SIAM J. SCI. COMPUT.
Vol. 42, No. 5, pp. A2701–A2730

STABILITY ANALYSIS OF INLINE ZFP COMPRESSION FOR FLOATING-POINT DATA IN ITERATIVE METHODS*

ALYSON FOX†, JAMES DIFFENDERFER†, JEFFREY HITTINGER†, GEOFFREY SANDERS†, AND PETER LINDSTROM†

Abstract. Currently, the dominating constraint in many high performance computing applications is data capacity and bandwidth, in both internode communications and even moreso in intranode data motion. A new approach to address this limitation is to make use of data compression in the form of a compressed data array. Storing data in a compressed data array and converting to standard IEEE-754 types as needed during a computation can reduce the pressure on bandwidth and storage. However, repeated conversions (lossy compression and decompression) introduce additional approximation errors, which need to be shown to not significantly affect the simulation results. We extend recent work [J. Diffenderfer et al., *SIAM J. Sci. Comput.*, 41 (2019), pp. A1867–A1898] that analyzed the error of a single use of compression and decompression of the ZFP compressed data array representation [P. Lindstrom, *IEEE Trans. Vis. Comput. Graph.*, 20 (2014), pp. 2674–2683; P. Lindstrom, *ZFP version* 0.5.5, May 2019] to the case of time-stepping and iterative schemes, where an advancement operator is repeatedly applied in addition to the conversions. We show that the accumulated error for iterative methods involving fixed-point and time evolving iterations is bounded under standard constraints. An upper bound is established on the number of additional iterations required for the convergence of stationary fixed-point iterations. An additional analysis of traditional forward and backward error of stationary iterative methods using ZFP compressed arrays is also presented. The results of several 1D, 2D, and 3D test problems are provided to demonstrate the correctness of the theoretical bounds.

Key words. lossy compression, error bounds, data-type conversion, iterative methods, floating-point representation

AMS subject classifications. 65G30, 65G50, 68P30

DOI. 10.1137/19M126904X

1. Introduction.

Applications in scientific computing often result in extremely large volumes of floating-point data; e.g., a simulation of turbulent fluid dynamics can produce on the order of 6 TB per time step [10]. Thus, a significant order cost is often the off-node and on-node data motion [3, 27]. Lossless and lossy compression algorithms have been considered to reduce the amount of data, thus reducing time in data transfer. Compression algorithms are most frequently considered for I/O operations (data and restart files) and in-memory storage of static data (e.g., tabulated material properties); however, there is potential for significant performance gains by using compressed data types within a simulation. In this approach, the solution state data could be stored in a compressed format, be decompressed, be operated on, and be recompressed inline during each time step or iteration of a numerical simulation. The trends in computer hardware are that processing power (the number of FLOPS) is

*Submitted to the journal's Methods and Algorithms for Scientific Computing section June 18, 2019; accepted for publication (in revised form) May 21, 2020; published electronically September 15, 2020. The U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes.
https://doi.org/10.1137/19M126904X
Funding: This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344 and was supported by the LLNL-LDRD Program under project 17-SI-004, LLNL-JRNL-769679.

†Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore, CA 94551 (fox33@llnl.gov, diffenderfer2@llnl.gov, hittinger1@llnl.gov, sanders29@llnl.gov, lindstrom2@llnl.gov).

growing much more rapidly than memory and network bandwidth or memory capacity. Thus, compressed data types would make more efficient use of bandwidth and storage.

The seemingly preferred choice would be a compressed data type based on lossless compression algorithms, such as FPC [21], SPDP [6], BLOSC [2], FPZIP [18], and other variants, as they reproduce the original data with no degradation. Unfortunately, lossless compression algorithms struggle to produce significant compression ratios for floating-point data [21, 18]. Lossy floating-point compression, e.g., SZ [7, 24, 15] and ZFP [16], on the other hand, only guarantees an approximation of the original data but produces a much higher ratio of data reduction than lossless compression. Since the solution state from a numerical simulation already contains a host of numerical approximation errors (e.g., floating-point round-off, truncation error, and iteration error), one can legitimately question the logic of requiring a lossless compressed data type. Data lost in lossy compression may not be meaningful, i.e., it represents errors, and a lossy compressed data type can be thought of as just another approximate representation of real numbers with a finite number of bits that introduces error that must be controlled.

The use of lossy compression in numerical simulation has been suggested before: it has been considered for checkpointing numerical simulations [5] and for inline compression of the solution state in [14]. In both cases, it was demonstrated that lossy compression can be used without causing significant changes to important physical quantities. However, neither application provided theoretical backing to ensure numerical stability, that is, to control the accumulated error resulting from repeated compression and decompression. In [25], the authors investigated the convergence of iterative methods when only one or possibly two checkpoints are used with lossy compression.

In this paper, we will consider the use of a lossy compressed data type in place of the standard IEEE-754 [1] floating point representation and analyze the impact of the associated compression error on two common applications in numerical simulation. Specifically, we consider the ZFP compressed data array, which was first described in [16] and modified in [17], as a lossy compression data type that individually compresses and decompresses small blocks of 4^d values from d-dimensional data. Due to the locality of the independently compressed blocks, ZFP compressed data arrays are ideal for storing simulation data, since only the block containing a particular data value needs to be uncompressed, which gives ZFP arrays a behavior similar to standard random access arrays. In prior work [8], extensive error analysis for a single cycle of ZFP compression and decompression was performed for all three of ZFP's encoding modes (fixed-rate, fixed-accuracy, and fixed-precision). Here we extend these results to the use of ZFP compressed data arrays under the action of an *advancement operator*, an operator that advances the numerical solution whether it be for an iterative or a time-stepping numerical method. Specifically, we consider the situation where data is stored in a ZFP compressed array, converted (decompressed) to IEEE double precision data types as needed, updated arithmetically under the action of the advancement operator, and then converted (compressed) back to the ZFP representation. This cycle will be repeated numerous times in a time-stepping or iterative algorithm.

Without loss of generality, we assume that the advancement operator is bounded, a common assumption for a wide class of problems. Either we assume the advancement operator is Lipschitz continuous, which is useful for both linear and nonlinear operators, or it is a Kreiss bounded linear operator (see section 3 for more details). In either case, without a bounded advancement operator, the numerical method may be

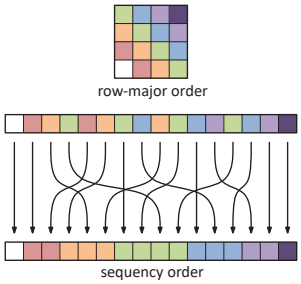


FIG. 1. *Total sequency ordering for a 2D array, which groups the diagonal elements together.*

unstable, resulting in poor approximations to the true solution. Under these assumptions, our results determine an upper bound on the accumulation of error introduced by the use of ZFP compressed arrays. Furthermore, our results allow the formulation of criteria for the parameters of ZFP that will ensure that the compression error remains below a user-defined tolerance. We also establish an upper bound on the number of additional iterations required to achieve convergence of a stationary iterative method when ZFP compression error accumulates. We also extend our error bounds for successive displacement fixed-point operators that are block diagonally dominant. Finally, we provide a forward and backward error analysis for stationary iterative methods using ZFP compressed arrays and compare the accumulated floating-point round-off error to the accumulated compression error caused by ZFP.

The remainder of this paper is structured as follows. In section 2, we will summarize the required notation and bounds established in [8]. In section 3, we provide the main results, which are upper bounds for the error introduced by using ZFP compressed arrays under repeated application of an advancement operator. Following the main results, we demonstrate the validity of the error bounds numerically in section 4.

2. Preliminary notation, definitions, and theorems. We first outline the ZFP compression algorithm as documented in [17] and provide the required notation and preliminary theorems, as defined in [8], that are necessary for this paper.

Consider a d-dimensional scalar array. This array is partitioned into *blocks* of 4^d scalars. Each block is then compressed independently by the following steps. The floating-point values, typically represented by IEEE, from each block are converted to a block-floating-point representation [19] using a common exponent then shifted and rounded into two's complement signed integers. A near orthogonal decorrelating transform, similar to the discrete cosine transform, is applied to the 4^d block. The idea is that the decorrelating transform removes redundancies in the data by a change in basis. From the change of basis, the magnitude of the coefficients tend to correlate with the location within the block; thus, ZFP reorders the coefficients by total sequency [17]. A 2D example can be seen in Figure 1.

As the sign-bit in two's complement does not provide any useful information without the leading one-bit, ZFP converts each integer into a negabinary representation [13] where the leading one-bit encodes both the sign and the approximate magnitude of the value. The block is then transposed so that it is ordered by bit-plane instead of by coefficient, from the most to the least significant bit; see Figure 2.

Each bit-plane is then losslessly compressed using an embedded coding scheme [16] which emits one bit at a time until some stopping criterion is satisfied. The stopping criterion for ZFP has three modes: fixed-precision, fixed-rate, and fixed-accuracy. For

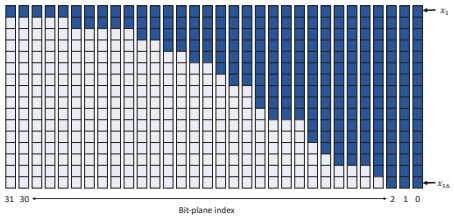


FIG. 2. *A 2D array represented with a 32-bit negabinary integer representation ordered by bit-plane, from the most to the least significant bit. Typically, the bits containing the information are concentrated in the higher transform coefficients as illustrated by the dark blue coloring (referred to as the energy compaction property* [20]*), where the light blue represents* 0 *and the dark blue represents either* 1 *or* 0*.*

more detailed information on ZFP see [16, 17].

The bounds presented in this paper are only for the fixed-precision mode, which retains a fixed number of bit-planes; however, the theorems can be extended to other modes using Theorems 5.3 and 5.4 in [8]. Each block of values can then be converted back to the source format (e.g., IEEE) by applying a decompression operator. Before discussing the round-off error bound between the compression of IEEE values to ZFP established in [8], we summarize a key vector space used in the analysis in [8].

2.1. Signed binary and negabinary bit-vector spaces. In [8], a vector space was introduced (which we will refer to as the *infinite bit vector space*) in order to express each step of the ZFP compression algorithm as an operator on binary or negabinary representations [13] of 4^d vectors. The components of each vector in this vector space are infinite sequences of zeros and ones that are restricted in a manner such that each real number has a unique representation as a binary sequence. For example, given $x \in \mathbb{R}$, there exist $c, d \in \mathbb{B}^\infty$ and $s \in \mathbb{B}$ such that x can be represented in signed binary and negabinary as

$$\text{(2.1)}\quad \text{Signed Binary: } x = (-1)^s \sum_{i=-\infty}^{\infty} c_i 2^i \quad \text{and} \quad \text{Negabinary: } x = \sum_{i=-\infty}^{\infty} d_i (-2)^i,$$

where $\mathbb{B} = \{0, 1\}$. Placing certain restrictions on the choices of c, d, and s such that each x has a unique representation in the form described in (2.1), we are able to form the infinite bit vector spaces for signed binary and negabinary representations, which we denote by $\mathcal{B}$ and $\mathcal{N}$, respectively. To imitate floating-point representations, the authors of [8] define subspaces $\mathcal{B}_k$ and $\mathcal{N}_k$ of $\mathcal{B}$ and $\mathcal{N}$, respectively, where k represents the maximum number of nonzero bits allotted for each representation, excluding the sign bit in the signed binary representation. As $\mathcal{B}_k$ and $\mathcal{N}_k$ are subsets but *not* subspaces of $\mathcal{B}$ and $\mathcal{N}$, all the analysis in [8] took place in $\mathcal{B}$, $\mathcal{N}$, or $\mathbb{R}$ using operators to imitate working with a fixed number of bits. For the full definition and underlying concepts of the infinite bit vector space, see section 3 in [8].

2.2. Single use error bound. An upper bound on the round-off error of a single use of ZFP compression from IEEE values was established in [8] by constructing an operator for each step of ZFP that acts on elements of an infinite bit vector space and by composing these operators to make a ZFP compression and decompression operator. Notation introduced in [8] will be used in this paper and will be summarized

here with respect to each step of ZFP (de)compression.

First, assume the d-dimensional data has been partitioned into arrays of dimension 4^d, which we refer to as *blocks*. Let a block $\boldsymbol{x} \in \mathbb{R}^{4^d}$ be given. For some precision $k \in \mathbb{N}$, assume every element in $\boldsymbol{x}$ can be represented with at most k-consecutive bits, including the leading one-bit. For frequently used IEEE single and double precision floating-point types, $k \in \{24, 53\}$, i.e., the number of mantissa bits. Each component in $\boldsymbol{x}$ is then converted to a block floating-point representation with respect to a common exponent. Note that error may occur when converting the block into signed integers and is taken into account in [8]. Let $q \in \mathbb{N}$ represent the maximum number of nonzero consecutive bits that can be used to represent each component in the block floating-point representation. In the ZFP implementation, $q = k + e - 2$, where $e \in \mathbb{N}$ is the maximum number of bits that represent the exponent in the IEEE single or double format, i.e., $e \in \{8, 11\}$ and $q \in \{30, 62\}$. Define $e_{max,\mathcal{B}}(\boldsymbol{x})$ and $e_{max,\mathcal{N}}(\boldsymbol{x})$ as the index of the leading nonzero bit in each corresponding infinite bit vector space (see section 3.4 in [8]). From Lemma 3.6 in [8], we have $\|\boldsymbol{x}\|_\infty \geq 2^{e_{max,\mathcal{B}}(\boldsymbol{x})}$, a useful relation between the magnitude of $\|\boldsymbol{x}\|_\infty$ and $e_{max,\mathcal{B}}(\boldsymbol{x})$.

Once the block is converted to signed integers represented in two's complement, the block is acted on by a linear transformation. In d dimensions, the transform operator, $\mathcal{T}$, is applied to each dimension separately, and the operator can be represented as a Kronecker product. For $A \in \mathbb{R}^{n_1,m_1}$ and $B \in \mathbb{R}^{n_2,m_2}$, the Kronecker product is defined as

$$A \otimes B = \begin{bmatrix} a_{1,1}B & a_{1,2}B & \cdots a_{1,m_1}B \\ \vdots & \ddots & \vdots \\ a_{n_1,1}B & a_{n_1,2}B & \cdots a_{n_1,m_1}B \end{bmatrix}.$$

Then the total forward transform operator used in of ZFP is defined as

$$\mathcal{T}_d = \underbrace{\mathcal{T} \otimes \mathcal{T} \otimes \cdots \otimes \mathcal{T}}_{(d-1)\text{-products}},$$

where $\mathcal{T} \in \mathbb{R}^{4\times 4}$ is defined by

$$\mathcal{T} = \frac{1}{16}\begin{bmatrix} 4 & 4 & 4 & 4 \\ 5 & 1 & -1 & -5 \\ -4 & 4 & 4 & -4 \\ -2 & 6 & -6 & 2 \end{bmatrix} \quad \text{and} \quad \mathcal{T}^{-1} = \frac{1}{4}\begin{bmatrix} 4 & 6 & -4 & -1 \\ 4 & 2 & 4 & 5 \\ 4 & -2 & 4 & -5 \\ 4 & -6 & -4 & 1 \end{bmatrix}. \tag{2.2}$$

The transformation could result in typical fixed-point round-off. Thus, the operator used in the implementation of the algorithm [17] will be defined as $\tilde{\mathcal{T}}_d$. From [8], we have the associated lemma bounding the round-off error.

LEMMA 2.1 (Lemma 4.4 in [8]). *Suppose $\boldsymbol{x} \in \mathbb{Z}^{4^d}$, such that $e_{max,\mathcal{B}}(\boldsymbol{x}) = q \in \mathbb{N}$ and $\boldsymbol{x} \neq \mathbf{0}$. Then*

$$\|\mathcal{T}_d\boldsymbol{x} - \tilde{\mathcal{T}}_d\boldsymbol{x}\|_\infty \leq k_{\mathcal{T}}\epsilon_q\|\boldsymbol{x}\|_\infty, \tag{2.3}$$

where $k_{\mathcal{T}} = \frac{7}{4}\left(2^d - 1\right)$ and $\epsilon_q = 2^{1-q}$.

After the forward transformation, each component in the block is converted losslessly from a two's complement representation to a negabinary representation. That is, a p-bit two's complement integer requires $p + 1$ bits in negabinary; however, as

ZFP no longer needs the guard bit that was used for the transformation, there is an extra bit of precision for the negabinary representation. Once each component of the block is converted to a negabinary representation, a deterministic permutation acts on the components of the block. The final compression step is then the bit stream truncation. For the fixed-precision mode, the fixed-precision parameter, denoted by $\beta \geq 0$, is provided by the user to ZFP. Here β represents the number of most significant bit planes to keep and any discarded bit plane is mathematically equivalent to replacing the bits with all-zero bits. Note that the encoding step is omitted from the discussion, as it is a lossless mapping. For the decompression operator, each block is decompressed by applying the inverse of each compression step in reverse order. It was shown in [8] that if $\beta \leq q-2d+2$, then no additional loss will occur in the decompression steps with the exception of potential round-off error due to the conversion back to the original source format (see section 4.2 in [8]).

For $r \in \mathbb{Z}$, we define the constant $\epsilon_r = 2^{1-r}$. Let the nonlinear operators $C : \mathbb{R}^{4^d} \to \mathcal{N}^{4^d}$ and $D : \mathcal{N}^{4^d} \to \mathbb{R}^{4^d}$ denote the ZFP compression and decompression operators as defined in [8]. We can now note the following pointwise error bound resulting from compressing and then decompressing a 4^d block using ZFP.

THEOREM 2.2 (Theorem 5.2 in [8]). *Assume $\boldsymbol{x} \in \mathbb{R}^{4^d}$ with $\boldsymbol{x} \neq 0$ such that for some k and e, each element in $\boldsymbol{x}$ is representable by k mantissa bits and e exponent bits as defined in* [8]. *Let $0 \leq \beta \leq q - 2d + 2$ be the fixed-precision parameter. Then*

$$\|DC(\boldsymbol{x}) - \boldsymbol{x}\|_\infty \leq K_\beta \|\boldsymbol{x}\|_\infty, \tag{2.4}$$

where $q \in \mathbb{N}$ is the precision for the block-floating point representation,

$$K_\beta := \left(\frac{15}{4}\right)^d \left((1+\epsilon_k)\left(\frac{8}{3}\epsilon_\beta + \epsilon_q\left(1+\frac{8}{3}\epsilon_\beta\right)(k_{\mathcal{T}}(1+\epsilon_q)+1)\right)+\epsilon_k\right), \tag{2.5}$$

and the constant $k_{\mathcal{T}} = \frac{7}{4}(2^d - 1)$.

Theorem 2.2 can easily be extended to any d-dimensional array of floating point values by taking the largest error from all blocks. Let $\hat{C}$ and $\hat{D}$ represent the compression and decompression operator for the whole d-dimensional array; then

$$\|\hat{D}\hat{C}(\boldsymbol{x}) - \boldsymbol{x}\|_\infty = \max_i \|DC(\boldsymbol{x}_i) - \boldsymbol{x}_i\|_\infty, \tag{2.6}$$

where each $\boldsymbol{x}_i$ represents a nonoverlapping 4^d block. When it is clear *from context* that the entire data field is being (de)compressed, the $\hat{D}$, $\hat{C}$ notation will be dropped. Now that the required notation and theorems have been presented, we proceed with our discussion of error bounds for inline use of ZFP compressed arrays.

3. Error bounds for inline use of ZFP conversion.

DEFINITION 3.1. *An operator $g(\cdot) : \mathbb{R}^n \to \mathbb{R}^n$ is* Lipschitz continuous *with constant $L_l < \infty$ if*

$$\|g(\boldsymbol{x}) - g(\boldsymbol{y})\|_\infty \leq L_l \|\boldsymbol{x} - \boldsymbol{y}\|_\infty \qquad \text{for all} \quad \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n. \tag{3.1}$$

DEFINITION 3.2. *A linear operator $A \in \mathbb{R}^{n\times n}$ is* Kreiss bounded [23] *with constant $L_k < \infty$ if*

$$\|A^i\|_2 \leq L_k \qquad \text{for all } i \in \mathbb{N}. \tag{3.2}$$

3.1. Lipschitz continuous advancement operator. Suppose that we generate a sequence $\{\boldsymbol{y}_t\}_{t=0}^{\infty}$ by

$$\boldsymbol{y}_{t+1} = g(\boldsymbol{y}_t), \tag{3.3}$$

where the advancement operator, $g : \mathbb{R}^n \to \mathbb{R}^n$, and initial value, $\boldsymbol{y}_0 \in \mathbb{R}^n$, are given. Additionally, consider the sequence $\{\boldsymbol{x}_t\}_{t=0}^{\infty}$ generated by

$$\boldsymbol{x}_{t+1} = g(DC(\boldsymbol{x}_t)), \tag{3.4}$$

where the initial point of the sequence is $\boldsymbol{x}_0 = \boldsymbol{y}_0$. In applications, the solution would be stored in the compressed state, $C(\boldsymbol{x}_t)$; however, the solution must be converted back to IEEE in order to analyze the error. We chose to consider the error after the application of the advancement operator; however, we could have instead considered the sequence $\boldsymbol{x}_{t+1} = DC(g(\boldsymbol{x}_t))$ and obtained similar results. The scheme given by (3.3) can either be viewed as a fixed-point or a time-stepping method, depending on the properties of $g(\cdot)$. For example, $g(\cdot)$ could represent a finite difference method of a PDE or a Jacobi iteration method. First, we will determine a bound on the value of $\|\boldsymbol{x}_{t+1} - \boldsymbol{y}_{t+1}\|_\infty$ for various properties of the advancement operator. Then we will investigate specifically the convergence for fixed-point iterative methods; we will determine the convergence properties for which the sequence $\{\boldsymbol{x}_t\}_{t=0}^{\infty}$ converges to the same fixed point as the sequence $\{\boldsymbol{y}_t\}_{t=0}^{\infty}$. Additionally, we will determine how many extra iterations, m, are needed for the sequence $\{\boldsymbol{x}_t\}_{t=0}^{T+m}$ to converge to the same accuracy as $\{\boldsymbol{y}_t\}_{t=0}^{T}$ with respect to the fixed point, given T iterations.

Theorem 3.3 provides a bound for the difference between the jth element of the sequences $\{\boldsymbol{y}_t\}_{t=0}^{j}$ and $\{\boldsymbol{x}_t\}_{t=0}^{j}$ for a general Lipschitz continuous advancement operator by using the single use error bound from Theorem 2.2. Since the only assumption on the advancement operator $g(\cdot)$ is Lipschitz continuity, Theorem 3.3 is not restricted to linear functions and is useful for applications involving Lipschitz continuous nonlinear functions.

THEOREM 3.3 (Lipschitz continuous advancement operator). *Suppose that $g(\cdot)$ is Lipschitz continuous with Lipschitz constant L_l. Then*

$$\|\boldsymbol{x}_{t+1} - \boldsymbol{y}_{t+1}\|_\infty \leq \sum_{j=0}^{t} L_l^{t-j+1} K_{\beta_j} \|\boldsymbol{x}_j\|_\infty, \tag{3.5}$$

where β_j is the number of bit planes kept at step j.

Proof. Since $g(\cdot)$ is Lipschitz continuous, it follows that

$$\begin{aligned} \|\boldsymbol{x}_{t+1} - \boldsymbol{y}_{t+1}\|_\infty &= \|g(DC(\boldsymbol{x}_t)) - g(\boldsymbol{y}_t)\|_\infty \\ &\leq L_l \|DC(\boldsymbol{x}_t) - \boldsymbol{y}_t\|_\infty \leq L_l \left(\|DC(\boldsymbol{x}_t) - \boldsymbol{x}_t\|_\infty + \|\boldsymbol{x}_t - \boldsymbol{y}_t\|_\infty\right) \end{aligned} \tag{3.6}$$

for all $t \geq 0$. Applying Theorem 2.2 yields the inequality

$$\|\boldsymbol{x}_{t+1} - \boldsymbol{y}_{t+1}\|_\infty \leq L_l \left(K_{\beta_t} \|\boldsymbol{x}_t\|_\infty + \|\boldsymbol{x}_t - \boldsymbol{y}_t\|_\infty\right). \tag{3.7}$$

Since $\boldsymbol{x}_0 = \boldsymbol{y}_0$ and (3.7) holds for all $t \geq 0$, the desired result now follows. ∎

It should be noted that if the advancement operator given by $g(\cdot)$ has a Lipschitz constant less than one, then (3.5) is a convergent sum. Otherwise, even if the advancement operator is stable (e.g., finite difference equations of hyperbolic PDEs),

the terms L_l^{t-j+1} in (3.5) will grow exponentially and will no longer provide a meaningful bound.

Simulations are fundamentally about approximation. For example, to solve a PDE one must use a finite difference or finite element method, which produces truncation error. Truncation error is typically the dominating error, and if the order of magnitude of the error is known, then one could find an approximate β value for all iterations that will ensure the truncation error remains the dominating error instead of the error caused by the repeated application of ZFP compression. Here we present a result that provides a lower bound on the choice of β to ensure that truncation error, or any other type of error resulting from the simulation, remains the dominant cost when using ZFP compressed data types.

LEMMA 3.4. *Suppose that $g(\cdot)$ is Lipschitz continuous with Lipschitz constant $L_l < 1$. Assume the sequence $\{\boldsymbol{x}_t\}_{t=0}^{\infty}$ defined by (3.4) is bounded. Assume $\epsilon \geq \frac{K_q \gamma}{1-L_l}$, where q is the number of bits used in the negabinary representation in ZFP and $\gamma = \max_{0\leq j\leq t} \|\boldsymbol{x}_j\|_\infty$. Then*

$$\|\boldsymbol{x}_{t+1} - \boldsymbol{y}_{t+1}\|_\infty \leq \epsilon \tag{3.8}$$

if

$$\tilde{\beta} \geq 1 - \log_2 \left(\left(\frac{4}{15} \right)^d \frac{3\epsilon(1-L_l)}{8\gamma(2^d+1)L_l(1-L_l^{t+2})} \right) \tag{3.9}$$

and $\tilde{\beta} \geq \beta_j$ for $1 \leq j \leq t$, where β_j is the number of bit planes kept at step j.

Proof. By dropping the lower-order terms, we can approximate K_{β_j} with respect to β_j as $K_{\beta_j} \approx \left(\frac{15}{4}\right)^d \frac{8}{3}(2^d+1)\epsilon_{\beta_j}$. Let $\gamma = \max_{0\leq j\leq t} \|\boldsymbol{x}_j\|_\infty$ and $K_{max} = \max_j K_{\beta_j}$. From (3.5), we have

$$\begin{aligned} \|\boldsymbol{x}_{t+1} - \boldsymbol{y}_{t+1}\|_\infty &\leq \gamma K_{max} \sum_{j=0}^{t} L_l^{t-j+1} \\ &\leq \gamma \left(\frac{15}{4} \right)^d \frac{8}{3}(2^d+1)\epsilon_{\beta'} \frac{L_l(1-L_l^{t+2})}{1-L_l}, \end{aligned} \tag{3.10}$$

where $\beta' \leq \min_j \beta_j$. Let $\tilde{\beta}$ be the minimal β' such that (3.8) is true. Then (3.8) implies

$$\gamma \left(\frac{15}{4} \right)^d \frac{8}{3}(2^d+1)\epsilon_{\tilde{\beta}} \frac{L_l(1-L_l^{t+2})}{1-L_l} \leq \epsilon \tag{3.11}$$

and the desired result follows. ∎

It is also of interest to consider convergence properties of the fixed-point iteration defined by (3.4). As the operator $g \circ D \circ C : \mathbb{R}^n \to \mathbb{R}^n$ may not be differentiable or even continuous, we cannot directly apply the fixed-point convergence theorem. However, we are able to use the error bound established in Theorem 2.2 for the compression and decompression operators to prove a similar convergence result when ZFP is used as a data type.

THEOREM 3.5. *Let B be a compact subset of $\mathbb{R}^n$. Suppose that $g(\cdot)$ is Lipschitz continuous on B with constant $L_l < 1$. Furthermore, if $L_l \in \left(0, \frac{1}{1+\tilde{K}_\beta}\right)$, where $\tilde{K}_\beta = \max_{0\le j\le t} K_{\beta_j}$, then*

$$\lim_{t\to\infty} \|\boldsymbol{x}_t - \boldsymbol{y}^*\| \le \frac{L_l \tilde{K}_\beta \|\boldsymbol{y}^*\|}{1-\theta}, \tag{3.12}$$

where $\theta := L_l(1+\tilde{K}_\beta)$ and $\boldsymbol{y}^*$ is the unique fixed point of $g(\cdot)$ in B.

Proof. As g is Lipschitz continuous with constant $L_l < 1$ on a compact set, there exists a unique point $\boldsymbol{y}^* \in B$ such that $g(\boldsymbol{y}^*) = \boldsymbol{y}^*$ [4]. Thus, the scheme in (3.3) will converge to $\boldsymbol{y}^*$ for all $\boldsymbol{y}_0 \in B$.

Now observe that

$$\begin{aligned}
\|\boldsymbol{x}_t - \boldsymbol{y}^*\| &= \|g(DC(\boldsymbol{x}_{t-1})) - g(\boldsymbol{y}^*)\| \\
&\le L_l \|DC(\boldsymbol{x}_{t-1}) - \boldsymbol{y}^*\| && \text{(Definition 3.1)} \\
&\le L_l \|DC(\boldsymbol{x}_{t-1}) - \boldsymbol{x}_{t-1}\| + L_l \|\boldsymbol{x}_{t-1} - \boldsymbol{y}^*\| \\
&\le L_l \tilde{K}_\beta \|\boldsymbol{x}_{t-1}\| + L_l \|\boldsymbol{x}_{t-1} - \boldsymbol{y}^*\| && \text{(Theorem 2.2)} \\
&= L_l \tilde{K}_\beta \|\boldsymbol{x}_{t-1} - \boldsymbol{y}^* + \boldsymbol{y}^*\| + L_l \|\boldsymbol{x}_{t-1} - \boldsymbol{y}^*\| \\
&\le \theta \|\boldsymbol{x}_{t-1} - \boldsymbol{y}^*\| + L_l \tilde{K}_\beta \|\boldsymbol{y}^*\|, && (3.13)
\end{aligned}$$

where $\theta := L_l(1+\tilde{K}_\beta)$. Applying (3.13) recursively yields that

$$\|\boldsymbol{x}_t - \boldsymbol{y}^*\| \le \theta^t \|\boldsymbol{x}_0 - \boldsymbol{y}^*\| + L_l \tilde{K}_\beta \|\boldsymbol{y}^*\| \sum_{j=0}^{t-1} \theta^j. \tag{3.14}$$

If $L_l \in \left(0, \frac{1}{1+\tilde{K}_\beta}\right)$, then $0 < \theta < 1$. We conclude that

$$\lim_{t\to\infty} \|\boldsymbol{x}_t - \boldsymbol{y}^*\| \le \frac{L_l \tilde{K}_\beta \|\boldsymbol{y}^*\|}{1-\theta}. \tag{3.15}$$

∎

Theorem 3.5 provides sufficient conditions under which the fixed point for the sequence (3.4) will be within some finite distance of the fixed point for the sequence (3.3). As another point of interest, we now consider the following result, which enumerates how many additional iterations are necessary for the sequence generated by (3.4) to achieve the same accuracy as the sequence generated by (3.3).

THEOREM 3.6. *Let B be a compact subset of $\mathbb{R}^n$. Suppose that $g(\cdot)$ is Lipschitz continuous on B with constant $L_l < 1$. Let $\boldsymbol{y}^*$ denote the fixed point of* (3.3). *Let the initial point of sequence* (3.3) *and* (3.4) *be equal, i.e., $\boldsymbol{x}_0 = \boldsymbol{y}_0$, such that $\boldsymbol{y}_0 \in B$. Let $t \in \mathbb{N}$ define the number of iterations for the sequence* (3.3), *and let $m \in \mathbb{N}$ be the number of additional iterations for sequence* (3.4), *i.e., $t+m$ iterations. Define $\tilde{K}_\beta = \max_j K_{\beta_j}$ for all $j = 1, \ldots, t+m$ and $C := \frac{\tilde{K}_\beta}{1-L_l}$. If $\frac{\tilde{K}_\beta}{(1-L_l)} \le L_l^t$ and*

$$m \ge \log_{L_l} \frac{L_l^{t+1} - C}{1-C} - (t+1), \tag{3.16}$$

then

$$\|\boldsymbol{x}_{t+m+1} - \boldsymbol{y}^*\|_\infty \le L_l^{t+1} \|\boldsymbol{y}_0 - \boldsymbol{y}^*\|_\infty. \tag{3.17}$$

Proof. Let $\boldsymbol{y}^*$ satisfy $\boldsymbol{y}^* = g(\boldsymbol{y}^*)$. Using standard fixed-point analysis techniques, we have that $\|\boldsymbol{y}_{t+1} - \boldsymbol{y}^*\|_\infty \leq L_l^{t+1}\|\boldsymbol{y}_0 - \boldsymbol{y}^*\|_\infty$. Similarly, we can find an expression for $\|\boldsymbol{x}_{t+m+1} - \boldsymbol{y}^*\|_\infty$ as follows:

$$\begin{aligned}\|\boldsymbol{x}_{t+m+1} - \boldsymbol{y}^*\|_\infty &= \|g(DC(\boldsymbol{x}_{t+m})) - g(\boldsymbol{y}^*)\|_\infty \\ &\leq L_l\|g(\boldsymbol{x}_{t+m}) - g(\boldsymbol{y}^*)\|_\infty + L_l K_{\beta_t}\|\boldsymbol{x}_{t+m}\|_\infty \\ &\leq L_l^{t+m+1}\|\boldsymbol{x}_0 - \boldsymbol{y}^*\|_\infty + \sum_{j=0}^{t+m} L_l^{t+m-j} K_{\beta_j}\|\boldsymbol{x}_j\|_\infty. \end{aligned} \tag{3.18}$$

Now define $\alpha := \max\{\max_{1\leq j\leq t+m+1}\|\boldsymbol{x}_j\|_\infty, \|\boldsymbol{y}_0 - \boldsymbol{y}^*\|_\infty\}$. By our choice that $\boldsymbol{x}_0 = \boldsymbol{y}_0$, we have $\|\boldsymbol{x}_0 - \boldsymbol{y}^*\|_\infty = \|\boldsymbol{y}_0 - \boldsymbol{y}^*\|_\infty$, which yields

$$\|\boldsymbol{x}_{t+m+1} - \boldsymbol{y}^*\|_\infty \leq \alpha\left(L_l^{t+m+1} + \tilde{K}_\beta \sum_{j=0}^{t+m} L_l^{t+m-j}\right). \tag{3.19}$$

As we wish to determine the additional number of iterations m such that $\boldsymbol{x}_{t+m+1}$ is sufficiently close to $\boldsymbol{y}^*$, assume that $\alpha\big(L_l^{t+m+1} + \tilde{K}_\beta \sum_{j=0}^{t+m} L_l^{t+m-j}\big) \leq L_l^{t+1}\|\boldsymbol{y}_0 - \boldsymbol{y}^*\|_\infty$. As $\|\boldsymbol{y}_0 - \boldsymbol{y}^*\|_\infty \leq \alpha$, by definition, we can consider the simplified inequality

$$L_l^{t+m+1} + \tilde{K}_\beta \sum_{j=0}^{t+m} L_l^{t+m-j} \leq L_l^{t+1}, \tag{3.20}$$

which, when solved for m, yields

$$m \geq \log_{L_l}\left(\frac{L_l^{t+1} - C}{1 - C}\right) - (t+1). \tag{3.21}$$

(3.22)

The assumption $\frac{\tilde{K}_\beta}{(1-L_l)} \leq L_l^{t+1}$ ensures that $\frac{L_l^{t+1}-C}{1-C} > 0$, which implies $\log_{L_l}\big(\frac{L_l^{t+1}-C}{1-C}\big)$ is real valued. Finally, we need to show that $m > 0$. By way of contradiction, assume $\log_{L_l}\big(\frac{L_l^{t+1}-C}{1-C}\big) - (t+1) < 0$. Then

$$\log_{L_l}\left(\frac{L_l^{t+1} - C}{1 - C}\right) < t+1, \tag{3.23}$$

$$\frac{L_l^{t+1} - C}{1 - C} > L_l^{t+1},$$

$$L_l^{t+1} - C > L_l^{t+1}(1 - C), \tag{3.24}$$

which implies $L_l^{t+1} > 1$, a contradiction to our hypothesis that $L_l < 1$. Thus,

$$\log_{L_l}\left(\frac{L_l^{t+1} - C}{1 - C}\right) - (t+1) > 0, \tag{3.25}$$

implying $m > 0$. ∎

Given t iterations, Theorem 3.6 lets us determine how many extra iterations are needed when a compressed ZFP data type is used to ensure the error is no larger than the worst case error bound for the traditional fixed-point method. It should be

noted that the number of iterations may be many fewer, as we have only determined a meaningful bound with respect to the traditional error bound for stationary iterative methods. Additionally, there is no guarantee that we can get as close to the original solution as desired due to the finite precision yielding from the lower bound from the choice of $\tilde{K}_\beta$.

3.2. Kreiss bounded linear advancement operator. Full discretization of many time-dependent PDEs takes the form of (3.3), where the iteration index plays the role of the time step. A common property in many time-stepping and fixed-point methods is the use of a linear advancement operator, say $A \in \mathbb{R}^{n\times n}$. This section will analyze iterative methods using compressed ZFP data types with respect to a bounded linear advancement operator. However, the action of (de)compression is a nonlinear operation, similar to traditional floating-point computations. As such, a similar analysis to floating-point error analysis will be used when considering the propagation of an error caused by ZFP in conjunction with a linear operator. The (de)compression operation produces a small perturbation from the original data; e.g., for $\boldsymbol{x} \in \mathbb{R}^n$ there exists $\delta\boldsymbol{x} \in \mathbb{R}^n$ such that

$$DC(\boldsymbol{x}) = \boldsymbol{x} + \delta\boldsymbol{x}. \tag{3.26}$$

From Theorem 2.2, we can quantify the magnitude of the perturbation vector $\delta\boldsymbol{x}$.

LEMMA 3.7 (nonlinear error). *Assume $\boldsymbol{x} \in \mathbb{R}^n$ with $\boldsymbol{x} \neq \boldsymbol{0}$. Then there exists $\delta\boldsymbol{x} \in \mathbb{R}^n$ such that*

$$DC(\boldsymbol{x}) = \boldsymbol{x} + \delta\boldsymbol{x} \quad \text{and} \quad \|\delta\boldsymbol{x}\|_\infty \leq K_\beta \|\boldsymbol{x}\|_\infty. \tag{3.27}$$

Now suppose that we are given the initial value, $\boldsymbol{y}_0$, and we generate the sequences $\{\boldsymbol{y}_t\}_{t=0}^\infty$ and $\{\boldsymbol{x}_t\}_{t=0}^\infty$ by

$$\boldsymbol{y}_{t+1} = A(\boldsymbol{y}_t) \quad \text{and} \quad \boldsymbol{x}_{t+1} = A(DC(\boldsymbol{x}_t)),$$

where $A : \mathbb{R}^n \to \mathbb{R}^n$ is used to denote a linear advancement operator and $\boldsymbol{x}_0 = \boldsymbol{y}_0$. Theorem 3.8 provides an error bound between the sequences $\{\boldsymbol{y}_j\}_{j=0}^{t+1}$ and $\{\boldsymbol{x}_j\}_{j=0}^{t+1}$ at some iterate $t+1$, i.e., a bound on the error introduced by applying inline ZFP (de)compression after t time steps.

THEOREM 3.8 (bounded linear advancement operator). *Suppose that A satisfies the hypothesis of the Kreiss matrix theorem* [23] *with Kreiss constant L_k. Then*

$$\|\boldsymbol{x}_{t+1} - \boldsymbol{y}_{t+1}\|_\infty \leq L_k \sum_{j=0}^{t} K_{\beta_j} \|\boldsymbol{x}_j\|_\infty, \tag{3.28}$$

where β_j is the number of bit planes kept at step j.

Proof. By the Kreiss matrix theorem, there exists a constant $L_k < \infty$ such that $\|A^i\|_2 \leq L_k$ for all $i \in \mathbb{N}$. From Lemma 3.7, $\boldsymbol{x}_{t+1}$ can be decomposed as

$$\boldsymbol{x}_{t+1} = A\left(DC(\boldsymbol{x}_t)\right) = A^{t+1}\boldsymbol{x}^0 + \sum_{j=0}^{t} A^{t-j+1}\delta\boldsymbol{x}_j. \tag{3.29}$$

By our choice of $\boldsymbol{x}_0 = \boldsymbol{y}_0$, it now follows that

$$\boldsymbol{x}_{t+1} - \boldsymbol{y}_{t+1} = \sum_{j=0}^{t} A^{t-j+1}\delta\boldsymbol{x}_j, \tag{3.30}$$

since $\boldsymbol{y}_{t+1} = A^{t+1}\boldsymbol{x}_0$. Finally, as $\|A^i\|_\infty \leq \|A^i\|_2 \leq L_k$ for all $i \in \mathbb{N}$, we conclude that

$$\|\boldsymbol{x}_{t+1} - \boldsymbol{y}_{t+1}\|_\infty = \left\| \sum_{j=0}^{t} A^{t-j+1} \delta \boldsymbol{x}_j \right\|_\infty \leq \sum_{j=0}^{t} \|A^{t-j+1}\|_\infty \|\delta \boldsymbol{x}_j\|_\infty \leq L_k \sum_{j=0}^{t} K_{\beta_j} \|\boldsymbol{x}_j\|_\infty. \quad (3.31)$$

∎

Similarly, Lemma 3.4 can be generalized for Kreiss bounded linear operators.

LEMMA 3.9. *Suppose that A is Kreiss bounded with Kreiss constant L_k, the sequence $\{\boldsymbol{x}_t\}_{t=0}^{\infty}$ defined by (3.4) is bounded, and $\epsilon \geq \frac{K_q \gamma}{1 - L_k}$, where q is the number of bits used in the negabinary representation in ZFP. Then*

$$\|\boldsymbol{x}_{t+1} - \boldsymbol{y}_{t+1}\|_\infty \leq \epsilon, \quad (3.32)$$

provided $\tilde{\beta} \geq 1 - \log_2\left(\left(\frac{4}{15}\right)^d \frac{3\epsilon}{8(2^d+1)\gamma L_k t}\right)$ and $\tilde{\beta} \geq \beta_j$ for all j, where β_j is the number of bit planes kept at step j and where $\gamma = \max_{0 \leq j \leq t} \|\boldsymbol{x}_j\|_\infty$.

Proof. The proof is similar to proof of Lemma 3.4. ∎

The implication of Theorems 3.3 and 3.8 is that the error introduced by repeated compression and decompression in conjunction with an advancement operator can be bounded by factors dependent only on the advancement operator, the error introduced by ZFP, and the dimensionality of the data. Lemmas 3.4 and 3.9 can be used to determine suitable choices for the magnitude of various parameters in ZFP to ensure that the error, relative to the solution represented in the IEEE floating point, is within some tolerance.

3.3. Successive displacement fixed-point methods. So far we have assumed that the solution state is completely decompressed before applying the advancement operator simultaneously to attain the next iterate. However, for fixed-point methods we could easily apply a Gauss–Seidel-type of operator, where each new element of the solution state is used in the computation of the next element. That is, depending on the ordering of the updates, the components of the new iterate will change. This is also known as successive displacement. If ZFP is used in conjunction with successive displacement methods, the additional compression error from each element will propagate into the computation of the next element during the iteration. In an effort to bound the error in a successive displacement setting, we will first study the simplest case for applying ZFP, i.e., where the successive displacement updates are aligned with our ZFP partitioned blocks. First, we will assume the solution states are formatted as a 1D problem (i.e., $d = 1$). Additionally, to simplify the analysis, we assume that the linear advancement operator, B, is partitioned into 4-by-4 blocks so that

$$B = \begin{bmatrix} B_{11} & B_{12} & B_{13} & \cdots & B_{1n_1} \\ B_{21} & B_{22} & B_{23} & \cdots & B_{2n_1} \\ \vdots & & \ddots & & \vdots \\ B_{n_1 1} & B_{n_1 2} & \cdots & & B_{n_1 n_1} \end{bmatrix},$$

where $B \in \mathbb{R}^{4n_1 \times 4n_1}$ and $B_{ij} \in \mathbb{R}^{4\times 4}$. If the operator B cannot be partitioned exactly into 4-by-4 blocks, one can pad the operator with up to three additional ghost variables by including additional zeros in the off diagonal entries and a one along the diagonal. By aligning the structure of the advancement operator with the ZFP blocks, the successive displacement steps can be applied without overlapping the advancement operators with the ZFP blocks, allowing for a simplified error analysis.

Assuming that each element is updated by successive displacement, we generate the sequences $\{\boldsymbol{y}_t\}_{t=0}^{\infty}$ and $\{\boldsymbol{x}_t\}_{t=0}^{\infty}$ by

$$\boldsymbol{y}_{i,t+1} = \sum_{j=1}^{i-1} B_{ij}\boldsymbol{y}_{j,t+1} + \sum_{j=i}^{n_1} B_{ij}\boldsymbol{y}_{j,t} \tag{3.33}$$

and

$$\boldsymbol{x}_{i,t+1} = \sum_{j=1}^{i-1} B_{ij}DC(\boldsymbol{x}_{j,t+1}) + \sum_{j=i}^{n_1} B_{ij}DC(\boldsymbol{x}_{j,t}), \tag{3.34}$$

respectively, where $\boldsymbol{y}_{j,t}$ denotes the j block of the solution state $\boldsymbol{y}_t$ at iterate t and where we assume $\boldsymbol{x}_0 = \boldsymbol{y}_0$. Using Theorem 3.8, equation (3.34) can be rewritten as

$$\boldsymbol{x}_{i,t+1} = \sum_{j=1}^{i-1} B_{ij}(\boldsymbol{x}_{j,t+1} + \delta\boldsymbol{x}_{j,t+1}) + \sum_{j=i}^{n_1} B_{ij}(\boldsymbol{x}_{j,t} + \delta\boldsymbol{x}_{j,t}), \tag{3.35}$$

where $\|\delta\boldsymbol{x}_{i,t}\|_\infty \le \max_j \|\delta\boldsymbol{x}_{j,t}\|_\infty \le \|\delta\boldsymbol{x}_t\|_\infty \le K_{\beta_t}\|\boldsymbol{x}_t\|_\infty$. Last, we will also assume the advancement operator B is block diagonally dominant (see Definition 1 in [9]), defined as

$$\|B_{kk}^{-1}\|_\infty^{-1} \ge \sum_{j=1, j\neq k}^{n_1} \|B_{jk}\|_\infty \tag{3.36}$$

for all $1 \le k \le n_1$, implying a stationary fixed-point method with a Lipschitz constant less than one. Additional analysis is needed to generalize Theorem 3.10 for more general operators and for $d > 1$.[1] Finally, it will be useful for the following analysis to define

$$\alpha_i := \sum_{j=1}^{i-1} \|B_{ij}\|_\infty, \quad \gamma_i := \sum_{j=i+1}^{n_1} \|B_{ij}\|_\infty, \quad \mu_i := \alpha_i + \gamma_i, \tag{3.37}$$

and $\mu := \max_{1\le i\le n_1} \mu_i$. Note that $\alpha_1 = \gamma_{n_1} = 0$.

THEOREM 3.10. *Assume that B is block diagonally dominant. If $\max_k \|B_{kk}^{-1}\|_\infty^{-1} < 1$ for all k, then*

$$\|\boldsymbol{x}_{t+1} - \boldsymbol{y}_{t+1}\|_\infty \le \eta \sum_{j=0}^{t+1} \nu^{t-j+1} K_{\beta_j} \|\boldsymbol{x}^j\|_\infty, \tag{3.38}$$

where $\nu = \max_{1\le i\le n_1} \frac{\gamma_i}{1-\alpha_i}$ and $\eta = 1 + \max_{1\le i\le n_1} \frac{\alpha_i}{1-\alpha_i}$.

[1] At first glance, the assumption in Theorem 3.10 seems highly restrictive. However, we claim that this is a less restrictive assumption when blocks of size 4^d are chosen spatially (internally connected), much as they often are for block Gauss–Seidel, than in common classical theory, which assumes symmetric positive definiteness or strict diagonal dominance. Padding with columns/rows of the identity is used for problems when it is not easy to find spatially connected blocks of exactly size 4^d. For a specific class of advancement operator, meeting the assumption could be analyzed, but this is beyond the scope of this paper.

Proof. As B is block diagonally dominant, we have

$$\left(\|B_{jj}^{-1}\|_\infty\right)^{-1} \geq \sum_{k=1,k\neq j} \|B_{jk}\|_\infty = \mu_j \quad \forall j, \tag{3.39}$$

implying that $\mu < 1$. Let $\hat{i} \in \{1,\ldots,n_1\}$ and $\hat{k} \in \{1,\ldots,4\}$ be the indices such that

$$\left|\left[\boldsymbol{x}_{\hat{i},t+1} - \boldsymbol{y}_{\hat{i},t+1}\right]_{\hat{k}}\right| = \max_{i,k} \left|\left[\boldsymbol{x}_{i,t+1} - \boldsymbol{y}_{i,t+1}\right]_k\right| = \|\boldsymbol{x}_{t+1} - \boldsymbol{y}_{t+1}\|_\infty. \tag{3.40}$$

Then

(3.41)

$$\|\boldsymbol{x}_{t+1} - \boldsymbol{y}_{t+1}\|_\infty = \max_{i,k} |\left[\boldsymbol{x}_{i,t+1} - \boldsymbol{y}_{i,t+1}\right]_k|$$

$$= \max_i \left| \left[\sum_{j=1}^{i-1} B_{ij}(\boldsymbol{x}_{j,t+1} - \boldsymbol{y}_{j,t+1} + \delta\boldsymbol{x}_{j,t+1}) + \sum_{j=i}^{n_1} B_{ij}(\boldsymbol{x}_{j,t} - \boldsymbol{y}_{j,t} + \delta\boldsymbol{x}_{j,t}) \right]_{\hat{k}} \right| \tag{3.42}$$

$$\leq \sum_{j=1}^{\hat{i}-1} \|B_{\hat{i}j}\|_\infty \|\boldsymbol{x}_{t+1} - \boldsymbol{y}_{t+1} + \delta\boldsymbol{x}_{t+1}\|_\infty + \sum_{j=\hat{i}}^{n_1} \|B_{\hat{i}j}\|_\infty \|\boldsymbol{x}_t - \boldsymbol{y}_t + \delta\boldsymbol{x}_t\|_\infty \tag{3.43}$$

$$\leq \frac{\gamma_{\hat{i}}}{1-\alpha_{\hat{i}}} \|\boldsymbol{x}_t - \boldsymbol{y}_t\|_\infty + \frac{\alpha_{\hat{i}}}{1-\alpha_{\hat{i}}} \|\delta\boldsymbol{x}_{t+1}\|_\infty + \frac{\gamma_{\hat{i}}}{1-\alpha_{\hat{i}}} \|\delta\boldsymbol{x}_t\|_\infty \tag{3.44}$$

$$\leq \left(1 + \frac{\alpha_{\hat{i}}}{1-\alpha_{\hat{i}}}\right) \sum_{j=0}^{t+1} \left(\frac{\gamma_{\hat{i}}}{1-\alpha_{\hat{i}}}\right)^{t-j+1} K_{\beta_j} \|\boldsymbol{x}_j\|_\infty \tag{3.45}$$

$$\leq \eta \sum_{j=0}^{t+1} \nu^{t-j+1} K_{\beta_j} \|\boldsymbol{x}_j\|_\infty. \tag{3.46}$$

Hence, for each i,

$$\alpha_i + \gamma_i - \frac{\gamma_i}{1-\alpha_i} = \frac{\alpha_i}{1-\alpha_i}(1 - (\alpha_i + \gamma_i)) \geq \frac{\alpha_i}{1-\alpha_i}(1-\mu) \geq 0, \tag{3.47}$$

which yields that $\nu \leq \mu < 1$. Thus, (3.46) is bounded provided that the solution $\|\boldsymbol{x}_t\|_\infty$ is bounded for all time steps. ∎

In the previous section, we were assuming exact arithmetic when applying the advancement operators; however, traditional floating-point error analysis assumes that round-off will occur in both the number representation and the arithmetic operations. In the next section, we will provide a detailed discussion of the forward and backward error analysis of the accumulated round-off error of both floating-point and ZFP round-off error. The analysis will follow traditional floating-point analysis, allowing us to study the relationship between the two types of error. All our analysis thus far has assumed that the error present in a simulation is aggregated together. Instead, we can further break down the round-off error into the floating-point arithmetic error and the ZFP compressed round-off error.

3.4. Forward and backward error analysis for linear stationary iterative methods. This section presents the forward and backward error analysis for linear stationary iterative methods with compressed data types for solving the system $A\boldsymbol{x} = \boldsymbol{b}$, where $A \in \mathbb{R}^{n_1 \times n_1}$ is nonsingular and $\boldsymbol{b} \in \mathbb{R}^{n_1}$. Decompose A such that $A = M - N$ and M is nonsingular. A stationary iterative method has the form

$$M\boldsymbol{y}_{t+1} = N\boldsymbol{y}_t + \boldsymbol{b}. \tag{3.48}$$

Define the matrices $G = M^{-1}N$ and $H = NM^{-1}$. Define $\sigma := \|G\|_\infty$ and $\omega := \|H\|_\infty$, and assume the spectral radius of G and H are bounded by one, i.e., $\sigma < 1$. As H and G are similar matrices, we have $\omega = \sigma$. Since $\sigma < 1$, one can show in exact arithmetic that the sequence $\{\boldsymbol{y}_t\}_{t=0}^{\infty}$ converges for an appropriate starting vector, $\boldsymbol{y}_0$. Detailed forward and backward error analysis for the sequence $\{\boldsymbol{y}_t\}_{t=0}^{\infty}$ can be found in [11]. We would like to perform a similar analysis for the sequence $\{\boldsymbol{x}_t\}_{t=0}^{\infty}$ defined by

$$M\boldsymbol{x}_{t+1} = N(DC(\boldsymbol{x}_t)) + \boldsymbol{b}, \tag{3.49}$$

where the solution vector $\boldsymbol{x}_t$ is compressed and decompressed before applying the update and where $\boldsymbol{x}_0 = \boldsymbol{y}_0$. Traditionally, in perturbation theory, the computed vectors satisfy the equality

$$(M + \Delta M_{t+1})\boldsymbol{y}_{t+1} = N\boldsymbol{y}_t + \boldsymbol{b} + \boldsymbol{f}_t, \tag{3.50}$$

where ΔM_t and $\boldsymbol{f}_t$ account for floating-point arithmetic errors in solving the linear system and forming the right-hand side, $N\boldsymbol{y}_t + \boldsymbol{b}$. Typically, $\|\Delta M_t\|_\infty \leq \epsilon_k b'_{n_1}\|M\|_\infty$ and $\|\boldsymbol{f}_t\|_\infty \leq \epsilon_k b_{n_1}(\|N\|_\infty\|\boldsymbol{y}_t\|_\infty + \|\boldsymbol{b}\|_\infty)$ for some constants b_{n_1} and b'_{n_1} (see sections 7 and 17 in [11] for details). For simplicity, define $c_{n_1} = \max\{b_{n_1}, b'_{n_1}\}$. The magnitude of c_{n_1} represents the accumulation of floating-point arithmetic errors, which is dependent on the size of the matrix as well as the conditioning of M^{-1}. In most cases, it can be assumed that the order of magnitude of c_{n_1} is with respect to n_1, where n_1 is the number of unknowns in the linear system. In [26], the constant c_{n_1} is dependent on the type of norm; however, it is shown to always be greater than one [12]. Similarly, the sequence $\{\boldsymbol{x}_t\}_{t=0}^{\infty}$ will satisfy the equality

$$(M + \Delta\tilde{M}_{t+1})\boldsymbol{x}_{t+1} = N\left(DC(\boldsymbol{x}_t)\right) + \boldsymbol{b} + \tilde{\boldsymbol{f}}_t. \tag{3.51}$$

Note that $\Delta\tilde{M}_{t+1}$ and $\tilde{\boldsymbol{f}}_t$ in (3.51) differ slightly from the corresponding quantities in (3.50). It follows that there exists a constant $\tilde{c}_{n_1}$ such that $\|\Delta\tilde{M}_t\|_\infty \leq \epsilon_k\tilde{c}_{n_1}\|M\|_\infty$ and $\|\tilde{\boldsymbol{f}}_t\|_\infty \leq \epsilon_k\tilde{c}_{n_1}(\|N\|_\infty\|\boldsymbol{x}_t\|_\infty + \|\boldsymbol{b}\|_\infty)$. However, we now have

$$\begin{aligned} \|\tilde{\boldsymbol{f}}_t\|_\infty &\leq \epsilon_k\tilde{c}_{n_1}\left(\|N\|_\infty\|DC(\boldsymbol{x}_t)\|_\infty + \|\boldsymbol{b}\|_\infty\right) \qquad (3.52)\\ &\leq \epsilon_k\tilde{c}_{n_1}\left((1+K_{\beta_t})\|N\|_\infty\|\boldsymbol{x}_t\|_\infty + \|\boldsymbol{b}\|_\infty\right). \qquad (\text{Theorem 2.2}) \end{aligned}$$

Using Lemma 3.7, equation (3.51) can be written as

$$M\boldsymbol{x}_{t+1} = N\boldsymbol{x}_t + \boldsymbol{b} + \zeta_t + N\delta\boldsymbol{x}_t, \tag{3.53}$$

where $\zeta_t = \tilde{\boldsymbol{f}}_t - \Delta\tilde{M}_{t+1}\boldsymbol{x}_{t+1}$. Let $\boldsymbol{x}^*$ represent the exact fixed-point solution of (3.48), and assume $\boldsymbol{x}^* \neq \boldsymbol{0}$. Then define $\gamma = \sup_i \frac{\|\boldsymbol{x}_i\|_\infty}{\|\boldsymbol{x}^*\|_\infty}$. Recall that the constant K_{β_t}, defined in (2.5), bounds the relative error caused by ZFP compression at iterate t and $\tilde{K}_\beta := \max_i K_{\beta_i}$. Then we have

$$\begin{aligned} \|\zeta_t\|_\infty &\leq \|\tilde{\boldsymbol{f}}_t\|_\infty + \|\Delta\tilde{M}_{t+1}\|_\infty\|\boldsymbol{x}_{t+1}\|_\infty \qquad (3.54)\\ &\leq \epsilon_k\tilde{c}_{n_1}\left((1+K_{\beta_t})\|N\|_\infty\|\boldsymbol{x}_n\|_\infty + \|M\|_\infty + \|A\boldsymbol{x}^*\|_\infty\right)\\ &\leq \epsilon_k\tilde{c}_{n_1}\left((1+\gamma)\left(\|N\|_\infty + \|M\|_\infty\right) + K_{\beta_t}\gamma\|N\|_\infty\right)\|\boldsymbol{x}^*\|_\infty. \end{aligned}$$

By applying the recurrence relation, we obtain

$$\boldsymbol{x}_{t+1} = G^{t+1}\boldsymbol{x}_0 + \sum_{j=0}^{t} G^{t-j}M^{-1}(\boldsymbol{b} + \zeta_j) + \sum_{j=0}^{t} G^{t-j+1}\delta\boldsymbol{x}_j. \tag{3.55}$$

Since $\boldsymbol{x}^*$ is the fixed point, we have

$$\boldsymbol{x}^* = G^{t+1}\boldsymbol{x}^* + \sum_{j=0}^{t} G^{t-j}M^{-1}\boldsymbol{b}, \tag{3.56}$$

and the error at $t+1$ is represented as

$$\boldsymbol{x}_{t+1} - \boldsymbol{x}^* = G^{t+1}(\boldsymbol{x}_0 - \boldsymbol{x}^*) + \sum_{j=0}^{t} G^{t-j}M^{-1}\zeta_j + \sum_{j=0}^{t} G^{t-j+1}\delta\boldsymbol{x}_j. \tag{3.57}$$

Define $\theta_{\mathrm{f},t} = \frac{1-\sigma^{t+1}}{1-\sigma}$. By applying norms and using the inequality (3.4),

$$\begin{aligned}
\|\boldsymbol{x}_{t+1} - \boldsymbol{x}^*\| &\le \sigma^{t+1}\|\boldsymbol{x}_0 - \boldsymbol{x}^*\|_\infty + \max_i \|\zeta_i\|_\infty \|M^{-1}\|_\infty \sum_{j=0}^{t} \|G^{t-j}\|_\infty \\
&\qquad + \tilde{K}_\beta \gamma \left(\sum_{j=0}^{t} \|G^{t-j+1}\|_\infty \right) \|\boldsymbol{x}^*\|_\infty \\
&\le \sigma^{t+1}\|\boldsymbol{x}_0 - \boldsymbol{x}^*\|_\infty + \left(\max_i \|\zeta_i\|_\infty \|M^{-1}\|_\infty + \tilde{K}_\beta \gamma \|G\|_\infty \right) \left(\sum_{j=0}^{t} \|G^{t-j+1}\|_\infty \right) \|\boldsymbol{x}^*\|_\infty \\
&\le \sigma^{t+1}\|\boldsymbol{x}_0 - \boldsymbol{x}^*\|_\infty + \Bigg(\underbrace{\epsilon_k \tilde{c}_{n_1}(1+\gamma)\|M^{-1}\|_\infty \left(\|N\|_\infty + \|M\|_\infty\right)}_{\text{Traditional Floating-Point Error}} \\
&\qquad + \underbrace{\tilde{K}_\beta \gamma \left(\epsilon_k \tilde{c}_{n_1} \|M^{-1}\|_\infty \|N\|_\infty + \sigma \right)}_{\text{ZFP Round-off}} \Bigg) \theta_{\mathrm{f},t} \|\boldsymbol{x}^*\|_\infty.
\end{aligned} \tag{3.58}$$

The bound can be seen as a sum of two error terms, one reflecting the traditional error caused by floating-point arithmetic,

$$\Phi_{\mathrm{FP}} := \epsilon_k \tilde{c}_{n_1}(1+\gamma)\|M^{-1}\|_\infty \left(\|N\|_\infty + \|M\|_\infty\right), \tag{3.59}$$

and the other term reflecting the error caused by ZFP round-off,

$$\Phi_{\mathrm{ZFP}} := \tilde{K}_\beta \gamma \left(\epsilon_k \tilde{c}_{n_1} \|M^{-1}\|_\infty \|N\|_\infty + \sigma \right). \tag{3.60}$$

To ensure the additional round-off error caused by applying ZFP repeatedly is less than the traditional forward error, we must have $\Phi_{\mathrm{FP}} \ge \Phi_{\mathrm{ZFP}}$. Lemma 3.11 provides the minimum value of $\tilde{\beta}$ so that $\Phi_{\mathrm{FP}} \ge \Phi_{\mathrm{ZFP}}$.

LEMMA 3.11. *Let $k \in \mathbb{N}$ represent the traditional floating-point precision, and define $\kappa_\infty(M) := \|M\|_\infty \|M^{-1}\|_\infty$ as the condition number of M. Assume $\tilde{c}_{n_1} \le \frac{1}{\epsilon_k \kappa_\infty(M)}$; then $\Phi_{FP} \ge \Phi_{ZFP}$ if*

$$\tilde{\beta} \ge k - \log_2 \left(\frac{1+\sigma}{\sigma} \tilde{c}_{n_1} \left(\frac{4}{15} \right)^d \left(\frac{3}{8(2^d+1)} \right) \right) \tag{3.61}$$

provided $\tilde{\beta} \ge \beta_j$ for all j, where β_j is the number of bit planes kept at step j.

Proof. By applying the inequality $\|AB\| \le \|A\|\|B\|$ for matrices A and B and the assumption $\tilde{c}_{n_1} \le \frac{1}{\epsilon_k \kappa_\infty(M)}$, we have that

$$\Phi_{\mathrm{FP}} \ge \epsilon_k \tilde{c}_{n_1} \gamma \|M^{-1}\|_\infty \left(\|N\|_\infty + \|M\|_\infty\right) \ge \epsilon_k \tilde{c}_{n_1} \gamma \left(\|G\|_\infty + 1\right) \ge \epsilon_k \tilde{c}_{n_1} \gamma \left(\sigma + 1\right) \tag{3.62}$$

and

(3.63)
$$\Phi_{\text{ZFP}} \leq \tilde{K}_\beta \gamma \left(\epsilon_k \tilde{c}_{n_1} \|M^{-1}\|_\infty \|N\|_\infty + \sigma\right) \leq \tilde{K}_\beta \gamma \left(\epsilon_k \tilde{c}_{n_1} \|M^{-1}\|_\infty \|M\|_\infty \|M^{-1}N\|_\infty + \sigma\right) \leq 2\tilde{K}_\beta \gamma \sigma.$$

Thus, if

$$2\tilde{K}_\beta \gamma \sigma \leq \epsilon_k \tilde{c}_{n_1} \gamma \left(\sigma + 1\right), \tag{3.64}$$

it follows that $\Phi_{\text{FP}} \geq \Phi_{\text{ZFP}}$. Without loss of generality, $\tilde{K}_\beta = \left(\frac{15}{4}\right)^d \frac{8}{3}(2^d + 1)\epsilon_{\tilde{\beta}}$, and the desired result follows. ∎

To serve as a visual aid and to help in the interpretation of Lemma 3.11, consider the contour plot in Figure 3a of $\tilde{\beta}$ given by (3.61) as a function of $\tilde{c}_{n_1}$ and σ. We would expect that, as the conditioning and the size of the matrix A grows, represented by the growth of $\tilde{c}_{n_1}$, ZFP is able to use a lower $\tilde{\beta}$ to compress more aggressively and still have the traditional floating-point arithmetic error dominate the ZFP round-off error. We see from Figure 3a that this is indeed true. If we have a poorly conditioned linear system, it is known that the floating-point round-off error will accumulate significantly. Lemma 3.11 and Figure 3a show that if $\tilde{\beta}$ is chosen appropriately, ZFP can represent the solution with fewer bits while retaining double-precision accuracy. For a well-conditioned system, typically, the floating-point round-off error will remain small, and Lemma 3.11 indicates that ZFP will be unable to compress aggressively if the only condition is that the ZFP error remains below the floating-point round-off error. However, in most computational simulations, errors other than floating-point round-off error will dominate, e.g., truncation error. In this situation, Lemmas 3.4 and 3.9 would be more useful, as $\tilde{\beta}$ can be determined from an approximation of the dominating simulation error.

We will now consider the backward error of the sequence $\{\boldsymbol{x}_t\}_{t=0}^{\infty}$. The backward error (see Chapter 7 in [11]) with respect to A and $\mathbf{b}$ is given by

$$\eta_{A,\boldsymbol{b}}(\boldsymbol{y}) = \frac{\|\boldsymbol{r}\|_\infty}{\|A\|_\infty \|\boldsymbol{y}\|_\infty + \|\boldsymbol{b}\|_\infty}, \tag{3.65}$$

where $\boldsymbol{r}$ is the residual defined by $\boldsymbol{r} := \boldsymbol{b} - A\boldsymbol{y}$, which contains both traditional floating-point arithmetic error and ZFP round-off error. Using (3.55), we have

$$\begin{aligned} \boldsymbol{r}_{t+1} &= AG^{t+1}A^{-1}(\boldsymbol{b} - A\boldsymbol{x}_0) - \sum_{j=0}^{t} AG^{t-j}M^{-1}(\boldsymbol{b} + \zeta_j) - \sum_{j=0}^{t} AG^{t-j+1}\delta\boldsymbol{x}_j \\ &= H^{t+1}(\boldsymbol{b} - A\boldsymbol{x}_0) - \sum_{j=0}^{t} H^{t-j}(I - H)\zeta_j + \sum_{j=0}^{t} H^{n-j+1}A\delta\boldsymbol{x}_j. \end{aligned} \tag{3.66}$$

Define $\theta_{\text{b},t} := \frac{1-\omega^{t+1}}{1-\omega}$. Then, by applying norms,

$$\begin{aligned} \|\boldsymbol{r}_{t+1}\|_\infty \leq \omega^{t+1}\|\boldsymbol{r}_0\|_\infty + \Bigg(& \underbrace{\|I - H\|_\infty \epsilon_k \tilde{c}_{n_1} (1 + \gamma) \left(\|M\|_\infty + \|N\|_\infty\right)}_{\text{Traditional Floating-Point Error}} \\ & + \underbrace{\tilde{K}_\beta \gamma \left(\epsilon_k \tilde{c}_{n_1} \|I - H\|_\infty \|N\|_\infty + \|A\|_\infty \omega\right)}_{\text{ZFP Round-off}} \Bigg) \theta_{\text{b},t} \|\boldsymbol{x}^*\|_\infty. \end{aligned} \tag{3.67}$$

Similar to the forward error bound, the backward error bound can be seen as a sum of two error terms, one reflecting the traditional error caused by floating-point arithmetic,

$$\Omega_{\text{FP}} := \|I - H\|_\infty \epsilon_k \tilde{c}_{n_1} (1+\gamma) \left(\|M\|_\infty + \|N\|_\infty\right), \tag{3.68}$$

and the other error caused by ZFP round-off,

$$\Omega_{\text{ZFP}} := \tilde{K}_\beta \gamma \left(\epsilon_k \tilde{c}_{n_1} \|I - H\|_\infty \|N\|_\infty + \|A\|_\infty \omega\right). \tag{3.69}$$

Similar to Lemma 3.11, Lemma 3.12 provides the minimum $\tilde{\beta}$ so that $\Omega_{\text{FP}} \geq \Omega_{\text{ZFP}}$.

LEMMA 3.12. *Let $k \in \mathbb{N}$ represent the traditional floating-point precision. Assume $\tilde{c}_{n_1} \leq \frac{1}{\epsilon_k}$; then $\Phi_{FP} \geq \Phi_{ZFP}$ if*

$$\tilde{\beta} \geq k - \log_2 \left(\frac{1-\omega}{(1+2\omega)} \tilde{c}_{n_1} \left(\frac{4}{15}\right)^d \left(\frac{3}{8(2^d+1)}\right) \right) \tag{3.70}$$

provided $\tilde{\beta} \geq \beta_j$ for all j, where β_j is the number of bit planes kept at step j.

Proof. By applying the inequality $\|A\| \leq \|M\| + \|N\|$ and applying the assumptions $\tilde{c}_{n_1} \leq \frac{1}{\epsilon_k}$ and $\omega < 1$, we have that

$$\Omega_{\text{FP}} \geq \|I - H\|_\infty \epsilon_k \tilde{c}_{n_1} \gamma \|A\|_\infty \geq \left| \|I\|_\infty - \|H\|_\infty \right| \epsilon_k \tilde{c}_{n_1} \gamma \|A\|_\infty \geq (1-\omega) \epsilon_k \tilde{c}_{n_1} \gamma \|A\|_\infty \tag{3.71}$$

and

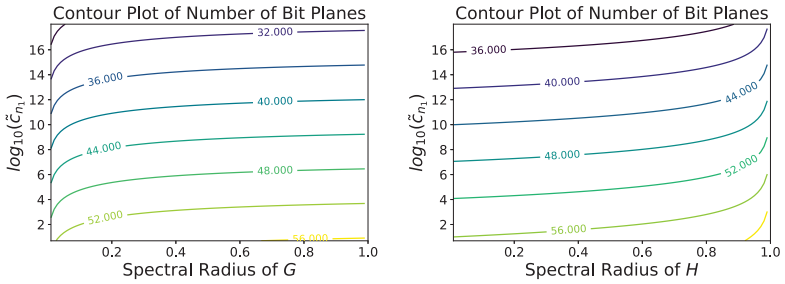
$$\Omega_{\text{ZFP}} \geq \tilde{K}_\beta \gamma \|A\|_\infty \left(\epsilon_k \tilde{c}_{n_1} (1+\omega) + \omega\right) \geq \tilde{K}_\beta \gamma \|A\|_\infty (1+2\omega). \tag{3.72}$$

Thus, if $\tilde{K}_\beta (1+2\omega) \leq (1-\omega)\epsilon_k \tilde{c}_{n_1}$, it follows that $\Omega_{\text{FP}} \geq \Omega_{\text{ZFP}}$. Without loss of generality, $\tilde{K}_\beta = \left(\frac{15}{4}\right)^d \frac{8}{3}(2^d+1)\epsilon_{\tilde{\beta}}$, and the desired result follows. ∎

Figure 3b represents $\tilde{\beta}$ as described in (3.70) as a function of $\tilde{c}_{n_1}$ and ω for the backward error analysis. Similar conclusions from Figure 3a for the forward error analysis can be concluded for the backward error. However, for the backward error, we see that $\tilde{\beta}$ is much more sensitive to the spectral radius.

For both cases, the forward and backward errors for the sequence defined by (3.55) contain traditional floating-point arithmetic error and round off error caused by ZFP. Lemmas 3.11 and 3.12 provide the $\tilde{\beta}$ value needed to use ZFP in a stationary iterative method while keeping the additional round-off error caused by ZFP less than the traditional round-off error caused by floating-point arithmetic. The contour plots, shown in Figures 3a and 3b, tell us a familiar story; the accuracy of the solution is dependent on the conditioning of the problem; thus, requiring a higher precision for our data representations than what is theoretically possible is fruitless. This is not a new phenomenon but one that has been well documented from studying floating-point arithmetic error [11]. Again, this section specifically studied the relationship between floating-point arithmetic error and ZFP round-off error. In many simulations, the floating-point arithmetic error is not the dominating error. We have shown that despite the possibilities of ZFP round-off error dominating floating-point arithmetic error that as long as the iterative method is stable, we can bound the ZFP round-off error, ensuring the method remains stable when used in conjunction with ZFP compressed data types.

Now that we have established bounds on the accumulated round-off error introduced by ZFP compression and decompression, we consider several numerical experiments to illustrate the tightness of these bounds.



(a) Forward Error (b) Backward Error

FIG. 3. *Contour plot of $\tilde{\beta}$, the number of bit-planes for ZFP, from Lemma* 3.11 *as a function of $\tilde{c}_{n_1}$, the conditioning of the linear operator, and the spectral radius of either G or H, respectively, as determined by the forward error or backward analysis in double precision ($k = 53$).* (a) *Contour plot of $\tilde{\beta}$ as a function of $\tilde{c}_{n_1}$ and σ, the spectral radius of G.* (b) *Contour plot of $\tilde{\beta}$ as a function of $\tilde{c}_{n_1}$ and ω, the spectral radius of H.*

4. Numerical results. As observed in section 3, repeated inline applications of ZFP with iterative methods will generate additional errors that can accumulate over time. The first two numerical experiments are designed to test how well the bounds established in Theorems 3.3 and 3.8 capture the compression error introduced by ZFP for different values of β, the fixed precision parameter. In each simulation, the value of the fixed precision parameter is chosen ahead of time and is held constant throughout each simulation. Again, due to hardware limitations, in order to apply the update, arithmetic operations are performed in IEEE double precision; that is, each ZFP solution vector is rounded to IEEE doubles and updated, and then the result is converted back to ZFP format. In each of the experiments, we will compare the additional error to the total numerical error to discuss the impact of the dominant error on the numerical solution, as an exact analytic solution is known. In particular, the total numerical error is determined by subtracting the exact analytic solution from the IEEE double-precision solution. For our chosen experiments, the total numerical error will be dominated by truncation error. We will observe that the round-off error introduced by ZFP will remain below the total numerical error for the correct choice of the fixed-precision parameter and that our bounds fully encapsulate the additional error. The final numerical experiment is designed to illustrate Theorem 3.6. Depending on the Lipschitz constant, the fixed-precision parameter, and the number of iterations required to solve the problem without inline compression, we can bound the number of extra iterations required by the ZFP scheme in (3.4) to achieve an iterate that is of similar accuracy to the exact solution of the noncompressed fixed-point scheme in (3.3).

In the interest of using notation common to the PDE literature, the notation we use in this section differs from the rest of the paper: x is a d-dimensional spatial variable, t is the temporal variable, n is the current iterate, u is the continuous solution, and $\boldsymbol{u}$ is an array used for the uncompressed solution to the discretized diffusion equations. For each example, the finite difference scheme can be written as $\boldsymbol{u}^{n+1} = A\boldsymbol{u}^n$, where $\boldsymbol{u}^n$ is the solution calculated in double precision with no inline compression at time step n and A is the advancement operator. Let $\boldsymbol{v}^{n+1} = A(DC(\boldsymbol{v}^n))$, where $\boldsymbol{v}^0 = \boldsymbol{u}^0$, denote the inline ZFP compression sequence, and let $\hat{\boldsymbol{u}}(x,t)$ denote the exact

solution evaluated in double precision. In Figures 4, 5, 6, 7, and 9a, the blue lines depict the error introduced by repeated compression of the solution at each time step relative to the true solution, $\|\boldsymbol{v}^{n+1} - \boldsymbol{u}^{n+1}\|_\infty / \|\hat{\boldsymbol{u}}^{n+1}\|_\infty$. The green lines represent the total numerical error relative to the true solution, which, for the mesh resolutions chosen, is dominated by truncation error, $\|\hat{\boldsymbol{u}}^{n+1} - \boldsymbol{u}^{n+1}\|_\infty / \|\hat{\boldsymbol{u}}^{n+1}\|_\infty$, where $\hat{\boldsymbol{u}}(x,t)$ is the exact solution evaluated in double precision. The red lines represent the theoretical bound from either Theorem 3.3 or 3.5. Finally, the black line represents the round-off error caused by IEEE double-precision representation relative to the true solution. Let $\tilde{\boldsymbol{u}}^n$ represent the numerical solution calculated in 80-bit precision at iterate n; then the black line represents $\|\boldsymbol{u}^{n+1} - \tilde{\boldsymbol{u}}^{n+1}\|_\infty / \|\hat{\boldsymbol{u}}^{n+1}\|_\infty$. We chose to present the total numerical error that is dominated by the truncation error, as it is important to understand when the error caused by floating-point round-off error or by the repeated use of ZFP, with appropriate parameters, is less than or greater than the dominating error. If the error caused by ZFP remains less than the total numerical error, in our example the bits that represent the ZFP error are meaningless for the solution. Additionally, as ZFP is a compression algorithm, we also present the compression ratio, i.e., the ratio between the uncompressed size and compressed size at time n, for varying β values.

4.1. Diffusion example. In the first example, we wish to test the bounds of the propagation of the additional error introduced by ZFP in an iterative method as derived in Theorem 3.3. The specific PDE we solve is

$$\partial_t u(x,t) = a\nabla^2 u(x,t), \qquad (x,t) \in [0,1]^d \times (0,1] \equiv \Omega, \tag{4.1}$$

with initial and boundary conditions,

$$u(x,t) = 0 \quad \forall x \in \delta\Omega \quad \text{and} \quad u(x,t=0) = \begin{cases} 1, & x = \hat{x}, \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

for $d = 2$, $\hat{x} = (1/2, 1/2)$, and constant $a = 1$. Using a forward time and central space finite difference approximation, the update is given by

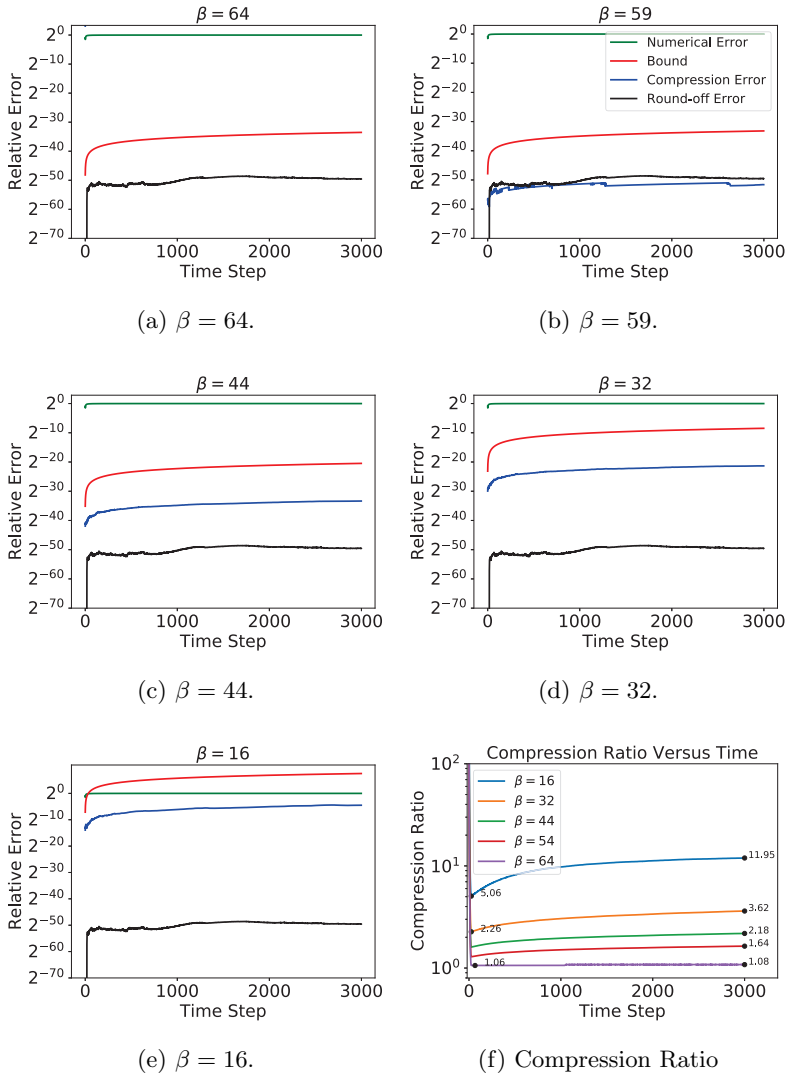
$$u_{i,j}^{n+1} = u_{i,j}^n + a\Delta t \left(\frac{u_{i,j+1}^n - 2u_{i,j}^n + u_{i,j-1}^n}{\Delta x_1^2} + \frac{u_{i+1,j}^n - 2u_{i,j}^n + u_{i-1,j}^n}{\Delta x_2^2} \right). \tag{4.3}$$

We choose a reasonable set of discretization parameters, $i, j = 1, \ldots, 99$, $\Delta x_1 = \Delta x_2 = 1/100$, $\Delta t = 0.0003125$, and the Dirichlet periodic boundary conditions are enforced by $u_{i,0}^n = u_{0,j}^n = 0$ for all i, j. The finite difference scheme can be written as $\boldsymbol{u}^{n+1} = A\boldsymbol{u}^n$, where $\boldsymbol{u}^n = [u_{i,j}^n]^T$ is the solution calculated in double precision with no inline compression at time step n with Lipschitz constant $L_l = \|A\|_\infty = 1$. Let $\boldsymbol{v}^{n+1} = A(DC(\boldsymbol{v}^n))$, where $\boldsymbol{v}^0 = \boldsymbol{u}^0$, denote the inline ZFP compression sequence. In this example, shown in Figure 4, the red lines represent the theoretical bound from Theorem 3.3,

$$\sum_{j=0}^{n} L_l^{n-j+1} K_{\beta_j} \|\boldsymbol{v}^j\|_\infty, \tag{4.4}$$

where $\beta_j = \{64, 59, 44, 32, 16\}$ is constant for each experiment presented.

For all β_j, the theoretical prediction bounds the relative error caused by the compression. As β_j decreases, the relative round-off error from ZFP compression and



(a) $\beta = 64$. (b) $\beta = 59$.

(c) $\beta = 44$. (d) $\beta = 32$.

(e) $\beta = 16$. (f) Compression Ratio

FIG. 4. *2D diffusion example: The blue line represents the compression error caused by ZFP, the green line represents the truncation-dominated total error, the red line represents the theoretical bound, and the black line represents the IEEE double-precision floating-point round-off error.*

the theoretical bound both increase slowly during the time-stepping iterations and trend in a similar way. The truncation-dominated numerical error is well above the compression error caused by ZFP, which means that the error is still dominated by the error caused by discretization. When $\beta = 64$, the error caused by compression is lower than the error caused by the double-precision round-off estimated by computing the solution using 80-bit precision, i.e., the total floating-point arithmetic error. Even though $\beta = 64$, the ZFP data type is using fewer bits, as ZFP uses an encoding scheme to represent the values more efficiently. From Lemma 3.11, assuming $\tilde{c}_{n_1} \approx 10,000$, i.e., the number of unknowns which can be used as an estimate of the condition number, we must have $\tilde{\beta} \geq 44$ for the round-off caused by ZFP to be less than the

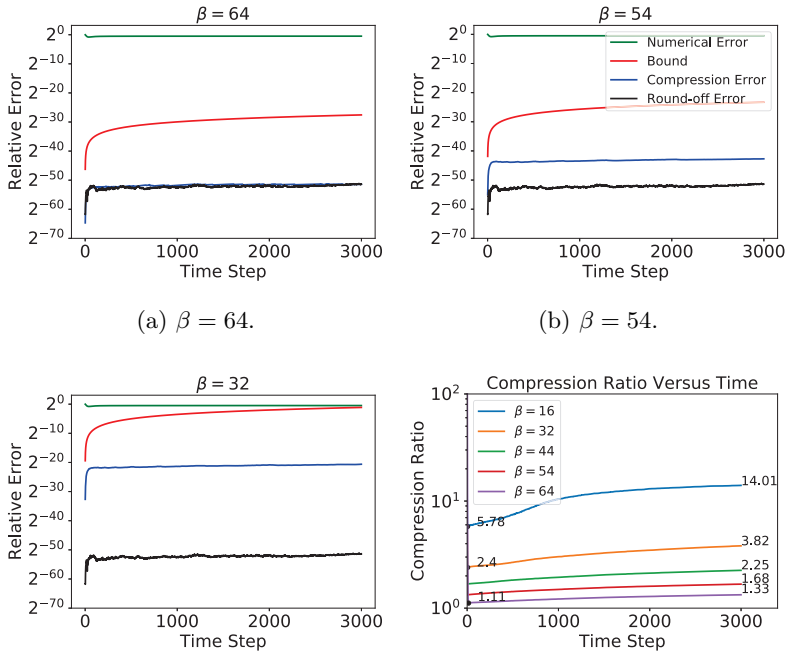
total floating-point arithmetic error using the forward error analysis. However, we see that this is not the case in Figure 4c. As A is a sparse well-conditioned matrix, $\tilde{c}_{n_1} \approx 10,000$ is likely a pessimistic approximation. If instead we assume $\tilde{c}_{n_1} \approx 1$, the most optimistic approximation, then from Lemma 3.11 we must have $\tilde{\beta} \geq 59$, which is consistent with Figure 4b. For each iterate, the error caused by ZFP remains below the floating-point arithmetic error validating Lemma 3.11. If instead we were concerned with the total numerical error, we could use Lemma 3.9 to estimate the value $\tilde{\beta}$ that will result in the error bound to be approximately the same magnitude as the truncation-dominated numerical error. The numerical error from the numerical experiment can be concluded to be $\approx 2^{-14}$ at $n = 3000$. From Lemma 3.9, we have that $\tilde{\beta} \geq 32$, which is depicted in Figure 4d. However, the actual accumulated error caused by ZFP will be much less than 2^{-14}. We can conclude that even though ZFP is introducing error into the numerical simulation, the method remains stable, as the theorems predict. When $\beta_j = 16$, depicted in Figure 4e, the total numerical error still dominates the ZFP round-off error. Our bound, though it encapsulates the introduced error, is more conservative than is necessary; 16 bit-planes are sufficient to obtain a solution with the same level of total numerical error.

Figure 4f presents the compression ratio, i.e., the ratio between the uncompressed size and compressed size at time n, for varying β values. For the first few time steps, ZFP is able to efficiently compress the data, as there is a single-point heat source while the remaining domain is approximately zero. However, there is a sharp decrease in the compression ratio until approximately $n \approx 80$, where the minimum compression ratio is attained for each β value. The minimum compression ratio is 1.09 at $n = 83$ with $\beta = 64$. Then, as $n \geq 83$ increases, the compression ratio increases. The final compression ratio at $n = 3000$ is plotted for each β value. For $\beta = 64$, the final compression ratio is 1.31.

ZFP is known to produce higher compression ratios for higher dimensions. Thus, we perform the same numerical example in three dimensions using the same setup as before with $\Delta x_1 = \Delta x_2 = \Delta x_3 = 1/40$, and $\Delta t = 2.5 \times 10^{-5}$. The Dirichlet periodic boundary conditions are enforced by $u^n_{i,j,0} = u^n_{0,j,h} = u^n_{i,0,h} = 0$ for all i, j, h, and the initial condition is $u_{19,19,19} = 1$. Figure 5 displays results similar to those of the 2D case for the 3D diffusion example, and similar conclusions from the 2D case can be drawn. The total numerical error from the numerical experiment can be concluded to be $\approx 2^{-14}$ at $n = 3000$. Similarly, from Lemma 3.9, we need $\tilde{\beta} \geq 32$ for the ZFP round-off error to remain below the numerical error, which is depicted in Figure 5c. Figure 5d presents the compression ratio for varying β values. As expected, the compression ratios tend to be higher in the 3D case than in the 2D case (although not substantially).

4.2. Advection example. In our next example, we have selected the hyperbolic advection equation in one dimension, as any error generated by ZFP compression will be less damped than in a system that involves explicit diffusion. Classical discretizations of this PDE, such as the Lax–Wendroff method, contain a small amount of implicit dissipation that helps to stabilize the scheme. However, the leading-order truncation error in this case is actually dispersive. Solving a discrete system with weak diffusion results in error that may grow over time like the worst-case theoretical bound of the iterative method expressed in (3.3) does, although the rate of growth is typically nowhere near the worst case for reasonably smooth initial data.

The specific PDE we solve in d dimensions, with its initial and boundary condi-



(a) $\beta = 64$. (b) $\beta = 54$.

(c) $\beta = 32$. (d) Compression Ratio

FIG. 5. *3D diffusion example: Figures* (a)–(c) *The blue line represents the compression error caused by ZFP, the green line represents the truncation-dominated total error, the red line represents the theoretical bound, and the black line represents the IEEE double-precision round-off error for to* $\beta = \{64, 54, 32\}$. (d) *The compression ratio for varying* $\beta = \{64, 54, 44, 32, 16\}$ *is displayed as a function of time.*

tions, is

$$\begin{array}{rcl} \partial_t u(x,t) + a\nabla u(x,t) &=& 0, \qquad (x,t) \in [0,1]^d \times (0,1], \\ u(x + me_d, t) &=& u(x,t), \qquad m = 0, \pm 1, \pm 2, \ldots, \\ u(x, t=0) &=& f(x) \end{array} \tag{4.5}$$

for constant a, where e_d is the Cartesian unit vector in the dth direction. For $d = 1$, the general analytic solution is $u(x,t) = f(x - at)$, representing a wave moving in the positive x-direction when $a > 0$. The Lax–Wendroff scheme is a second-order accurate finite differencing method, expressed as

$$u_i^{n+1} = u_i^n - a\frac{\Delta t}{2\Delta x}\left[u_{i+1}^n - u_{i-1}^n\right] + a^2\frac{\Delta t^2}{2\Delta x^2}\left[u_{i+1}^n - 2u_i^n + u_{i-1}^n\right] \tag{4.6}$$

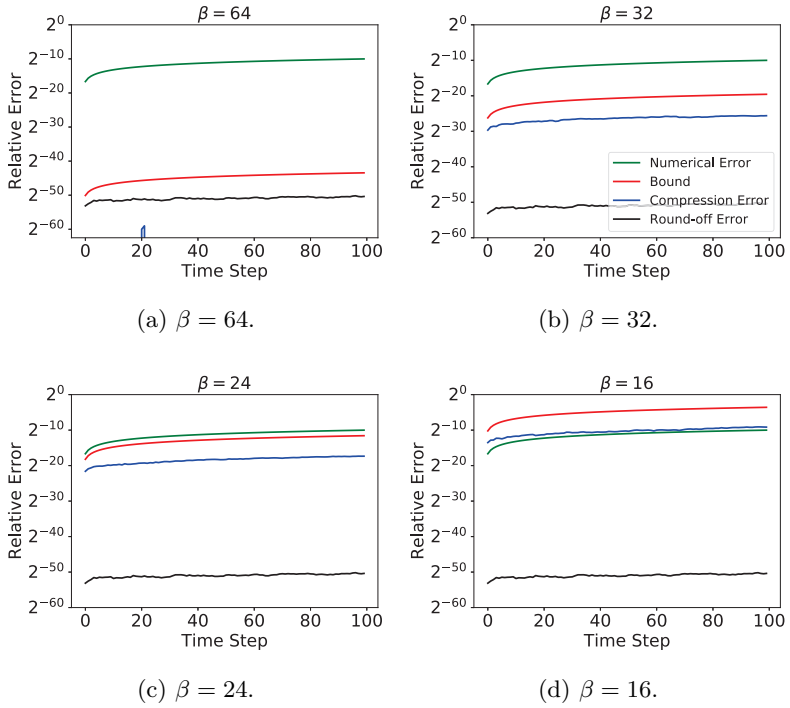
for $i = 1, \ldots, 100$ and periodicity enforced by $u_0^n := u_{100}^n$ and $u_{101}^n := u_1^n$.

For initial condition $u(x,0) = \sin(2\pi x)$, let $a = 1$, $\Delta x = 1/90$, $\Delta t = 1/100$, such that the CFL number is $\sigma = a\Delta t/\Delta x = 9/10$. The Lax–Wendroff scheme can again be written as $\boldsymbol{u}^{n+1} = A\boldsymbol{u}^n$, where $\boldsymbol{u}^n = [u_1^n, \ldots, u_{100}^n]^T$. In this case, the Lipschitz constant is $\|A\|_\infty = |1 - \sigma^2| + |\sigma| \approx 1.09$, so for any $0 < \sigma < 1$, $\|A\|_\infty > 1$, the bound of Theorem 3.3 will grow exponentially with each time step (iteration). However, Theorem 3.8 only depends on the Kreiss constant, which is more appropriate. It is known that for the Lax–Wendroff scheme the Kreiss constant is $L_k = 2$ (see p. 89 in

[22]); thus, we can investigate the validity of Theorem 3.8. In this example, the red lines represent the theoretical bound from Theorem 3.8,

$$\sum_{j=0}^{n} L_k K_{\beta_j} \|\boldsymbol{u}^j\|_\infty, \tag{4.7}$$

where $\beta_j = \{64, 32, 24, 16\}$ is constant for each experiment presented in Figure 6.



(a) $\beta = 64$. (b) $\beta = 32$.

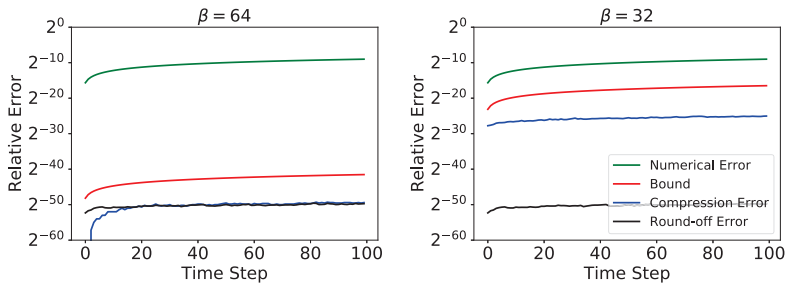
(c) $\beta = 24$. (d) $\beta = 16$.

FIG. 6. *1D Lax–Wendroff example: The blue line represents the compression error caused by ZFP, the green line represents the truncation-dominated total error, the red line represents the theoretical bound, and the black line represents the IEEE double-precision round-off error estimated by using a solution calculated in* 80*-bit precision.*

For all β_j, the error caused by the compression algorithm is less than the total numerical error dominated by discretization, and the theoretical bound is able to fully capture the error caused by the compression. For $\beta_j = 64$, ZFP produces no discernible additional error. The maximum truncation-dominated total error is approximately $\approx 2^{-8}$ at $n = 1000$; thus, from Lemma 3.9, we have $\tilde{\beta} \approx 24$. Figure 6c depicts the same plots as before where $\beta_j = 24$ is held constant. One can see that, at time step $n = 1000$, the theoretical bound is less than the total numerical error, but the actual error caused by the compression is much smaller than both the bound and the total numerical error. Unlike the diffusion example, when $\beta = 16$ the total numerical error is less than the ZFP round-off error. However, the use of ZFP compressed data types still did not destabilize the method and our theoretical bound still bounds the additional error, ensuring the method remained stable. Figure 8a represents the compression ratio for varying β values. Since the initial condition is a sine wave with periodic boundary conditions, the sine wave propagates as time varies.

The shape of the sine wave remains nearly constant, and thus the compression ratio remains constant as time varies.

As ZFP does not produce significant compression ratios in one dimension, we must consider ZFP in higher dimensions. Thus, we perform the same numerical example in two dimensions. Let $a = 1$, $\Delta x_1 = \Delta x_2 = 1/90$, and $\Delta t = 1/100$. For initial condition $u(x, 0) = \cos(2\pi(x_1 + x_2))$, we used the two-step Lax–Wendroff scheme [28]. Figure 7 displays similar results for the 2D advection example, and similar conclusions to the 1D case can be drawn. The total numerical error from the numerical experiment is $\approx 2^{-10}$ at $n = 100$. From Lemma 3.9, we need $\tilde{\beta} \geq 26$ for the ZFP round-off error to remain below the total numerical error, which is depicted in Figure 7c. Figure 8b presents the compression ratio for varying β values. As expected, the compression ratios tend to be higher in the 2D case than in the 1D case. When $\beta = 16$, round-off error caused by ZFP does not destabilize the method, which still converges to a solution. However, since the round-off error is over an order of magnitude greater than the total numerical error, the compression ratio degrades over time, as seen in Figure 8b.

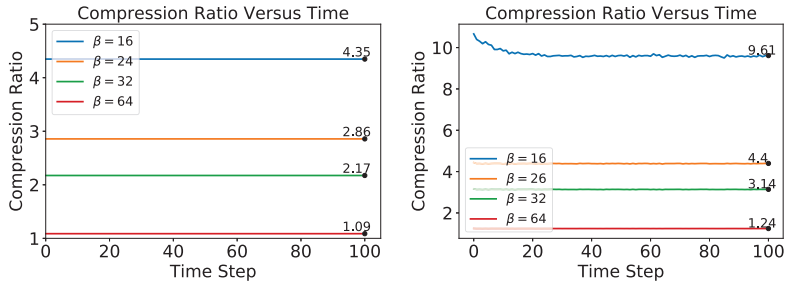


(a) $\beta = 64$. (b) $\beta = 32$.

(c) $\beta = 26$. (d) $\beta = 16$.

FIG. 7. *2D Lax–Wendroff example: The blue line represents the compression error caused by ZFP, the green line represents the truncation-dominated total error, the red line represents the theoretical bound, and the black line represents the IEEE double-precision round-off error estimated by using a solution calculated in* 80*-bit precision.*

5. Poisson example. In our last example, we have selected the Poisson equation in two dimensions, which will be solved using the Jacobi method. In this example, we wish to test Theorem 3.6, as the discrete Poisson equation is a linear system with a fixed-point solution and the Jacobi method is a stationary iterative method. Using



(a) 1D Lax–Wendroff Example (b) 2D Lax–Wendroff Example

FIG. 8. *The compression ratio for varying* $\beta = \{64, 32, 24, 16\}$ *is displayed over time.* (a) 1*D Lax–Wendroff example:* (b) 2*D Lax–Wendroff example: The minimum compression ratio is* 1.24 *when* $\beta = 64$.

notation similar to that in the previous example, let x, y be the spatial variables, u the continuous solution to the Poisson equation, and $\boldsymbol{u}$ the uncompressed solution to the discretized Poisson equation. The specific PDE we solve, with its initial and boundary conditions, is

$$\partial_x^2 u(x,y) + \partial_y^2 u(x,y) = 4, \quad (x,y) \in [0,1] \times [0,1], \tag{5.1}$$

with initial condition

$$u(x,y) = \begin{cases} 1+y^2, & x = 0, \\ 1+x^2, & y = 0, \\ 2+y^2, & x = 1, \\ 2+x^2, & y = 1, \\ 0 & \text{otherwise.} \end{cases} \tag{5.2}$$

Using central differences in the x- and y-directions, the algebraic system is expressed as

$$\frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{\Delta x_1^2} + \frac{u_{i,j-1} - 2u_{i,j} + u_{i,j+1}}{\Delta x_2^2} = 4. \tag{5.3}$$

Let $\Delta x_1 = \Delta x_2$. Using Jacobi iteration, we have

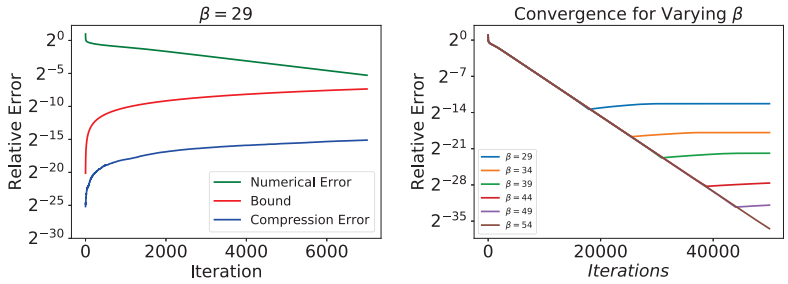
$$u_{i,j}^{n+1} = \frac{1}{4}\left(u_{i-1,j}^n + u_{i+1,j}^n + u_{i,j-1}^n + u_{i,j+1}^n - 4\Delta x_1^2\right). \tag{5.4}$$

Then the finite difference scheme for Poisson using Jacobi can be written as $\boldsymbol{u}^{n+1} = A\boldsymbol{u}^n$, with Lipschitz constant $L_l \approx .9996$ for $\Delta x_1 = 0.01$. Define $\boldsymbol{v}^{n+1} = A(DC(\boldsymbol{v}^n))$, where $\boldsymbol{v}^0 = \boldsymbol{u}^0$.

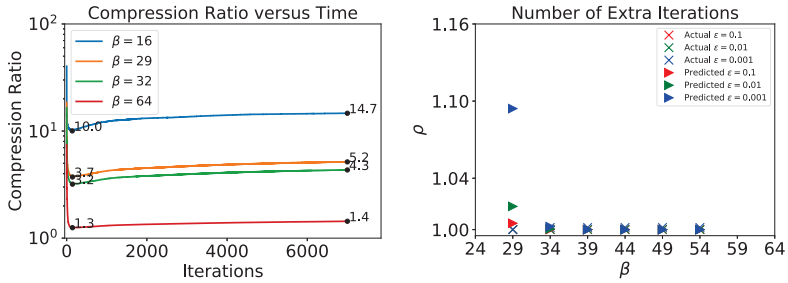
In this example, the red lines represent the theoretical bound from Theorem 3.5:

$$\sum_{j=0}^{n} L_l^{n-j+1} K_{\beta_j} \|\boldsymbol{u}^j\|_\infty, \tag{5.5}$$

where $\beta_j = 29$ is held constant. In this example, the Lipschitz constant is less than one, and thus Theorem 3.3 implies Theorem 3.8. Figure 9b displays the convergence



(a) $\beta = 29$. (b) Convergence for Varying β.



(c) Compression Ratio (d) Number of Extra Iterations.

FIG. 9. *Poisson equation example:* (a) *The blue line represents ZFP error, the green line represents the total numerical error, and the red line represents the theoretical bound.* (b) *The error between the true solution, i.e.,* $\hat{u}(x, y) = 1 + x^2 + y^2$*, is calculated in double precision and the ZFP solution for varying* β *values.* (c) *The compression ratio for varying* $\beta = \{64, 32, 29, 16\}$ *is displayed over time.* (d) *For a user-defined error tolerance,* $\epsilon = \{0.1, 0.01, 0.001\}$*, the ratio of the number of extra iterations (*ρ_{actual} *and* $\rho_{predicted}$*) provided by Theorem* 3.6 *is displayed as a function of the fixed-precision parameter. From the assumptions of Theorem* 3.6*,* β *must be greater than* 28.

of the ZFP solution to the fixed point, $\|\boldsymbol{v}^{n+1} - \hat{\boldsymbol{u}}\|_\infty$, for varying β values. For each β, the convergence stalls once $\|\boldsymbol{v}^{n+1} - \hat{\boldsymbol{u}}\|_\infty \approx \mathcal{O}\big(\frac{K_\beta}{1-L_l}\big)$. Figure 9c represents the compression ratio for varying β values. Due to the initial conditions, we again see a sharp decrease in the compression ratio at the start of the simulation; then the ratio slightly increases as $n \geq 144$ increases.

Since this is a fixed-point numerical example, we can study the validity of Theorem 3.6. For some ϵ, let n and n_{actual} be the index such that $\|\boldsymbol{u}^{n+1} - \hat{\boldsymbol{u}}\| \leq \epsilon$ and $\|\boldsymbol{v}^{n_{actual}+1} - \hat{\boldsymbol{u}}\| \leq \epsilon$, respectively. Define the constants

$$\rho_{actual} = \frac{n_{actual}}{n} \qquad \text{and} \qquad \rho_{predicted} = \frac{n+m}{n},$$

where $m = \log_{L_l} \frac{L_l^{n+1} - C}{1-C} - (n+1)$ and $C = \frac{K_{\tilde{\beta}}}{1-L_l}$, as defined in Theorem 3.6. For the bound in Theorem 3.6 to be valid, it must be the case that $\rho_{actual} \leq \rho_{predicted}$. Figure 9d displays ρ_{actual} and $\rho_{predicted}$ for varying $\epsilon = \{0.1, 0.01, 0.001\}$ and β values. For all β values, the actual number of extra iterations, represented by ρ_{actual}, is approximately zero. As β decreases, an exponential increase occurs for the predicted number of extra iterations, as represented by $\rho_{predicted}$. However, even for $\beta = 29$, $\rho_{predicted} \leq 1.16$, meaning that only 16% extra iterations are required to ensure the

error caused by ZFP is less than the theoretical worst-case bound for the double-precision solution. The required assumption from Theorem 3.6, that $K_\beta \leq L_l^{n+1}(1 - L_l)$, informs us that if $n = 7,000$, then we must use a $\beta \geq 28$ to produce a meaningful result.

6. Conclusion. In this paper, we addressed the accumulated round-off error introduced by the use of a compressed array data type in fixed-point and time-stepping methods. An important contribution of this paper is the extension of the single use round-off error bound that was first formulated in [8] to bound the accumulated error introduced by ZFP to both time-evolving and fixed-point iterative methods. Under reasonable assumptions on the advancement operators, Theorems 3.3 and 3.8 bound the error between the double-precision IEEE solution state and the ZFP solution state at some iterate n for general Lipschitz continuous operators and Kreiss bounded linear operators. Lemmas 3.4 and 3.9 extend the theorems to predict the fixed-precision parameter, β, to obtain a certain accuracy over the course of the simulation. Theorem 3.5 proves that if the fixed-precision parameter is chosen with respect to the Lipschitz constant, then fixed-point methods on ZFP compressed arrays will converge to the same fixed point as fixed-point methods without ZFP. Theorem 3.6 provides a bound on the number of extra iterations required to obtain the same accuracy as the noncompressed solution. Our results indicated that we can achieve the required precision with an appropriate fixed-precision parameter. For comparison purposes, IEEE double precision was used for all the numerical examples presented. It can be seen that when $\beta = 64$, the error caused by the repeated use of compression was less than the round-off error caused by IEEE floating-point round-off; however, the compression ratio remained above one, indicating that the ZFP solution always used fewer bits than the IEEE double-precision solution. To assist in choosing a suitable β value, we provided Lemmas 3.11 and 3.12, which state the minimal fixed-precision parameter, β, required to ensure that the round-off error caused by ZFP is less than traditional floating-point arithmetic error. To conclude, we have presented the theoretical rationale that the continued investigation and research into ZFP is advantageous to the HPC community.

We limited our results and analysis to the fixed-precision implementation of ZFP. However, the error bounds from [8] for the fixed accuracy and fixed-rate implementation can be used to extend the bounds presented in our paper. The use of new efficient number representations, lossy compression algorithms, and mixed precision algorithms are all potentially beneficial methods to reduce the memory capacity and bandwidth demands in simulation codes. The key advantage of using ZFP as a number representation over mixed precision algorithms in IEEE is that we can achieve bandwidth reduction without changing the underlying structure of the algorithm, whereas mixed-precision algorithms typically require restructuring of the code in order to obtain the same accuracy. In addition, due to the flexibility of precision in ZFP via the fixed-precision parameter, mixed-precision algorithms can be easily achieved using ZFP with access to any level of precision desired without the restriction to double, single, and half precision, as in IEEE.

REFERENCES

[1] *IEEE standard for floating-point arithmetic*, IEEE Std 754-2008, (2008), pp. 1–70, https://doi.org/10.1109/IEEESTD.2008.4610935.

[2] *BLOSC version* 1.16.3, 2019, https://github.com/Blosc/c-blosc.

[3] D. L. Brown, P. Messina, D. Keyes, J. Morrison, R. Lucas, J. Shalf, P. Beckman,

R. Brightwell, A. Geist, J. Vetter, B. L. Chamberlain, E. Lusk, J. Bell, M. S. Shephard, M. Anitescu, D. Estep, B. Hendrickson, A. Pinar, and M. A. Heroux, *Scientific Grand Challenges: Crosscutting Technologies for Computing at the Exascale*, Tech. report, U.S. Department of Energy, Washington, D.C., 2010.

[4] R. Burden and J. Faires, *Numerical Analysis*, Brooks/Cole, Cengage Learning, Boston, MA, 2011.

[5] J. Calhoun, F. Cappello, L. Olson, M. Snir, and W. Gropp, *Exploring the feasibility of lossy compression for PDE simulations*, Int. J. High Perform. Comput. Appl., (2018), https://doi.org/10.1177/1094342018762036.

[6] S. Claggett, S. Azimi, and M. Burtscher, *SPDP: An automatically synthesized lossless compression algorithm for floating-point data*, in Proceedings of the Data Compression Conference, 2018, pp. 335–344, https://doi.org/10.1109/DCC.2018.00042.

[7] S. Di and F. Cappello, *Fast error-bounded lossy HPC data compression with SZ*, in Proceedings of the IEEE International Parallel and Distributed Processing Symposium (IPDPS), 2016, pp. 730–739, https://doi.org/10.1109/IPDPS.2016.11.

[8] J. Diffenderfer, A. L. Fox, J. A. Hittinger, G. Sanders, and P. G. Lindstrom, *Error analysis of ZFP compression for floating-point data*, SIAM J. Sci. Comput., 41 (2019), pp. A1867–A1898, https://doi.org/10.1137/18M1168832.

[9] D. Feingold and R. Varga, *Block diagonally dominant matrices and generalizations of the Gershgorin theorem*, Pacific J. Math., 12 (1962), https://doi.org/10.2140/pjm.1962.12.1241.

[10] S. Hamilton, R. Burns, C. Meneveau, P. Johnson, P. Lindstrom, J. Patchett, and A. Szalay, *Extreme event analysis in next generation simulation architectures*, in Proceedings of the ISC High Performance Conference, 2017, pp. 277–293.

[11] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd ed., SIAM, Philadelphia, 2002, https://doi.org/10.1137/1.9780898718027.

[12] N. Higham and P. Knight, *Componentwise error analysis for stationary iterative methods*, in Linear Algebra, Markov Chains, and Queueing Models, C. D. Meyer and R. J. Plemmons, eds., Springer, New York, 1993, pp. 29–46.

[13] D. E. Knuth, *The Art of Computer Programming, Volume* 2: *Seminumerical Algorithms*, 3rd ed., Addison-Wesley, Reading, MA, 1997.

[14] D. Laney, S. Langer, C. Weber, P. Lindstrom, and A. Wegener, *Assessing the effects of data compression in simulations using physically motivated metrics*, in Proceedings of the ACM International Conference on High Performance Computing, Networking, Storage and Analysis (SC '13), 2013, 76, https://doi.org/10.1145/2503210.2503283.

[15] X. Liang, S. Di, D. Tao, S. Li, S. Li, H. Guo, Z. Chen, and F. Cappello, *Error-controlled lossy compression optimized for high compression ratios of scientific datasets*, in Proceedings of the IEEE International Conference on Big Data, 2018, pp. 438–447, https://doi.org/10.1109/BigData.2018.8622520.

[16] P. Lindstrom, *Fixed-rate compressed floating-point arrays*, IEEE Trans. Vis. Comput. Graph., 20 (2014), pp. 2674–2683, https://doi.org/10.1109/TVCG.2014.2346458.

[17] P. Lindstrom, *ZFP version* 0.5.5, May 2019, https://computation.llnl.gov/projects/floating-point-compression.

[18] P. Lindstrom and M. Isenburg, *Fast and efficient compression of floating-point data*, IEEE Trans. Vis. Comput. Graph., 12 (2006), pp. 1245–1250, https://doi.org/10.1109/TVCG.2006.143.

[19] A. Mitra, *On finite wordlength properties of block-floating-point arithmetic*, International Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering, 2 (2008), pp. 1709–1714.

[20] K. R. Rao and P. Yip, *Discrete Cosine Transform: Algorithms, Advantages, Applications*, Academic Press, San Diego, CA, 1990.

[21] P. Ratanaworabhan, J. Ke, and M. Burtscher, *Fast lossless compression of scientific floating-point data*, in Proceedings of the IEEE Data Compression Conference (DCC '06), 2006, pp. 133–142, https://doi.org/10.1109/DCC.2006.35.

[22] R. D. Richtmyer and K. W. Morton, *Difference Methods for Initial-Value Problems*, Krieger, Malabar, FL, 1994.

[23] J. Strikwerda and B. Wade, *A survey of the Kreiss matrix theorem for power bounded families of matrices and its extensions*, in Linear Operators, Banach Center Publ. 38, Institute of Mathematics of the Polish Academy of Sciences, Warsaw, Poland, 1997, pp. 339–360, https://doi.org/10.4064/-38-1-339-360.

[24] D. Tao, S. Di, Z. Chen, and F. Cappello, *Significantly improving lossy compression for scientific data sets based on multidimensional prediction and error-controlled quantization*,

in Proceedings of the IEEE International Parallel and Distributed Processing Symposium (IPDPS), 2017, pp. 1129–1139, https://doi.org/10.1109/IPDPS.2017.115.

[25] D. Tao, S. Di, X. Liang, Z. Chen, and F. Cappello, *Improving performance of iterative methods by lossy checkpointing*, in Proceedings of the 27th International Symposium on High-Performance Parallel and Distributed Computing (HPDC '18), 2018, pp. 52–65, https://doi.org/10.1145/3208040.3208050.

[26] J. H. Wilkinson, *Rounding Errors in Algebraic Processes*, Dover, New York, 1994.

[27] S. Williams, A. Waterman, and D. Patterson, *Roofline: An insightful visual performance model for multicore architectures*, Commun. ACM, 52 (2009), pp. 65–76, https://doi.org/10.1145/1498765.1498785.

[28] G. Zwas, *On two step Lax-Wendroff methods in several dimensions*, Numer. Math., 20 (1972), pp. 350–355, https://doi.org/10.1007/BF01402557.