Fixed-point iterative linear inverse solver with extended precision

Zheyuan Zhu*, Andrew B. Klein, Guifang Li & Sean Pang

Solving linear systems, often accomplished by iterative algorithms, is a ubiquitous task in science and engineering. To accommodate the dynamic range and precision requirements, these iterative solvers are carried out on floating-point processing units, which are not efficient in handling large-scale matrix multiplications and inversions. Low-precision, fixed-point digital or analog processors consume only a fraction of the energy per operation than their floating-point counterparts, yet their current usages exclude iterative solvers due to the cumulative computational errors arising from fixed-point arithmetic. In this work, we show that for a simple iterative algorithm, such as Richardson iteration, using a fixed-point processor can provide the same convergence rate and achieve solutions beyond its native precision when combined with residual iteration. These results indicate that power-efficient computing platforms consisting of analog computing devices can be used to solve a broad range of problems without compromising the speed or precision.

Floating-point digital processing is the standard computing paradigm in virtually all scientific and engineering applications, such as image processing, compressive sensing, and predictive control systems. The cores of these applications often involve large-scale linear systems, which require multiplying and/or inverting matrices with thousands to millions of elements. A conventional digital floating-point processor, such as a single CPU core, processes the matrix multiplication as sequential multiply-accumulate (MAC) operations, which are not efficient in handling large-scale linear systems and could yield minutes to hours of computation time even for a moderate-scale problem.

Driven by the growing demand for data processing power, digital accelerators that implement a large array of parallel, half-precision floating-point processing units (FPUs), such as Google TPU and the Tensor Cores in NVIDIA Volta GPU, have been developed, particularly for artificial intelligence applications. Due to the on-chip power consumption limit, fixed-point digital MAC unit that requires only a fraction of the energy of the FPU becomes increasingly favorable in large-scale digital accelerators. Recently, analog accelerators that utilize natural physical processes for array multiplications and summations have shown to be even faster and more energy-efficient for matrix processing, with implementations on both electronic and photonic platforms. Yet, analog accelerators are susceptible to signal artifacts arising from noises and device imperfections, and are unable to achieve floating-point precision. As a result, fixed-point is the de-facto data format in both analog and digital accelerators.

Currently, the applications of analog or digital fixed-point accelerators are limited to problems robust against computing errors, such as neural network inference, adaptive filtering, and binary optimization. On the other hand, many computing tasks in science and engineering are inverse problems without analytical solutions, which must be solved iteratively. For these problems, floating-point precision is typically required to cache the intermediate results, because low-precision matrix calculations can introduce cumulative errors and stall the iterative process. The precision requirement excludes many resource-demanding computing applications from deployment on analog or digital fixed-point accelerators.

Extending the iterative linear solver to fixed-point data format can effectively leverage the high speed and efficiency of analog accelerators for a broad range of computing problems in science and engineering. However, the errors of fixed-point arithmetic and the precision of the result in both the forward and inverse processes of a linear system have not been quantitatively studied. This work examines the properties of an iterative linear inverse algorithm (Richardson iteration) in which the matrix computations are performed with fixed-point arithmetic. Using residual iterations for error correction, the precision of the iterative solution can be theoretically extended to any desired level. Similar methods have been previously proposed for correcting the computing errors in reduced precision floating-point processors or hybrid analog–digital processors, but quantitative analyses on the error and convergence of a fixed-point solver are still absent. Here we introduce two theorems describing the convergence criteria and error of a fixed-point Richardson solver. The theorems have two major implications:

CREOL, College of Optics and Photonics, University of Central Florida, Orlando, FL 32816, USA. *email: zyzhu@knights.ucf.edu
(1) the fixed-point solver does not compromise the convergence rate, and (2) using residual iterations, the fixed-point solver can reach solutions beyond its native precision limit. We verify the two theorems with three examples, namely, matrix inversion, image deconvolution, and tomography reconstruction. We also present a prototype fixed-point iterative solver implemented on a field programable gate array (FPGA), which reaches the solution in the same number of iterations as a floating-point CPU and reduces the calculation time by two folds.

**Theory**

**Iterative solution of linear system.** The problem of solving a linear system aims to deduce the inputs, \( \mathbf{x} \), that generate the observations, \( \mathbf{y} \), given the physical model of the system, expressed by a matrix \( \mathbf{A} \). Due to the size of the system matrix and noises in the measurement process, direct inverse of the observations \( \mathbf{y} \) is usually impractical and, in many cases, impossible. As a result, many inverse algorithms rely on iterative refinements to approximate the solution \(^3\). Assuming that the measurements are dominated by Gaussian noise, finding the solution \( \hat{\mathbf{x}} \) given the observations \( \mathbf{y} \) amounts to minimizing the objective function \(^3\),

\[
\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \frac{1}{2} ||\mathbf{y} - \mathbf{Ax}||^2_2, \tag{1}
\]

where \( ||\cdot||_2 \) denotes the \( \ell^2 \) norm. The solver starts with an initial guess, \( \mathbf{x}_0 \), and moves along the gradient direction, \( \partial L/\partial \mathbf{x} \), in steps of size \( \tau \) to update the estimate. Given the objective function in Eq. (1), the update from step \( k \) to \( k+1 \) can be expressed as follows,

\[
\mathbf{x}_{k+1} = \mathbf{x}_k - \tau \mathbf{A}^T (\mathbf{Ax}_k - \mathbf{y}). \tag{2}
\]

Equation (2) can be rearranged into the form of Richardson iteration \(^4\) (Eq. (3)) by introducing a pre-calculated vector \( \mathbf{b} = \tau \mathbf{A}^T \mathbf{y} \),

\[
\mathbf{x}_{k+1} = \left( \mathbf{I} - \tau \mathbf{A}^T \mathbf{A} \right) \mathbf{x}_k + \mathbf{b}, \tag{3}
\]

where each iteration comprises two matrix–vector multiplications and two vector additions. In the following sections, we discuss the error and convergence of the iterative linear solver when Eq. (3) is carried out in fixed-point precision available to analog or digital low-precision accelerators.

**Fixed-point format.** In the fixed-point data format, a vector, matrix, or tensor is represented by an array of signed mantissas, and a global exponent is shared among all the elements. Each element, \( x_i \) in the \( L \)-bit fixed-point representation, \( \hat{x} \), of an \( N \)-element array \( \mathbf{x} \in \mathbb{R}^N \) can be expressed as,

\[
\hat{x}_i = \text{sign}(x_i) \times \text{mant}_i \times 2^{\text{expo}-(L-1)}. \tag{4}
\]

Here expo denotes the exponent indicating the number of bits above decimal point. To avoid overflow, the exponent is usually determined by the maximum element of the array,

\[
\text{expo} = \left\lfloor \log_2 \max_i (\text{abs}(x_i)) \right\rfloor, \tag{5}
\]

where \( x \) denotes the rounding to the nearest integer larger than \( x \). The \((L-1)\)-bit mantissas \( \text{mant}_i \), are calculated as,

\[
\text{mant}_i = \left\lfloor \text{abs}(x_i) \times 2^{L-1-\text{expo}} \right\rfloor, \tag{6}
\]

where \( x \) denotes the rounding to the nearest integer smaller than \( x \).

The multiplication between a fixed-point matrix \( \mathbf{M} \) and a fixed-point vector \( \hat{x} \) can be simplified as integer arithmetic between the mantissas, accompanied by bit-shifting to match the exponent of the result. Compared with the floating-point matrix–vector product \( \mathbf{y} = \mathbf{Mx} \), the \( \ell^2 \) norm of the error, \( \delta \mathbf{y} \), of the fixed-point matrix–vector product \( \hat{\mathbf{y}} = \hat{\mathbf{M}} \hat{\mathbf{x}} \) is bounded by,

\[
||\delta \mathbf{y}||_2 = ||\hat{\mathbf{M}} \hat{\mathbf{x}} - \mathbf{Mx}||_2 \leq \eta ||\mathbf{M}||_2 ||\mathbf{x}||_2, \tag{7}
\]

Here \( ||\mathbf{M}||_2 \) denote the \( \ell^2 \) norm of the operator. An upper bound of the coefficient \( \eta \) is given by \( \zeta_v + \zeta_m + 3 \zeta_v \zeta_m \) in Section S1 of the supplementary materials. The coefficients \( \zeta_v \) and \( \zeta_m \) are defined as:

\[
\zeta_v := ||\mathbf{x} - \hat{\mathbf{x}}||_2/||\mathbf{x}||_2, \tag{8}
\]

\[
\zeta_m := ||\mathbf{M} - \hat{\mathbf{M}}||_2/||\mathbf{M}||_2, \tag{9}
\]

which represent the normalized rounding errors of the fixed-point vector and matrix, respectively, and are dependent on the bit widths and exponents of the arrays. In most cases, \( \eta \) is much lower than this upper bound, depending on the distribution of elements in \( \mathbf{M} \) and \( \mathbf{x} \).

**Error and convergence criteria of fixed-point Richardson iteration.** The solution \( \mathbf{x}^* \) to a linear system \( \mathbf{y} = \mathbf{Ax} \) is unique if matrix \( \mathbf{A} \) has full rank. To ensure that the intermediate estimations in the Richardson iteration \( \{x_k\} \) converge to the solution \( \mathbf{x}^* \) as the iteration number \( k \to \infty \), all the eigenvalues of matrix \( \left( \mathbf{I} - \tau \mathbf{A}^T \mathbf{A} \right) \)
must fall within \((-1, 1)\). Since the largest and smallest eigenvalues are \(1 - \frac{\tau|A^T A|_2}{\kappa}\) and \(1 - \frac{\tau|A^T A|_2}{\kappa}\), respectively, where the operator norm \(|A^T A|_2\) equals the largest eigenvalue, and \(\kappa\) is the condition number of \(A^T A\), the choice of the step size, \(\tau\), is confined by:

\[
0 < \tau < \frac{2}{|A^T A|_2}. \tag{9}
\]

In practice, to ensure the convergence while expediting the iterative solver, the maximum step size, \(\tau_{\text{max}}\), is often selected with a safety margin \(0 < \chi \ll 1\) as follows:

\[
\tau_{\text{max}} = \frac{2 - \chi}{|A^T A|_2}. \tag{10}
\]

For fixed-point Richardson iterations, Theorem 1 below summarizes the error and convergence criteria based on the fixed-point matrix multiplication error \(\eta\).

**Theorem 1:** For a linear system \(Ax = y\) solved using fixed-point Richardson iterations (Eq. (3)) with step size \(\tau\), under the conditions in Eq. (11),

\[
0 < \tau < \frac{2}{|A^T A|_2} \quad \text{and} \quad \eta < \frac{\tau|A^T A|_2}{\kappa - \tau|A^T A|_2}, \tag{11}
\]

the asymptotic error, \(\theta\), between the estimation, \(x_k\), and the solution, \(x^*\), is bounded by:

\[
\theta = \lim_{k \to \infty} \sup \frac{|x^* - x_k|_2}{|x^*|_2} \leq \eta \left(\frac{\kappa}{\tau|A^T A|_2} - 1\right), \tag{12}
\]

where \(\kappa\) is the condition number of \(A^T A\), and \(\eta\) is the normalized \(\ell^2\) error of the two matrix–vector multiplications in each iteration.

**Algorithm I. Fixed-point residual iteration for solving the linear system \(y = Ax\)**

| Residual iteration algorithm |
|-----------------------------|
| \(x^{(0)} \leftarrow 0, r^{(0)} \leftarrow y\), |
| for \(l\) from 1 to \(M\) |
| \(\delta x_0 \leftarrow 0\) |
| \(b^{(0)} \leftarrow -r A^T r^{(0)}\) |
| // Solve \(r^{(0)} = A\delta x^{(0)}\) with fixed-point Richardson iterations |
| repeat: |
| \(\delta x_k \leftarrow \delta x_{k-1} - A^T (A \delta x_{k-1}) + b^{(0)}\) |
| until \(|\delta x_k - \delta x_{k-1}|_2 < \varepsilon\) |
| \(\delta x^{(0)} \leftarrow \delta x_k\) |
| // Solution and residue update |
| \(x^{(0)} \leftarrow x^{(l-1)} + \delta x^{(0)}\) |
| \(r^{(l)} = y - Ax^{(0)}\) |

Theorem 1 is proved in Section S2 of the supplementary materials by examining the \(\ell^2\) norm of the error of intermediate estimations, \(|x^* - x_k|_2\), as \(k \to \infty\). The proof shows a linear convergence with rate \(|I - \tau A^T A/\kappa|_2\), which is independent on the error of the fixed-point matrix–vector product, \(\eta\). Because a typical problem involves a value of \(\kappa\) on the order of \(10^2 \sim 10^3\), using the choice of step size \(\tau\) in Eq. (10), \(\tau|A^T A|_2 = 2 - \chi \ll \kappa\), we can approximate

\[
|I - \tau A^T A/\kappa|_2^k \approx \exp(-\gamma k), \tag{13}
\]

where the convergence rate \(\gamma = (2 - \chi)/\kappa\).

**Fixed-point residual iteration.** The asymptotic error \(\theta\) in Theorem 1 is independent of the solution \(x^*\), which implies that the fixed-point iterative solver can stagnate at an estimation, \(x_\infty\), closer to the solution \(x^*\) in terms of \(\ell^2\) distance, if \(x^*\) has a lower \(\ell^2\) norm. Hence, constructing a sequence of fixed-point iterative solvers in which the solutions have decreasing \(\ell^2\) norm becomes the natural choice to approach the unbiased solution of the inverse problem.
The proposed residual iteration algorithm described in Algorithm 1 leverages the adjustable exponent of the fixed-point format. The algorithm contains $M$ sets of fixed-point Richardson iterations as the inner loops. The acceptance criterion of the inner loop solution is:

$$|\delta x_k - \delta x_{k-1}|_2 < \varepsilon,$$

where $\varepsilon$ is a small number of the order $\zeta$. Let $x^{(M)} = \sum_{l=1}^{M} \delta x^{(l)}$ denote the accumulated solutions from $M$ fixed-point Richardson iterative solvers. The residual iteration algorithm aims to reduce the norm of $r^{(l)} := y - Ax^{(l)}$ at each outer loop $l$ by recursively solving the linear system $r^{(l)} = A\delta x^{(l)}$ with the updated exponents in the next inner loop of the fixed-point Richardson iterations.

It is worth noting that similar precision refinement methods have been employed in iterative solvers using mixed full- and half-precision floating-point data formats. Yet, the floating-point precision refinement typically requires far fewer residual loops, $M$, than Algorithm 1. In addition, floating-point format allows a different exponent for every element in the matrix–vector product. As a result, the error analysis in floating-point precision refinement does not apply to fixed-point residual iterations. Here we present the upper bound of the normalized error of the solution $x^{(M)}$ after $M$ fixed-point residual iterations in Theorem 2.

**Theorem 2:** For a full-rank linear system $Ax = y$ with its solution $x^*$ obtained from the residual iteration algorithm (Algorithm 1), the asymptotic error of the estimation after $M$ residue updates, $x^{(M)}$, is bounded by:

$$\theta^{(M)} \leq \theta^M,$$

where $\theta$ is the asymptotic error of the solution obtained from fixed-point Richardson solver (Eq. (12)).

Theorem 2 is established based on the decreasing $\ell^2$ norm of the solution $x^*$ in each fixed-point Richardson iteration. The upper bound of the solution error after $M$ residue updates can be derived recursively from Eq. (12) in Theorem 1. Detailed proof is presented in Section S3 of the supplementary materials.

For Eq. (15) to be valid, the exponents of $b^{(l)}$ and $A^T A \delta x_k$ should be updated after every iteration of the inner loop. In practice, due to the computational cost of distribution-based estimation of the infinity norm (Eq. (19)), the exponents are adjusted after the completion of each inner loop. As a result, the errors may not always satisfy the bound predicted by Eq. (15).

![Figure 1. Two 4 × 4 matrices (a) and their inverses (b) in fixed-point iterative matrix inversion solver. $A_1$ has a $\kappa$ of 25, and $A_2$ has a $\kappa$ of 11.1. All matrices are stored in floating-point format but displayed with two decimal points. (c) Normalized error of the iterative solutions $\theta_k$ vs. the iteration number $k$ for the inversion of $A_1$ ($\kappa = 25.0$) and $A_2$ ($\kappa = 11.1$) using 8-bit fixed-point Richardson iterations.](image-url)
Fixed-point iterative solver experiments

Matrix inversion. We first verify the convergence rate and the residue error in theorem 1 with a fixed-point matrix inversion solver. The two matrices to be inverted, $A_1$ and $A_2$, shown in Fig. 1a, are constructed from the $4 \times 4$ discrete cosine transfer matrix by applying different linear scaling factors to its eigenvalues. The condition numbers $\kappa$ of $A_1$ and $A_2$ are 25.0 and 11.1, respectively. The analytical inversions $A^{-1}_1$ and $A^{-1}_2$, shown in Fig. 1b, are calculated with floating-point LU decomposition.

The solver uses signed 8-bit fixed-point format. The step sizes, $\tau$, are calculated from Eq. (10) with a safety margin, $\chi=0.2$, for the inversions of $A_1$ and $A_2$. To estimate the normalized $\ell^2$ error $\eta$ of the matrix–vector product, $A^T A_x$, the arrays $A^T A$ and $b$ in Eq. (3) are pre-calculated and converted into fixed-point. We enumerate 200 normally distributed $\tilde{x}$ with mean $b$ and standard deviation $\sqrt{\frac{|b-x^*|^2}{2}}$, where $x^*$ is the solution. The enumerated $\tilde{x}$ are converted to fixed-point format and multiplied with the fixed-point matrix $\tilde{A}^T A$. We then obtain $\eta=0.019$ from the average normalized root-mean-square error (RMSE) of the multiplication results.

Figure 1c plots the normalized errors, $\theta_k$, in log-scale compared with the analytical inverses, $A^{-1}_1$ and $A^{-1}_2$, as the iterations progress. The asymptotic errors, $\theta$, of the 8-bit iterations are 0.21 and 0.083 for the inversions of $A_1$ and $A_2$, respectively, both of which are consistent with the error bounds given by Eq. (12). The convergence rate, $\gamma$, is obtained from the exponential fitting of Fig. 1c with a 95% confidence level. For the inversions of $A_1$ and $A_2$, $\gamma$ is $0.040 \pm 0.001$ and $0.085 \pm 0.004$ respectively, which are inversely proportional to $\kappa$, as predicted by Eq. (13). The asymptotic errors and convergence rates are summarized in Table 1.

For the same condition number, Fig. 2 compares the fixed-point Richardson iterations with the 8-bit, 7-bit, 6-bit, and floating-point precisions. The convergence rates are summarized in Table 2, which confirms the implication of Eq. (13) that the fixed-point iterative solver maintains the same convergence rate, $\gamma$, as the floating-point counterpart. The independence of convergence rate on the precision suggests that fixed-point iterative solvers do not compromise the convergence speed. The asymptotic errors of the 8-bit, 7-bit, and 6-bit solutions are 0.083, 0.18, and 0.33, respectively, which are all below the upper bound established in Eq. (12).

### Table 1

| Matrix to invert | $\kappa$ | $\theta$ | Upper bound of $\theta$ | Convergence rate $\gamma$ |
|------------------|----------|----------|-------------------------|---------------------------|
| $A_1$            | 25.0     | 0.21     | 0.24                    | $0.040 \pm 0.001$         |
| $A_2$            | 11.1     | 0.083    | 0.098                   | $0.085 \pm 0.004$         |

### Table 2

| Precision | $\eta$ | $\theta$ | Upper bound of $\theta$ | Convergence rate $\gamma$ |
|-----------|--------|----------|-------------------------|---------------------------|
| 8-bit     | 0.019  | 0.083    | 0.098                   | $0.085 \pm 0.002$         |
| 7-bit     | 0.036  | 0.18     | 0.19                    | $0.085 \pm 0.004$         |
| 6-bit     | 0.072  | 0.33     | 0.37                    | $0.088 \pm 0.009$         |
| Floating-point | N.A   | N.A    | N.A                     | $0.083 \pm 0.003$         |
Image deconvolution. Deconvolution is a classic linear inverse problem that has broad applications in signal/image filtering and processing. For a linear, shift-invariant system, the convolution output, \( y \) (i.e., filtered signal, or blurred image) is the discrete convolution between the input signal or image \( x \) and the kernel \( K \), i.e., \( y = K \ast x \), which is defined elementwise as:

\[
y_{ij} = \sum_{p,q=0}^{M,N} K_{p,q} x_{i-p,j-q},
\]

where \( M \) and \( N \) are the sizes of the kernel along row and column directions, respectively. Here, we have assumed a two-dimensional convolution, as in the case of image processing. This example demonstrates the iterative Richardson-Lucy deconvolution, which recovers images from blurry measurements given the convolution kernel.

The discrete convolution can be formulated as the product between the vectorized input, \( x \), and a Toeplitz matrix, \( A \), constructed from the kernel \( K \), producing the output \( y = Ax \) in the vectorized form. To estimate the condition number, we rewrite Eq. (16) as Eq. (17) following the convolution theorem,

\[
y = \mathcal{F}^\dagger \text{diag}(\tilde{K}) \mathcal{F} x,
\]

where \( \mathcal{F} \) and \( \mathcal{F}^\dagger \) represent the forward and inverse two-dimensional Fourier transforms, both of which are unitary operators, \( \dagger \) denotes the conjugate transpose of a complex matrix, and \( \text{diag}(\tilde{K}) \) is the diagonal matrix constructed from the Fourier transform of the kernel \( K \). The condition number, \( \kappa \), of \( A^T A \) can be obtained from the maximum and minimum elements in \( \text{diag}(\tilde{K})^\dagger \text{diag}(\tilde{K}) \).

Figure 3 shows an example based on Richardson-Lucy deconvolution. The input, \( x^* \), is a test image downsampled from the MATLAB image “Saturn”, shown in Fig. 3 (a). The image contains 128 × 102 pixels in signed 4-bit fixed-point format with range (-1,1). The convolution kernel, \( K \), follows the Gaussian profile, where the parameter \( \sigma \) determines the width of the kernel, and \( K_0 \) is a normalization constant that ensures that \( \sum_{p,q} |K_{p,q}|^2 = 1 \). Four example kernels with \( \sigma = 0.70, 0.75, 0.80, \) and \( 0.85 \), are plotted in Fig. 3b. Figure 3c shows the normalized eigenvalue spectra of the four Toeplitz matrices, calculated from the Fourier transform of the kernels. The condition numbers, \( \kappa \), of the four kernels are 8.38, 16.59, 35.07, and 78.50, respectively. If \( \sigma \) is increased beyond \( 0.90 \), \( \kappa \) would violate the criteria in Eq. (11) and lead to a non-converging deconvolution result. Similar phenomenon has been reported in hybrid analog–digital solvers.

The measurements, shown in Fig. 4a, are calculated from convolving the input image, \( x^* \), with the four kernels in Fig. 3b, and then quantized to 8-bit to simulate the digitization error of an ordinary CCD or CMOS camera. The deconvolution problems are solved using 8-bit fixed-point iterations, with \( \eta \) estimated to be 0.014.

---

**Figure 3.** Fixed-point iterative solver for a discrete Richardson-Lucy deconvolution problem. (a) The 128 × 102 signed 4-bit input image “Saturn”. (b) Four convolution kernels with \( \sigma = 0.70, 0.75, 0.80, \) and \( 0.85 \), respectively. (c) Eigenvalue spectra of the Toeplitz matrices calculated from the Fourier transforms of the convolution kernels. The condition numbers, \( \kappa \), of \( A^T A \) are 8.38, 16.59, 35.07, and 78.50, respectively.
Step sizes with the safety margin $\chi = 0.2$ are used for all four iterative deconvolutions. Figure 4b plots the deconvolved images after 200 iterations. The normalized errors as functions of iteration steps are plotted in Fig. 4c. The asymptotic errors, $\theta$, are summarized in Table 3, and are all below the error bound predicted by Eq. (12).

Tomographic reconstruction with residual iteration. Theorem 2 indicates that the residual iteration can overcome the stagnation of the fixed-point Richardson solver and reach a solution beyond the fixed-point limit. To verify the performance of the residual iterations, we examine a tomographic reconstruction of a 16 × 16 “Super Mario” pixel art from its pencil-beam projections, shown in Fig. 5a. The pixels of the input image, $x^*$, are stored in the signed 3-bit fixed-point format with range (−1, 1), giving a quantization interval of 0.25 between two adjacent intensity levels. The measurements consist of 60 projections between 0 and 180º at 4º intervals, each containing 31 uniformly spaced beams. The system matrix, $A$, of the tomographic model is constructed from the distance-driven method 38. Figure 5b plots the eigenvalue spectrum of the tomographic model. The condition number, $\kappa$, of $A^T A$ is 101.80. Assuming the maximum step size, $\tau$, is used in the Richardson iterations, the computation error $\eta$ must be less than 0.020 to ensure convergence, according to Eq. (11). For the size and sparsity of this system matrix, we find that a minimum of $L = 8$ bits (with an estimated $\eta = 0.015$) are required for fixed-point iterations to converge.

Fixed-point Richardson and residual iterations with 8-bit, 9-bit, and 10-bit precisions are performed to reconstruct the “Super Mario” pixel art from the tomographic projections. The step sizes, $\tau$, with safety margin $\chi = 0.3$ are used in all the solvers. The system matrix $A$ is quantized to the solver precision with expo = 0. The measurement, $y$, is serialized to a vector, multiplied by $\tau A^T$, and quantized to the solver precision with expo = 0 to pre-calculate the fixed-point vector $b$ for the iterations. The fixed-point Richardson iteration sets the exponent of the multiplication between $A^T A$ and $x_k$ to 2. For residual iterations, the exponents of matrix–vector multiplications are decremented by 1 every 80 steps from 2 to −2.

Figure 6 plots the reconstructions from Richardson and residual iterations with 8-bit, 9-bit, and 10-bit precisions. Table 4 summarizes the normalized errors of the fixed-point Richardson reconstructions, all of which are below the error bound given by Eq. (12). The normalized error, $\theta$, is plotted as a function of the Richardson iteration number.

---

**Figure 4.** 8-bit fixed-point Richardson solver of a discrete Richardson-Lucy deconvolution problem. (a) Convolution measurements from the four kernels with $\sigma$ of (a1) 0.70, (a2) 0.75, (a3) 0.80, and (a4) 0.85. (b1-b4) Deconvolved images using 8-bit fixed-point iterations from (a1-a4). (c) Normalized error vs. iteration steps for the reconstructions in (b1-b4).

**Table 3.** Normalized errors of Richardson-Lucy deconvolution results.

| $\sigma$ | $\kappa$ | $\theta$ | Upper bound of $\theta$ |
|---------|----------|----------|-------------------------|
| 0.70    | 8.38     | 0.012    | 0.056                   |
| 0.75    | 16.59    | 0.016    | 0.12                    |
| 0.80    | 35.07    | 0.023    | 0.25                    |
| 0.85    | 78.50    | 0.037    | 0.53                    |
iteration number in Fig. 6c1. Note that the same convergence rate is present regardless of the precision of the solver. The normalized errors of the residual iterations are shown in Fig. 6c2. After $M=5$ residue updates, the 8-bit, 9-bit, and 10-bit solvers all reach normalized errors below 0.1. At this level of $\theta$, the pixelwise differences between $x^{(5)}$ and $x^* \Delta x_{\delta}$ are below the quantization interval, 0.25, of the true image $x^*$. Therefore, the reconstructions in Fig. 6b1–b3 would exhibit no visual differences from the true image if displayed in 3-bit grayscale colormap.

Table 4. Normalized errors of 8-bit, 9-bit, and 10-bit Richardson iterations.

| Precision | $\eta$ | $\theta$ | Upper bound of $\theta$ |
|-----------|--------|----------|-------------------------|
| 8-bit     | 0.015  | 0.49     | 0.88                    |
| 9-bit     | 0.0056 | 0.32     | 0.33                    |
| 10-bit    | 0.0028 | 0.19     | 0.16                    |

Faster convergence is possible via adaptive adjustments of the exponents of $b^{(l)}$ and $A^T A \Delta x_{\delta}$. Figure 7 shows the error of the 8-bit residual iteration executed on the hardware prototype solver.
a fixed-point iterative solver"). The exponent is updated every 5 steps based on the distribution of the elements in a fixed-point array x. The exponents are calculated using:

\[ \text{expo} = \left\lceil \log_2 \left( |\mu_x| + 3\sigma_x \right) \right\rceil \]

where \( \mu_x \) and \( \sigma_x \) denote the mean and standard deviation, respectively, of all the elements in x. The adaptive residual iteration achieves the same convergence speed as the floating-point solver, with both methods achieving errors below the quantization interval of the true solution after 28 iterations.

**Hardware prototype for a fixed-point iterative solver**

We have built a hardware prototype to perform the fixed-point Richardson iteration and the adjustment of exponent. The core component of the solver, i.e., the fixed-point matrix multiplication, is implemented on an FPGA development board (Xilinx ZC706) and communicates with the host PC via PCIe bus. Figure 8 illustrates the block diagram of the logic design within the FPGA. The systolic multiplier array performs 512 MAC operations in a single clock cycle. The inputs of the multiplier array consist of a 16 × 16 \( W \) block and a 16 × 2 \( X \) block. The precisions of both input blocks are signed 8-bit. The multiplication outputs are cached in signed 32-bit and wired to a 16 × 2 array of accumulators. For the matrix–vector multiplication involving a matrix \( W \) larger than 16 × 16, the matrix \( W \) is zero-padded to an integer number of 16 × 16 \( W \) blocks, and the batch vector \( X \) is zero-padded to an integer number of 16 × 2 \( X \) blocks, along the row dimension only. All the decomposed \( W \) and \( X \) blocks are cached in on-chip block random access memory (BRAM). The multiplier array reads the \( W \) blocks along the rows and the \( X \) blocks along the columns. The blocks of 16 × 2 results are summed in the accumulators. After completion of one row of \( W \) blocks, the multiplier array reads the \( W \) blocks in the next row and cycles through all the blocks of \( X \) again.

The host PC converts \( W \) and \( X \) into fixed-point, and then decomposes the mantissas into 16 × 16 \( W \) blocks and 16 × 2 \( X \) blocks. The decomposed \( W \) and \( X \) blocks are streamed to the FPGA through PCIe. Once all the blocks are multiplied and accumulated in FPGA, the results \( Y \) are streamed back to host PC in 32-bit integer format. The host PC masks the results back to 8-bit, with the most significant bit (MSB) and least significant bit (LSB) selected by the exponents. The host PC updates the exponents according to Eq. (19) every five iterations.

For the FPGA design in Fig. 8 implemented on the ZC706 evaluation board, the Xilinx synthesis and implementation tools estimate a power consumption of 2.047W, of which 1.253W is consumed by the transceivers for PCIe communication with the host PC, and the remaining 0.794W is consumed by the logics, including the clocks, systolic multiplier array, BRAM cache, and control units that generate the addresses of \( W \) and \( X \) blocks. The clock rate is 250 MHz. Considering the total number of 512 MACs within a single clock cycle, the energy consumption by the fixed-point computation is 6.2pJ/MAC, which is on the same order as that of datacenter TPUs \(^{39} \).

We have tested the speed of the iterative solver implemented on our hardware prototype and CPU (Intel Core i3-4130, dual-core, 3.40 GHz clock rate). The total CPU time of the fixed-point Richardson iterations, excluding exponent adjustment, is measured, and then divided by the total number of iterations in all residual updates.
to obtain the time per iteration, which is indicative of the calculation speed. The fixed-point solver on CPU takes 1.7 ± 0.2 ms per iteration, while that on FPGA prototype takes 0.76 ± 0.05 ms per iteration, representing more than two times of improvement in speed. Errors represent the variance of the calculation time within five repeated tests.

Conclusion

We have demonstrated a fixed-point iterative solver that computes high-precision solutions for linear inverse problems beyond the precision limit of the hardware. We have described the convergence and error of the fixed-point iterations in two theorems. Theorem 1 shows that the convergence rate is independent of the precision of the solver, which implies that fixed-point iteration does not compromise the convergence speed of the solver. Theorem 2 resolves the stagnation of a fixed-point solver with residual iterations, which correct the error and refine the solution to a higher precision level. We have presented three examples of linear inverse problems, namely, matrix inversion, Richardson-Lucy deconvolution, and tomographic reconstruction, to verify both theorems.

The combination of residual iteration with adaptive exponent adjustment achieves the same rate constant and error as a floating-point Richardson solver. We have prototyped the fixed-point solver with a fixed-point systolic multiplier array on the FPGA and fixed-point arithmetic libraries on host PC. The prototype solver achieves more than twice the calculation speed of a CPU and takes the same number of iterations to reach the solution as its floating-point counterpart. We envision that an array of low-precision, analog matrix processing cores can supersede the current FPGA systolic array in solving numerous large-scale, high-performance computing applications, including but not limited to convex optimizations, solving differential equations, and training of artificial neural networks. Certain nonlinear problems could also be incorporated into our iterative solver framework in conjunction with digital co-processing to calculate the local gradient on host PC.

Data availability

The data and software that support the findings of this study are available from the corresponding author upon reasonable request.

Received: 23 January 2023; Accepted: 26 March 2023
Published online: 30 March 2023

References

1. Biemond, J., Lagendijk, R. L. & Mersereau, R. M. Iterative methods for image deblurring. Proc. IEEE 78(5), 856–883. https://doi.org/10.1109/5.354303 (1990).
2. A. C. Kak and M. Slaney, Principles of Computed Tomographic Imaging. IEEE press New York, 2001.
3. Harmany, Z. T., Marcia, R. F. & Willett, R. M. This is SPIRAL-TAP: Sparse poisson intensity reconstruction Alogrithms—theory and practice. IEEE Trans. Image Process. 21(3), 1084–1096. https://doi.org/10.1109/TIP.2011.2168410 (2012).
4. Donoho, D. L. Compressed sensing. IEEE Trans. Inf. Theory 52(4), 1289–1306. https://doi.org/10.1109/TIT.2006.871582 (2006).
5. R. Baraniuk and P. Steeghs, Compressive Radar Imaging, In 2007 IEEE Radar Conference, Apr. 2007, pp. 128–133. doi: https://doi.org/10.1109/RADAR.2007.374203.
6. Tsai, T.-H., Luih, P., Yuan, X., Carin, L. & Brady, D. J. Spectral-temporal compressive imaging. Opt. Lett. 40(17), 4054–4057. https://doi.org/10.1364/OL.40.004054 (2015).
7. Wang, Y. & Boyd, S. Fast model predictive control using online optimization. IEEE Trans. Control Syst. Technol. 18(2), 267–278. https://doi.org/10.1109/TCST.2009.2107934 (2010).
8. Morari, M., Garcia, C. E. & Prett, D. M. Model predictive control: Theory and practice. IFAC Proc. 21(4), 1–12. https://doi.org/10.1016/B978-0-08-035735-5.00008-1 (1988).
9. J. L. Jerez, G. A. Constantinides, and E. C. Kerrigan, “An FPGA implementation of a sparse quadratic programming solver for constrained predictive control,” in Proceedings of the 19th ACM/SIGDA international symposium on Field programmable gate arrays - FPGA ‘11, 2011, p. 209. doi: https://doi.org/10.1145/1950413.1950454.
10. Choi, Y. & Cong, J. Acceleration of EM-based 3D CT reconstruction using FPGA. IEEE Trans. Biomed. Circuits Syst. 10(3), 754–767. https://doi.org/10.1109/TBCAS.2015.2471813 (2016).
11. J. K. Kim, Z. Zhang, and J. A. Fessler, “Hardware acceleration of iterative image reconstruction for X-ray computed tomography,” in 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), May 2011, no. 1, pp. 1697–1700. doi: https://doi.org/10.1109/ICASSP.2011.5946827.
12. Y. E. Wang, G.-Y. Wei, and D. Brooks, “Benchmarking tpu, gpu, and cpu platforms for deep learning,” arXiv Prepr. arXiv1907.10701, 2019.
13. S. Markidis, S. W. Der Chen, E. Laure, T. R. Peng, and J. S. Vetter, NVIDIA tensor core programmability, performance & precision, In Proceedings of 2018 IEEE 32nd International Parallel Distributed Processing Symposium Workip (IPDPSW) 2018, pp. 522–531, 2018. doi: https://doi.org/10.1109/IPDPSW.2018.00091.
14. Sze, V., Chen, Y.-H., Yang, T.-J., & Emer, J. S. Efficient processing of deep neural networks: A tutorial and survey. Proc. IEEE 105(12), 2293–2329. https://doi.org/10.1109/JPROC.2017.2761740 (2017).
15. M. Horowitz, Computing's energy problem. In Proceedings of 2018 IEEE 32nd International Parallel Distributed Processing Symposium Workip (IPDPSW) 2018, pp. 522–531, 2018. doi: https://doi.org/10.1109/IPDPSW.2018.00091.
16. Haensch, W., Gokmen, T., & Puri, R. The next generation of deep learning hardware: Analog Computing. Proc. IEEE 107(1), 108–122. https://doi.org/10.1109/JPROC.2018.2871057 (2019).
17. Li, C. et al. Efficient and self-adaptive in-situ learning in multilayer memristor neural networks. Nat. Commun. 9(1), 7–14. https://doi.org/10.1038/s41467-018-04484-2 (2018).
18. Xoo, P. et al. Fully hardware-implemented memristor convolutional neural network. Nature 577(7792), 641–646. https://doi.org/10.1038/s41586-020-1942-2 (2020).
19. Hameery, R., Bernstein, L., Sludds, A., Soljačić, M. & Englund, D. Large-scale optical neural networks based on photonic multiplexing. Phys. Rev. X 9(2), 1–12. https://doi.org/10.1103/physrevx.9.021032 (2019).
20. Shen, Y. et al. Deep learning with coherent nanophotonic circuits. Nat. Photonics 11(7), 441–446. https://doi.org/10.1038/nphoton.2017.93 (2017).
21. Tsai, H., Ambrogio, S., Narayanan, P., Shelby, R. M. & Burr, G. W. Recent progress in analog memory-based accelerators for deep learning. J. Phys. D. Appl. Phys. 51(28), 415. https://doi.org/10.1088/1361-6463/aac8a5 (2018).
Acknowledgements
The authors would like to thank Dr. Stephen Becker (Department of Applied Mathematics, University of Colorado Boulder) for helpful discussions. This work is supported in part by the National Science Foundation (1932858), in part by the Army Research Office (W911NF2110321), and in part by the Office of Naval Research (N00014-20-1-2441).

Author contributions
G. L. and S. P. initiated and supervised the project. Z. Z. and S. P. conceived the idea of fixed-point residue iteration. Z. Z. and A. B. built the hardware prototype and performed the numerical experiments. All authors participated in the discussion and writing of the manuscript.

Competing interests
The authors declare no competing interests.

Additional information

Supplementary Information
The online version contains supplementary material available at https://doi.org/10.1038/s41598-023-32338-5.

Correspondence
Correspondence and requests for materials should be addressed to Z.Z.

Reprints and permissions information
Reprints and permissions information is available at www.nature.com/reprints.

Publisher’s note
Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

© The Author(s) 2023