP) [2]. This leads to a higher computational complexity $O(n^{3.5})$ than that of LP [8]. In this paper, we aim to develop an ultra-efficient solver for robust CS.

Previous research efforts [1]–[7] focused on software-based approaches for sparse signal recovery, with the support of CPUs. However, the conventional implementation of CS algorithms running on CPUs suffers from limited scalability due to the relatively high computational complexity and memory requirements, which thereby limits the applicability of CS solutions to big data problems. Therefore, it is attractive to design a low-complexity and ultra-efficient hardware architecture for CS algorithms. Beyond traditional CMOS designs, this paper utilizes memristor crossbars (one type of non-volatile memory device) to achieve significant speed-up for robust CS compared to the state-of-the-art software approach.

The memristor was recently invented by HP Labs [9] and has received significant attention as a potential building block for neuromorphic computing systems [10], based on its unique property to change and record its own state. Memristors offer the ability to construct a dense, continuously programmable, and reasonably accurate cross-point array architecture, known as memristor crossbar [11]. It has be shown in [12] that a memristor crossbar can be utilized to effectively perform matrix-vector multiplications and solve a system of linear equations with an astonishing O(1) complexity. The computational merit of memristor crossbars has been stressed in various applications such as linear solvers [12], dictionary learning [13], autoencoder [14], and unsupervised learning [15].

In this work, we propose a low-complexity and ultra-efficient memristor-based CS solver. In spite of its computational advantages, there exist two challenges from the perspective of hardware implementation. First, the memristor crossbar can only perform computations involving *square* matrices with *nonnegative* entries, since the memristance is always nonnegative. Second, the memristor crossbar suffers from *hardware variations*, which degrade the accuracy of CS solutions. To the best of our knowledge, it is the first time that memristors are being used for CS, and the aforementioned hardware obstacles are addressed from the perspective of algorithm-hardware co-optimization.

The major contributions of this paper are two folds. First, we successfully implement memristor crossbars to solve robust CS problems with the help of an operator splitting method, the alternating directions method of multipliers (ADMM). The advantage of ADMM is that we are able to extract subproblems that involve the solution of systems of linear equations from the solution of robust CS. Memristor crossbars can then be utilized to design the required linear solver. This is different from the literature [14], [15], where memristor crossbars were recently adopted for implementing the gradient descent algorithm. Second, we propose an efficient method to construct a valid memristor conductance matrix by eliminating its negative entries without loss of performance. We also show that the proposed memristor crossbar based CS solver is quite robust to hardware variations and achieves $O(n^{2.5})$ speed-up compared to the conventional SOCP running on CPUs.

II. PRELIMINARIES: ROBUST CS AND MEMRISTOR CROSSBAR

In this section, we briefly introduce the problem of robust compressive sensing and the utility of memristor crossbars.

A. Robust CS

Let $\mathbf{x}_* \in \mathbb{R}^n$ be a sparse or compressible vector (e.g., a digital signal or image) to be recovered. We have access to measurements $\mathbf{y} \in \mathbb{R}^m$ via

$$\mathbf{y} = \mathbf{A}\mathbf{x}_* + \mathbf{v}, \tag{1}$$

where $\mathbf{A} \in \mathbb{R}^{m \times n}$ is a given measurement matrix (e.g., random Gaussian matrix), $\mathbf{v}$ is a stochastic or deterministic error term with bounded energy $\|\mathbf{v}\|_2 \leq \epsilon$, and $\|\cdot\|_2$ denotes the ℓ_2 norm of a vector. Here we focus on the scenario in which $m \ll n$, rendering $\mathbf{A}$ non-invertible.

S. Liu was with Syracuse University. Now he is with University of Michigan. This work was supported in part by ARO grant number W911NF-14-1-0339 and NSF CCF-1637559.

The major task of robust CS is to stably recover the unknown sparse signal $\mathbf{x}_*$ from the noisy measurements $\mathbf{y}$. It has been shown in [16] that a stable recovery can be achieved in polynomial time by solving the convex optimization problem

$$\begin{array}{ll} \text{minimize} & \|\mathbf{x}\|_1 \\ \text{subject to} & \|\mathbf{A}\mathbf{x}-\mathbf{y}\|_2 \le \epsilon, \end{array} \tag{2}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the optimization variable, and $\|\cdot\|_1$ denotes the ℓ_1 norm of a vector. We note that problem (2) is a special instance of a second order cone program [17], which is typically solved by the interior-point algorithm [2].

B. Memristor Crossbar

A Memristor has the unique property to be able to record the historical profile of the excitations on the device. More specifically, the state (memristance) of a memristor will change when a certain voltage higher than a threshold voltage is applied at its two terminals. Otherwise, the memristor behaves like a resistor. This unique property makes it an ideal candidate for non-volatile memory and matrix computations [18], [19]. Physical memristors can be fabricated in a high density grid, known as a crossbar [11].

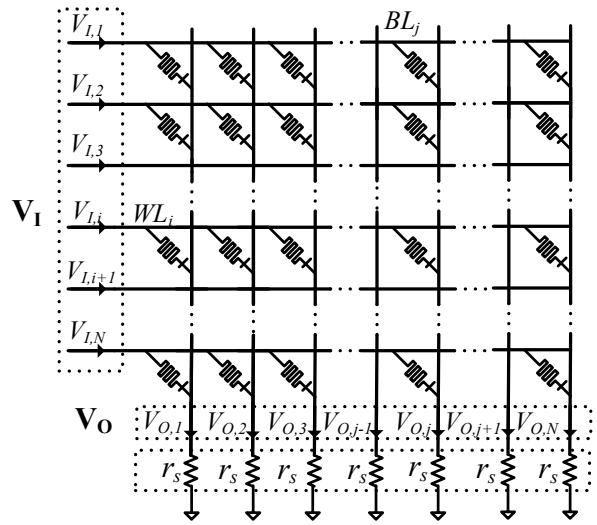

Fig. 1: A memristor crossbar.

A typical $N \times N$ memristor crossbar is illustrated in Fig. 1, where a memristor is connected between each pair of horizontal word-line (WL) and vertical bit-line (BL) [20]. This structure could be implemented with a small footprint and is capable of re-programming each memristor to different resistance states by properly applying biasing voltages at its two terminals [21], [22]. To demonstrate the matrix computation functionality, we apply a vector of input voltages $\mathbf{V}_{\rm I}$ on WLs and collect the current through each BL by measuring the voltage across resistor r_s with conductance of g_s. Assume that the memristor at the connection between WL_i and BL_j has a conductance of $g_{i,j}$. Then the output voltages can be represented by

$$\mathbf{V}_{\rm O} = \mathbf{C}\mathbf{V}_{\rm I}, \tag{3}$$

where $\mathbf{C} \in \mathbb{R}^{N\times N}$ is determined by the conductance of memristors (g_s and $\{g_{i,j}\}$); see [20, Eq. 5]. It is clear from (3) that a programmed memristor crossbar (with certain conductance matrix $\mathbf{C}$) can be used to perform matrix-vector multiplication with a constant-time complexity $O(1)$. Note that the entries of $\mathbf{C}$ are restricted to nonnegative values.

As the reverse operation, the memristor crossbar can be used to solve a system of linear equations. A voltage vector $\mathbf{V}_{\rm O}$ is applied on each r_s of BL, and the resulting current flow through each BL can be approximated as $I_{{\rm O},j} = g_s V_{{\rm O},j}$. On the other hand, the current $I_{{\rm O},j}$ through BL_j can also be calculated as $I_{{\rm O},j} = \sum_i V_{{\rm I},i} g_{i,j}$. Hence, for each BL_j, equation $V_{{\rm O},j} = (1/g_s)\sum_i V_{{\rm I},i} g_{i,j}$ is mapped. Namely, the system of linear equations (3) is mapped back to the memristor crossbar, where the solution $\mathbf{V}_{\rm I}$ can be determined by measuring voltages on the WLs. This implies that the memristor crossbar can be used as a linear solver under the conditions that $\mathbf{C}$ is a square matrix with nonnegative entries.

III. Memristor Crossbar Based Robust CS via ADMM

The key step to successfully apply memristor crossbars into CS is to extract subproblems that solve systems of linear equations leading to the solution of the CS problem. In this section, we will show that ADMM provides a suitable optimization framework to effectively split problem (2) into a sequence of subproblems that require the solution of systems of linear equations.

We begin by reformulating problem (2) in a way that lends itself to the application of ADMM,

$$\begin{array}{ll} \text{minimize} & \|\mathbf{w}\|_1 + \mathcal{I}_1(\mathbf{u}) + \mathcal{I}_2(\mathbf{x},\mathbf{s}) \\ \text{subject to} & \mathbf{x}-\mathbf{w}=\mathbf{0}, \quad \mathbf{s}-\mathbf{u}=\mathbf{0}, \end{array} \tag{4}$$

where $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{w} \in \mathbb{R}^n$, $\mathbf{u} \in \mathbb{R}^m$ and $\mathbf{s} \in \mathbb{R}^m$ are optimization variables, and $\mathcal{I}_1$ and $\mathcal{I}_2$ are indicator functions corresponding to constraints of problem (2), namely,

$$\mathcal{I}_1(\mathbf{u}) = \begin{cases} 0 & \|\mathbf{u}\|_2 \le \epsilon \\ \infty & \text{otherwise}, \end{cases} \tag{5}$$

and

$$\mathcal{I}_2(\mathbf{x},\mathbf{s}) = \begin{cases} 0 & \mathbf{A}\mathbf{x}-\mathbf{s}-\mathbf{y}=0 \\ \infty & \text{otherwise}. \end{cases} \tag{6}$$

As will be evident later, the newly introduced optimization variables $\mathbf{w}$, $\mathbf{u}$ and $\mathbf{s}$ together with the indicator functions (5)–(6) in (4) help to extract the character of ℓ_1 norm, Euclidean ball constraint, and linear equality constraints from the original problem (2).

ADMM is performed based on the augmented Lagrangian [23] of problem (4), which is given by

$$\begin{aligned} \mathcal{L}(\mathbf{x},\mathbf{s},\mathbf{w},\mathbf{u},\boldsymbol{\mu},\boldsymbol{\nu}) &= \|\mathbf{w}\|_1 + \mathcal{I}_1(\mathbf{u}) + \mathcal{I}_2(\mathbf{x},\mathbf{s}) + \boldsymbol{\mu}^T(\mathbf{x}-\mathbf{w}) \\ &\quad + \frac{\rho}{2}\|\mathbf{x}-\mathbf{w}\|_2^2 + \boldsymbol{\nu}^T(\mathbf{s}-\mathbf{u}) + \frac{\rho}{2}\|\mathbf{s}-\mathbf{u}\|_2^2, \end{aligned} \tag{7}$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ are Lagrangian multipliers (also known as dual variables), and $\rho > 0$ is a given regularization parameter associated with quadratic augmented terms.

ADMM iteratively executes the following three steps [23] for iteration $k = 1, 2, \ldots$

$$\{\mathbf{x}^{k+1}, \mathbf{s}^{k+1}\} = \arg\min_{\mathbf{x},\mathbf{s}} \mathcal{L}(\mathbf{x},\mathbf{s},\mathbf{w}^k,\mathbf{u}^k,\boldsymbol{\mu}^k,\boldsymbol{\nu}^k) \tag{8}$$

$$\{\mathbf{w}^{k+1}, \mathbf{u}^{k+1}\} = \arg\min_{\mathbf{w},\mathbf{u}} \mathcal{L}(\mathbf{x}^{k+1},\mathbf{s}^{k+1},\mathbf{w},\mathbf{u},\boldsymbol{\mu}^k,\boldsymbol{\nu}^k) \tag{9}$$

$$\boldsymbol{\mu}^{k+1} = \boldsymbol{\mu}^k + \rho(\mathbf{x}^{k+1}-\mathbf{w}^{k+1}), \ \boldsymbol{\nu}^{k+1} = \boldsymbol{\nu}^k + \rho(\mathbf{s}^{k+1}-\mathbf{u}^{k+1}),$$

until the stopping conditions are satisfied, $\|\mathbf{x}^k-\mathbf{w}^k\|_2 + \|\mathbf{s}^k-\mathbf{u}^k\|_2 \le \xi$ and $\|\mathbf{x}^{k+1}-\mathbf{x}^k\|_2 + \|\mathbf{s}^{k+1}-\mathbf{s}^k\|_2 \le \xi$, where ξ is a given stopping tolerance parameter.

We emphasize that the crucial property of ADMM is that, as we demonstrate in the rest of this section, the solution of problem (8) can be found by using memristor crossbars with low complexity, and the solution of problem (9) only involves elementary vector operations.

A. Memristor-based solution to problem (8)

After completing squares with respect to $\mathbf{x}$ and $\mathbf{s}$ in (7), problem (8) becomes

$$\begin{array}{ll}\underset{\mathbf{x},\mathbf{s}}{\text{minimize}} & \frac{\rho}{2}\|\mathbf{x}-\mathbf{a}\|_2^2+\frac{\rho}{2}\|\mathbf{s}-\mathbf{b}\|_2^2 \\ \text{subject to} & \mathbf{A}\mathbf{x}-\mathbf{s}=\mathbf{y},\end{array} \tag{10}$$

where $\mathbf{a}:=\mathbf{w}^k-(1/\rho)\boldsymbol{\mu}^k$, and $\mathbf{b}:=\mathbf{u}^k-(1/\rho)\boldsymbol{\nu}^k$.

The optimality condition of problem (10) can be obtained by setting the first-order derivative of the Lagrangian function of problem (10) equal to zero. This yields

$$\rho\mathbf{x}+\mathbf{A}^T\boldsymbol{\lambda}=\rho\mathbf{a},\ \rho\mathbf{s}-\boldsymbol{\lambda}=\rho\mathbf{b},\ \mathbf{A}\mathbf{x}-\mathbf{s}=\mathbf{y}, \tag{11}$$

where $\boldsymbol{\lambda}\in\mathbb{R}^m$ is the Lagrangian multiplier. From (11), the solution of problem (10) is then given by

$$\begin{bmatrix}\rho\mathbf{I}_n & \mathbf{0} & \mathbf{A}^T\\ \mathbf{0} & \rho\mathbf{I}_m & -\mathbf{I}_m\\ \mathbf{A} & -\mathbf{I}_m & \mathbf{0}\end{bmatrix}\begin{bmatrix}\mathbf{x}\\ \mathbf{s}\\ \boldsymbol{\lambda}\end{bmatrix}=\begin{bmatrix}\rho\mathbf{a}\\ \rho\mathbf{b}\\ \mathbf{y}\end{bmatrix},\Longleftrightarrow\ \mathbf{C}\mathbf{z}=\mathbf{d}, \tag{12}$$

where $\mathbf{I}_n$ denotes an identity matrix of size n, $\mathbf{z}:=[\mathbf{x}^T,\mathbf{s}^T,\boldsymbol{\lambda}^T]^T$ is the vector of primal and dual variables of problem (10), and $\mathbf{d}:=[\rho\mathbf{a}^T,\rho\mathbf{b}^T,\mathbf{y}^T]^T$. In (12), $\mathbf{C}$ corresponds to the coefficient matrix of the left equation, which is invertible (due to the theory of Schur complement) and is independent of the ADMM iteration.

Note that the analytical solution of (12) is given by $\mathbf{z}=\mathbf{C}^{-1}\mathbf{d}$, which requires $O(n^3)$ arithmetic operations due to the presence of matrix inversion. To further reduce the computational complexity, a memristor crossbar could be utilized to solve (12) by configuring memristance values according to the matrix $\mathbf{C}$ and applying $\mathbf{d}$ at the output of the memristor crossbar in Fig. 1. Since there may exist negative entries in $\mathbf{C}$, in what follows we propose a novel matrix representation that eliminates those negative entries and maintains valid memristance values.

By introducing auxiliary variables $\bar{\mathbf{z}}$, we rewrite (12) as the following system of linear equations

$$\begin{bmatrix}(\mathbf{C})_+ & \mathbf{B}\\ \mathbf{D} & \mathbf{I}_{\bar{n}}\end{bmatrix}\begin{bmatrix}\mathbf{z}\\ \bar{\mathbf{z}}\end{bmatrix}=\begin{bmatrix}\mathbf{d}\\ \mathbf{0}_{\bar{n}}\end{bmatrix}, \tag{13}$$

where $\bar{\mathbf{z}}\in\mathbb{R}^{\bar{n}}$, $\bar{n}$ is the number of columns of $\mathbf{C}$ that contain negative elements, $(x)_+=\max\{0,x\}$ is a positive operator defined in an elementwise fashion for a matrix argument, $\mathbf{B}\in\mathbb{R}^{n\times\bar{n}}$ is a submatrix of $(-\mathbf{C})_+$ after the columns of all zeros are removed, $\mathbf{D}\in\mathbb{R}^{\bar{n}\times n}$ is a submatrix of $\mathbf{I}_n$ after the rows, indexed by the columns of $\mathbf{C}$ that only contain nonnegative elements, are removed, and $\mathbf{0}_{\bar{n}}$ is a zero vector of size $\bar{n}$. We note that if $\bar{n}=0$, the linear system (13) reduces to (12).

For ease of notation, we define $\mathbf{Q}:=\begin{bmatrix}(\mathbf{C})_+ & \mathbf{B}\\ \mathbf{D} & \mathbf{I}_{\bar{n}}\end{bmatrix}$. We illustrate the construction of $\mathbf{Q}$ through the following example with $n=3$,

$$\mathbf{C}=\begin{bmatrix}2 & -0.1 & 0.1\\ -0.1 & 2 & 0.1\\ 0.1 & 0.1 & 2\end{bmatrix}\Rightarrow(\mathbf{C})_+=\begin{bmatrix}2 & 0 & 0.1\\ 0 & 2 & 0.1\\ 0.1 & 0.1 & 2\end{bmatrix},$$

$$(-\mathbf{C})_+=\begin{bmatrix}0 & 0.1 & 0\\ 0.1 & 0 & 0\\ 0 & 0 & 0\end{bmatrix},\ \mathbf{B}=\begin{bmatrix}0 & 0.1\\ 0.1 & 0\\ 0 & 0\end{bmatrix},\ \mathbf{D}=\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\end{bmatrix},$$

$$\mathbf{I}_{\bar{n}}=\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix},\ \mathbf{Q}=\begin{bmatrix}2 & 0 & 0.1 & 0 & 0.1\\ 0 & 2 & 0.1 & 0.1 & 0\\ 0.1 & 0.1 & 2 & 0 & 0\\ 1 & 0 & 0 & 1 & 0\\ 0 & 1 & 0 & 0 & 1\end{bmatrix}. \tag{14}$$

Based on (13), we can configure the memristance values (according to $\mathbf{Q}$) and utilize the memristor crossbar to obtain the solution of problem (8). As demonstrated in Sec. II-B, a programmed memristor crossbar only requires a constant-time complexity $O(1)$ to solve a system of linear equations.

B. Solution to problem (9)

After completing squares with respect to $\mathbf{w}$ and $\mathbf{u}$ in (7), problem (9) becomes

$$\underset{\mathbf{w},\mathbf{u}}{\text{minimize}}\quad \|\mathbf{w}\|_1+\mathcal{I}(\mathbf{u})+\frac{\rho}{2}\|\mathbf{w}-\mathbf{p}\|_2^2+\frac{\rho}{2}\|\mathbf{u}-\mathbf{q}\|_2^2, \tag{15}$$

where $\mathbf{p}:=\mathbf{x}^{k+1}+(1/\rho)\boldsymbol{\mu}^k$ and $\mathbf{q}:=\mathbf{A}\mathbf{x}^{k+1}-\mathbf{y}+(1/\rho)\boldsymbol{\nu}^k$.

Problem (15) can be further decomposed into two subproblems

$$\text{minimize}\quad \|\mathbf{w}\|_1+\frac{\rho}{2}\|\mathbf{w}-\mathbf{p}\|_2^2, \tag{16}$$

and

$$\text{minimize}\ \|\mathbf{u}-\mathbf{q}\|_2^2,\quad \text{subject to}\ \|\mathbf{u}\|_2\le\epsilon. \tag{17}$$

The solution to problem (16) is given by the soft thresholding operator [24]

$$\mathbf{w}=(\mathbf{p}-1/\rho\mathbf{1})_+-(-\mathbf{p}-1/\rho\mathbf{1})_+, \tag{18}$$

where $\mathbf{1}\in\mathbb{R}^n$ is a vector of all ones.

The solution of problem (17) is given by projecting $\mathbf{q}$ onto an Euclidean ball,

$$\mathbf{u}=\min\{\epsilon,\|\mathbf{q}\|_2\}\frac{\mathbf{q}}{\|\mathbf{q}\|_2}. \tag{19}$$

We remark that the difficulty of hardware-based calculation of (18)–(19) lies in calculating the ℓ_2-norm of a vector, which can be realized using peripheral circuits including analog multipliers and summing amplifiers in the analog domain [25], [26]. Or we can convert the vector to the digital domain and then calculate its norm.

Following (8) and (9), the solution of robust CS is readily obtained by using memristor crossbars and elementary hardware-based vector computations. In the next section, we will elaborate on the algorithm complexity and the impact of hardware variations on the memristor-based CS solution.

IV. Computational Complexity and Device Variations

ADMM has a linear convergence rate $O(1/k)$ for general convex optimization problems [27], where k is the number of iterations. In other words, given the stopping tolerance ξ, ADMM requires $O(1/\xi)$ iterations to converge. However, it is often the case that ADMM converges faster to provide modest accuracy that is sufficient for many applications [23], [24], [28]–[30]. At each iteration of ADMM, the computational complexity is approximated by $O(n)$, where we obtain a) $O(1)$ complexity by utilizing memristor crossbars to solve a system of linear equations (13), and b) $O(n)$ complexity for vector operations in (18) and (19). In the big data context with large n, the memristor crossbar based CS solver yields the complexity $O(n)$. Compared to the software approach, e.g., SOCP with complexity $O(n^{3.5})$, the proposed solver achieves $O(n^{2.5})$ speed-up.

The hardware design of the proposed memristor crossbar-based robust CS solver also exhibits relatively low complexity, and mainly consists of two parts. The first part is the memristor crossbar-based linear system solver to solve problem (8), in which only a single memristor crossbar is needed. The second part is the digital (or

analog) circuit-based solver of problem (9), which uses a multiplier, an accumulator, and a divider when solving iteratively in the digital domain. In addition, the updating of dual variables μ^{k+1} and γ^{k+1} involves standard addition/subtraction calculations in digital or analog domains. We stress that the solution framework exhibits low hardware complexity because only one memristor crossbar is required.

It is known from [13] that the memory devices relying on the ion/defects motion typically show noticeable device variations from cell-to-cell and from cycle-to-cycle. Therefore, to turn the memristor crossbar based CS framework into real-life applications, we face the following obstacle. The presence of hardware variations leads to a reduced reading accuracy of $\mathbf{Q}$ configured in memristor crossbar arrays, where $\mathbf{Q}$ is introduced in (14). As a result, the matrix used in ADMM is actually modified to $\tilde{\mathbf{Q}} = \mathbf{Q} + \tau\mathbf{\Sigma}$, where $\tau\mathbf{\Sigma}$ represents the hardware variation, τ governs the variation strength, and $\mathbf{\Sigma}$ is τ-dependent. Based on Taylor series expansion, we obtain that

$$\tilde{\mathbf{Q}}^{-1} = \mathbf{Q}^{-1} + \tau\mathbf{Q}^{-1}\mathbf{\Sigma}\mathbf{Q}^{-1} + O(\tau^2), \tag{20}$$

which holds when τ is a small number. It is clear from (20) that the matrix inversion $\tilde{\mathbf{Q}}^{-1}$ is valid when the hardware variation does not dominate the matrix $\mathbf{Q}$. It is often the case that τ is small since $(\|\tilde{\mathbf{Q}} - \mathbf{Q}\|_F / \|\mathbf{Q}\|_F)$ is typically less than $5\% \sim 10\%$, where $\|\cdot\|_F$ denotes the Frobenius norm of a matrix. Our numerical results in the next section will show that although there exists a slight decrease of recovery accuracy, the obtained sparse support (i.e., sparsity pattern) of the signal of interest is almost the same for different levels of hardware variations.

V. Numerical Results

In this section, we consider a sparse signal $\mathbf{x}_*$ of length $n = 1024$ with s nonzero entries. These s nonzero spike positions are chosen randomly, and their values are chosen independently from the standard normal distribution. To specify the observation model (2), we consider $m = 300$ and generate a random $m \times n$ matrix $\mathbf{A}$ with i.i.d. entries from the standard normal distribution. The vector of measurement noises $\mathbf{v}$ is drawn from the Gaussian distribution $\mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$, where $\sigma = 0.01$. The accuracy tolerance ϵ is set to $\epsilon^2 = \sigma^2(m + 2\sqrt{2m})$, which provides a likely upper bound on $\|\mathbf{v}\|_2$ [31]. We adopt $\|\mathbf{x} - \mathbf{x}_*\|_2$ (averaged over 50 numerical trials) to measure the accuracy of the recovered sparse signal. The ADMM parameters are set to $\xi = 10^{-3}$ and $\rho = 10$.

In Fig. 2, we present the error in sparse signal recovery as a function of signal sparsity s for different levels of hardware variations. We recall from (20) that $(\|\tilde{\mathbf{Q}} - \mathbf{Q}\|_F / \|\mathbf{Q}\|_F)$ provides the level of hardware variations. For comparison, we also employ the commonly-used orthogonal matching pursuit (OMP) algorithm [32] (MATLAB toolbox designed by [33]) to recover the sparse signal $\mathbf{x}_*$. In both ADMM and OMP, we set the maximum number of iterations as 1000. We observe that the recovery accuracy improves as s decreases (namely, the number of nonzero entries in the sparse signal decreases). This is not surprising, since the sparsity prior plays a key role in stable signal recovery at a rate much smaller than what is commonly prescribed by Shannon-Nyquist [2]. We also note that the recovery accuracy of using the memristor-based CS solver is comparable to that of using OMP even in the presence of hardware variations. By fixing s, we see that the recovery accuracy decreases while increasing the level of hardware variations. To clearly show the performance loss, in Fig. 3 we compare the recovered signal under 0% or 5% hardware variation with the original sparse signal to be recovered when $s = 105$. Clearly, the recovered signal yields almost the same sparse support as that of the original signal even if there exists 5%

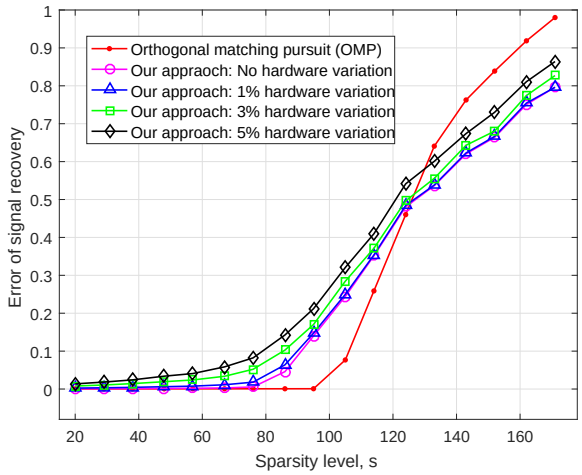

Fig. 2: Recovery accuracy versus signal sparsity s for different levels of hardware variations.

hardware variation. The promising results show that the memristor-based CS solver is quite robust to hardware variations, and is able to provide reliable signal recovery performance compared to the original sparse signal and the software solution.

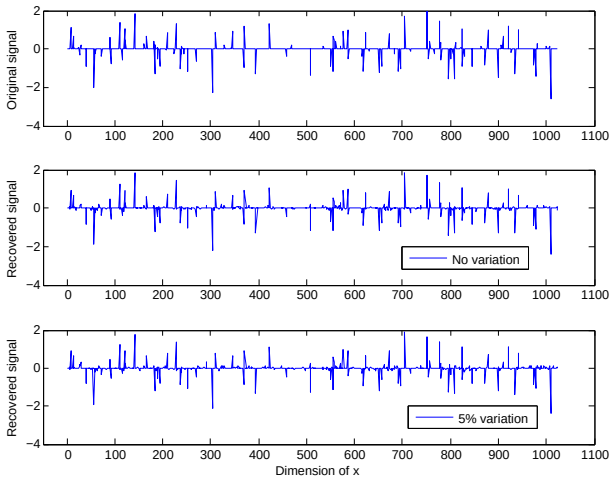

Fig. 3: Recovered signals versus original sparse signal for different levels of hardware variations.

VI. Conclusion

In this paper, we have developed a novel CS framework using memristor crossbars. To be more specific, we first employed ADMM to extract systems of linear equations from the solution of robust CS problems. A memristor crossbar array is then utilized to solve the resulting linear equations with astonishing $O(1)$ time complexity. Moreover, we elaborated on the impact of device variations on the memristor-based CS solver. We demonstrated that the proposed CS solver is robust to hardware variations and achieves $O(n^{2.5})$ speed-up compared to the conventional software approach. In future work, we would like to develop a unified memristor-based CS solver that addresses the problem of robust CS as well as 1-bit CS. We also would like to develop a real hardware system on chip for CS using memristor devices.

REFERENCES

[1] S. Qaisar, R. M. Bilal, W. Iqbal, M. Naureen, and S. Lee, "Compressive sensing: From theory to applications, a survey," *Journal of Communications and Networks*, vol. 15, no. 5, pp. 443–456, Oct. 2013.

[2] E. Candès, J. Romberg, and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements," *Comm. Pure and Applied Math.*, vol. 59, no. 8, pp. 1207–1223, 2006.

[3] D. L. Donoho, "Compressed sensing," *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, April 2006.

[4] E. J. Candes, J. Romberg, and T. Tao, "Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information," *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 489–509, Feb. 2006.

[5] W. Dai and O. Milenkovic, "Subspace pursuit for compressive sensing signal reconstruction," *IEEE Transactions on Information Theory*, vol. 55, no. 5, pp. 2230–2249, May 2009.

[6] W. Xu and B. Hassibi, "Efficient compressive sensing with deterministic guarantees using expander graphs," in *Information Theory Workshop, 2007. ITW '07. IEEE*, Sept. 2007, pp. 414–419.

[7] E. J. Candes and M. B. Wakin, "An introduction to compressive sampling," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 21–30, March 2008.

[8] A. Nemirovski, "Interior point polynomial time methods in convex programming," 2012 [Online], Available: *http://www2.isye.gatech.edu/~nemirovs/Lect_IPM.pdf*.

[9] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *Nature*, vol. 453, no. 7191, pp. 80–83, 05 2008.

[10] D. Chabi, W. Zhao, D. Querlioz, and J. O. Klein, "Robust neural logic block (nlb) based on memristor crossbar array," in *2011 IEEE/ACM International Symposium on Nanoscale Architectures*, June 2011, pp. 137–143.

[11] S. H. Jo, K.-H. Kim, and W. Lu, "High-density crossbar arrays based on a si memristive system," *Nano Letters*, vol. 9, no. 2, pp. 870–874, 2009.

[12] I. Richter, K. Pas, X. Guo, R. Patel, J. Liu, E. Ipek, and E. G. Friedman, "Memristive accelerator for extreme scale linear solvers," 2015.

[13] P. Y. Chen, D. Kadetotad, Z. Xu, A. Mohanty, B. Lin, J. Ye, S. Vrudhula, J. S. Seo, Y. Cao, and S. Yu, "Technology-design co-optimization of resistive cross-point array for accelerating learning algorithms on chip," in *Proc. 2015 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2015, pp. 854–859.

[14] B. Li, Y. Wang, Y. Wang, Y. Chen, and H. Yang, "Training itself: Mixed-signal training acceleration for memristor-based neural network," in *2014 19th Asia and South Pacific Design Automation Conference (ASP-DAC)*, Jan 2014, pp. 361–366.

[15] L. Chen, C. Li, T. Huang, Y. Chen, and X. Wang, "Memristor crossbar-based unsupervised image learning," *Neural Computing and Applications*, vol. 25, no. 2, pp. 393–400, 2014.

[16] E. Candès, "Compressive sampling," in *Proc. International Congress of Mathematicians*, 2006.

[17] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, Cambridge, 2004.

[18] S. H. Jo, T. Chang, I. Ebong, B. B. Bhadviya, P. Mazumder, and W. Lu, "Nanoscale memristor device as synapse in neuromorphic systems," *Nano Letters*, vol. 10, no. 4, pp. 1297–1301, 2010.

[19] M. Di Ventra, Y. V. Pershin, and L. O. Chua, "Circuit elements with memory: Memristors, memcapacitors, and meminductors," *Proceedings of the IEEE*, vol. 97, no. 10, pp. 1717–1724, Oct. 2009.

[20] M. Hu, H. Li, Q. Wu, G. S. Rose, and Y. Chen, "Memristor crossbar based hardware realization of bsb recall function," in *Proc. The 2012 International Joint Conference on Neural Networks (IJCNN)*, June 2012, pp. 1–7.

[21] A. Heittmann and T. G. Noll, "Limits of writing multivalued resistances in passive nanoelectronic crossbars used in neuromorphic circuits," in *Proceedings of the Great Lakes Symposium on VLSI*, 2012, pp. 227–232.

[22] D. Kadetotad, Z. Xu, A. Mohanty, P.-Y. Chen, B. Lin, J. Ye, S. Vrudhula, S. Yu, Y. Cao, and J. Seo, "Neurophysics-inspired parallel architecture with resistive crosspoint array for dictionary learning," in *Proc. IEEE Biomedical Circuits and Systems Conference (BioCAS)*, Oct 2014, pp. 536–539.

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

[24] N. Parikh and S. Boyd, "Proximal algorithms," *Foundations and Trends in Optimization*, vol. 1, no. 3, pp. 123–231, 2013.

[25] M. Hu, H. Li, Y. Chen, Q. Wu, and G. S. Rose, "Bsb training scheme implementation on memristor-based circuit," in *Proc. IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, April 2013, pp. 80–87.

[26] W. Wen, C. R. Wu, X. Hu, B. Liu, T. Y. Ho, X. Li, and Y. Chen, "An eda framework for large scale hybrid neuromorphic computing systems," in *Proc. 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, June 2015, pp. 1–6.

[27] B. He and X. Yuan, "On the $O(1/n)$ convergence rate of the douglas-rachford alternating direction method," *SIAM Journal on Numerical Analysis*, vol. 50, no. 2, pp. 700–709, 2012.

[28] S. Liu, S. Kar, M. Fardad, and P. K. Varshney, "Sparsity-aware sensor collaboration for linear coherent estimation," *IEEE Transactions on Signal Processing*, vol. 63, no. 10, pp. 2582–2596, May 2015.

[29] S. Liu, M. Fardad, E. Masazade, and P. K. Varshney, "Optimal periodic sensor scheduling in networks of dynamical systems," *IEEE Trans. Signal Process.*, vol. 62, no. 12, pp. 3055–3068, June 2014.

[30] F. Lin, M. Fardad, and M. R. Jovanović, "Augmented lagrangian approach to design of structured optimal state feedback gains," *IEEE Transactions on Automatic Control*, vol. 56, no. 12, pp. 2923–2929, Dec. 2011.

[31] E. Candes, M. Wakin, and S. Boyd, "Enhancing sparsity by reweighted ℓ_1 minimization," *Journal of Fourier Analysis and Applications*, vol. 14, pp. 877–905, 2008.

[32] J. A. Tropp and A. C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Trans. Inf. Theor.*, vol. 53, no. 12, pp. 4655–4666, 2007.

[33] S. Becker, "Cosamp and omp for sparse recovery," *https://www.mathworks.com/matlabcentral/fileexchange/32402-cosamp-and-omp-for-sparse-recovery/content/OMP.m*, Aug. 2011.