BEMC: A Searchable, Compressed Representation for Large Seismic Wavefields

Julio López[1], Leonardo Ramírez-Guzmán[2], Jacobo Bielak[2], and David O'Hallaron[1]

[1] Parallel Data Laboratory, Carnegie Mellon University
[2] Computational Seismology Laboratory, Carnegie Mellon University

Abstract. State-of-the-art numerical solvers in Earth Sciences produce multi terabyte datasets per execution. Operating on increasingly larger datasets becomes challenging due to insufficient data bandwidth. Queries result in difficult to handle I/O access patterns. BEMC is a new mechanism that allows querying and processing wavefields in the compressed representation.
This approach combines well-known spatial-indexing techniques with novel compressed representations, thus reducing I/O bandwidth requirements. A new compression approach based on boundary integral representations exploits properties of the simulated domain. Frequency domain representation further compresses the data by eliminating temporal redundancy found in wave propagation data.
This representation enables the transformation of a large I/O workload into a massively-parallel CPU-intensive computation. Queries to this representation result in largely sequential I/O accesses. Although, decompression places heavy demands on the CPU, it exhibits parallelism well-suited for many-core processors. We evaluate our approach in the context of data analysis for the Earth Sciences datasets.

1 Introduction

Massive datasets representing multi-dimensional *fields* are common in many disciplines of science [17]. Fields describe N-dimensional continuum spaces by assigning scalar or vector quantities to each point in the space. Field datasets found in computational sciences correspond to discrete representations of continuum fields. Advances in simulation methodologies coupled with enabling technological trends, such as faster multicore processors and increased storage capacity, allow scientists and engineers to collect, generate and store increasingly larger fields. As a result, we have become very good at generating gigantic datasets. For example, computations running on current high-performance computing platforms generate datasets with sizes in the order of tens of terabytes (TB) and soon petabytes (PB) [3].

On the downside, we have outstripped our capacity for analyzing these datasets. The main goal of high-resolution numerical simulation is to provide insight about physical

This work was sponsored in part by NSF under grant IIS-0429334; in part by a subcontract from the Southern California Earthquake Center (SCEC) as part of NSF ITR EAR-0122464; and in part by a grant from the Moore Foundation in the eScience project.
The original publication is availabe at www.springerlink.com

phenomena. As the resolution and dataset sizes increase, it becomes difficult to store these datasets at simulation time. Similarly, moving, querying, processing and analyzing these datasets becomes extremely hard. To work around some of these challenges, often a large portion of the data is discarded and instead just a few selected data points or small regions are stored by the simulation for later analysis and publication.

We propose a new mechanism, named *BEMC*, for compactly representing, storing and querying large simulation-generated seismic wavefields. Wavefields are four dimensional vector fields that describe wave propagation phenomena. BEMC is specifically designed to support and enable new data analytic applications in Earth Sciences. The aim is two-fold. First, reduce the I/O bandwidth and storage capacity requirements at simulation time through a compact wavefield representation. Second, facilitate post-simulation analysis through efficient queries and reconstructions of subsets of the wavefield, especially for analytics performed in the frequency domain. Many analysis operations only require portions of a large wavefield at a time. A user might be interested in studying a phenomenon confined to a region of interest. BEMC represents large wavefields as compressed data structures that support spatial queries. Only a relatively small portion of the compressed dataset needs to be accessed and decompressed when analyzing a subset of the wavefield.

BEMC is a domain-specific compression scheme that takes advantage of spatial redundancy present in many wavefields. BEMC uses a novel approach based on boundary integral equations and their corresponding discretization into the boundary element method (BEM) [10,26]. It is coupled with a frequency-based encoding method to further push the limits of wavefield compression while keeping the ability to perform spatial searches on the wavefield.

The basic idea behind BEMC is to only store the computed solution (wave displacement values) for carefully chosen points in the wavefield. These points lie at the boundary of various regions in the simulation input model. In a 3D domain, the region boundaries are the 2D surfaces that wrap each of the 3D regions. At query time a *BEM microsolver* performs a numerically intensive computation to reconstruct the data for a query point from the solutions at the boundary of the region that contains the query point.

The BEM microsolver computation is highly parallel and the compressed data is laid out to exploit the sequential streaming bandwidth of storage systems. The result of the combined approach is that I/O intensive analysis tasks become highly-parallel compute-intensive tasks. With the advent of many-core processors, with hundreds of processing elements per chip, we believe this compression approach will be extremely useful in alleviating I/O bandwidth constraints.

BEMC yields up to 3:1 data size reduction when applied to wavefields based on unstructured octree meshes. Combining BEMC with a frequency-based wave compression results in up to 10:1 factors on these datasets. The location of boundary points is known prior to the simulation. This results in a 66% reductions in the I/O bandwidth requirements at simulation time, since the simulation needs to output values for boundary points only. A post-simulation step transforms the boundary values to a frequency-domain compressed representation to achieve the final 10X compression ratio.

BEMC is independent of the simulation method used in the generation of the synthetic wavefield. It can be used to represent wavefields produced by simulation approaches such as Finite Differences Methods (FD) [27], octree-based Finite Element Methods (FEM) [3], unstructured tetrahedral FEM [8] and Boundary Element Methods (BEM) [26]. BEMC can be applied to other wave propagation problems, and in general to other domains where the problem can be formulated in terms of boundary integral equations. For explanation purpose and without loss of generality, we present BEMC in the context of wavefields generated by octree-based FEM numerical solver.

2 Seismic Wavefield Analysis

The goal of large-scale ground-motion simulations is to enhance our understanding of how the ground shakes during strong earthquakes. Simulating physical processes, such as seismic wave propagation during strong earthquakes, involves computing the solution to a set of governing equations. Non-trivial scenarios have no closed-form solution and the solution is obtained through numerical methods such as FD [27] and FEM [6]. These simulations take as input a model of the ground (material model), construct a discrete simulation *mesh* from the material model and solve a set of partial differential equations on the mesh (See Fig. 1). The simulation produces a large 4-dimensional wavefields and a corresponding mesh. Wavefields are spatio-temporal datasets that describe wave propagation processes by assigning a quantity to each point in space. In the context of ground-motion simulations, a seismic wavefield describes ground properties such as displacement, velocity and acceleration of points in the ground at different moments in time during an earthquake [3]. The analysis of these wavefields has great value for scientists and the community in general. Data analysis produce derived data for applications such as: verification and validation of the simulation process; analysis of interaction with buildings and other man-made structures; real-time damage estimation; material model inversion; and, visualization. Efficient access to these wavefields is key for such analysis applications. The mechanisms presented here help alleviate the I/O requirements needed to store the wavefield (Dataset 3 in Fig. 1) at simulation time and provide a searchable representation for the data analysis tasks.

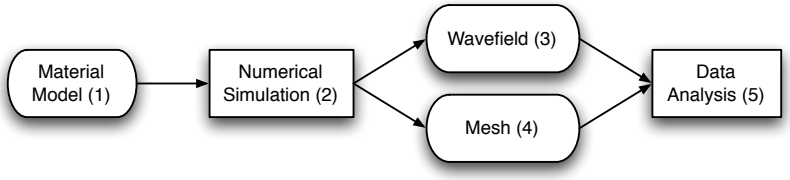


Fig. 1: Physical simulation process. A material model (1) is the input to a numerical simulation process (2), which produces an output wavefield (3) and a corresponding simulation mesh (4). The produced datasets (3) and (4) are used for post-simulation data analysis (5).

3 Compactly Storing Wavefields

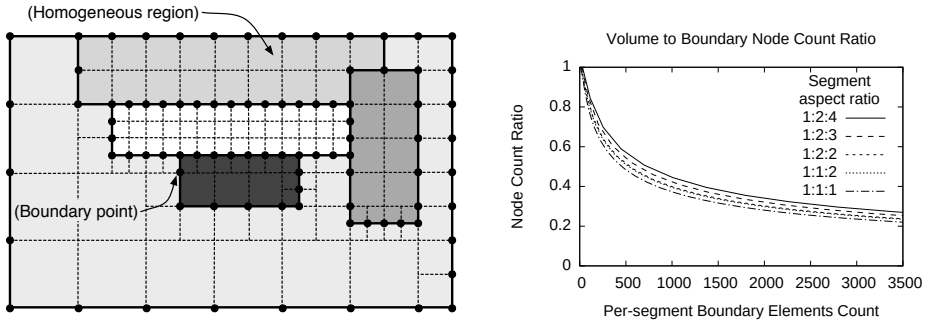
Many datasets from various domains in computational sciences exhibit spatial coherence. The values represented by the field vary in a smooth manner across points in space and time. This is certainly the case for simulation-generated ground-motion wavefields where displacement values vary smoothly in a volume of the wavefield. Despite their spatial coherence, wavefields generated by FEM numerical simulations are difficult to compress since they contain floating point (FP) values associated with the nodes of an irregular mesh. Compressing FP values is challenging because small differences in absolute value yield large changes in the FP bit representation. Moreover, unstructured meshes used in FEM simulations adapt to variations in the properties of the input model parameters, thus greatly eliminating spatial redundancy in the corresponding output wavefield. FEM octree-based meshes and corresponding output wavefields commonly require 1/8 of the space needed to represent a FD mesh for the same problem. It is hard to use image compression techniques in this case due to the irregular nature of the meshes used to generate the wavefields, as many image compression approaches operate on regularly spaced grids. The question is then: How can spatio-temporal coherence be exploited to compress massive wavefields? BEMC exploits wavefield spatial coherence through domain-specific formulation based on Boundary Integral Equations and Green's functions. This approach can be used to compress datasets generated using FEM, FDs or other method. The main idea is to make use of a property of the input material model (Dataset 1 in Fig. 1) to compress the output wavefield (Dataset 3). This is based on the observation that the *input* material models used in simulation (Dataset 1) have large homogeneous regions. Then, for any point inside a homogeneous region in the input (Dataset 1), we can compute the corresponding output displacement values (in Dataset 3) in terms of the (output) displacement values of the points that lie on the region boundary.

3.1 Model Homogeneity

Material models used for simulations comprise large homogeneous regions. This is depicted in Fig. 2 (a), where 5 homogeneous regions (different shades of gray) make up a simulation domain. Simulating physical phenomena in complex structures requires the use of full-domain methods such as FEM and FD. Meshes for these methods divide the domain into small discrete mesh elements depicted by dotted squares in the figure. The element corners are referred to as mesh nodal points or simply nodes. Homogeneous regions in the input model are also divided into small mesh elements to satisfy numerical stability requirements for the simulation. Finer resolution meshes increase simulation accuracy. For example, simulating higher wave frequencies requires finer meshes. As a mesh becomes finer, more elements are needed to fill up the same homogeneous region. In the simulation output wavefield (Dataset 3 in Fig. 1), the wave displacement values at mesh nodal points inside homogeneous regions vary relatively smoothly within each region. Although the values have small differences from node to node, the variations are significant enough and thus need to be explicitly stored.

BEMC reduces wavefield storage requirements by exclusively keeping values for nodes located on region boundaries, i.e., lying on the thick lines in Fig. 2 (a). A discrete

representation of region boundaries is made up of 2D *boundary elements* (BE) embedded in a 3D space. In the 2D illustration in Fig. 2 (a), the BEs correspond to the thick lines between two mesh points at the boundary of the regions.



(a) Material model homogeneity. **(b)** Node Count Reduction Ratio.

Fig. 2: (**a**) Material model homogeneity. This 2D sketch shows the composition of a hypothetical material model for ground-motion simulation. This corresponds to Dataset 1 in Fig. 1. It contains various homogeneous regions (different shades of gray), separated by boundaries (thick lines). The size and properties of these regions can vary by various orders of magnitude.
(**b**) Node Count Ratio. This is the achievable compression for regions of different sizes and aspect ratios. The X axis is the number of boundary elements needed to wrap the segment. The Y axis shows the reduction in the number of mesh nodes compared to a regular volume mesh with the same resolution as the boundary mesh.

A boundary mesh representation only requires nodes at the boundary. Values for nodes inside a region can be safely discarded. Figure 2 (b) shows the storage reduction obtained by only storing boundary node values. The X axis is the boundary mesh size in number of elements. The Y axis shows the ratio of boundary node count to volume node count for a boundary mesh of the given size. Different lines show the ratio for hexahedral regions for various 3D aspect ratios — 1:1:1, 1:1:2, etc. Compression ratios below 0.3 are possible for regions with more than 1500 boundary elements.

3.2 Compression Steps

The BEMC comprises the steps shown in Figure 3: Model Segmentation (6), Boundary mesh generation (7) and Wavefield data extraction (8). These steps are explained below. The inputs for BEMC are: the full domain wavefield to be compressed (3), the input material model used to generate the wavefield (1) and the full 3D domain simulation mesh (4). The BEMC process produces a compact output representation comprised of a segment index (9), a boundary mesh (10) and the wave data associated with nodes along the boundaries of homogeneous regions (11). The segment index and boundary mesh are used to support spatial range queries. The segment index is used as a spatial index, and the boundary mesh provides the mapping from the segment boundary to wave data associated with points along the boundary for the segments. At query time, the data

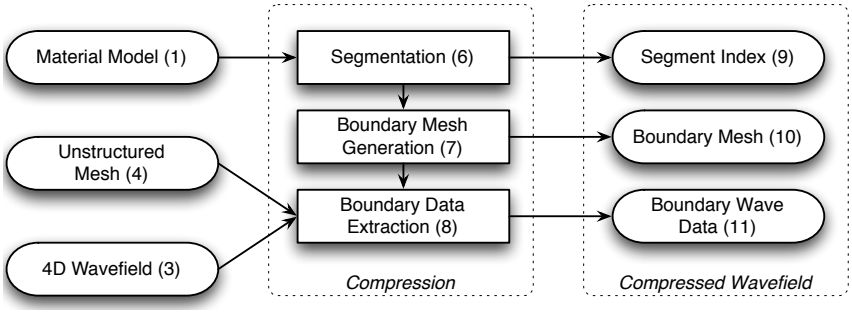


Fig. 3: Steps for the BEMC compression phase. Its inputs are a wavefield (3), the corresponding unstructured mesh (4) and the material model (1) for the region covered by the wavefield. This phase consists of the following processes: Segmentation (6), Boundary Mesh Generation (7) and Boundary Data Extraction (8). They produce a compressed boundary wavefield with three components: a segment index (9), a boundary mesh (10) and associated wave data (11).

for a query point inside a segment is reconstructed from the data associated with the boundary nodes.

Model Segmentation. The goal of this step is to find a set of homogeneous regions or *segments* in the input model and produce a spatially-indexed segment set. A region is considered to be homogeneous if all of its material properties remain constant in its volume. Extracting homogeneous regions becomes challenging due to the large model sizes, currently in the order of tens to hundreds of gigabytes. Segmentation is an active area of research with applications to medical imaging and computer vision [34]. Most readily available implementations of algorithms for image segmentation target the case where the image fits in main-memory [14]. Researchers in the area of scientific visualization have proposed methods for iso-surface extraction [12], model simplification [13] and near real-time rendering [19] of large spatial datasets.

In order to deal with large 3D material models, we developed a simplified out-of-core segmentation procedure [20] that leverages the internal representation of material models and octree meshes used in FEM ground-motion simulation [3,30]. The general idea is to use an octree FEM mesh as a segmentation template, and check the mesh against the material model to eliminate any over-segmentation –whenever possible coarsen the mesh where it is too fine. These octree-based meshes are stored on disk as indexed *linearized octrees* using the CMU etree library [29]. The mesh elements are cubes that correspond to nodes of the octree. The order for the elements' on-disk representation corresponds to a *pre-order traversal* of the corresponding octree. The segmentation process performs a pre-order traversal of the octree mesh elements and checks the homogeneity of a set of 8 sibling octants elements by querying to the material model. If the octants are homogeneous they are coalesced into a single parent octant. If the octants cannot be coalesced, they are output as individual segments.

This algorithm has complexity $O(n)$ and requires little in-memory state. This approach requires a single pass over the mesh elements. It uses the streaming bandwidth of storage devices very efficiently, as the pre-order traversal of the octree structure re-

sults in sequential I/O accesses. The implementation has a very small footprint. It only requires state for eight siblings octree nodes[3] in the path from the tree root to the current leaf node. The state size has an upper bound of $8 \times \texttt{tree\_depth}$. The octree depth for commonly used seismic meshes varies between 12 and 20 [20].

The resulting segments are constrained to simple shapes, simplifying the generation of the corresponding boundary mesh. However, this approach does not fully take advantage of model homogeneity as it does not coalesce homogeneous segments that do not align properly across octree boundaries, i.e., when they are not children of the same parent octant.

Boundary Mesh Generation. This stage reads the material segment set and the full-domain mesh in order to generate discrete *boundary meshes* (BM) for the homogeneous regions. The resulting BMs discretize the segment boundaries such that they can be later used to reconstruct any point inside using a numerical computation. A BM comprises a set of 2D boundary elements (BE) embedded in a 3D space. The corresponding element corners are the boundary mesh nodes (BN).

This procedure reads each segment and subdivides each segment face into smaller rectangular boundary elements according to the material properties of the segments on both sides of the face. The size of the BE is chosen such that it satisfies the numerical requirements for the numerical computation in the reconstruction process. Whenever possible, the BE size is the same as the size of the faces of the elements in the unstructured mesh. Matching the BE size to the one of the FEM elements can be done based on two simulation parameters (points per wavelength and maximum wave frequency), thus access to the complete unstructured FEM mesh is not needed. Boundary mesh nodes are generated in a second pass over the generated boundary mesh elements. This is an involved process that assigns two identifiers to each node: a segment-local id and a mesh global id. The corresponding mapping is stored as part of the boundary mesh output. The global node identifiers are used for laying out and addressing the wave data associated with boundary nodes.

Wavefield Data Extraction. This process extracts the actual wave data associated with boundary nodes. It accounts for the bulk of the data size reduction. The first step is to build an auxiliary structure for mapping between boundary mesh node ids and full-domain mesh node ids. If the BM generation above produces meshes with BEs that are aligned with the FEM elements, then there is a one-to-one mapping between boundary mesh nodes and nodes in the full domain mesh. Additional interpolation operations on the associated boundary data are avoided when the boundary mesh nodes are aligned with full domain mesh nodes. This process iterates over the boundary nodes and uses the node mapping structure to locate and extract the corresponding data from the 4D wavefield.

The wave data is transposed from the *space domain* to the *time domain*. Popular ground-motion solvers produce wave data one time step at a time and store it in a space domain representation (space $\times$ time), where the dataset is a collection of volumes each corresponding to a snapshot in time [3,22]. Given a time step t, values for all the mesh nodes are adjacently stored, then all the values for time step $t+1$, and so on. To further compress the data and facilitate its reconstruction at query time, the wave data is stored

[3] Child nodes with the same common parent.

in a time domain representation (time $\times$ space). In the time domain, the data is stored as a sequence of time series. The extracted wave data is transposed with an efficient out-of-core scheme optimized for matrices with extremely large aspect ratios, e.g., 1000:1 [20].

Simulation-generated seismic wavefields often are *band-limited*, i.e., they contain useful wave data in a limited frequency range. In particular, the numerical solver produces wave data up to a specified maximum frequency for the simulation. This offers an opportunity to achieve higher compression. We developed a frequency-based compression scheme named *effective-zero* encoding. First, the wave data is transformed to the *frequency domain* using an available FFT implementation. Then, an *effective zero* value is independently computed for each node wave spectrum [20]. In the frequency spectrum, a wave number has an effective-zero value if its magnitude is below a user defined threshold relative to the cummulative energy of the spectrum. Only non-zero values are stored in the frequency spectrum for a boundary node.

4 Wavefield Reconstruction

Wavefield reconstruction is formulated as queries for a point q in the wavefield. The query set can include arbitrary points in the domain–it is not limited to points in the original full-domain mesh. Spatial range queries are possible due to the chosen storage representation and can be satisfied by reading only the portion of the dataset covered by the query range. The first step involves looking up q in the segment index to find the homogeneous region containing q (See Fig. 4). Reconstructing the data for q is achieved through one of the following methods:

1. If q *coincides* with a boundary mesh node, then return the data associated with the mesh node.
2. When q is *on* or *near* the boundary, interpolate from the data associated with the corners of the closest boundary element.
3. When q is *far* from the boundary, execute a *microsolver* to compute the wave data from the values at the region boundary.

A query point is near the boundary if its distance to the closest BE is less than a user specified parameter, which by default is $1/4$ of the BE edge length. In all the cases above, satisfying the query only requires access to small subsets of the compressed wavefield, thus reducing I/O bandwidth usage and computation. The compressed layout is such that it induces sequential I/O accesses.

4.1 Microsolver Computation

For cases where q is far from the region boundary, the answer is computed by applying the Boundary Element Method (BEM) [10,26]. A microsolver carries out the required numerical computation using as input the compressed boundary wavefield. This process is divided into two steps: Phi computation (13) and Wave data computation (14). The BEM mathematical formulation and the corresponding steps are explained below.

BEM Formulation. Without loss of generality, assume we have a query point $q \in \Omega, \Omega \subset \Re^3$, where Ω is a homogeneous 3D region with a boundary S (See Fig. 5 for

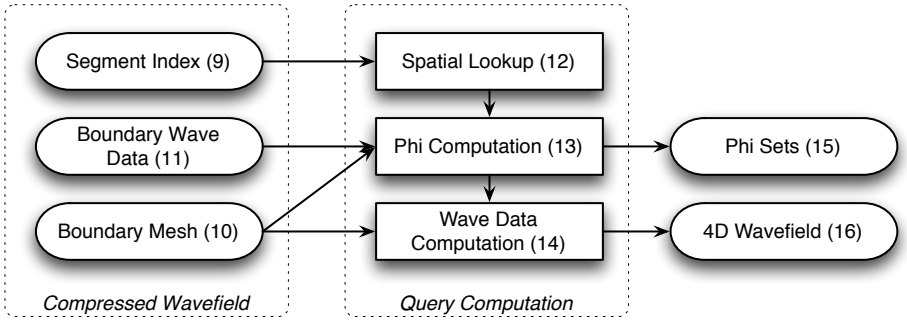


Fig. 4: BEM decompression phase. At query time, the data for query points is reconstructed from the boundary wavefield representation. The computation involves three steps: Spatial Index Lookup (12), Phi Computation (13) and Query Wave Data Computation (14).

an illustration in two dimensions). Reconstructing the displacement values for a point inside Ω involves numerically computing the boundary integral shown in Eq. 1.

$$u(q) = \int_S \phi(\xi) G(q,\xi) dS_\xi \tag{1}$$

The discrete approximation is shown in Equation 2. This is roughly equivalent to dividing the boundary S into discrete boundary elements ξ_i and summing the contributions of all boundary elements ξ_i along the boundary S as shown in Fig. 5.

$$u(q) = \sum_{k=0}^{n-1} \phi(\xi_k) G(q,\xi_k) \times A(\xi_k) \tag{2}$$

Here, $u(q)$ is a vector containing the displacement in the frequency domain for a point q, $\phi(\xi_k)$ weights the contribution to $u(q)$ from ξ relative to other points (or BEs) on the surface S. $A(\xi_k)$ is the area of the BE ξ_k. $G(q,\xi_k)$ is the Green's function [5] between q and a boundary element ξ_k. $G(q,\xi_k)$ can be analytically computed based on the material properties of the region Ω and the coordinates of q and ξ. See Appendix A for details.

Phi Computation. The phi values $\phi(\xi)$ in equations 1 and 2 are initially unknown. The first phase of the microsolver computes the $\phi(\xi)$ terms for the Ω region. The computed $\phi(\xi)$ depend exclusively on the region material properties and boundary displacement values. Once the phi values are computed, they can be reused across other query points in the same region.

Equation 1 also holds for boundary points as shown in Fig. 6. Thus, computing the phi values is achieved by using Eq. 2 with points at the boundary, by letting $q = \xi_i$. In this case, $u(\xi_i)$ are known values stored in the boundary wavefield (Dataset 11 in Fig. 4) and $G(\xi_i,\xi_j)$ can be analytically computed. This enables setting up a system of N linear equations with N unknown $\phi(\xi_i)$ values, where N is the number of boundary elements.[4]

[4] More precisely, the number of equations and unknowns is $3N$ once $u(\xi_i)$, $\phi(\xi_i)$ and $G(\xi_i,\xi_j)$ are expanded into their individual 3D components.

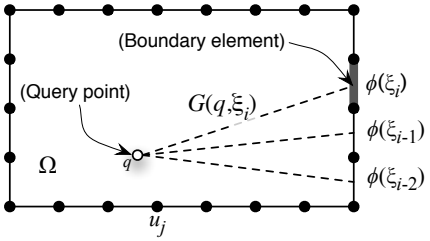


Fig. 5: Query point computation. Computing values associated with a query point q, in a homogeneous region Ω, from the $\phi(\xi_i)$ values associated with boundary elements $\xi_i \in S$ (outer line).

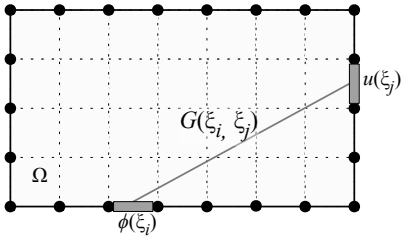


Fig. 6: Phi computation. The $\phi(\xi_i)$ values are unknown. They are computed from the known displacement values at the boundary $u(\xi_j)$ by solving a system of linear equations.

$$\overbrace{\begin{bmatrix} G(\xi_0\xi_0) & \cdots & G(\xi_0\xi_i) & \cdots & G(\xi_0\xi_{n-1}) \\ G(\xi_1\xi_0) & \cdots & G(\xi_1\xi_i) & \cdots & G(\xi_1\xi_{n-1}) \\ \vdots & & \vdots & & \vdots \\ G(\xi_j\xi_0) & \cdots & G(\xi_j\xi_i) & \cdots & G(\xi_j\xi_{n-1}) \\ \vdots & & \vdots & & \vdots \\ G(\xi_{n-1}\xi_0) & \cdots & G(\xi_{n-1}\xi_i) & \cdots & G(\xi_{n-1}\xi_{n-1}) \end{bmatrix}}^{\xi_i} \times \overbrace{\begin{bmatrix} \phi_{\xi_0} \\ \phi_{\xi_1} \\ \vdots \\ \phi_{\xi_i} \\ \vdots \\ \phi_{\xi_{n-1}} \end{bmatrix}}^{\phi} = \overbrace{\begin{bmatrix} u(\xi_0) \\ u(\xi_1) \\ \vdots \\ u(\xi_i) \\ \vdots \\ u(\xi_{n-1}) \end{bmatrix}}^{u}$$

Fig. 7: Matrices representing the linear system of equations corresponding to the $\mathcal{G}_\omega \times \Phi_\omega = U_\omega$ equation. This system is solved to obtain the unknown ϕ values for a frequency ω. $\mathcal{G}_\omega$ contains the (known) coefficients of the Green's tensors, Φ_ω is the vector of unknown variables and U_ω is the vector of known displacement values obtained from the simulation wavefield.

Figure 7 shows the matrix system resulting from these equations. The discretization details can be found elsewhere [20,26]. Equation 1 is undefined for the entries along the matrix diagonal, more precisely $G(\xi_i,\xi_i)$ is undefined. These singularities are resolved through numerical integration using quadratures of odd order [24]. All these computations are carried out in the frequency domain. A system of equations needs to be solved for each frequency (wave number) that has a non-zero $u(\xi)$ value. The last step in the compression phase eliminates frequencies with effective zero values. This also reduces the number of systems of equations that need to be resolved at query time.

Query Value Computation. After computing the phi values, the data for q is obtained from Eq. 2. The data for q is the sum of the products between the $\phi(\xi_i)$ and the corresponding Green's tensor $G(q,\xi_j)$. Figure 5 illustrates the process.

5 Evaluation

The goal of this evaluation is to answer the following questions: (1) *What level of compression can be achieved with BEMC*? and (2) *What is the computational cost of reconstructing a query point with the BEM microsolver?* These questions are answered below.

5.1 Quake Unstructured Wavefields

Table 1 shows the characteristics of the wavefields used in the experiments. These are seismic wavefields generated using a state-of-the-art numerical solver for seismic wave propagation problems. They cover a $100\,\text{km} \times 100\,\text{km} \times 37.5\,\text{km}$ region in Southern California that includes Los Angeles and San Fernando basins. The first table column contains the dataset name, which corresponds to the maximum resolved wave frequency. The second and third columns respectively show the number of mesh elements and nodes. The fourth column displays the wavefield size in Gigabytes.

Table 1: Unstructured 4D Wavefields.

Wavefield	Mesh		Size
Name	Elements	Nodes	(GB)
LA 0.50 Hz	8,026,868	8,634,452	116
LA 0.70 Hz	17,970,403	19,372,567	260
LA 1.00 Hz	64,128,816	66,548,707	893

Table 2: BEMC Compression Ratio.

Wavefield	BEMC	BEMC + freq.
LA 0.50 Hz	0.71	0.14
LA 0.70 Hz	0.51	0.13
LA 1.00 Hz	0.27	0.07

5.2 BEMC Compression

In order to find out the effectiveness of BEMC, we applied the method on the wavefields shown in Table 1. We constructed boundary meshes for these wavefields and compared the number of mesh nodal points versus the number of nodes in the FEM hexahedral mesh for the same wavefield. The boundary mesh is constructed using the approach described in Section 3. Remember that this approach might not capture all the homogeneity of the model, thus not allowing the BEMC compression to take advantage of all the spatial redundancy.

The compression ratio (`output_size` / `input_size`) for the different wavefields is shown in Table 2. Column 2 contains the compression ratio achieved exclusively from the dimensionality reduction –storing only values at the boundary. As the wavefield simulation frequency increases, BEMC produces better compression. This is expected as an FEM mesh needs to subdivide a homogeneous region with higher resolution as the maximum simulation frequency increases. Compressing the LA-0.5Hz wavefield with BEMC produces only 30% storage savings, however for the LA-1.0Hz the savings are 73%, this is a 3.7:1 compression factor. The implication is that the number of boundary elements per region is larger for the LA-1Hz wavefield, which has an effect on computation performed by the BEM microsolver at query time.

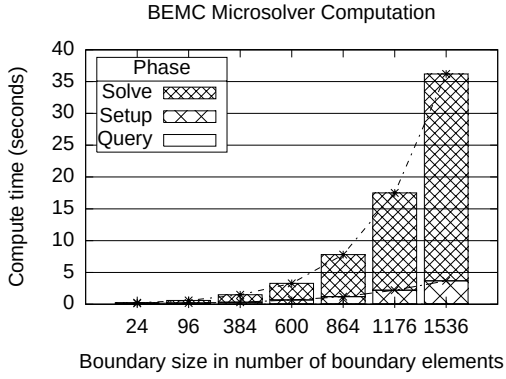
The third column in Table 2 shows the compression ratio obtained when applying the frequency-based encoding technique (described in Section 3.2) to the boundary wavefields. The combined scheme achieves a compression ratio of 0.07 (14.3:1 factor) in the best case on the LA-1Hz wavefield, and a 0.14 ratio (7:1 factor) in the worst case for the LA-0.5Hz wavefield.

Combining these two approaches, not only reduces storage requirements, but also computation requirements. At query time, the BEM microsolver only needs to compute values for frequencies that are non-zero. This represents a 5X reduction of computation for the best case (LA-0.5Hz) and a 4X reduction for the worst case (LA-0.7Hz).

5.3 BEM Microsolver Computation Cost

The following micro-benchmark was used to determine the computational cost of the BEM microsolver. For regions of different sizes (measured in the number of elements along the boundary), we measured the time needed to compute the data for a query point inside the region. The boundary mesh and wave data were already in memory, allowing to measure the actual compute cost. The microsolver is implemented as a library of C++ classes. In the phi computation phase, the system of linear equations is solved by calling LAPACK [4] functions with low-level BLAS [9] routines provided by ATLAS [32]. Theses experiments were carried out on dual SMP machines with 3.6 GHz x86_64 processors, 2 MB L2 cache per processor and 8 GB of memory running version 2.6.15 of the Linux™kernel. The numerical libraries were configured to use 2-processors. The system setup and query phases are done sequentially on a single processor.

Figure 8 shows the microsolver compute time per frequency. The X axis indicates the region size in number of boundary elements, the Y axis contains the elapsed time in seconds. Each bar is divided into the following BEM microsolver phases: Setup, Solve, and Query computation. The table in Fig 8 shows the corresponding values in seconds.



BEMC Microsolver Time (s)

BE count	Setup	Solve	Query	Total
24	0.13	0.12	0.002	0.25
96	0.20	0.40	0.005	0.60
384	0.30	1.19	0.008	1.50
600	0.66	2.61	0.011	3.29
864	1.19	6.61	0.014	7.81
1176	2.19	15.29	0.019	17.50
1536	3.67	32.51	0.024	36.20

Fig. 8: BEMC microsolver compute time in seconds.

Not surprisingly, the query time grows linearly with the region size, setup time is $O(n^2)$ and solving time is $O(n^3)$. This puts a practical limit on the achievable compression ratio, in the sense that homogeneous regions cannot grow too large, otherwise the query computation time becomes too high.

A typical homogeneous region contains in the order of a 100 wave numbers in its frequency spectrum. Thus the total required computation is about 100X larger. However, with the increase in the number of available processor cores, the solving time for the system of equations can be reduced. Moreover, the computation for different frequencies (wave numbers) can proceed independently in parallel in separate chips or compute nodes. The setup and compute phases are performed once per region, introducing a one-time cost. Subsequent query points for the same region can be computed from the already obtained solution in linear time, thus amortizing the setup and solving costs.

We can trade off the achieved compression ratio to reduce the query latency by pre-computing the phi values and storing them on disk. The raw size of the phi set is 2X the size of the compressed displacement field for the boundary points. However, both additional storage size and computational reductions can be obtained by devising a multi-resolution scheme for storing the phi set, such that the representation for lower frequencies has a coarser boundary element mesh (i.e., fewer phi values) and higher frequencies have finer-resolution boundary meshes.

6 Related Work

The goal of data compression is to reduce the number of bits needed to represent the data. Compression techniques exploit data features such as redundancy. Clever encoding schemes reduce the number of bits required to represent a data symbol. Such techniques include arithmetic coding, Huffman coding, Golomb-Rice codes among others [1]. Lelewer and Hirschberg present a good survey of these and other techniques [18]. Run-length encoding (RLE) is commonly used in many applications, including the compression of non-photographic images, that is, diagrams produced using drawing tools. BEMC employs a form of implicit RLE encoding to avoid storing the effective zero values of a frequency spectrum.

There are various approaches that exploit data features commonly found in text files. These include Lempel-Ziv-Welch (LZ and LZW) and the Burrows-Wheeler transform (BWT) [11]. The LZW and BWT methods are respectively used in the popular `gzip` and `bzip2` compression tools. These approaches work well on text data and binary executable programs.

Lossy compression schemes achieve higher compression factors at the cost of some information loss. Upon decompression, the output is an approximation of the input. Image formats such as JPEG [31] use lossy compression methods based on a discrete cosine transform [2]. JPEG compression takes advantage of the fact that photographic images have smooth variations from one pixel to the next one. Audio compression techniques exploit the fact that the stream of data to compress corresponds to sound waves [21,33]. Approaches for compressing floating-point values have been recently developed [15,16,25]. These approaches are complementary to BEMC, as they could

be used as a post-processing pass to effectively encode the residual floating point data for points at the boundary.

7 Conclusion

Generating, querying and otherwise analyzing simulation-generated wavefield datasets becomes difficult as their data size increases. We presented BEMC, a novel approach for compactly representing large seismic wavefields. A key feature of BEMC is that it enables spatial range queries in the compressed domain, only a small portion of the data needs to be retrieved from storage at query time. BEMC uses a domain-specific approach based on the boundary element method (BEM) to reconstruct the wave data. A query-time microsolver is used to carry out the BEM numerical computation. We believe that this approach can be generalized and applied to other wave propagation problems and in general to other domains that can be formulated in terms of boundary integral equations.

Our evaluation shows that dimensionality reduction alone offers up to a 3.7X compression factor when applied to large seismic wavefields. The combined BEMC approach yields compression factors up to 14X. The frequency-based encoding used in BEMC contributes to the reduction of both storage and query-time computational requirements.

A Boundary Integral Equations

$$u_i(p) = \int_S \sum_{j=0}^{2} \phi_j(\xi) G_{ij}(p,\xi) dS_\xi \tag{3}$$

$$G_{ij}(x,\xi) = [f_2 \delta_{ij} + (f_1 - f_2)\gamma_i \gamma_j]/4\pi\mu r \tag{4}$$

The following is the corresponding expansion for the terms in the Green's function ($G_{ij}(x,\xi)$). The parameters involved in the equations are shown as well.

$$f_1 = \frac{\beta^2}{\alpha^2}\left[1 - \frac{2i}{qr} - \frac{2}{(qr)^2}\right] e^{-iqr} + \left[\frac{2i}{kr} + \frac{2}{(kr)^2}\right] e^{-ikr}$$

$$f_2 = \frac{\beta^2}{\alpha^2}\left[\frac{i}{qr} + \frac{1}{(qr)^2}\right] e^{-iqr} + \left[1 + \frac{i}{kr} - \frac{1}{(kr)^2}\right] e^{-ikr}$$

$$\gamma_j = (p_j - \xi_j)/r$$

$$r^2 = (p_0 - \xi_0)^2 + (p_1 - \xi_1)^2 + (p_2 - \xi_2)^2$$

$$\mu = \rho\beta^2$$

$$q = \omega/\alpha = \text{P-wave number}$$

$$k = \omega/\beta = \text{S-wave number}$$

$$\delta_{ij} = 1 \text{ if } i = j; 0 \text{ otherwise}$$

	Description
p	Point (3D coordinates)
$u(p)$	3D displacement at p
$\phi(\xi)$	Phi vector for BE ξ
δ_{ij}	Kronecker's delta
q	P-wave number
α	P-wave velocity
k	S-wave number
β	S-wave velocity
ρ	Region's mass density

References

1. Abramson, N.: Information Theory and Coding. McGraw-Hill, New York (1963)
2. Ahmed, N., Natarajan, T., Rao, K.R.: Discrete cosine transform. Trans. on Computers (TOC) C(23), 90–93 (1974)
3. Akcelik, V., Bielak, J., Biros, G., Ipanomeritakis, I., Fernandez, A., Ghattas, O., Kim, E., López, J., O'Hallaron, D., Tu, T., Urbanic, J.: High resolution forward and inverse earthquake modeling on terascale computers. In: Proc. Supercomputing SC'03 (2003)
4. Anderson, E., et al.: LAPACK Users' Guide. SIAM, Philadelphia, 2nd edn. (1995)
5. Arfken, G.: Mathematical Methods for Physicists, chap. Nonhomogeneous Equation–Green's Function, pp. 480–491. Academic Press, 3rd edn. (1985)
6. Babuska, I., Strouboulis, T.: The Finite Element Method and Its Reliability. Oxford University Press, USA, 1st edn. (Dec 2001)
7. Bao, H., et al.: Large-scale simulation of elastic wave propagation in heterogeneous media on parallel computers. Computer Methods in Applied Mechanics and Engineering 152 (1998)
8. Blackford, L., et al.: An updated set of basic linear algebra subprograms (BLAS). ACM Trans. Math. Soft 28(2), 135–151 (2002)
9. Bonnet, M.: Boundary Integral Equation Methods for Solids and Fluids. John Wiley & Sons, 1st edn. (June 1999), iSBN:0-471-97184-7
10. Burrows, M., Wheeler, D.J.: A block-sorting lossless data compression algorithm. Tech. Rep. 124, Digital Equipment Corporation (DEC) (1994)
11. Chiang, Y.J., Farias, R., Silva, C.T., Wei, B.: A unified infrastructure for parallel out-of-core isosurface extraction and volume rendering of unstructured grids. In: Proc. Symp. on Parallel and Large-data Visualization and Graphics. pp. 59–66. IEEE, IEEE Press (2001)
12. Hoppe, H.: Smooth view-dependent level-of-detail control and its application to terrain rendering. In: Proc. Int. Conf. Visualization VIS'98. pp. 35–42. IEEE (Oct 1998)
13. Ibáñez, L., Schroeder, W., Ng, L., Cates, J.: The ITK Software Guide. Kitware, Inc (November 2005), iSBN: 1930934157
14. Ibarria, L., Lindstrom, P., Rossignac, J., Szymczak, A.: Out-of-core compression and decompression of large n-dimensional scalar fields. In: Proc. Eurographics (2003)
15. Isenburg, M., Lindstrom, P.: Fast and efficient compression of floating-point data. IEEE Transactions on Visualization and Computer Graphics (TOVCG) 12(5), 1245–1250 (2006)
16. Landau, L.D., Lifshitz, E.M.: Classical Theory of Fields, The Course of Theoretical Physics, vol. 2. Pergamon, London, 3rd edn. (1976)
17. Lelewer, D.A., Hirschberg, D.S.: Data compression. ACM Comput. Surveys (CSUR) 19(3), 261–296 (1987)
18. Lindstrom, P.: Out-of-core construction and visualization of multiresolution surfaces. In: Proceedings of the 2003 ACM Interactive 3D Graphics Conference (2003)
19. López, J.: Methods for Querying Compressed Wavefields. Ph.D. thesis, Department of Electrical and Computer Engineering, Carnegie Mellon University (May 2007)
20. (MPEG), I.J.M.P.E.G.: ISO/IEC-11172 MPEG-1 standard, Coding Of Moving Pictures And Audio. International Organisation For Standardisation (ISO), 1 edn. (1996)
21. Olsen, K.: Three-dimensional ground motion simulations for large earthquakes on the san andreas fault with dynamic and observational constraints. Comp. Acoust. 9(3), 1203–1215 (2001)
22. Press, W.H., Teukolsky, S.A., (Contributor), W.T.V.: Numerical Recipes in C: The Art of Scientific Computing. Cambridge University Press (1992), 1020 pages
23. Ratanaworabhan, P., Ke, J., Burtscher, M.: Fast lossless compression of scientific floating-point data. In: Proc. Data Compression Conference (DCC) (March 2006), pp. 133 – 142

24. Sanchez-Sesma, F.J., Luzon, F.: Seismic response of three-dimensional alluvial valleys for incident p, s, and rayleigh waves. Bulletin of the Seismological Society of America 85(1), 269–284 (Feb 1995)
25. Smith, G.D.: Numerical Solution of Partial Differential Equations: Finite Difference Methods (Third Edition). Oxford University Press, New York, NY (1985)
26. Tu, T., López, J., O'Hallaron, D.: The Etree library: A system for manipulating large octrees on disk. Tech. Rep. CMU-CS-03-174, Carnegie Mellon, Computer Science Dept. (2003)
27. Tu, T., O'Hallaron, D., López, J.: Etree – a database-oriented method for generating large octree meshes. In: Proc. 11th Int. Meshing Roundtable. pp. 127– 138. Ithaca, NY (Sep 2002)
28. Wallace, G.K.: The JPEG still picture compression standard. Communications of the ACM (CACM) 34(4), 30–44 (Apr 1991)
29. Whaley, R.C., Dongarra, J.: Automatically Tuned Linear Algebra Software (ATLAS). In: Ninth SIAM Conference on Parallel Processing for Scientific Computing (1999)
30. Yang, D., Moriya, T., Liebchen, T.: A lossless audio compression scheme with random access property. In: Proc. Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP'04). vol. 3, pp. iii– 1016–1019. IEEE (May 2004)
31. Yoo, T.S. (ed.): Insight into Images: Principles and Practice for Segmentation, Registration, and Image Analysis. AK Peters, Ltd (July 2004), iSBN:1568812175