1 **Reducing Disk Storage of Full-3D Seismic Waveform Tomography (F3DT) Through**
2 **Lossy Online Compression**

3

4 ¹*Peter Lindstrom*, ²*Po Chen (pchen@uwyo.edu)*, ³*En-Jui Lee*

5

6 ¹*Lawrence Livermore National Laboratory, USA*

7 ²*Department of Geology and Geophysics, University of Wyoming, USA*

8 ³*Department of Earth Sciences, National Cheng Kung University, Taiwan*

9

10

Abstract

Full-3D seismic waveform tomography (F3DT) is the latest seismic tomography technique that can assimilate broadband, multi-component seismic waveform observations into high-resolution 3D subsurface seismic structure models. The main drawback in the current F3DT implementation, in particular the scattering-integral implementation (F3DT-SI), is the high disk storage cost and the associated I/O overhead of archiving the 4D space-time wavefields of the receiver- or source-side strain tensors. The strain tensor fields are needed for computing the data sensitivity kernels, which are used for constructing the Jacobian matrix in the Gauss-Newton optimization algorithm. In this study, we have successfully integrated a lossy compression algorithm into our F3DT-SI workflow to significantly reduce the disk space for storing the strain tensor fields. The compressor supports a user-specified tolerance for bounding the error, and can be integrated into our finite-difference wave-propagation simulation code used for computing the strain fields. The decompressor can be integrated into the kernel calculation code that reads the strain fields from the disk and compute the data sensitivity kernels. During the wave-propagation simulations, we compress the strain fields before writing them to the disk. To compute the data sensitivity kernels, we read the compressed strain fields from the disk and decompress them before using them in kernel calculations. Experiments using a realistic dataset in our California statewide F3DT project have shown that we can reduce the strain-field disk storage by at least an order of magnitude with acceptable loss, and also improve the overall I/O performance of the entire F3DT-SI workflow significantly. The integration of the lossy online compressor may potentially

- 1 open up the possibilities of the wide adoption of F3DT-SI in routine seismic tomography
- 2 practices in the near future.
- 3
- 4

1. Introduction

Seismic tomography has been one of the most effective means for imaging the internal structure of the Earth in the past few decades (e.g., Anderson & Dziewonski, 1984; Nolet, 1987a; Iyer, 1993; Nolet, 2008; Nolet, 2012). The techniques used in seismic tomography have been constantly improving. Recent advances in computing technology has drastically reduced the computational cost for solving the 3D (visco)elastic seismic wave equation, which has enabled full-3D tomography (F3DT) (e.g., Chen et al., 2007ab; Fichtner et al., 2009; Tape et al., 2009; Lee et al., 2014ab; Chen & Lee, 2015). In F3DT, the starting seismic structural model can be fully three-dimensional and the Fréchet (sensitivity) kernels are computed by numerically solving the inhomogeneous equations of motion for a heterogeneous, (an)elastic solid. It accounts for the nonlinearity of waveform inversions through iterated cycles of wave-propagation simulations, misfit measurements, sensitivity kernel calculations and inversions.

There are two complementary F3DT implementations: the adjoint-wave-field method (F3DT-AW), which constructs the gradient of the objective function using the adjoint method and solves the optimization problem using gradient-based algorithms (e.g., Fichtner et al., 2009; Tape et al., 2009), and the scattering-integral method (F3DT-SI), which sets up the Jacobian of the objective function by calculating and storing the data sensitivity (Fréchet) kernel for each misfit measurement and solves the optimization problem using the Gauss-Newton algorithm (e.g., Chen et al., 2007a; Lee et al., 2014a). These two types of implementations are based on the same physics, but their

1 computational requirements can be highly different (Chen et al., 2007b). For tomography
2 problems involving a large number of seismic sources F3DT-SI may significantly reduce
3 the total amount of computing time at the expense of substantially higher disk storage
4 cost. For the F3DT inversion in Southern California (Lee et al., 2014a), the peak disk
5 storage for F3DT-SI was about 39 TB, while the peak disk storage for F3DT-AW was
6 only about 200 GB, a nearly 200 times difference. The high disk storage cost of F3DT-SI
7 is becoming the main obstacle to the wide adoption of F3DT-SI in routine seismic
8 tomography, especially on small to medium-sized shared computer clusters without large
9 amounts of high-speed disk storage.

10
11 In this paper, we describe a potential solution for significantly reducing the disk storage
12 of F3DT-SI through lossy but error-bounded online compression. Our compression
13 algorithm, named *zfp*, provides high compression ratios with minimal CPU overhead and
14 can work inside the wave-propagation simulation code and the sensitivity kernel
15 calculation code in a streaming setting during the I/O stage (Lindstrom, 2014). Whereas
16 *zfp* was originally designed for fixed-rate compression in order to support random access,
17 we use its fixed-accuracy (variable-rate) mode in order to limit compression-induced
18 errors. Although this mode sacrifices the ability to perform constant-time random access,
19 we require only sequential reads and writes of entire strain fields. Moreover, relaxing the
20 fixed-rate constraint can significantly improve the quality per bit of compressed storage.
21 Preliminary experiments using realistic simulations in our California statewide F3DT-SI
22 inversion show highly promising results. On average, the disk space for storing the 4D
23 strain tensor fields can be reduced by at least an order of magnitude with much improved

I/O performance. The kernels computed from the compressed strain fields have negligible differences from those computed using the raw strain fields. By integrating zfp , we expect to make F3DT-SI much more affordable on small clusters.

2. Disk Storage Cost of F3DT-SI

F3DT is often implemented using gradient- or Hessian-based iterative optimization algorithms. The discretized earth structural model $\mathbf{m}$ is iteratively updated through a finite series of perturbations,

$$\mathbf{m}_{k+1} = \mathbf{m}_k + \mathbf{D}\mathbf{m}_k, \quad k = 0, 1, 2, \dots, K \quad (1)$$

where k is the iteration index. The perturbation for the k^{th} iteration, $\mathbf{D}\mathbf{m}_k$, can be obtained by minimizing an objective function,

$$C^2(\mathbf{m}, \mathbf{m}_k) = \mathbf{d}^T(\mathbf{m}, \mathbf{m}_k) \mathbf{C}_d^{-1} \mathbf{d}(\mathbf{m}, \mathbf{m}_k) + (\mathbf{m} - \mathbf{m}_k)^T \mathbf{C}_m^{-1} (\mathbf{m} - \mathbf{m}_k), \quad (2)$$

where $\mathbf{m}$ is the “target” structural model, $\mathbf{d}(\mathbf{m}, \mathbf{m}_k)$ is a column-vector composed of misfit measurements that quantify the discrepancies between the i^{th} -component observed seismogram generated by the s^{th} seismic source and recorded at the r^{th} receiver, $\bar{u}_i^s(\mathbf{x}_r, t)$, and the corresponding synthetic seismogram $u_i^s(\mathbf{x}_r, t)$ computed using the latest structural model $\mathbf{m}_k$, $\mathbf{C}_d$ and $\mathbf{C}_m$ are respectively the data and model covariance matrices.

In F3DT-SI, the objective function in eq. (2) is minimized using the Gauss-Newton algorithm, which requires the solution of the Gauss-Newton normal equation

$$\begin{bmatrix} \mathbf{C}_d^{-1/2} \mathbf{A}_k \\ \mathbf{C}_m^{-1/2} \end{bmatrix} \mathbf{D} \mathbf{m}_k = \begin{bmatrix} \mathbf{C}_d^{-1/2} \mathbf{d}_k \\ \mathbf{0} \end{bmatrix}, \quad (3)$$

where $\mathbf{A}_k = \mathbf{\nabla} \mathbf{d}_k / \mathbf{\nabla} \mathbf{m}_k$ is the Jacobian matrix for the k^{th} iteration. In F3DT-SI, the Jacobian matrix is explicitly constructed and equation (3) is solved using the scalable parallel LSQR algorithm (Lee et al., 2013).

Each row of the Jacobian is a discretized data sensitivity kernel, which can be computed using the 4D strain fields from the source and those from the receiver. Equations for constructing the data sensitivity kernels using the receiver-side strain Green's tensors (RSGTs) have been given in (e.g., Zhao et al., 2005; Zhao et al., 2006; Chen et al., 2007ab; Chen & Lee, 2015). The calculation involves temporal convolution between the strain field from the source and the RSGT for the corresponding receiver, which can be computed by placing a point impulsive source at the receiver location (Zhao et al., 2006). To construct the kernels for all misfit measurements, we need to store either the RSGTs or the source-side strain fields. When seismic sources outnumber receivers, it is more economical to store the RSGTs.

In practice, the kernels are usually smoother than the strain fields and we often regularize

1 the inverted model perturbation $\mathbf{Dm}_k$ through smoothness damping. Therefore we can
2 sample the kernels on a mesh that is sparser than the mesh used for the wave-propagation
3 simulations. The accuracy of the temporal convolution is usually sufficient if we have 10
4 time samples per dominant period. Because of these considerations, the disk space for
5 storing strain fields can be reduced significantly through decimation in space and time.
6 But even after decimation, the disk storage for all strain fields used in a realistic inversion
7 can still be significant.

9 **3. Lossy Online Compression of Strain Fields**

10
11 Previous studies on seismic data compression mainly focused on the compression of
12 observed active-source seismic data in the space-time domain in an off-line setting (e.g.,
13 Wood, 1974; Jonsson & Spanias, 1990; Mandyam et al., 1996; Villasenor et al., 1996;
14 Wang & Wu, 2000; Averbuch et al., 2001). Blind application of traditional lossless
15 compression algorithms on observed active-source seismic wavefields can only provide
16 low compression ratios of around 2 (e.g., Villasenor et al., 1996), while applications of
17 lossy compression algorithms were able to achieve compression ratios ranging from ~20
18 to over 100 with acceptable losses of useful seismic information (e.g., Villasenor et al.,
19 1996; Wang & Wu, 2000; Averbuch et al., 2001). For the 4D synthetic RSGTs from the
20 California statewide inversion considered in this study, perfectly lossless compression
21 using a state-of-the-art floating-point lossless compressor *FPZIP* (Lindstrom & Isenburg,
22 2006) provided a compression ratio of merely 1.55. For F3DT purposes, a lossless
23 compression of synthetic strain fields is both unnecessary and inefficient. A more

1 desirable compression scheme is a lossy algorithm that can work in an online I/O setting
2 from within the wave-propagation simulation code with minimal CPU overhead and can
3 achieve significant compression ratios without introducing significant artifacts into the
4 data sensitivity kernels.

5
6 Lossy compression of observed active-source seismic data has been extensively studied
7 in the past two decades (e.g., Levik et al., 1996; Villasenor et al., 1996; Vassiliou &
8 Wickerhouser, 1997; Wang & Wu, 2000; Averbuch et al., 2001; Al-Moohimeed, 2004;
9 Wang et al., 2004; Aparna & David, 2006; Wu et al., 2006; Xie & Qin, 2009; Aqrawi &
10 Elster, 2011; Zheng & Liu, 2012; Fajardo et al., 2015). The majority of the compression
11 algorithms usually follow a 3-stage process: de-correlating transformation, quantization
12 and coding. In the transformation stage, suitable basis functions can lead to a much
13 sparser representation of the original data in the transformed domain. Wavelets, wavelet
14 packets, (adaptive) local trigonometric functions and various combinations of the above
15 have been widely used in previous studies (e.g., Villasenor et al., 1996; Al-Moohimeed,
16 2004; Wang & Wu, 2000; Wu et al., 2006; Zheng & Liu, 2012). The floating-point
17 transform coefficients are then mapped to a set of integers in the quantization stage. The
18 majority of previous studies adopted uniform quantization schemes (e.g., Levik et al.,
19 1996; Al-Moohimeed, 2004; Aparna & David, 2006; Wu et al., 2006; Fajardo et al.,
20 2015). In general, as the number of quantization bits decreases, the compression ratio, as
21 well as the loss of useful information, increases. In the final stage, the quantized
22 transform coefficients are encoded using a lossless coding scheme, usually an entropy
23 encoding technique such as Huffman coding (e.g., Villasenor et al., 1996; Vassiliou &

Wickerhouser, 1997; Wang et al., 2004; Aqrawi & Elster, 2011; Fajardo et al., 2015) or arithmetic coding (e.g., Levik et al., 1996; Averbuch et al., 2001; Wu et al., 2006; Xie & Qin, 2009; Fajardo et al., 2015). The primary goal in previous studies was to increase the compression ratio while minimizing the loss of useful seismic information.

When integrating a lossy compressor into the F3DT-SI workflow in an online fashion, we face new challenges. (1) Our wave-propagation simulation code has been carefully optimized and is highly efficient (Cui et al., 2009). For the California statewide simulation, evolving the entire 9-component wavefield (i.e., 3 particle-velocity components and 6 stress components, about 6.75 GB data in memory) by one time step takes around 0.163 s of wall-time on 256 cores, which amounts to about 165.64 MB/s/core of calculation speed. It is desirable if the speed of the compressor is comparable to the calculation speed. (2) The memory size per core is usually too small to allow buffering the entire time history of the simulated wavefields, which means that the online compressor has to work with the 3D spatial wavefields at only one or a few buffered time steps. This limitation prevents the online compressor from fully exploiting the data redundancy in the time dimension, which can be significant for simulated wavefields. Therefore it is harder for an online compressor to reach compression ratios comparable to those obtained by off-line compressors that can work with the full 4D wavefields. (3) Realistic F3DT-SI applications usually involve a large number of data sensitivity kernels. At the current stage of the California statewide F3DT, each F3DT-SI iteration requires ~24,000 kernel runs to construct 893,911 kernels. Each kernel run requires reading two 4D strain fields from disk repeatedly. The kernel code is heavily I/O

bound. A typical kernel run spends over 90% of time on reading the two 4D strain fields repeatedly. If the compressed 4D strain fields are sufficiently small and the decompressor is sufficiently fast, we can then read the two compressed 4D strain fields once and replace slow disk reads with decompression operations to reduce the I/O overhead.

The *zfp* open source compressor (Lindstrom, 2014) adopted in this study was originally inspired by ideas from fixed-rate texture compression algorithms widely implemented in graphics hardware (e.g., Iourcha et al., 1999; Fenney, 2003; Ström & Akenine-Möller, 2005; Nystad et al., 2012), but specialized for high-precision floating-point data. We used *zfp* version 0.4.1, which improves upon the original version (Lindstrom, 2014) in both speed and compression ratio, and also supports fixed-precision and fixed-accuracy modes for setting relative respectively absolute error tolerances.

The wave-propagation simulation code is the parallel Anelastic-Wave-Propagation (AWP) code, which solves the 3D (visco)elastic wave-equation in the velocity-stress form using the 4th-order staggered-grid finite-difference method (Olsen, 1994; Cui et al., 2009). At each saved time step we obtain the strain fields from the stress fields using the constitutive relation (Zhao et al., 2005; 2006).

Each component of the strain tensor is represented as a 3D array of floating-point numbers inside the memory. Following most texture compression formats, as well as image and video compression formats like JPEG and MPEG, we divide the 3D array into blocks of dimensions 4×4×4. (Blocks near array boundaries are padded if the array

1 dimensions are not multiples of four.) The following 6 steps are then conceptually
2 applied to each block independently:

- 3 1. Block-floating-point transform,
- 4 2. De-correlating transform,
- 5 3. Coefficient reordering,
- 6 4. Conversion from signed to unsigned integer,
- 7 5. Embedded coding of integers,
- 8 6. Bit stream truncation.

9 The truncated bit streams for all blocks of the 6 independent strain components are then
10 aggregated and written to disk at the end of each saved time step. Steps 1, 2 and 6 are
11 potentially lossy, while the others are guaranteed lossless. Each step is described in more
12 detail below.

14 **3.1. Block-floating-point transform**

16 In this step, all the sixty-four 32-bit floating-point values in a block are normalized by
17 expressing them with respect to the largest floating-point exponent $\hat{e}$ in the block. For
18 efficiency of subsequent operations, each floating-point value $x = (-1)^s 2^e (1 + m)$, with
19 mantissa $0 \leq m < 1$, is rewritten as a 31-bit signed integer y times some power of two,
20 i.e., $y = \text{round}(2^{29-\hat{e}} x)$, such that $-2^{30} < y < 2^{30}$. A single-precision floating-point number
21 has 24 bits of mantissa (including the implicit leading one) plus one sign bit. By directly
22 transferring the 24-bit mantissa plus the sign bit to the 31-bit signed integer, we actually
23 have gained as many as 6 bits of precision in this step. However, if the ratio of the largest

value to the smallest value within a block exceeds $2^6 = 64$, then some loss of least significant bits is possible in the smaller-magnitude number. The same type of loss also happens in floating-point hardware during floating-point calculations, such as those in the finite-difference simulations. This type of loss is inherent to the floating-point format and its machine representation and can usually be ignored in practical seismic modeling. Before moving on to the next step, *zfp* transmits the 8-bit common exponent $\hat{e}$ verbatim.

3.2. De-correlating transform

The signed integers obtained from step 1 are transformed with respect to a set of basis functions to reduce their correlation. For suitably chosen basis functions, this transform often results in many near-zero coefficients, which are particularly amenable to compression. For regularly gridded 3D data, such as a component of the strain field computed in *AWP*, we adopt a separable 3D transform that can be implemented as three 1D transforms along each spatial axis. The 3D separable basis function can then be expressed as

$$B_{ijk}(x, y, z) = b_i(x)b_j(y)b_k(z), \quad (4)$$

where $0 \leq i, j, k \leq 3$ and $\|B_{ijk}\| = 1$. The indices i, j, k correspond to the polynomial degrees of the 1D basis functions $b_i(x)$, $b_j(y)$ and $b_k(z)$. In this study we have adopted the orthogonal Gram polynomial basis functions and an efficient lifted implementation of the 1D transform, similar to (Lindstrom, 2014). An important advantage of using those

basis functions is that the matrix-vector multiplications involved in the 1D transforms can be implemented very efficiently through in-place bit-shift and addition operations. The complete 3D transform based on this *lifting scheme* involves 4.5 bit shifts and 7.5 additions (amortized) per compressed value, which compares very favorably with the 64 multiplications and 63 additions per compressed value for implementations based on conventional matrix-vector multiplications. This integer transform preserves the range of $(-2^{30}, 2^{30})$, but involves a number of divisions by two (implemented as bit shifts) that can potentially cause loss of the least significant bit. But since we have gained as many as 6 bits from the previous step, this potential loss is usually not a problem.

3.3. Coefficient reordering

The indices i, j, k of the basis functions in equation (7) can be considered as proxies for the wavenumber (i.e., spatial frequency) of the strain field. We use the summation of the three indices $0 \leq i + j + k \leq 9$ to represent the total degree of the 3D basis function. The corresponding coefficients resulting from the de-correlating transform in step 2 are then reordered according to the total degree of the 3D basis function, much akin to how 2D JPEG DCT coefficients are arranged in $i + j$ zig-zag order. The motivation for this reordering is based on the observation that the power of the strain field tends to decrease with increasing wavenumber. Thus by reordering the coefficients according to the total degree, we obtain a list of coefficients sorted roughly by their magnitude.

3.4. Conversion to unsigned integer

1

2 Before encoding the integer coefficients, we first transform them from two's complement
3 signed integers to an unsigned representation. We have chosen the negabinary
4 representation with base -2 , for three reasons: (1) This mapping can be computed very
5 efficiently using only one integer addition and one bitwise exclusive or. (2) Compared to
6 other representations, it results in improved rate distortion when the bit stream is
7 truncated, i.e., when zeroing some of the least significant bits during quantization. (3)
8 The lack of a dedicated sign bit reduces conditionals and simplifies the logic of the
9 encoder. Bit k in negabinary has place value $(-2)^k$, and the sign thus alternates between
10 consecutive bits. This lossless transformation shifts the range of values from $(-2^{30}, 2^{30})$
11 to $[0, 7/8 \cdot 2^{32})$.

12

13 **3.5. Embedded coding**

14

15 The reordered, unsigned coefficients of a block can be transformed into a sequence of
16 bits (i.e., a bit stream) that can be easily transmitted or stored. Several embedded coding
17 strategies exist, and the one adopted in this study is a more efficient variation on the one
18 documented in (Lindstrom, 2014). Because the de-correlating transform is orthogonal, all
19 the coefficient bits in the same “bit plane” (i.e., a set of coefficient bits at the same bit
20 position) have the same impact on L_2 error, and are thus equally “important.” We
21 therefore encode one bit plane at a time, from most to least significant bit, using a *group*
22 *testing* procedure (Hong & Ladner, 2002). Anticipating that the coefficients are arranged
23 roughly in decreasing order of magnitude, and with most of them having many leading

zeros, we achieve (lossless) compression by identifying large groups of zero-bits. We begin by testing if all bits within a bit plane are zero (a negative group test), and if so emit a zero-bit and move on to the next bit plane. Otherwise (a positive group test), we output a one-bit followed by a string of coefficient value bits, in order of increasing polynomial degree, until we encounter and emit a one-bit. We then perform another group test on the remaining bits and repeat until either all remaining bits in this bit plane are zero or none remain. In this manner, the first $0 \leq n \leq 64$ value bits of the current bit plane are explicitly output; the remaining $64 - n$ zero-bits are encoded with a single bit. To encode the next bit plane, we first emit its first n bits, which are presumed random, before we proceed with group testing the remaining bits, incrementing n as necessary from one bit plane to the next. This coding strategy losslessly encodes all 64×32 integer coefficient bits, and usually using far fewer bits.

3.6. Bit stream truncation

By initializing all coefficients to zero on the decoder side, we note that any prefix of the resulting bit stream allows the decoder to produce a valid reconstruction of some number of leading bits of each coefficient, which after the inverse transformation results in an approximate representation of the original floating-point values. Such truncation of the bit stream affords great flexibility and granularity in choosing the compression ratio or quality. In *zfp*'s *fixed-rate* mode, the bit stream for each block is truncated at a fixed number of bits to ensure fixed-size storage for each block and random access at block granularity. (In practice, bit stream truncation is implemented by pre-empting the encoder

once the bit budget has been reached.) Alternative truncation strategies include *fixed precision*—by encoding a fixed number of bit planes—and *fixed accuracy*—by encoding all bit planes up to some given bit plane number q , which is equivalent to applying uniform quantization with step size 2^q . Note that in fixed-accuracy mode, the number of encoded bit planes depends both on q and on the block exponent $\hat{e}$. For a given q , a worst-case analysis reveals that the maximum reconstruction error in 3D is at most 2^{q+6} . We may, thus, choose q so as to guarantee that the decompressed data falls within a user-specified error tolerance. In this study, we used this fixed-accuracy mode of *zfp*.

4. Results

We have successfully applied our lossy online compression algorithm to significantly reduce the disk storage in the current California statewide F3DT-SI inversions. At the current stage we have carried out 4 iterations following the tomographic navigation approach as discussed in Lee et al. (2014a).

Figure 1 shows the S-wave velocity at 10 km depth in our starting model CAF3D00 and the model CAF3D04 after the 4th iteration. About 24,000 seismic waveforms have been used in our inversion. The misfit function used in the 4 iterations was defined as the L2-norm of the frequency-dependent group-delay anomalies measured at frequencies ranging from 0.03 Hz to 0.1 Hz. Figure 2 shows about 40% misfit reduction of about 893,000 measurements after the 4 iterations.

Synthetic waveforms computed using CAF3D04 show substantially better fit to observed waveforms than those computed using CAF3D00. We measured the differences between an observed waveform $\bar{u}(t)$ and its corresponding synthetic waveform $u(t)$ within a time window $[t_0, t_1]$ using the relative waveform misfit (RWM) defined by (e.g., Zhu & Helmberger, 1996; Tape et al., 2010; Lee et al 2014b)

$$RWM = \frac{\int_{t_0}^{t_1} [\bar{u}(t) - u(t)]^2 dt}{\sqrt{\int_{t_0}^{t_1} [\bar{u}(t)]^2 dt \int_{t_0}^{t_1} [u(t)]^2 dt}}. \quad (5)$$

Figure 3 shows the histograms of the RWMs for CAF3D00 and CAF3D04.

Through the 4 iterations, the median of the RWM (mRWM) has dropped from 1.4 to 0.73 (Figure 4) and the median absolute deviation (MAD) (Hoaglin et al., 1983) has dropped from 0.65 to 0.35. The mRWM and the MAD provide robust measures of the location and the spread of the RWM distributions (Lee et al., 2014b).

The inversion is being carried out on the IBM System X cluster, “Mount Moran” (284 nodes, each with two 8-core Intel E5-2670), at University of Wyoming. For this study, each *AWP* wave-propagation simulation runs on 256 CPU cores for ~35 minutes. Each F3DT-SI iteration requires 227 *AWP* simulations (~33,900 core-hours). Each kernel calculation takes ~1.2 minutes on 256 cores. We have ~24,000 kernel calculations per

iteration (~122,880 core-hours). The total amount of core-hours for one F3DT-SI iteration is roughly 160,000.

4.1. Disk storage without compression

The *AWP* simulations are carried out on a uniform mesh with ~201.3 million grid points with 1-km grid spacing. The simulation volume is 1000-km wide, 2000-km long and 100-km deep (Figure 1). Each *AWP* simulation computes the strain and particle-velocity fields for 12,282 time steps (~700 seconds). The synthetic wavefields computed using this spatial-temporal discretization set-up is accurate for frequencies up to 0.12 Hz. We are not considering visco-elastic attenuation. At the current stage, we are using the vertical components of 227 stations from the California Integrated Seismic Network (CISN) and the Transportable Array (TA) of the EarthScope project.

Along the time axis, we decimate by a factor of 10, which gives ~18 samples per dominant period of the simulated fields. Table 1 lists the disk storage with and without compression for all RSGTs with 6 different spatial decimation settings: (a) $D_x = D_y = 8$, $D_z = 2$; (b) $D_x = D_z = 8$, $D_y = 2$; (c) $D_x = D_y = 4$, $D_z = 2$; (d) $D_x = D_z = 4$, $D_y = 2$; (e) $D_x = D_y = D_z = 2$; (f) no decimation. Here $D_{x,y,z}$ are decimation rates in x, y and z, respectively. At each saved grid point and time step, we store the 6 independent components of the strain tensor and the 3 components of the particle velocity in single-precision. At the current stage we are using setting (c).

4.2. Error tolerance

1

2 The average compression ratio mainly depends upon two factors: (1) the criteria used for
 3 truncating the bit stream (section 3.6) and (2) the spatial smoothness (i.e., redundancy) of
 4 the wavefields. When truncating the bit stream, we used an error tolerance of $\epsilon = 2^t$. All
 5 coefficient bits lying on a bit plane below 2^{t-6} are discarded. The effective precision in
 6 terms of the number of bits used for representing each integer coefficient is
 7 $p = (32 - 29) + \hat{e} - (t - 6) = \hat{e} - t + 9$ restricted to $0 \leq p \leq 32$, where $\hat{e}$ is the exponent
 8 associated with the block.

9

10 In practice, we determine the error tolerance ϵ through trial-and-error based on the
 11 relative kernel error (*RKE*) defined as

12

$$13 \quad RKE = \frac{\int_V dV(\mathbf{x}) [\bar{K}(\mathbf{x}) - K(\mathbf{x})]^2}{\sqrt{\int_V dV(\mathbf{x}) \bar{K}^2(\mathbf{x}) \int_V dV(\mathbf{x}) K^2(\mathbf{x})}}. \quad (6)$$

14

15 Here $K(\mathbf{x})$ and $\bar{K}(\mathbf{x})$ are the data sensitivity kernels computed using the strain fields
 16 with and without compression, respectively. Figure 5 shows the average *RKE* (averaged
 17 over 1000 kernels with respect to the two Lamé parameters) as a function of ϵ . We
 18 computed the average *RKE* using 4 different error tolerances: $\epsilon = 2^{-40} \gg 9.09 \cdot 10^{-13}$,
 19 $\epsilon = 2^{-44} \gg 5.68 \cdot 10^{-14}$, $\epsilon = 2^{-47} \gg 7.11 \cdot 10^{-15}$, $\epsilon = 2^{-50} \gg 8.88 \cdot 10^{-16}$. For the strain fields
 20 used in our inversion, when $\epsilon = 2^{-44} \gg 5.68 \cdot 10^{-14}$, the average *RKE* is $\sim 10^{-6}$ (Fig 5).
 21 Examples of the frequency-dependent group-delay kernels with respect to the Lamé

parameter μ computed using the strain fields without compression and those computed using the strain fields compressed with the 4 different ϵ values are shown in Figure 6. The compression ratios in Table 1 were computed using $\epsilon = 2^{-44}$. Compression-introduced kernel errors have negligible effects on the inverted structure model for two reasons: (1) the relative kernel errors (Fig. 5) are much smaller than the estimated noise level in the misfit measurements (a few percent for the dataset used in our current inversion); (2) compression-introduced errors are random and tend to cancel each other's effects during the LSQR inversion step (Zhao et al., 2005; 2006).

Even though *zfp* is a block-based compression algorithm, it is more resilient to block artifacts than traditional block-based compression algorithms, such as JPEG. For the error tolerances we have tried, no block artifacts were observed in the compressed strain fields or the kernels. Figure 7 shows a snapshot of the *zz*-component of the strain tensor at 2-km depth with no compression and those compressed using the 4 different error tolerances. This lack of block artifacts is due to (1) the smaller block size ($4 \times 4 \times 4$ in *zfp* compared with 8×8 in JPEG), which reduces the mismatch across block boundaries; (2) the fixed-accuracy mode in *zfp*, which bounds the error across blocks to a uniform value; and (3) *zfp*'s uniform quantization of transform coefficients by bit plane (Lindstrom 2014).

4.3. Compression ratios

Compression ratios for the 6 grid decimation settings are summarized in Table 1 and Figure 8. At lower spatial decimation rates the correlation among the strain values on

1 neighboring saved grid points is higher and the de-correlating transform allows us to
2 compress more redundant information efficiently, which increases the average
3 compression ratio. As shown in Table 1 and Figure 8, the average compression ratio
4 increases from about 15 for the highest decimation rate to about 38 (i.e., less than one bit
5 per value) for the lowest decimation rate.

6
7 The total number of saved grid points for decimation setting (a) is identical to that in
8 setting (b). In setting (a), the decimation rate is 2 on the z -axis and 8 on the y -axis, while
9 in setting (b) the decimation rate is 8 on the z -axis and 2 on the y -axis. The average
10 compression ratios for these two settings are similar, indicating that on average the
11 redundancy (smoothness) of the strain fields in the horizontal dimensions is similar to
12 that in the vertical dimension. For decimation settings (c) and (d), we observe similar
13 results.

14 **4.4. Compression performance in wave-propagation simulations**

15
16
17 The *AWP* simulation code was parallelized using the message-passing interface (MPI)
18 and a domain decomposition approach. The uniform mesh used in the finite-difference
19 simulations was divided evenly into a number of sub-meshes and each CPU core handles
20 the computation, compression and output of the wavefields on one sub-mesh.

During the simulation we measured the time used for compressing the wavefields and for writing the compressed fields to disk on each core. The wall time for each run was obtained by taking the maximum value of the estimates generated by all the CPU cores.

Figure 9 shows the performance improvements for the *AWP* simulations using our online compressor with 4 different error tolerances. The amount of time used for compression combined with writing the compressed wavefields to disk is less than the amount of time used for writing the uncompressed wavefields to disk. The compression time increases with decreasing error tolerance. For smaller error tolerance, more bit planes need to be encoded, therefore more time is spent on compression. For $\epsilon = 2^{-44} \gg 5.68 \times 10^{-14}$, the compression time was ~8.87 s and the time spent on writing the compressed wavefields to disk was ~2.36 s. The compression-output combined time, 11.23 s, is about one fourth of the time spent on writing the raw wavefields to disk (Fig 9). The average speed of the online compressor was ~116.80 MB/s/core. For the higher error tolerance $\epsilon = 2^{-40} \gg 9.09 \times 10^{-13}$, the number of bit planes need to be encoded is fewer and the speed of the compressor was ~137.35 MB/s/core. For the lower error tolerance $\epsilon = 2^{-50} \gg 8.88 \times 10^{-16}$, the compression speed was ~87.86 MB/s/core.

4.5. Decompression performance in kernel calculations

The kernel calculation involves temporal convolutions of two strain fields at every saved grid point. The kernel code, named *ker*, is parallelized using a domain decomposition approach, in which each MPI process computes the kernel on the saved grid points of one

1 sub-mesh. The number of sub-meshes (i.e., MPI processes) is determined by the *AWP*
2 simulation. If each process has access to sufficient amount of memory, only two disk-
3 read operations per process are needed to load the two 4D strain fields on one sub-mesh.

4

5 The memory-per-core on Mount Moran (~2GB), and also many other small- to medium-
6 sized clusters, is often not sufficient to hold the entire time history of two 4D strain fields.

7 In our inversion, the 4D wavefields on all saved grid points (decimation setting *c*) and all
8 saved time steps of one sub-mesh occupies about 1.1 GB without compression. To hold

9 two 4D strain fields on one sub-mesh, we would need 2.2 GB/core, not including

10 memory for other data.

11

12 The approach adopted in *ker* is to divide each sub-mesh into smaller portions so that the
13 entire time history of the strain fields on each portion can fit inside the memory of each

14 process. For each disk-read, we load the entire time history of the strain fields on one

15 portion of a sub-mesh, compute the kernels on that portion and then repeat on every

16 portion. The number of disk-reads depends upon the memory size per core. The code *ker*

17 worked well on supercomputers with powerful I/O sub-systems but its performance

18 deteriorates significantly on smaller clusters.

19

20 The compressed strain fields have much smaller sizes than the raw strain fields. For the

21 California statewide F3DT, even at the smallest error tolerance ($e = 2^{-50} \gg 8.88 \times 10^{-16}$),

22 the maximum size of the 4D strain field on a sub-mesh is about 364 MB. It is now

23 feasible to load the entire time history of two compressed 4D strain fields of one sub-

1 mesh into the memory of one core, which takes two disk-reads per process. We still
2 divide each sub-mesh into a number of smaller portions, whose sizes depend upon the
3 remaining memory. We decompress the 4D strain fields on one portion of the sub-mesh,
4 compute the kernels on that portion and repeat on every portion. We therefore replace
5 multiple disk-reads for loading the strain fields of all the portions with the same number
6 of decompression operations. The speed of the decompressor is therefore critical for the
7 overall performance of our new kernel code *kerz*.

8
9 Figure 10 shows the performance of *kerz* compared with the performance of *ker* for the 4
10 compression error tolerances. For $\epsilon = 2^{-44} \gg 5.68 \times 10^{-14}$ used in our inversion, the speed
11 of *kerz* is about 5.3 times that of *ker*. The amount of time spent on computing the kernels
12 is roughly the same in *kerz* at the 4 different error tolerance values and also in *ker*. The
13 amount of time spent on reading the compressed 4D wavefields from disk increases
14 slightly with decreasing error tolerance due to the fact that the size of the compressed
15 wavefields increases with decreasing error tolerance. The amount of time used for
16 decompressing the two 4D wavefields also increases with decreasing error tolerance. For
17 $\epsilon = 2^{-44} \gg 5.68 \times 10^{-14}$ the decompression speed is ~ 353.52 MB/s/core.

19 5. Summary and Discussion

20
21 In this study, we have shown that by integrating lossy compression into the F3DT-SI
22 workflow, we can substantially reduce disk storage and improve I/O performance, which
23 makes it feasible to adopt F3DT-SI to solve practical F3DT problems of realistic size on

1 small clusters. The main idea of our implementation was to utilize the compute cycles
2 previously wasted in waiting for the I/O of strain fields to compress/decompress the
3 strain fields. Because the online compressor can be implemented very efficiently, we not
4 only reduced disk storage substantially but also improved the overall I/O performance of
5 the entire F3DT-SI workflow significantly.

6

7 The effectiveness of our compressor can be further enhanced. At the current stage, only
8 the redundancies in the 3 spatial dimensions have been exploited by the compressor. It is
9 also possible to reduce the redundancy in the temporal dimension by using a 4D
10 separable basis function in the de-correlating transform.

11

12 Even though the compressor can effectively reduce the redundancy within the wavefields,
13 it does not account for smoothing during kernel calculations and through smoothness
14 damping when solving the linear system in equation (3). The redundancy introduced
15 through those smoothing processes can be reduced more effectively through a judicious
16 choice of the spatial decimation rates. In practice, combining spatial decimation with
17 online compression can provide higher overall storage reduction without compromising
18 the accuracy of the inverted structure model.

19

20 **Acknowledgment**

21

22 The work performed at University of Wyoming was supported by the United States
23 Geological Survey under grant number G10AP00032 and Crust LLC. This work was

performed in part under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research.

Reference:

Al-Moohimeed, M. A. (2004, August). Towards an efficient compression algorithm for seismic data. In *Radio Science Conference, 2004. Proceedings. 2004 Asia-Pacific* (pp. 550-553). IEEE.

Anderson, D. L., & Dziewonski, A. M. (1984). Seismic tomography. *Scientific American*, 251, 60-68.

Aparna, P., & David, D. S. (2006, December). Adaptive local cosine transform for seismic image compression. In *Advanced Computing and Communications, 2006. ADCOM 2006. International Conference on* (pp. 254-257). IEEE.

Aqrawi, A., & Elster, A. C. (2011, May). Bandwidth reduction through multithreaded compression of seismic images. In *Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on* (pp. 1730-1739). IEEE.

1 Averbuch, A. Z., Meyer, F., Strömberg, J. O., Coifman, R., & Vassiliou, A. (2001). Low
2 bit-rate efficient compression for seismic data. *Image Processing, IEEE Transactions on*,
3 *10*(12), 1801-1814.
4

5 Bao, H., Bielak, J., Ghattas, O., Kallivokas, L., O'Hallaron, D., Shewchuk, J. & Xu, J.
6 (1998). Large-scale simulation of elastic wave propagation in heterogeneous media on
7 parallel computers. *Computational Methods in Applied Mechanics and Engineering* 152,
8 85-102.
9

10 Brocher, T. M., 2005a. Empirical Relations Between Elastic Wave-speeds And Density
11 In The Earth's Crust, *Bulletin Of The Seismological Society Of America*, 95(6), 2081–
12 2092.

13 Brocher, T. M., 2005b. A Regional View Of Urban Sedimentary Basins In Northern
14 California Based On Oil Industry Compressional-Wave Velocity And Density Logs,
15 *Bulletin Of The Seismological Society Of America*, 95(6), 2093–2114.

16 Brocher, T. M., 2008. Compressional And Shear-Wave Velocity Versus Depth Relations
17 For Common Rock Types In Northern California, *Bulletin Of The Seismological Society*
18 *Of America*, 98(2), 950–968.

19 Chen, P., Jordan, T. H., & Zhao, L. (2005). Finite-moment tensor of the 3 September
20 2002 Yorba Linda earthquake. *Bulletin of the Seismological Society of America*, 95(3),
21 1170-1180.
22

1 Chen, P., Zhao, L., & Jordan, T. H. (2007a). Full 3D tomography for the crustal structure
2 of the Los Angeles region. *Bulletin of the Seismological Society of America*, 97(4), 1094-
3 1120.

4

5 Chen, P., Jordan, T. H., & Zhao, L. (2007b). Full three-dimensional tomography: a
6 comparison between the scattering-integral and adjoint-wavefield methods. *Geophysical*
7 *Journal International*, 170(1), 175-181.

8

9 Chen, P., Jordan, T. H., & Zhao, L. (2010). Resolving fault plane ambiguity for small
10 earthquakes. *Geophysical Journal International*, 181(1), 493-501.

11

12 Chen, P., & Lee, E. J. (2015). *Full-3D Seismic Waveform Inversion: Theory, Software*
13 *and Practice*. Springer, 511pp.

14

15 Cui, Y., Olsen, K., Chourasia, A., Moore, R., Maechling, P., & Jordan, T. (2009). The
16 TeraShake Computational Platform for Large-Scale Earthquake Simulations. In *Advances*
17 *in Geocomputing* (pp. 229-277). Springer Berlin Heidelberg.

18

19 Fajardo, C., M Reyes, O., & Ramirez, A. (2015). Seismic Data Compression Using 2D
20 Lifting-Wavelet Algorithms. *Ingeniería y Ciencia*, 11(21), 221-238.

21

1 Fenney, S. (2003, July). Texture compression using low-frequency signal modulation.
2 In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics*
3 *hardware* (pp. 84-91). Eurographics Association.
4

5 Fichtner, A., Kennett, B. L., Igel, H., & Bunge, H. P. (2009). Full seismic waveform
6 tomography for upper-mantle structure in the Australasian region using adjoint
7 methods. *Geophysical Journal International*, 179(3), 1703-1725.
8

9 Gauthier, O., Virieux, J., & Tarantola, A. (1986). Two-dimensional nonlinear inversion
10 of seismic waveforms: Numerical results. *Geophysics*, 51(7), 1387-1403.
11

12 Graves, R. (1996). Simulating seismic wave propagation in 3D elastic media using
13 staggered-grid finite differences. *Bulletin of the Seismological Society of America* 86(4),
14 1091-1106.
15

16 Hoaglin, D. C., F. Mosteller, and J. W. Tukey (1983). *Understanding Robust and*
17 *Exploratory Data Analysis*, John Wiley & Sons, 404– 414, ISBN: 0-471-09777-2.
18

19 Hong, E. S., & Ladner, R. E. (2002). Group testing for image compression. *Image*
20 *Processing, IEEE Transactions on*, 11(8), 901-911.
21

22 Iourcha, K. I., Nayak, K. S., & Hong, Z. (1999). *U.S. Patent No. 5,956,431*. Washington,
23 DC: U.S. Patent and Trademark Office.

1

2 Iyer, H. M. (1993). *Seismic tomography: Theory and practice*. Springer Science &
3 Business Media.

4

5 Jonsson, S. B., & Spanias, A. S. (1990, March). Seismic data compression. In *Computers*
6 *and Communications, 1990. Conference Proceedings., Ninth Annual International*
7 *Phoenix Conference on* (pp. 276-279). IEEE.

8

9 Käser, M. & Dumbser, M. (2006). An arbitrary high-order discontinuous Galerkin
10 method for elastic waves on unstructured meshes-I. the two- dimensional isotropic case
11 with external source terms. *Geophysical Journal International* 166(2), 855-877.

12

13 Komatitsch, D. & Tromp, J. (1999). Introduction to the spectral element method for
14 three-dimensional seismic wave propagation. *Geophysical Journal International* 139, 806-
15 822.

16

17 Lee, E. J., Chen, P., Jordan, T. H., & Wang, L. (2011). Rapid full-wave centroid moment
18 tensor (CMT) inversion in a three-dimensional earth structure model for earthquakes in
19 Southern California. *Geophysical Journal International*, 186(1), 311-330.

20

21 Lee, E. J., Chen, P., Jordan, T. H., Maechling, P. B., Denolle, M. A., & Beroza, G. C.
22 (2014a). Full- 3- D tomography for crustal structure in Southern California based on the

1 scattering- integral and the adjoint- wavefield methods. *Journal of Geophysical*
2 *Research: Solid Earth*, 119(8), 6421-6451.

3
4 Lee, E. J., Chen, P. and Jordan, T. H., (2014b) Testing waveform predictions of 3D
5 velocity models against two recent Los Angeles earthquakes. *Seismological Research*
6 *Letters*, 85(6), doi: 10.1785/0220140093.

7
8 Lee, E. J., Huang, H., Dennis, J. M., Chen, P., & Wang, L. (2013). An optimized parallel
9 LSQR algorithm for seismic tomography. *Computers & Geosciences*, 61, 184-197.

10
11 Lervik, J. M., Rosten, T., & Ramstad, T. (1996, September). Subband seismic data
12 compression: optimization and evaluation. In *Digital Signal Processing Workshop*
13 *Proceedings, 1996., IEEE* (pp. 65-68). IEEE.

14
15 Lin, G., Thurber, C. H., Zhang, H., Hauksson, E., Shearer, P. M., Waldhauser, F.,
16 Brocher, T. M., & Hardebeck, J., 2010. A California statewide three-dimensional seismic
17 velocity model from both absolute and differential times, *Bulletin of the Seismological*
18 *Society of America*, 100(1), 225–240.

19
20 Lindstrom, P., & Isenburg, M. (2006). Fast and Efficient Compression of Floating-Point
21 Data. *Visualization and Computer Graphics, IEEE Transactions on*, 12(5), 1245-1250.

1 Lindstrom, P. (2014). Fixed-Rate Compressed Floating-Point Arrays. *Visualization and*
2 *Computer Graphics, IEEE Transactions on*, 20(12), 2674-2683.

3

4 Liu, Q., & Tromp, J. (2006). Finite-frequency kernels based on adjoint methods. *Bulletin*
5 *of the Seismological Society of America*, 96(6), 2383-2397.

6

7 Madariaga, R. (1976). Dynamics of an expanding circular fault. *Bulletin of the*
8 *Seismological Society of America*, 65, 163-182.

9

10 Mandyam, G., Magotra, N., & McCoy, W. (1996, May). Lossless seismic data
11 compression using adaptive linear prediction. In *Geoscience and Remote Sensing*
12 *Symposium, 1996. IGARSS'96. 'Remote Sensing for a Sustainable Future.'*,
13 *International* (Vol. 2, pp. 1029-1031). IEEE.

14

15 Nolet, G. (1987). Seismic wave propagation and seismic tomography. In *Seismic*
16 *tomography* (pp. 1-23). Springer Netherlands.

17

18 Nolet, G. (1987). Waveform tomography. In *Seismic Tomography* (pp. 301-322).
19 Springer Netherlands.

20

21 Nolet, G. (2008). A Breviary of Seismic Tomography. *A Breviary of Seismic*
22 *Tomography, by Guust Nolet, Cambridge, UK: Cambridge University Press, 2008, 1.*

23

1 Nolet, G. (Ed.). (2012). *Seismic tomography: with applications in global seismology and*
2 *exploration geophysics* (Vol. 5). Springer Science & Business Media.

3

4 Nystad, J., Lassen, A., Pomianowski, A., Ellis, S., & Olson, T. (2012, June). Adaptive
5 scalable texture compression. In *Proceedings of the Fourth ACM*
6 *SIGGRAPH/Eurographics conference on High-Performance Graphics* (pp. 105-114).
7 Eurographics Association.

8

9 Olsen, K. B. (1994). Simulation of three-dimensional wave propagation in the salt lake
10 basin (Doctoral dissertation, University of Utah, Salt Lake City, UT).

11

12 Pratt, R. G. (1999). Seismic waveform inversion in the frequency domain, Part 1: Theory
13 and verification in a physical scale model. *Geophysics*, 64(3), 888-901.

14

15 Ström, J., & Akenine-Möller, T. (2005, July). i PACKMAN: High-quality, low-
16 complexity texture compression for mobile phones. In *Proceedings of the ACM*
17 *SIGGRAPH/EUROGRAPHICS conference on Graphics hardware* (pp. 63-70). ACM.

18

19 Tape, C., Liu, Q., Maggi, A., & Tromp, J. (2010). Seismic tomography of the southern
20 California crust based on spectral-element and adjoint methods. *Geophysical Journal*
21 *International*, 180(1), 433-462.

22

1 Tarantola, A. (1988). Theoretical background for the inversion of seismic waveforms
2 including elasticity and attenuation. *Pure and Applied Geophysics*, 128(1-2), 365-399.
3

4 Villasenor, J. D., Ergas, R. A., & Donoho, P. L. (1996, March). Seismic data
5 compression using high-dimensional wavelet transforms. In *Data Compression*
6 *Conference, 1996. DCC'96. Proceedings* (pp. 396-405). IEEE.
7

8 Vassiliou, A. A., & Wickerhouser, M. V. (1997, October). Comparison of wavelet image
9 coding schemes for seismic data compression. In *Optical Science, Engineering and*
10 *Instrumentation'97* (pp. 118-126). International Society for Optics and Photonics.
11

12 Wang, X. Z., Teng, Y. T., Gao, M. T., & Jiang, H. (2004). Seismic data compression
13 based on integer wavelet transform. *Acta Seismologica Sinica*, 17(1), 123-128.
14

15 Wang, Y., & Wu, R. S. (2000). Seismic data compression by an adaptive local
16 cosine/sine transform and its effects on migration. *Geophysical Prospecting*, 48(6), 1009-
17 1031.
18

19 Wood, L. C. (1974). Seismic data compression methods. *Geophysics*, 39(4), 499-525.
20

21 Wu, W., Yang, Z., Qin, Q., & Hu, F. (2006, July). Adaptive seismic data compression
22 using wavelet packets. In *Geoscience and Remote Sensing Symposium, 2006. IGARSS*
23 *2006. IEEE International Conference on* (pp. 787-789). IEEE.

1

2 Xie, X., & Qin, Q. (2009, May). Fast lossless compression of seismic floating-point data.
3 In *Information Technology and Applications, 2009. IFITA'09. International Forum on*
4 (Vol. 1, pp. 235-238). IEEE.

5

6 Xu, Z., Chen, P., & Chen, Y. (2013). Sensitivity kernel for the weighted norm of the
7 frequency-dependent phase correlation. *Pure and Applied Geophysics*, 170(3), 353-371.

8

9 Zhao, L., Jordan, T. H., Olsen, K. B., & Chen, P. (2005). Fréchet kernels for imaging
10 regional earth structure based on three-dimensional reference models. *Bulletin of the*
11 *Seismological Society of America*, 95(6), 2066-2080.

12

13 Zhao, L., Chen, P., & Jordan, T. H. (2006). Strain Green's tensors, reciprocity, and their
14 applications to seismic source and structure studies. *Bulletin of the Seismological Society*
15 *of America*, 96(5), 1753-1763.

16

17 Zheng, F., & Liu, S. (2012, October). A fast compression algorithm for seismic data from
18 non-cable seismographs. In *Information and Communication Technologies (WICT), 2012*
19 *World Congress on* (pp. 1215-1219). IEEE.

20

21 Zhu, L., and D. V. Helmberger (1996). Advancement in source estimation techniques
22 using broadband regional seismograms, *Bulletin of the Seismological Society of America*.
23 86, no. 5, 1634–1641.

- 1 **Table 1. RSGT disk storage with and without compression for different decimation settings.**
- 2 **D_x , D_y and D_z are decimation rates in the two horizontal axes x , y and the vertical axis z ,**
- 3 **respectively.**

Spatial decimation rates			Number of saved grid points	Saved time steps	Number of RSGTs	Disk storage without compression (TB)	Disk storage with compression (TB)	Average compression ratio
D_x	D_y	D_z						
8	8	2	1,572,864	1,228	227	14.36	0.98	14.69
8	2	8	1,572,864	1,228	227	14.36	0.99	14.56
4	4	2	6,291,456	1,228	227	57.42	2.89	20.03
4	2	4	6,291,456	1,228	227	57.42	2.91	19.83
2	2	2	25,165,824	1,228	227	229.69	8.95	25.86
1	1	1	201,326,592	1,228	227	1837.50	48.09	38.37

- 4
- 5

Figure Captions:

Figure 1. Seismic shear-wave velocity (V_s) at 10 km depth in the starting model CAF3D00 (left) and in the latest updated model CAF3D04. Warm colors: lower V_s ; cold colors: higher V_s . Coast lines, state boundaries and major faults are shown in black solid lines. Black dots: locations of the 227 broadband seismic stations used in this study.

Figure 2. Histograms show the distributions of the frequency-dependent group-delay time misfits for the starting model CAF3D00 (black) and the updated model CAF3D04 (red). The misfits quantify the discrepancies between synthetic seismograms computed using the CAF3D00 or CAF3D04 and the corresponding observed seismograms. Smaller misfits indicate better agreement between synthetic and observed seismograms.

Figure 3. Histograms show the distribution of the RWM for the starting model CAF3D00 (black) and the updated model CAF3D04 (red). The RWM quantifies the energy of the waveform differences between observed and synthetic seismograms normalized by the geometrical mean of the observed and synthetic waveform energy. Smaller RWM indicates better waveform fit.

Figure 4. The median of the RWM (mRWM) at different iteration number. At iteration 0, synthetic seismograms were computed using the starting model CAF3D00 and at iteration 4, synthetics were computed using the updated model CAF3D04.

Figure 5. The average relative kernel error (*RKE*) as defined in equation (6) in the text as a function of the error tolerance ϵ used by the compressor for truncating the bit streams. A larger error tolerance indicates that more bits in the bit stream are being discarded. Larger average *RKE* indicates larger discrepancies between the kernels computed using the uncompressed strain fields and those computed using the compressed strain fields.

Figure 6. An example of the data sensitivity kernels computed using raw strain fields without compression (a) and those computed from strain fields compressed at error tolerance $\epsilon = 2^{-50} \gg 8.88 \times 10^{-16}$ (b), $\epsilon = 2^{-47} \gg 7.11 \times 10^{-15}$ (c), $\epsilon = 2^{-44} \gg 5.68 \times 10^{-14}$ (d) and $\epsilon = 2^{-40} \gg 9.09 \times 10^{-13}$ (e). Map-view plots of the data sensitivity kernels are at 10 km depth. Warm colors: negative sensitivity; cold colors: positive sensitivity. The kernel is computed for a frequency-dependent group-delay misfit measured at 0.1 Hz on a vertical-component Rayleigh wave with respect to the second Lamé parameter μ .

Figure 7. Comparison of the snapshots of 4D *zz*-component strain fields at 2-km depth without compression (a), compressed at error tolerance $\epsilon = 2^{-50} \gg 8.88 \times 10^{-16}$ (b), error tolerance $\epsilon = 2^{-47} \gg 7.11 \times 10^{-15}$ (c), error tolerance $\epsilon = 2^{-44} \gg 5.68 \times 10^{-14}$ (d) and error tolerance $\epsilon = 2^{-40} \gg 9.09 \times 10^{-13}$ (e).

Figure 8. Box plot of the RSGT compression ratios for the 6 different grid decimation settings discussed in the text at the error tolerance $\epsilon = 2^{-44} \gg 5.68 \times 10^{-14}$. The red line inside each box indicates the median over the 227 RSGTs. The top and bottom edges of each box indicate the 75th and 25th percentile, respectively. The whiskers extend to

1 approximately ± 2.7 standard deviations and points lying outside of the ranges are plotted
2 as red plus symbols.

3

4 **Figure 9.** The average time for compressing and writing the strain fields in *AWP* wave-
5 propagation simulations for the spatial decimation setting (c). From left to right, the
6 columns are for error tolerance values $e = 2^{-40} \gg 9.09 \cdot 10^{-13}$, $e = 2^{-44} \gg 5.68 \cdot 10^{-14}$,
7 $e = 2^{-47} \gg 7.11 \cdot 10^{-15}$, $e = 2^{-50} \gg 8.88 \cdot 10^{-16}$ and no compression. Blue: compression
8 time; red: disk-writing time.

9

10 **Figure 10.** The average time for reading the compressed strain fields from disk (blue),
11 decompression (green) and kernel computation (red) in the new kernel code *kerz* and the
12 original kernel code *ker*. From left to right, the columns are for error tolerance values
13 $e = 2^{-40} \gg 9.09 \cdot 10^{-13}$, $e = 2^{-44} \gg 5.68 \cdot 10^{-14}$, $e = 2^{-47} \gg 7.11 \cdot 10^{-15}$,
14 $e = 2^{-50} \gg 8.88 \cdot 10^{-16}$ and no compression.

15

1 **Fig 1.**

2

3

4

5

6

7

8

9

1 **Fig 2.**

2

3

4

5

1 **Fig 3.**

2

1 **Fig 4.**

2

1 **Fig 5.**

2

3

4

1 **Fig 6.**

2

3

4

1 **Fig 7.**
2

1

2 **Fig 8.**

3

4

5

6

7

8

1 **Fig 9.**

2

3

4

1 **Fig 10.**

2

3

4

5