Geophysical Prospecting

EAGE EUROPEAN ASSOCIATION OF GEOSCIENTISTS & ENGINEERS

Geophysical Prospecting, 2011, **59**, 857–873 doi: 10.1111/j.1365-2478.2011.00982.x

On 3D modeling of seismic wave propagation via a structured parallel multifrontal direct Helmholtz solver

Shen Wang*, Maarten V. de Hoop and Jianlin Xia

Center for Computational and Applied Mathematics, Purdue University, 150 N. University Street, West Lafayette IN 47907, USA

Received January 2011, revision accepted March 2011

ABSTRACT

We consider the modeling of (polarized) seismic wave propagation on a rectangular domain via the discretization and solution of the inhomogeneous Helmholtz equation in 3D, by exploiting a parallel multifrontal sparse direct solver equipped with Hierarchically Semi-Separable (HSS) structure to reduce the computational complexity and storage. In particular, we are concerned with solving this equation on a large domain, for a large number of different forcing terms in the context of seismic problems in general, and modeling in particular. We resort to a parsimonious mixed grid finite differences scheme for discretizing the Helmholtz operator and Perfect Matched Layer boundaries, resulting in a non-Hermitian matrix. We make use of a nested dissection based domain decomposition, and introduce an approximate direct solver by developing a parallel HSS matrix compression, factorization, and solution approach. We cast our massive parallelization in the framework of the multifrontal method. The assembly tree is partitioned into local trees and a global tree. The local trees are eliminated independently in each processor, while the global tree is eliminated through massive communication. The solver for the inhomogeneous equation is a parallel hybrid between multifrontal and HSS structure. The computational complexity associated with the factorization is almost linear in the size, n say, of the matrix, viz. between $O(n \log n)$ and $O(n^{4/3} \log n)$, while the storage is almost linear as well, between $O(n)$ and $O(n \log n)$. We exploit the use of a regular (Cartesian) mesh common in many seismic applications.

Key words: Helmholtz, HSS structure, massively parallel, modeling, multifrontal, rectangular domain.

INTRODUCTION

We consider the discretization and approximate solution of the inhomogeneous Helmholtz equation in 3D. In particular, we are concerned with solving this equation on a large domain, for a large number of different forcing terms in the context of modeling seismic wave propagation with applications in so-called (local optimization based) full waveform inversion (FWI) in mind. The direct method of choice for solving this problem is the multifrontal factorization algorithm. The central idea of the multifrontal algorithm is to reorganize the sparse factorization into a series of dense local factorizations. The algorithm is used together with the method of nested dissection to obtain a nested hierarchical structure and generate the LU factorization from the bottom up to minimize fill-ins.

In this paper, we follow the approach developed by Xia; Xia *et al.* (2010) of integrating the multifrontal method with structured matrices. The main implication is that the fill-in blocks of the factorization are highly compressible using the framework of hierarchically semiseparable (HSS) matrices.

*E-mail: wang273@purdue.edu

We resort to a parsimonious mixed grid finite differences scheme for discretizing the Helmholtz operator, which yields a compact stencil, and Perfect Matched Layer (PML) boundaries. We note that the resulting matrix is non-Hermitian, indefinite, and relatively poorly conditioned. In a previous paper we introduced and numerically analyzed this approach in 2D in the context of RTM-based inverse scattering (Wang, de Hoop & Xia 2010). We include attenuation and 'acoustic' (VTI and HTI) anisotropy.

We present a (discrete) problem-specific massively parallel structured multifrontal solver in the setting of 3D regular Cartesian grids. In view of the chosen finite-difference stencil, nested dissection ordering of the grid can be conducted fast in $O(\log n)$ operations, where n is the number of grid points or variables. The separators are fixed layers of planes or lines. However, we can also extend the algorithm to separators of variable thickness. This extension enables higher accuracy and accommodates TTI (Tilted Transverse Isotropy) at the cost of increased (roughly a factor of 4 in 2D, 8 in 3D) communication. In the multifrontal factorization, dense intermediate matrices (called frontal and update matrices) are approximated by HSS matrices and are factorized with an *ULV* scheme. Related HSS algorithms are used (Chandrasekaran, Gu and Pals 2006; Xia *et al.* 2010; Chandrasekaran *et al.* 2006; Xia *et al.* 2009, 2010; Wang *et al.* 2010).

We obtain estimates for the associated computational complexity to be in the range from $O(n \log n)$ to $O(n^{4/3} \log n)$ flops, in a fully structured implementation. The solution cost is between $O(n)$ and $O(n \log n)$. In our current version, we implemented limited levels of compression in the HSS approximations of the dense intermediate matrices. The algorithm is highly parallelizable. We present a massively parallel implementation of the multifrontal algorithm together with HSS structures. The method has nice scalability and data locality, as illustrated by the numerical experiments. We compare the performance with the one of MUMPS on a common computational platform. We tested the performance of our algorithm for mid-range frequencies on models approaching the size of SEAM on the TeraGrid clusters.

In view of the sheer size of the matrix in the equation generated by discretizing the Helmholtz operator in 3D, most developments to date have been restricted to iterative solvers (Erlangga, Oosterlee and Vuik 2006; Riyanti *et al.* 2006; Plessix 2007; Riyanti *et al.* 2007). In general, iterative solvers lack the efficiency to deal with many forcing terms, and suffer from decreasing convergence rates with increasing frequency, though sophisticated preconditioners have been developed to address this issue in principle. Direct factorization methods have the natural advantage that the matrix factorization needs be carried out only once (per frequency), the factors being used in solving fastly the equation for multiple right-hand-sides (Operto *et al.* 2007, 2009).

In many problems following the discretization of linear partial differential equations from (geo)physics, the dense intermediate matrices in the direct solution also have the low-rank property. See e.g., Bebendorf and Hackbusch (2003) and Bebendorf (2005). Also, the solver is not restricted to a particular numerical method such as finite differences. We summarize how the approach developed in this paper directly applies to certain finite-element (Galerkin) discretizations of the Helmholtz equation. These form a natural setting for including discontinuities, for example.

The multi-frequency formulation of the seismic inverse problem aids in developing multi-scale regularization methods to mitigate its nonlinear nature. The Helmholtz-equation approach to FWI is computationally efficient for the low- to mid-frequency range, and can be viewed as complimentary to wave-packets based approaches which are efficient for the finer scales.

HELMHOLTZ OPERATOR DISCRETIZATION IN 3D

We consider the Helmholtz equation in three dimensions,

$$\left[-\rho(\mathbf{x})\,\frac{\partial}{\partial x}\left(\frac{1}{\rho(\mathbf{x})}\frac{\partial}{\partial x}\right)-\rho(\mathbf{x})\,\frac{\partial}{\partial y}\left(\frac{1}{\rho(\mathbf{x})}\frac{\partial}{\partial y}\right)-\rho(\mathbf{x})\frac{\partial}{\partial z}\left(\frac{1}{\rho(\mathbf{x})}\frac{\partial}{\partial z}\right)-\frac{\omega^2}{c^2(\mathbf{x})}\right]u(\mathbf{x},\omega)=f(\mathbf{x},\omega),\quad \mathbf{x}\in\mathbb{R}^3, \tag{1}$$

in which ρ is the density of mass, c is the compressional wavespeed, ω is the temporal frequency, f is the seismic forcing term, and u is the wavefield. We denote the Laplace operator part of the Helmholtz operator by $\Gamma = \Gamma(\mathbf{x}, \partial_{\mathbf{x}})$; $c^{-2}(\mathbf{x})\omega^2$ can be viewed as a potential. To model a seismic survey, (1) needs to be solved for many right-hand sides, which motivates the approach developed in this paper.

Simple acoustic anisotropy

We generalize the standard Helmholtz equation to an anisotropic form (Operto *et al.* 2009)

$$\begin{aligned}\Gamma(., \partial_{\mathbf{x}}) = {} & C_{xx}\ \rho\ \frac{\partial}{\partial x}\frac{1}{\rho}\frac{\partial}{\partial x} + C_{yy}\ \rho\ \frac{\partial}{\partial y}\frac{1}{\rho}\frac{\partial}{\partial y} + C_{zz}\ \rho\ \frac{\partial}{\partial z}\frac{1}{\rho}\frac{\partial}{\partial z} \\ & + C_{xy}\ \rho\ \frac{\partial}{\partial x}\frac{1}{\rho}\frac{\partial}{\partial y} + C_{yz}\ \rho\ \frac{\partial}{\partial y}\frac{1}{\rho}\frac{\partial}{\partial z} + C_{zx}\ \rho\ \frac{\partial}{\partial z}\frac{1}{\rho}\frac{\partial}{\partial x} \\ & + C_{yx}\ \rho\ \frac{\partial}{\partial y}\frac{1}{\rho}\frac{\partial}{\partial x} + C_{zy}\ \rho\ \frac{\partial}{\partial z}\frac{1}{\rho}\frac{\partial}{\partial y} + C_{xz}\ \rho\ \frac{\partial}{\partial x}\frac{1}{\rho}\frac{\partial}{\partial z} \\ & + C_{xyz}\ \rho\ \frac{\partial}{\partial z}\frac{1}{\rho}\frac{\partial}{\partial z}\left[\rho\ \frac{\partial}{\partial x}\frac{1}{\rho}\frac{\partial}{\partial x} + \rho\ \frac{\partial}{\partial y}\frac{1}{\rho}\frac{\partial}{\partial y}\right] \\ & + C_{zxy}\ \left[\rho\ \frac{\partial}{\partial x}\frac{1}{\rho}\frac{\partial}{\partial x} + \rho\ \frac{\partial}{\partial y}\frac{1}{\rho}\frac{\partial}{\partial y}\right]\rho\ \frac{\partial}{\partial z}\frac{1}{\rho}\frac{\partial}{\partial z}. \end{aligned} \tag{2}$$

Here, $C_{xx}, \ldots, C_{zxy}$ are coefficients which can be dependent on $\mathbf{x}$. The principal symbol of this Helmholtz operator can reproduce dispersion relaton for qP polarized elastic waves. For example, in the case of a medium with vertically transverse isotropy (VTI),

$$C_{zz} = 1, \quad C_{xx} = C_{yy} = 1 + 2\epsilon,$$
$$C_{xy} = C_{yx} = C_{yz} = C_{zy} = C_{zx} = C_{xz} = 0,$$
$$C_{xyz} = C_{zxy} = \epsilon - \delta,$$

while c is replaced by the qP wave velocity along the symmetry axis. In the case of a medium with horizontally transverse isotropy (HTI), we permute the z and x components or the z and y components. Here, $\epsilon = \epsilon(\mathbf{x})$ and $\delta = \delta(\mathbf{x})$ stand for Thomsen's parameters.

We introduce a perfectly matched layer (PML) (Turkel and Yefet 1998) contained in the computational domain, $[0, L_x] \times [0, L_y] \times [0, L_z]$ say: Let $0 < L_{x1} < L_x$, then the damping function S_x is defined as

$$S_x = S_x(x, \omega) = \begin{cases} 1 & \text{if } 0 \le x \le L_{x1}, \\ 1 - \mathrm{i}\dfrac{\sigma_0}{\omega}\cos^2\left(\dfrac{\pi}{2}\dfrac{x - L_{x1}}{L_x - L_{x1}}\right) & \text{if } L_{x1} < x \le L_x; \end{cases} \tag{3}$$

similar definitions hold for $S_y = S_y(y, \omega)$ and $S_z = S_z(z, \omega)$. Here, σ_0 is an appropriately chosen constant. The PML, or complex scaling, is incorporated by adjusting the partial derivatives: $\frac{\partial}{\partial x}$ is replaced by $\frac{1}{S_x}\frac{\partial}{\partial x}$ and similarly for the partial derivatives with respect to y and z. For example, the term

$$C_{xy}\ \rho\ \frac{\partial}{\partial x}\left(\frac{1}{\rho}\frac{\partial}{\partial y}\right) \quad \text{becomes} \quad C_{xy}\ \frac{\rho}{S_x}\ \frac{\partial}{\partial x}\left(\frac{1}{\rho S_y}\frac{\partial}{\partial y}\right).$$

We note that Γ becomes frequency dependent, that is, now $\Gamma = \Gamma(\mathbf{x}, \partial_{\mathbf{x}}, \omega)$.

Attenuation

Upon incorporating a PML, the Helmholtz operator is no longer self adjoint. It is, hence, natural to also consider attenuation, resulting in a complex-valued wavespeed. A common model used in seismic processing is given by

$$\frac{1}{\widehat{c}(\mathbf{x}, \omega)} = \frac{1}{c_r(\mathbf{x})}\left[1 + \frac{1}{\pi\, Q_r(\mathbf{x})}\ \log\left|\frac{\omega_r}{\omega}\right| + \mathrm{i}\frac{\mathrm{sgn}(\omega)}{2\, Q_r(\mathbf{x})}\right], \tag{4}$$

a non-causal approximation (with frequency-independent Q_r) to the models by Kolsky and Futterman; ω_r is a reference frequency. Here,

$$Q \approx Q_r + \frac{1}{\pi}\log\left|\frac{\omega_r}{\omega}\right|,$$

is frequency dependent. Essentially, such a model is obtained from the (causal) models by Futterman (1962) as $|\omega/\omega_r|$ becomes large, upon identifying

$$\frac{1}{c_0} = \frac{1}{c_r}\left[1 - \frac{1}{\pi\, Q_0}\ \log\left|\frac{\omega_r}{\omega}\right|\right]^{-1}, \qquad \frac{1}{Q_0} = \frac{1}{Q_r}\frac{c_0}{c_r}.$$

For comparison, a causal frequency-independent Q model was given by Kjartansson (1979),

$$\frac{1}{\widehat{c}(\mathbf{x}, \omega)} = a(\mathbf{x})\,|\omega|^{-\beta}\left[\cot\left(\frac{\pi\beta}{2}\right) + \mathrm{i}\,\mathrm{sgn}(\omega)\right], \quad \beta \in (0, 1),$$

with $Q \approx \cot(\pi\ \beta)$. Naturally β can be $\mathbf{x}$ dependent. For an overview of models, see Ursin and Toverud (2002).

Parsimonious mixed grid finite difference scheme

We follow the work of Operto *et al.* (2007) discussing a 27-point compact parsimonious mixed grid finite difference stencil. The corresponding scheme has been proven to be of at least 4^{th} order (Hustedt, Operto and Virieux 2004); in practice, this implies that a sampling rate of 5 grid points per wavelength yields sufficiently accurate results. The mixed grid stencil incorporates coordinate rotations, and hence numerical anisotropy is, to the given order, minimized. The conversion from subscripts to linear index is chosen to be,

$$\mathbf{u}_{(k-1)N_xN_y+(j-1)N_x+i}(\omega) = u(x_i, y_j, z_k, \omega),$$
$$x_i = (i-1)\,h_x, \quad y_j = (j-1)\,h_y, \quad z_k = (k-1)\,h_z,$$
$$i = 1, \ldots, N_x, \quad j = 1, \ldots, N_y, \quad k = 1, \ldots, N_z,$$

with $(N_x - 1)\,h_x = L_x$, $(N_y - 1)\,h_y = L_y$, $(N_z - 1)\,h_z = L_z$, N_x, N_y, $N_z \approx N$, which applies to f in a similar fashion, and cast the discretized Helmholtz equation in corresponding matrix

form:

$$\mathbf{A}(\omega)\,\mathbf{u}(\omega) = \mathbf{f}(\omega). \tag{5}$$

Naturally, the matrix $\mathbf{A}(\omega)$ is of size $\sim N^3 \times N^3$, $n = N^3$, and shares the same nonzero pattern for different values of ω. The matrix is non-definite, non-Hermitian, and poorly conditioned. Our approach addresses the complications associated with these properties.

MULTIFRONTAL METHOD WITH NESTED DISSECTION

For each ω, system (5) needs to be solved for many right-hand sides, while for different ω the matrix $\mathbf{A}(\omega)$ has the same nonzero pattern, say, block tri-diagonal. Hence, a direct solver is the natural method of choice. However, direct solvers are generally expensive due to catastrophic fill-in. For example, a straightforward factorization of $\mathbf{A}(\omega)$ in 3D costs $\mathcal{O}(n^{7/3})$ flops with $\mathcal{O}(n^{5/3})$ storage, where n is the size of $\mathbf{A}(\omega)$. In addition, one can not control the solution accuracy of an exact direct solver, even though we only need a mild solution accuracy about three or four digits for the Helmholtz problem.

Here, we discuss an approximate direct Helmholtz solver based on certain sparse matrix techniques and structured matrix methods, which enable us to approximately adapt the solution accuracy to a predefined tolerance via exploiting the low rankness, that is to say, we can control the solution accuracy. More details are given in Section 4.3. We first reorder the matrix with nested dissection, then we factorize the matrix with the multifrontal method where the intermediate dense matrices are approximated by structured matrices. This section is devoted to these sparse matrix techniques; in the next section we discuss the employment of structured matrices.

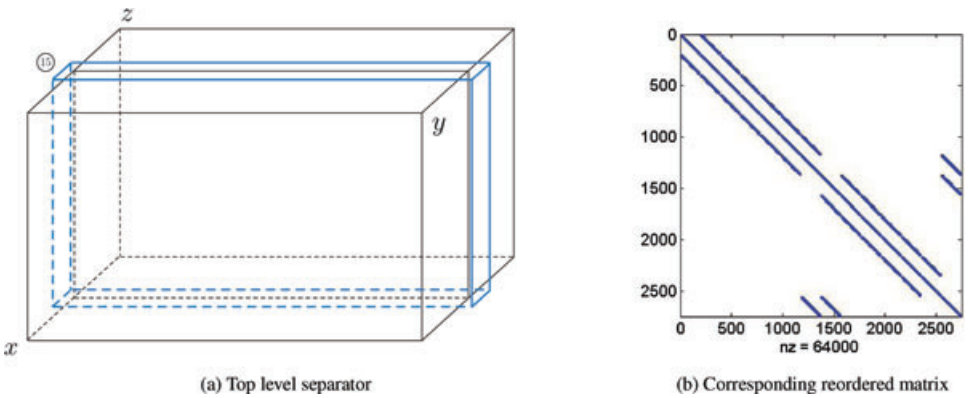
Matrix reordering with 3D nested dissection

To reduce fill-in, the matrix is to be reordered. Each unknown variable corresponds to a mesh point, so do each row and each column of $\mathbf{A}(\omega)$. The reordering of $\mathbf{A}(\omega)$ can be done in terms of the ordering of the mesh points. One of the most important reordering methods is nested dissection (George 1973; Liu 1992). With such reordering, the matrix $\mathbf{A}(\omega)$ in 3D can be factorized in $\mathcal{O}(n^2)$ flops with $\mathcal{O}(n^{4/3})$ storage.

Nested dissection recursively divides the mesh corresponding to $\mathbf{A}(\omega)$ into subdomains with separators (Teng 1997). A separator is defined as a set of mesh points the removal of which divides the mesh into two disjoint subregions.

At the top level or level 0, a separator divides the original mesh into three parts: two subdomains and the separator itself. The grid points associated with the separator are ordered after those associated with the subdomains. The separator and the reordered matrix are shown in Fig. 1.

Each subdomain is then recursively ordered following the same pattern. This is illustrated in Fig. 2. Lower level separators are ordered before upper level ones. After reordering of $\mathbf{A}(\omega)$, the matrix pattern is illustrated in Fig. 3 (left–middle). Moreover, we note that each separator in 3D mesh is a 2D plane. In the process of lower level partition, this plane also manifests a substructure of 2D nested dissection. For example, in Fig. (2), if we focus on the z direction separator, we note this separator has 2D substructure when y and x direction partitions take place. Therefore, we incorporate a 2D nested dissection reordering within each separator for 3D problems. The other reason that we adopt such a 2D nested dissection for each separator is the numerical rank will be lowered (Chandrasekaran, Dewilde and Somasunderam 2010), which we will discuss in detail in Section 4. Nested dissection at these two levels, an outer level and an inner level, forms the overall



(a) Top level separator (b) Corresponding reordered matrix

Figure 1 A top-level separator divides the mesh into subregions and is then ordered after the subregions.

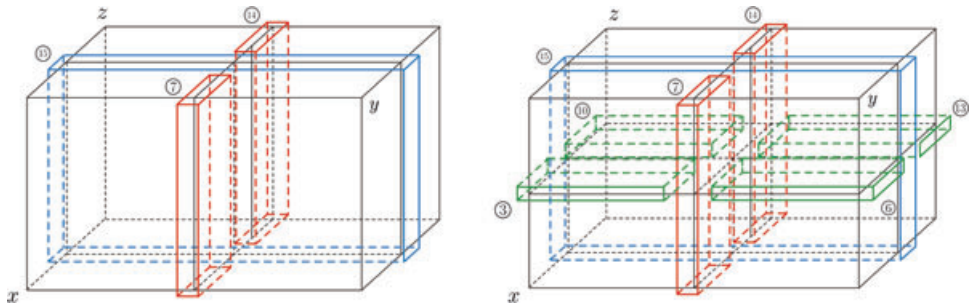


Figure 2 Two more levels of separators and partitions in 3D nested dissection.

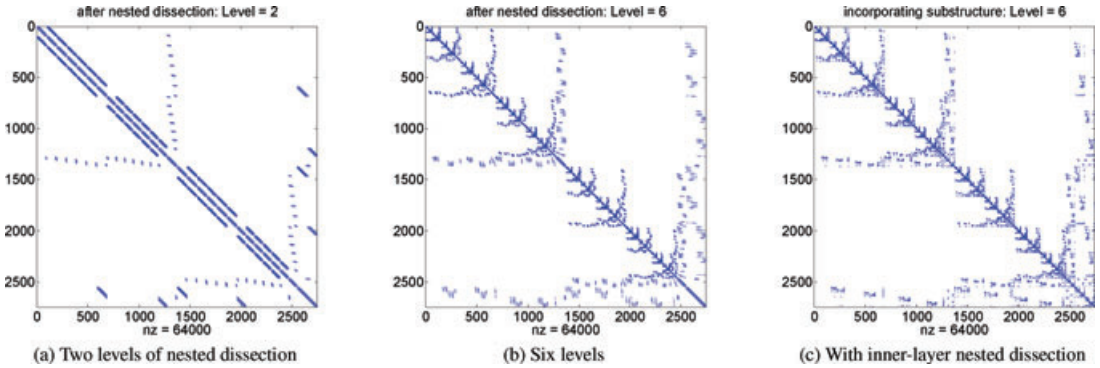


Figure 3 The nested dissection reordering of the global matrix $\mathbf{A}(\omega)$, where (c) shows both the inner- and outer-layer nested dissection ordering.

ordering. The corresponding matrix pattern is shown in Fig. 3 (right). (It is shown in Hoffman, Martin and Rose (1973)) that for 2D problems, nested dissection reordering provides the optimal complexity for *LU* factorizations among all the ordering techniques.)

After nested dissection, the factorization of the reordered matrix can be conducted following the traversal of a binary tree. Each node of this tree represents an (outer-level) separator. See Fig. 4. (Later by a separator, we mean an outer-level separator unless otherwise specified.)

Factorization with the multifrontal method

The factorization is arranged with the multifrontal method (Duff and Reid 1983; Liu 1992), which reorganizes the overall factorization into partial updates and factorizations of smaller

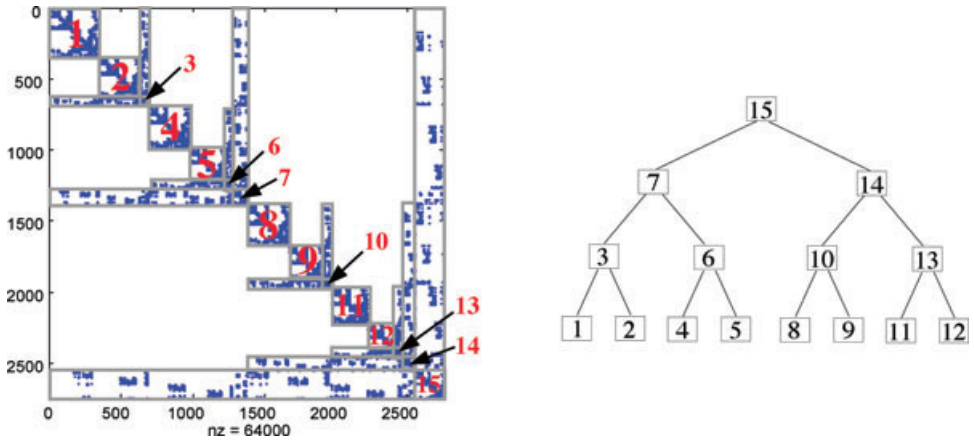


Figure 4 Reordered $\mathbf{A}(\omega)$ as in Fig. 3(c) and the tree structure for the elimination.

dense matrices according to a tree structure called *elimination tree* or *assembly tree* (Liu 1992; Gilbert 1993; Liu 1990; Eisenstat and Liu 2005). The method has nice data locality and is suitable for parallelization. Here, after the nested dissection ordering, we can treat the separator tree (Fig. 4) as the assembly tree.

Let i index a separator, and $\mathcal{N}_i$ be the set of upper level neighbor separators of i. Here, the neighbors are connect to i due to the fact the elimation of lower level separators mutually connects the neighbors of these lower level separators. We use $A_{i,j}$ to denote the matrix entries of $\mathbf{A}(\omega)$ that have row and column indices given by the grid points in separators i and j, respectively.

Then, if i is a leaf (bottom level nodes) in the assembly tree, we define a matrix

$$\mathcal{F}_i = \begin{pmatrix} A_{i,i} & A_{i,\mathcal{N}_i} \\ A_{\mathcal{N}_i,i} & 0 \end{pmatrix}.$$

The elimination of separator i corresponds to the partial factorization

$$\mathcal{F}_i = \begin{pmatrix} L_{i,i} & 0 \\ A_{\mathcal{N}_i,i}U_{i,i}^{-1} & I \end{pmatrix} \begin{pmatrix} U_{i,i} & L_{i,i}^{-1}A_{i,\mathcal{N}_i} \\ 0 & \mathcal{U}_i \end{pmatrix}, \tag{6}$$

where $\mathcal{U}_i = -(A_{\mathcal{N}_i,i}U_{i,i}^{-1})(L_{i,i}^{-1}A_{i,\mathcal{N}_i})$ is the Schur complement.

If i is a non-leaf node, we assume that its child nodes c_1 and c_2 have been eliminated so that the update matrices $\mathcal{U}_{c_1}$ and $\mathcal{U}_{c_2}$ are obtained. Then we form a matrix as

$$\mathcal{F}_i = \begin{pmatrix} A_{i,i} & A_{i,\mathcal{N}_i} \\ A_{\mathcal{N}_i,i} & 0 \end{pmatrix} \oplus \mathcal{U}_{c_1} \oplus \mathcal{U}_{c_2},$$

where the symbol $\oplus$ denotes an assembly operation called *extend-add* (Liu 1992) which reorders and adds matrix entries according to the grid points that the entries correspond to. Partitioning $\mathcal{F}_i$ conformably and carrying out a partial LU factorization, yields

$$\mathcal{F}_i = \begin{pmatrix} F_{i,1,1} & F_{i,1,2} \\ F_{i,2,1} & F_{i,2,2} \end{pmatrix} = \begin{pmatrix} L_{i,i} & 0 \\ F_{i,2,1}U_{i,i}^{-1} & I \end{pmatrix} \begin{pmatrix} U_{i,i} & L_{i,i}^{-1}F_{i,1,2} \\ 0 & \mathcal{U}_i \end{pmatrix},$$

where the Schur complement or the update matrix is given by

$$\mathcal{U}_i = F_{i,2,2} - \left(F_{i,2,1}U_{i,i}^{-1}\right)\left(L_{i,i}^{-1}F_{i,1,2}\right).$$

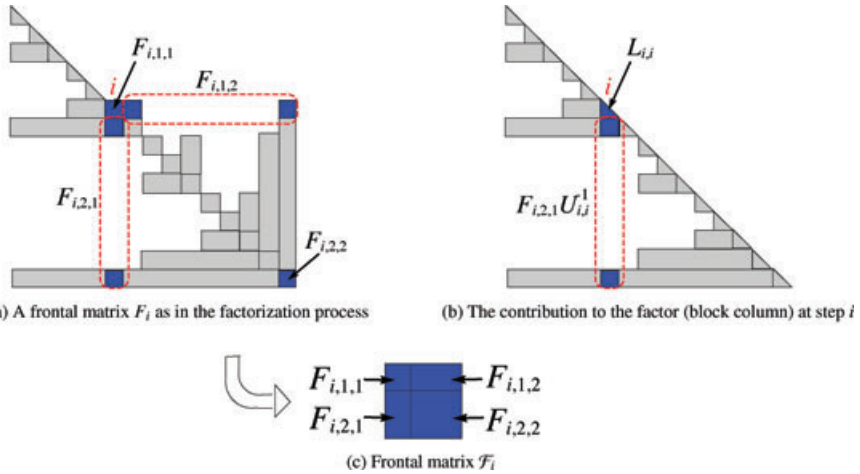
This is the process of eliminating separator i. The process then repeats until the root separator is eliminated. Here, the matrices $\mathcal{F}_i$ and $\mathcal{U}_i$ for a (leaf or non-leaf) node i are called a *frontal matrix* and an *update matrix*, respectively (Liu 1992). The update matrix $\mathcal{U}_i$ represents the contribution from node i to its parent.

The overall factors are given by the matrices $L_{i,i}$, $U_{i,i}$, $A_{\mathcal{N}_i,i}U_{i,i}^{-1}$, $L_{i,i}^{-1}A_{i,\mathcal{N}_i}$ as in equation (6) for all separators i; see Fig. 5. Unlike the classical LU factorization, such a factorization method is a left-looking method (Bai *et al.* 2000), where there is no global Schur complement computation.

The solution with substitution consists of two stages, a forward stage and a backward one:

$$Ly = b, \quad Ux = y.$$

The forward and backward substitutions correspond to the bottom-up and top-down traversals of the assembly tree, respectively. For example, in the forward stage, associated with



(a) A frontal matrix F_i as in the factorization process

(b) The contribution to the factor (block column) at step i

(c) Frontal matrix $\mathcal{F}_i$

Figure 5 A frontal matrix in the elimination, and the corresponding block column in the triangular factor.

each node of the assembly tree, we solve a system of the form

$$\begin{pmatrix} L_{i,i} & 0 \\ L_{\mathcal{N}_i,i} & I \end{pmatrix} \begin{pmatrix} y_i \\ \widetilde{b}_{\mathcal{N}_i} \end{pmatrix} = \begin{pmatrix} \widetilde{b}_i \\ b_{\mathcal{N}_i} \end{pmatrix}.$$

Thus,

$$y_i = L_{i,i}^{-1}\widetilde{b}_i, \quad \widetilde{b}_{\mathcal{N}_i} = b_{\mathcal{N}_i} - L_{\mathcal{N}_i,i}\, y_i,$$

where $\widetilde{b}_i$ and $\widetilde{b}_{\mathcal{N}_i}$ represent pieces of the updated right-hand side during the substitution, and $\widetilde{b}_{\mathcal{N}_i}$ is used in later solution steps associated with the nodes in $\mathcal{N}_i$.

Pivoting is also included in the frontal matrix $\mathcal{F}_i$ factorization stages. As for the global matrix $\mathbf{A}(\omega)$, we observed no global pivoting is needed based on the nested dissection strategy we adopted for Helmholtz problems. However, we admit that global pivoting can be conducted the similar way as sparse direct solver packages like MUMPS or SuperLU.

Speedup thanks to the regularity of 3D mesh geometry

The mesh in our problem is regular, which means that the mesh can be represented in a Cartesian fashion; the outer- and inner-level separators are planes and lines, respectively. In this case, nested dissection can be efficiently implemented. Indeed, we can straightforwardedly pinpoint the locations of separators according to the regularity of the mesh geometry. For instance, let us initiate the partitioning of the domain from z direction (see Fig. 1); the first level separator can be fully represented by a fixed value of z, yielding that the two disjoint subdomains can be uniquely represented by two end vertices on their diagonals respectively. Then we move on to the second level of nested dissection along y direction (see Fig. 2). The resulting four subdomains can also be uniquely represented by two end vertices on their diagonals, respectively. We follow the same principle up to a prescribed level, yielding that the separators living on each level are uniquely represented. Additionally, the subdomains living on the bottom level (leaf nodes of the entire assembly tree) are uniquely determined by two end vertices of their diagonals.

We note that the number of nodes on each level of the assembly tree is always a certain power of 2, due to the fact that two subdomains are generated out of one separator according to the regularity of the mesh. This guarantees that the assembly tree formed out of the 3D regular mesh is always a complete binary tree. The nested dissection process can be summarized as the following:

`subroutine nd3d(mesh(1:`N_x`, 1:`N_y`, 1:`N_z`))`
  find the axis $a(=x, y,$ or $z)$ according to max (N_x, N_y, N_z), and let the other two axes be b and c
  choose the plane with cordinate $N_1 = \lfloor \frac{N_a+1}{2} \rfloor$ in the a direction to be the (outer) separator
  `call nd2d(mesh(1:`N_b`, 1:`N_c`))`
  `call nd3d(mesh(1:`$N_1 - 1$`, 1:`N_b`, 1:`N_c`))`
  `call nd3d(mesh(`$N_1 + 1$`:`N_a`, 1:`N_b`, 1:`N_c`))`
  `end subroutine`

`subroutine nd2d(mesh(1:`N_x`, 1:`N_y`))`
  find the axis $a(=x$ or $y)$ according to max (N_x, N_y), and let the other axis be b
  choose the line with cordinate $N_1 = \lfloor \frac{N_a+1}{2} \rfloor$ in the a direction to be the (inner) separator
  `call nd2d(mesh(1:`$N_1 - 1$`, 1:`N_b`))`
  `call nd2d(mesh(`$N_1 + 1$`:`N_a`, 1:`N_b`))`
  `end subroutine`

The total complexity for the regular mesh nested dissection described above is at most $\mathcal{O}(\log^2 n)$, since there are $\mathcal{O}(\log n)$ outer levels of partition and at most $\mathcal{O}(\log n)$ coordinates to be visited at each level. In comparison, the nested dissection cost for a general mesh is at least $\mathcal{O}(n)$ (Lipton and Tarjan 1979; Teng 1997). We gain a significant speedup $\mathcal{O}(n)/\mathcal{O}(\log^2 n)$ at the stage of nested dissection by exploiting this mesh regularity, which will be further demonstrated in Section 6.

After the nested dissection, we conduct the traversal of the assembly tree to carry out the factorization with the multifrontal method. The conventional assembly tree is not necessarily a complete binary tree, which is why the load balance is highly impacted by the configuration of the assembly tree. However, in our problem, we are guaranteed that the assembly tree is always a complete binary tree. Thus the entire parallel task scheduling is partitioned into local factorization and global factorization with a two-two communication prototype. The load balance is perfect. We gain another speedup at the factorization stage by exploiting the completeness of the assembly tree. In Section 5 we present the details.

FRONTAL AND UPDATE MATRICES: APPROXIMATION BY STRUCTURED MATRICES

The efficiency of the multifrontal method can be further improved by using structured matrix methods. It has been observed by researchers that, during the direct factorization of sparse discretized matrices from various partial differential equations problems, certain off-diagonal blocks of the

intermediate dense matrices have small numerical ranks (Bebendorf and Hackbusch 2003; Bebendorf 2005; Chandrasekaran *et al.* 2010). Such a *low-rank property* is also observed in the direct solution of Helmholtz problems (Engquist and Ying; Wang *et al.* 2010). In the multifrontal method for Helmholtz equations in 2D, the off-diagonal numerical ranks of the frontal and update matrices are discussed in Wang *et al.* (2010).

We leave the discussion of the low-rank property for Helmholtz problems in 3D to Subsection 4.2. We first discuss the way to take advantage of the low-rank property. That is, we approximate the dense intermediate frontal matrices $\mathcal{F}_i$ and update matrices $\mathcal{U}_i$ in the multifrontal method by structured matrices. Here, we review the basic ideas following the discussions in (Xia *et al.* 2009, 2010).

HSS structured approximations and HSS operations

We can approximate a dense matrix $\mathcal{F}$ whose off-diagonal blocks have small numerical ranks by rank-structured matrices such as $\mathcal{H}$-matrices (Hackbusch 1999; Hackbusch and Börm 2002; Hackbusch and Khoromskij 2000), $\mathcal{H}^2$-matrices (Börm *et al.* 2003; Börm and Hackbusch; Hackbusch and Sauter 2000), semiseparable matrices (Chandrasekaran *et al.* 2006, 2005) and quasiseparable matrices (Eidelman and Gohberg 1989). Here, we use hierarchically semiseparable (HSS) structures (Chandrasekaran *et al.* 2006; Xia *et al.* 2010; Chandrasekaran *et al.* 2006; Xia *et al.* 2009), where the off-diagonal blocks are hierarchically represented by compressed forms. The block partition corresponds to a binary tree structure, as shown in Fig. 6.

The HSS structure is an effective way to study the low-rank property. Many efficient and stable HSS algorithms have been proposed. In this paper, we are particularly interested in two important HSS procedures: parallel HSS structure generation and parallel *ULV*-type HSS factorization (Xia *et al.* 2010).

The first procedure is to approximate a dense matrix $\mathcal{F}$ by an HSS form, where the off-diagonal blocks of $\mathcal{F}$ have small numerical ranks. This procedure uses a fundamental operation called *compression*. That is, we compute a compact form $B \approx QS$ for an off-diagonal block B with methods such as truncated SVD and rank-revealing QR factorizations (Xia *et al.* 2010). To construct an HSS representation, we hierarchically compress the off-diagonal blocks. This is done in parallel by hierarchically compressing all block rows at all levels, and then all block columns.

The next procedure is to compute an *ULV*-type factorization of the HSS matrix (Xia *et al.* 2010). Using the HSS form, we efficiently introduce zeros into the off-diagonal blocks and partially eliminate the diagonal blocks. The remaining blocks are then merged and eliminated recursively.

If the order of $\mathcal{F}$ is K, then the HSS construction and ULV factorization cost $\mathcal{O}(rK^2)$ and $\mathcal{O}(rK)$ flops, respectively, where r is the maximum numerical rank of all related off-diaogonal blocks. Moreover, according to (Xia), the numerical ranks of the off-diagonal blocks at different levels can be allowed to increase as a function $r(K_i)$, where K_i is the row dimension of the off-diagonal block rows. For example, if $r(K_i) = \mathcal{O}(\log K_i)$, the HSS construction and factorization costs are still $\mathcal{O}(K^2)$ and $\mathcal{O}(K)$, respectively. If $r(K_i) = \mathcal{O}(K_i^{1/2})$, then the costs are $\mathcal{O}(K^2 \log K)$ and $\mathcal{O}(K^{3/2})$ respectively.

Structured multifrontal solution of 3D Helmholtz problems

Here, we summarize our implementation of the structured multifrontal method where the intermediate dense Schur complements are approximated by HSS forms (Xia *et al.* 2009). At certain lower levels of the assembly tree, we use standard dense LU factorizations for the frontal matrices. After a switching level, structured factorizations are used. There are three stages. The first stage is to approximate a frontal matrix $\mathcal{F}_i$ by an HSS matrix, using the HSS construction algorithm. The

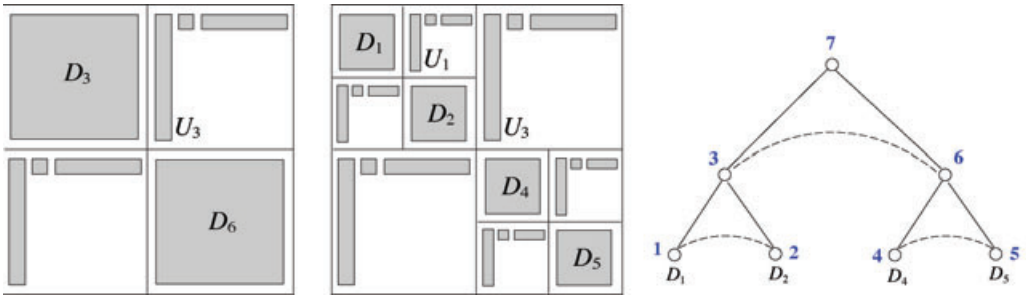


Figure 6 Pictorial representations of an HSS matrix in terms of two levels of generators.

second stage is to partially factorize $\mathcal{F}_i$. Let

$$\mathcal{F}_i = \begin{pmatrix} F_{i,1,1} & F_{i,1,2} \\ F_{i,2,1} & F_{i,2,1} \end{pmatrix},$$

where the partition is conformable to equation (6), and $F_{i,1,1}$ corresponds to $L_{i,i}\, U_{i,i}$ in the exact factorization. Here, we replace equation (6) by an *ULV* factorization which fully eliminates $F_{i,1,1}$, but not $F_{i,2,1}$. Then $F_{i,1,2}$ and $F_{i,2,1}$ will also be updated to data-sparse representations, which will be used to efficiently compute the Schur complement or update matrix $\mathcal{U}_i$. In the third stage, the extend-add operation can remain dense and is thus the same as in the exact factorization. This can be improved by keeping $\mathcal{U}_i$ in a data-sparse form, since $\mathcal{U}_i$ can be understood as a sparse matrix plus low-rank updates.

In general, we can partition $\mathcal{F}_i$ into arbitrary number of diagonal and off-diagonal blocks, which implies arbitrary level of HSS structure. In this current work, we implemented an HSS structured form where $\mathcal{F}_i$ has two levels of partition, that is, $F_{i,1,1}$ has been partitioned into two diagonal blocks together with off-diagonal blocks.

Complexity and accuracy

During the factorization of discretized partial differential equations, such as elliptic and Helmholtz problems (Chandrasekaran *et al.* 2010; Engquist and Ying), the intermediate dense Schur complements have small numerical ranks, and the structured multifrontal solver can be very efficient. However, the maximum off-diagonal numerical rank may also be as large as $\mathcal{O}(N)$ for 3D problems, where the mesh is assumed to be $N \times N \times N$. This is true for elliptic equations (Chandrasekaran *et al.* 2010). We expect a similar result to hold also for Helmholtz equations, although a theoretical justification is not yet available. If this holds, we can still use the multifrontal method where the dense frontal and update matrices are approximated by HSS matirces. Then it can be shown that the total cost of such a structured multifrontal method is between $\mathcal{O}(n \log n)$ and $\mathcal{O}(n^{4/3} \log n)$, depending on the implementation (Xia), where $n = N^3$. Here, the nearly optimal (factorization) complexity is achieved if the bottom level standard *LU* factorization cost is equal to the upper level structured factorizations. The cost of the solution with substitution is between $\mathcal{O}(n)$ and $\mathcal{O}(n \log n)$. The memory requirement is about $\mathcal{O}(n \log n)$. We notice that the solution cost is roughly linear in n, which is especially important for our problem where many right-hand sides present.

The performance of the structured factorization is observed to be relatively insensitive to the frequency, as discussed in (Wang *et al.* 2010). One reason is that we allow the numerical rank to vary in a large range while achieving the similar order of complexity. In comparison, for iterative methods, it usually requires effective preconditioners to enable fast convergence, which is often difficult. The convergence often depends on the condition number, which becomes large especially for relative high frequency Helmholtz problems (Plessix 2007).

The accuracy of the structured solver depends on the compression used in HSS construction. In conventional HSS operations (Chandrasekaran *et al.* 2006; Xia *et al.* 2010), the solution accuracy (forward error) is roughly of the same order as the compression accuracy in the HSS construction, even though a theoretical justification of the dependence of the solution accuracy on the compression accuracy for the Helmholtz problem is not available (The errors for both accuracies are measured in the spectral norm). This is also observed to hold for the sparse structured multifrontal solution, where the solution accuracy is close to the tolerance in the HSS construction for the frontal matrices. For instance, factorizing a 3D mesh with $n = 512^3$ on 256 computing nodes with 32 GB per node, compression dictated by memory limitations yields an accuracy of 3 to 4 digits in the matrix solutions. Such an accuracy is in concert with requirements in seismic applications.

ASPECTS OF THE MASSIVELY PARALLEL IMPLEMENTATION

In this section, we discuss the parallel implementation of the structured multifrontal method introduced in Sections 3 and 4 in the context of solving the Helmholtz equation.

During the process of 3D domain decomposition of the regular mesh, the assembly tree, which governs the entire multifrontal factorization and solution processes, is formed top-down (Fig. 7). Each node on the assembly tree represents a certain region of the entire 3D mesh. The key observation here is the regularity of the 3D mesh guarantees that the assembly tree is a complete binary tree, which implies the number of nodes on each level of the assembly tree is a certain power of 2, and hence guarantees the load balance of the parallelization, provided that the number of processors *Nprocs* is also a certain power of 2. Therefore, we have the following definition: the parallel level l_p is defined as the level on which the number of nodes is equal to the number of processors (*Nprocs*) in the assembly tree: $l_p = \lfloor \log_2(Nprocs) \rfloor + 1$.

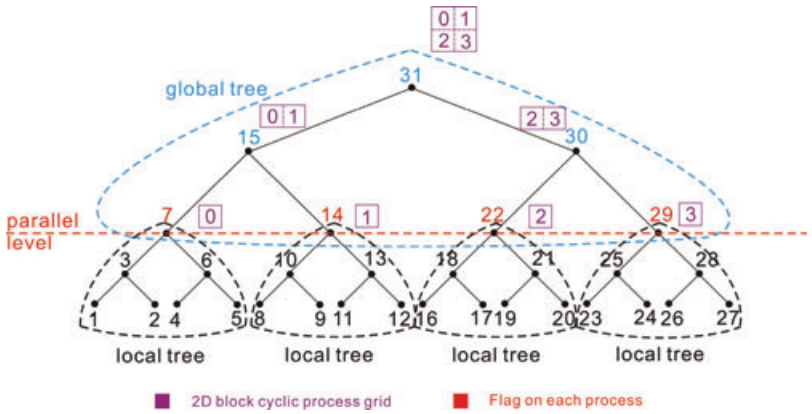


Figure 7 Illustration of the 3D massively parallel multifrontal method. The red dashed line indicates the level (l_p) that parallelization comes into play. The entire assembly tree, shown in Fig. (4) (right), is partitioned into two parts. The `first` part is the local tree stored in each individual processor, indicated by those separate dashed boxes below the red dashed line. These local trees are eliminated first locally, resulting in a dense update matrix eventually. The second part is the global tree stored in all processors, indicated by a big dashed box above the red dashed line. Two-two communication prototypes only happen at this stage. In the second part, we adopt the 2D-block-cyclic strategy in `Scalapack` to create sub-process grids on each node of the global tree.

The introduction of the parallel level, l_p, yields that the assembly tree can be partitioned into several local trees which are stored in each processor separately, and a global tree which is stored on all processors jointly. Figure 7 illustrates an example of an assembly tree, and how the entire assembly tree is partitioned into a global tree and local trees. It illustrates how we implement the multifrontal method in the context of massive parallelization. Below the parallel level, each processor eliminates a local subtree in parallel, ending up with a local update matrix which will participate in the later on extend-add operation. Up to this stage, no communications occur between processors.

Above the parallel level, the frontal matrix will be stored and factorized over multiple processors. We observe that the communication possesses a unique feature, *two-two communication* prototype, which means that each node on the global tree only communicates with its sibling node to form and factorize its parent frontal matrix, thanks to the completeness of the assembly tree, thus the global tree. This also yields that at each node of the global tree, only those processors associated with its descendants down to the parallel level l_p will participate in its factorization process. Additionally, the number of processors which participate in the factorization of each node in the global tree is also guaranteed to be a certain power of two. This two-two communication prototype indicates that we can construct *sub-communicators* (or *sub-contexts* in `Scalapack` (www.netlib.org/scalapack)) out of the entire communicator, say, `MPI_COMM_WORLD`, at each node of the global tree. For example, the frontal and update matrices associated with node 15 will be stored in processor 0 and 1 associated with node 7 and 14 respectively. Communications only happen between processors 0 and 1. However, the frontal matrix associated with node 31 will be stored in four processors, 0, 1, 2, and 3. We note that the root node involves all the processors in the final stage of the factorization. Therefore, in summary, at the stage of global tree factorization, we

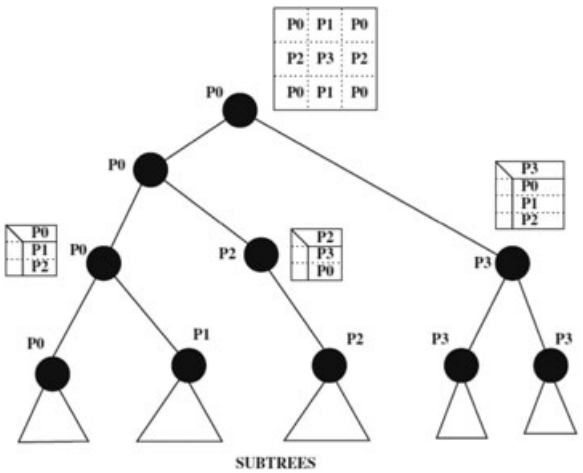


Figure 8 Parallel task scheduling assembly tree adopted by MUMPS. Figure courtesy of (Agullo *et al.* 2008). *P*0, *P*1, *P*2 means the processor 0, 1 and 2. As you can see, the assembly tree inside MUMPS is not guaranteed to be a complete binary tree. MUMPS only takes advantage of 2D-block-cyclic data distribution at the root node of the assembly tree, and use master-slave prototype for the remainder nodes.

Table 1 The benchmark comparison tests with MUMPS on a 100 × 100 × 100 mesh, on the same computing platform: 64 cores on `coates.rcac.purdue.edu`. $T_{analysis}$ is the wall time for nested dissection; T_{fact} is the wall time for factorization after nested dissection; T_{sol} is the wall time for the solution for one RHS after factorization

	$T_{analysis}(s)$	$T_{fact}(s)$	$T_{sol}(s)$
MUMPS	19.05	704.17	9.46
Hsolver	0.002	100.45	0.25

traverse the global tree bottom-up level-wise instead of node-wise. No processor is idle, and hence the perfect load balance is achieved thanks to the completeness of the assembly tree, and thus thanks to the regularity of the mesh geometry.

We utilize the coding libraries from `MKL` (Math Kernel Library), which contains `BLAS` (Basic Linear Algebra Subprograms) (www.netlib.org/blas), `PBLAS` (Parallel Basic Linear Algebra Subprograms) (www.netlib.org/scalapack), `BLACS` (Basic Linear Algebra Communication Subprograms) (www.netlib.org/blacs), `Lapack` (Linear Algebra Package) (www.netlib.org/lapack), and `Scalapack` (Scalarized Lapack) (www.netlib.org/scalapack).

In the local tree eliminations on each processor we can take advantage of `BLAS` and `Lapack`, because there is no communication at this stage. After the frontal matrix in Figure 5 is formed via the extend-add operation, we use the subroutine `CGETRF` to factorize the block $F_{i,1,1}$. Afterwards, the blocks $F_{i,1,2}$ and $F_{i,2,1}$ are updated via calling the subroutine `CTRTRS`. Then, we use `BLAS3` subroutine `CGEMM` to update the Schur complement block $F_{i,2,2}$. There is one final update matrix stored on each processor after the local tree is completely eliminated.

In the global tree elimination, we take advantage of `BLACS`, `PBLAS` and `Scalapack`, because of the inherent massive communications at this stage. To be specific, we use the `BLACS` 2D-block-cyclic data distribution features, which have been proved to attain optimized load balancing and minimized overhead. In `BLACS`, any matrix is assumed to be laid out on a 2D process grid, which is associated with a *context*. This context creates a sub-communicator out of `MPI_COMM_WORLD`. In the multifrontal method, the parent frontal matrix is generated out of two update matrices of its two children. Thus only two contexts associated with its two children participate in forming the parent frontal matrix. Each node on the global tree has a context associated with it (see Fig. 7). We use `BLACS_GET` and `BLACS_GRIDMAP` to generate such contexts on each node. Because the parent context is the union of two children's contexts, the update matrices stored on each child's context should be re-distributed before the extend-add operation. We use a particular `Scalapack` subroutine, `PCGEMR2D`, to carry out such a parallel extend-add operation. We use the parallel subroutines `PCGETRF`, `PCTRTRS` and `PCGEMM` to partially factorize the frontal matrix and generate the update matrix on each context. We note that contexts on the same level of the global tree are conducted in parallel.

Table 2 The comparison of CPU wall time between MUMPS and Hsolver on a 80 × 80 × 80 mesh, for an increasing number of processors

Nprocs	1	2	4	8	16	32	64
MUMPS(s)	261.65	361.94	391.84	505.98	471.56	415.88	375.88
Hsolver(s)	960.77	491.76	275.93	156.22	93.03	53.09	45.43

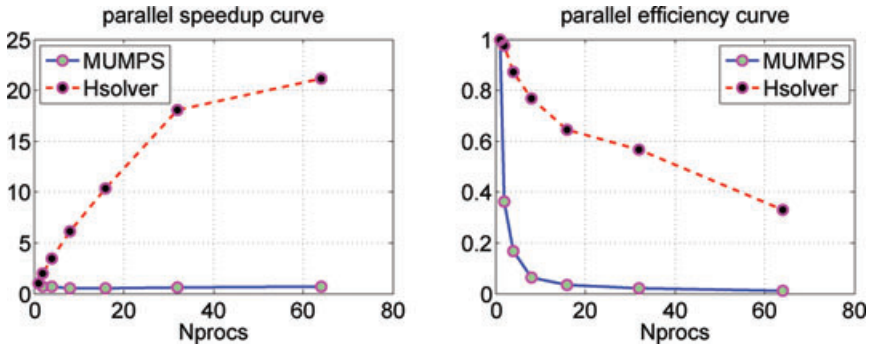


Figure 9 Left: the parallel speedup curve; right: the parallel efficiency curve. The data is from Table 2.

Table 3 The factorization complexity, solution complexity and storage comparison between the uncompressed and the compressed versions of the Hsolver. The preset tolerance for Hsolver with compression is 10^{-4}

N	32^3	64^3	128^3	256^3
flops_fact ($\times 10^{10}$)	1.43	91.13	5955.77	386370.81
flops_fact_compr ($\times 10^{10}$)	1.13	53.98	2185.47	76862.98
flops_sol ($\times 10^{8}$)	0.30	4.86	78.17	1292.93
flops_sol_compr ($\times 10^{8}$)	0.23	3.15	38.20	407.32
storage ($\times 10^{8}$)	0.23	3.79	63.05	1025.59
storage_compr ($\times 10^{8}$)	0.19	2.66	32.48	378.28

In comparison, MUMPS does not take advantage of the regularity of the mesh, because its input is the matrix $\mathbf{A}(\omega)$ instead of mesh geometry. It does take advantage of 2D-block-cyclic data distribution only at the root node of the assembly tree. However, for the remainder of the nodes, it adopts the master-slave prototype for the task scheduling. Figure 8 (Agullo *et al.* 2008) illustrates that a general purpose sparse direct solver such as MUMPS might have to handle elimination trees that are not completely binary, depending on the sparsity pattern and reordering strategy selected for a general sparse matrix. Even if a nested dissection reordering is applied in advance to a sparse matrix arising from a 3D 27-point stencil, the

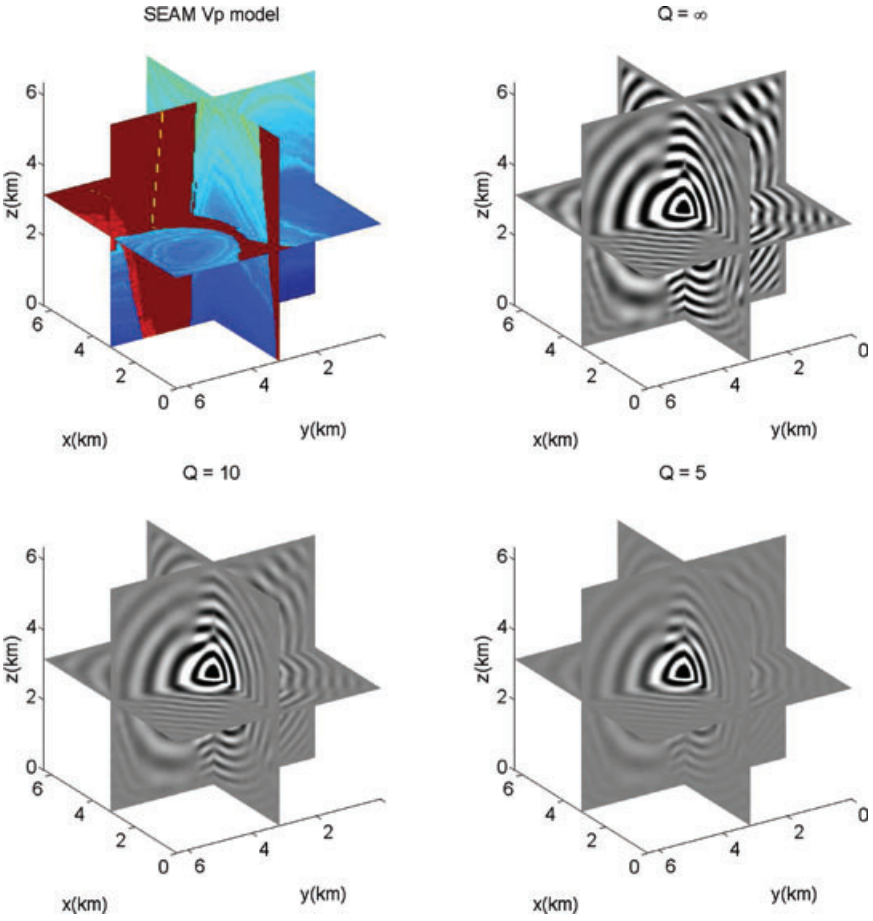


Figure 10 Part of the SEAM velocity model on a 256 × 256 × 256 mesh and its corresponding 7.5 Hz time-harmonic acoustic wavefields with attenuation effects. The mesh step size is $h_x = h_y = h_z = 25$ m. The physical domain is 0-6.4 km × 0-6.4 km × 0-6.4 km. The source location is at $\mathbf{x}_s = (3.15, 3.15, 3.15)$ km. Upper left: partial SEAM velocity model; upper right: 7.5Hz time-harmonic wavefield with quality factor $Q = \infty$; lower left: 7.5Hz time-harmonic wavefield with quality factor $Q = 10$; lower right: 7.5Hz time-harmonic wavefield with quality factor $Q = 5$. The computation is conducted on 16 nodes (128 cores) on `coates.rcac.purdue.edu`.

elimination tree might not be as well-balanced as the one resulting from a 'hand-made' geometrical nested dissection of a rectangular mesh.

PERFORMANCE TESTS AND EXAMPLES

Performance

We start with a set of benchmark tests comparing the performance of our solver with the one of MUMPS. The following experiments are all conducted on the same platform (`coates.rcac.purdue.edu`); we use only 8 nodes, each one with a dual 2.5 GHZ quad-core AMD 2380 and 32 GB memory. To make comparisons fair, we input the same nested dissection reordering arising from our Helmholtz solver to MUMPS.

We compute the 3D solution for $N = 100, n = 10^6$. The CPU wall time is listed in Table 1. $T_{analysis}$ is the wall time for nested dissection which is also referred to as the analysis stage; T_{fact} is the wall time for factorization following the nested dissection; T_{sol} is the wall time for the solution for one RHS after factorization. From the results we conclude that our Helmholtz solver, which we will refer to as Hsolver, is at least one order of magnitude faster than MUMPS at the analysis, factorization and solution stages. We observe that there is a significant difference in $T_{analysis}$; this can be attributed to the fact that MUMPS still conducts the analysis stage internally although the same nested dissection reordering is input to both MUMPS and Hsolver. The factorization and solution stages are hence strongly impacted by the analysis stage.

Secondly, we fix a 3D $N = 80$ mesh, and let the number of processors increase from 1 to 64. CPU wall times are recorded in Table 2. We plot the speedup curve in Fig. 9 (left), and the efficiency curve in Fig. 9 (right). We note that both the parallel speedup and the efficiency of Hsolver is better than that of MUMPS.

We can gain about a factor 5 speedup by blocking together a certain number of right-hand-sides at the parallel solution stage, using `BLAS3` (rather than `BLAS2`).

Thirdly, we show the comparison of the factorization complexity, solution complexity and storage between the compressed and uncompressed versions of Hsolver in Table 3. The preset tolerance for Hsolver with compression is 10^{-4}. We note that the statistics reflect the theoretical complexities.

SEAM

In this subsection, we show some numerical examples using the 3D SEAM (SEG Advanced Modeling) velocity model.

The first example is the modeling with attenuations. Figure 10 displays part of the SEAM *P*-wave velocity model on a 256 × 256 × 256 mesh and its corresponding 7.5Hz time-harmonic acoustic wavefield with attenuation effects. The mesh step size is $h_x = h_y = h_z = 25m$, whence the physical domain is 0-6.4 km × 0-6.4 km × 0-6.4 km. The source location is at $\mathbf{x}_s = (3.15, 3.15, 3.15)$ km. The number of unknowns is over 16 million. We use 16 nodes (8 cores per node, 4GB per core) to carry out this computation. The CPU wall time for the factorization is 1362 seconds, and the CPU time for solution for one RHS is 7.6 seconds. Figure 10 (upper right) displays the partial SEAM velocity model. Figure 10 (upper right) displays the 7.5 Hz time-harmonic wavefield with quality factor $Q = \infty$. Figure 10 lower left displays the 7.5 Hz time-harmonic wavefield with quality factor $Q = 10$ (computed as if Q were spatially varying). Figure 10 (lower right) displays the 7.5 Hz time-harmonic wavefield with quality factor $Q = 5$.

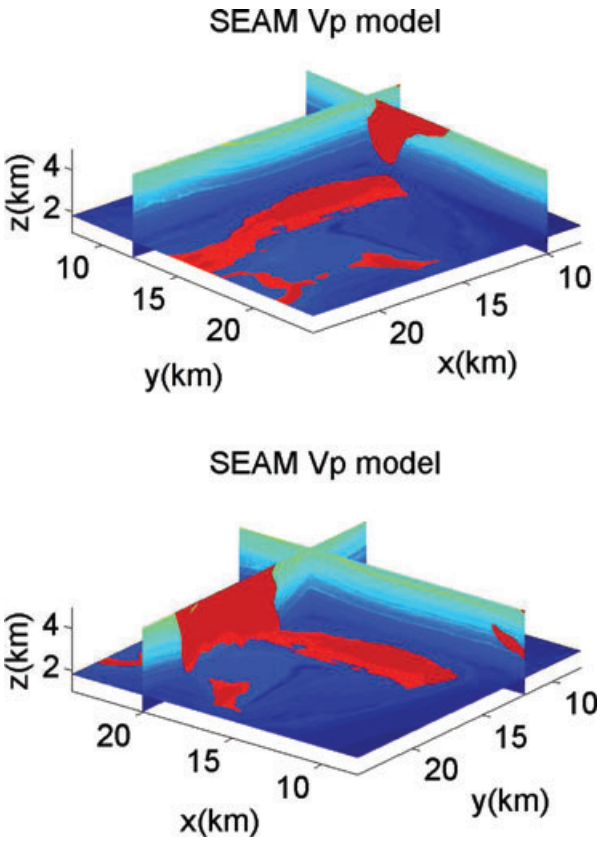


Figure 11 Part of the SEAM velocity model on a 401 × 401 × 201 mesh. Mesh step size is $h_x = h_y = 40$ m, $h_z = 20$ m. The physical domain is 8-24 km × 8-24 km × 1-5 km. Upper: viewing angle 1; lower: viewing angle 2.

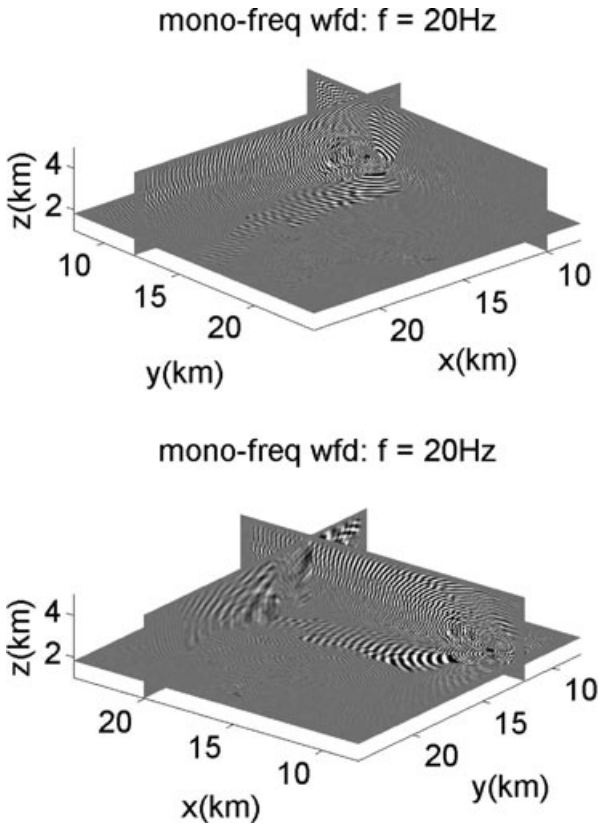


Figure 12 20 Hz time-harmonic acoustic wavefields computed using the Hsolver. The source location is at $\mathbf{x}_s = (20, 12, 1.8)$ km. The velocity model is from Fig. (11). The computation is conducted on 64 nodes (8 cores, 32 GB per node) on ConocoPhillips internal cluster. Upper: viewing angle 1; lower: viewing angle 2.

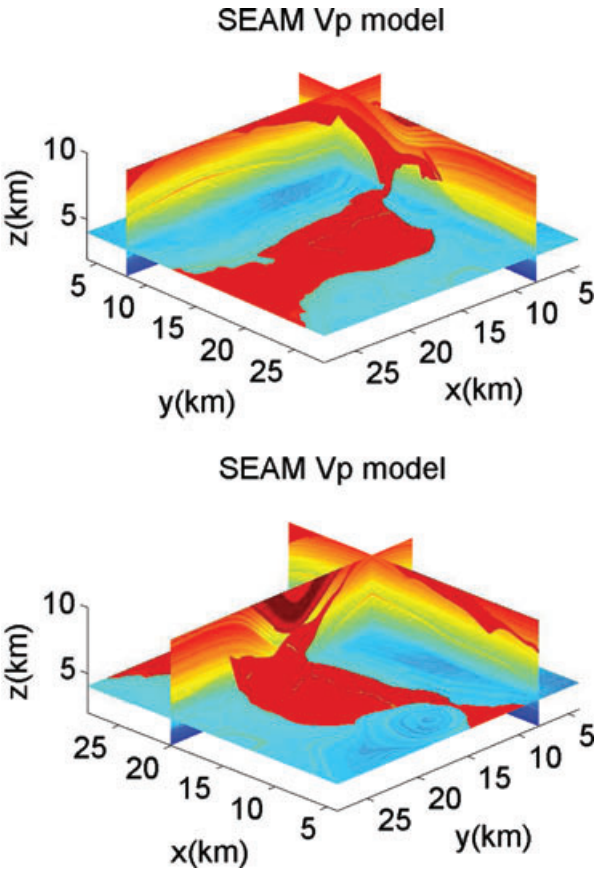


Figure 13 Part of the SEAM velocity model on a 601 × 601 × 301 mesh. Mesh step size is $h_x = h_y = 40m$, $h_z = 25m$. The physical domain is 4-29 km × 4-29 km × 2.5-10 km. Upper: viewing angle 1; lower: viewing angle 2.

In the second example, Fig. 11 displays part of the SEAM *P*-wave velocity model discretized on a 401 × 401 × 201 mesh. The number of unknowns is more than 32 million. The mesh step size is $h_x = h_y = 40$ m, $h_z = 20$ m, whence the physical domain is 8-24 km × 8-24 km × 1-5 km. Figure 12 displays the 20Hz time-harmonic acoustic wavefield computed using Hsolver. The source location is at $\mathbf{x}_s = (20, 12, 1.8)$ km. The velocity model is from Fig. 11. The computation is conducted on 64 nodes (8 cores, 32 GB per node). The CPU wall time for factorization is 7638 seconds, and the CPU wall time for solution for one RHS is 16 seconds.

In the last example, Fig. 13 displays another even larger part of the SEAM *P*-wave velocity model, covering a 601 × 601 × 301 mesh; the number of unknowns is more than 108 million. The mesh step size is $h_x = h_y = 40$ m, $h_z = 25$ m, whence the physical domain is 4-29 km × 4-29 km × 2.5-10 km. Figure 14 displays the 5 Hz time-harmonic acoustic wavefield computed using H-solver. The source location is at $\mathbf{x}_s = (20.0, 8.0, 4.0)$ km. Figure 15 displays the 15Hz time-harmonic acoustic wavefield computed using Hsolver. The source location is also at $\mathbf{x}_s = (20.0, 8.0, 4.0)$ km. The velocity model is from Fig. 13. We use 512 nodes with 4 cores per node on a TeraGrid cluster `kraken.nics.tennessee.edu`, to carry out this computation. The memory of each node on Kraken is 16 GB. The CPU wall time for factorization is 8163 seconds and the CPU time for the solution for one RHS is 39 seconds.

DISCUSSION

We briefly mention that the approach and algorithm developed here in the context of finite-difference approximations also applies to certain finite-element discretizations of the Helmholtz equation. We choose a regular tetrahedral triangulation. Following the (continuous) Galerkin method, we

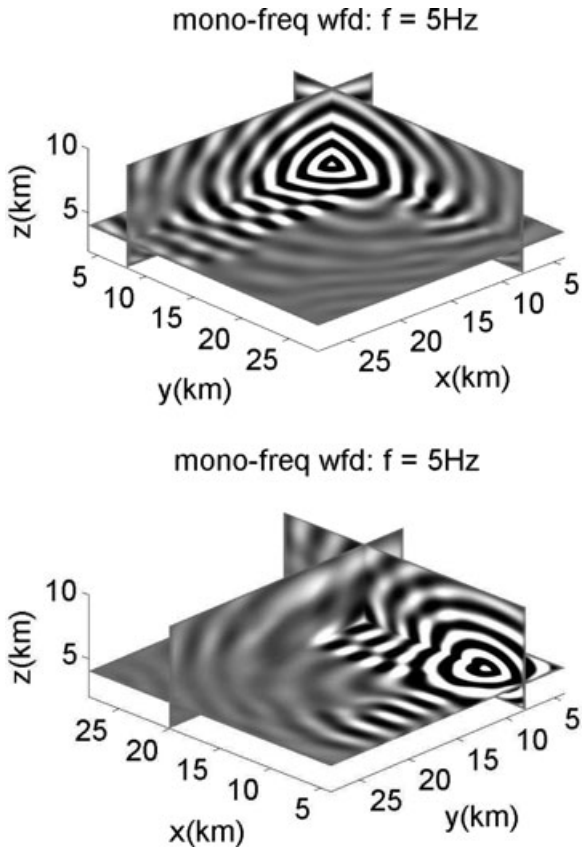


Figure 14 5Hz time-harmonic acoustic wavefields computed using the Hsolver. The source location is at $\mathbf{x}_s = (20.0, 8.0, 4.0)$ km. The velocity model is from Fig. (13). The computation is conducted on 512 nodes (4 cores, 16GB per node) on TeraGrid cluster `kraken.nics.tennessee.edu`. Upper: viewing angle 1; lower: viewing angle 2.

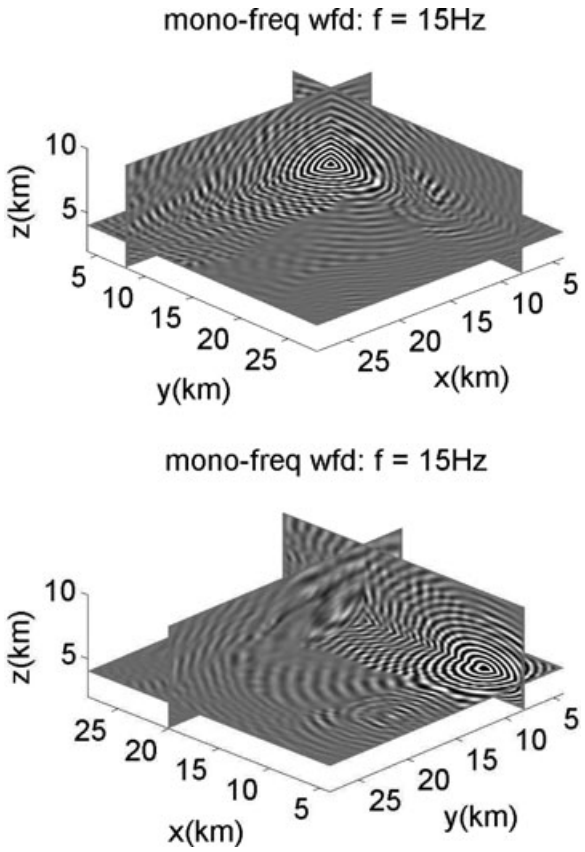


Figure 15 15Hz time-harmonic acoustic wavefields computed using the Hsolver. The source location is at $\mathbf{x}_s = (20.0, 8.0, 4.0)$ km. The velocity model is from Fig. (13). The computation is conducted on 512 nodes (4 cores, 16GB per node) on TeraGrid cluster `kraken.nics.tennessee.edu`. Upper: viewing angle 1; lower: viewing angle 2.

discretize the Helmholtz equation on such a mesh with piecewise linear, continuous finite element (nodal) basis functions, $\{\varphi_\alpha(\mathbf{x})\}$ say. Each $\varphi_\alpha(\mathbf{x})$ attains the value 1 at vertex α and 0 on the other vertices. The stiffness matrix, $\mathbf{M}(\omega)$, is then given by

$$(\mathbf{M}(\omega))_{\alpha\beta} = \int_{[0,L_x]\times[0,L_y]\times[0,L_z]} \left[\frac{1}{\rho(\mathbf{x})} \frac{\partial\varphi_\alpha}{\partial x}\frac{\partial\varphi_\beta}{\partial x} + \frac{1}{\rho(\mathbf{x})} \frac{\partial\varphi_\alpha}{\partial y}\frac{\partial\varphi_\beta}{\partial y} + \frac{1}{\rho(\mathbf{x})} \frac{\partial\varphi_\alpha}{\partial z}\frac{\partial\varphi_\beta}{\partial z} - \frac{\omega^2}{\rho(\mathbf{x})c^2(\mathbf{x})}\varphi_\alpha\varphi_\beta \right] \mathrm{d}\mathbf{x}, \tag{7}$$

while the source is: $(\mathbf{g}(\omega))_\alpha = \int_{[0,L_x]\times[0,L_y]\times[0,L_z]} f(\mathbf{x})\,\varphi_\alpha \mathrm{d}\mathbf{x}$. The pattern of the stiffness matrix $\mathbf{M}(\omega)$ in the finite element approach is then the same as the pattern of the matrix $\mathbf{A}(\omega)$.

Figure 16 illustrates the strategy that we adopt to carry out the domain decomposition of the finite element mesh based on the principles of nested dissection. This strategy is closely related to the one we use in Section 3. To simplify the illustration, we show the 2D case. A nodal basis function is indicated by the black box on the upper right of Fig. 16. The form of a nodal basis function in 3D is shown in Fig. 17. Similar to the finite-difference scenario, the separators in the finite element mesh are defined relative to those nodal basis functions which divide the remaining nodal basis functions into two subgroups which do not have any overlap region with one another. For example, in Fig. 16 mesh, the first level separator is formed out of those nodal basis functions whose central grid point coincide with those grid points in the first level separator of finite difference scenario. Figure 16 use alternating colors to illustrate the overlapping between nodal basis functions. Different levels of separators are also displayed in Fig. 16 using different color stripes.

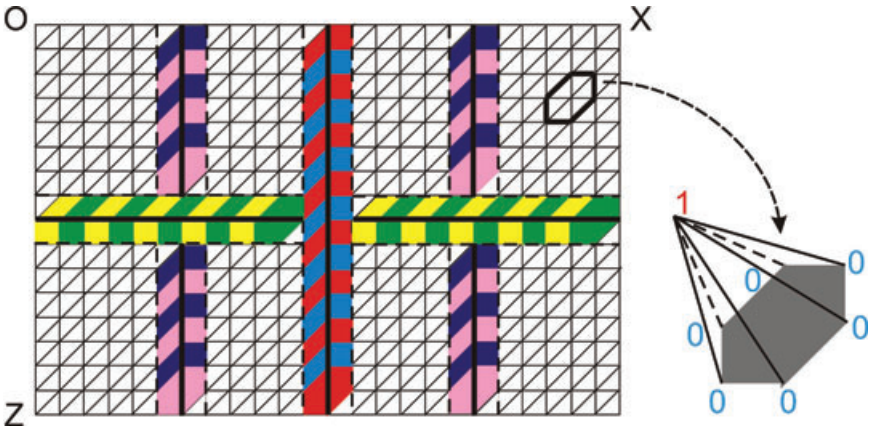


Figure 16 The nested dissection based domain decomposition of a 2D regular finite-element mesh. The upper right black box indicates the support of a piece-wise linear nodal basis function, which has value 1 at the center grid point and value 0 at corner points. The counterpart of the 3D nodal basis function is illustrated in Fig. (17). The solid lines surrounding with stripes indicate different levels of separators, on which the nodal basis functions are displayed alternately.

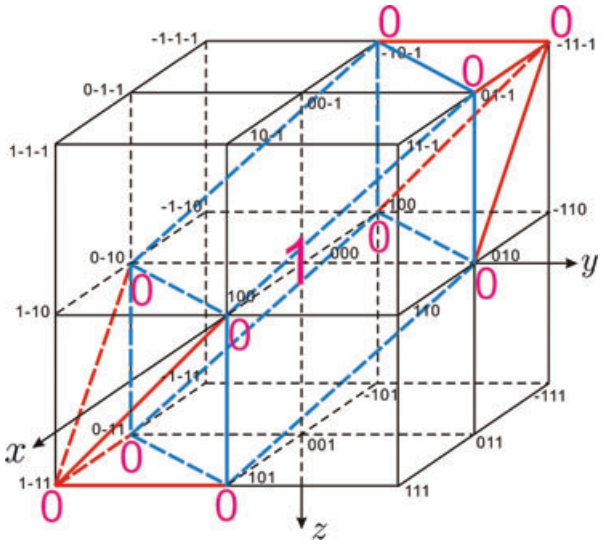


Figure 17 The nodal basis function in the 3D finite element mesh. It has the value 1 at the center grid point 000, and value 0 at vertices illustrated in the figure.

The parallel multifrontal strategy of this paper applies to factorizing $\mathbf{M}(\omega)$.

ACKNOWLEDGMENTS

The authors would like to thank the members, ConocoPhillips, ExxonMobil, PGS, Statoil and Total, of the Geo-Mathematical Imaging Group for financial support. The authors also thank Xiaoye Li from Lawrence Berkeley National Laboratory and Yingchong Situ from Purdue University for interactive discussions and valuable suggestions. S. Wang would like to thank ConocoPhillips especially for partial support while developing the parallel implementation of the algorithm presented in this paper. The research of Jianlin Xia was supported in part by NSF grant CHE-0957024.

REFERENCES

Agullo E., Guermouche A. and L'Excellent J. 2008. A parallel out-of-core multifrontal method: Storage of factors on disk and analysis of models for an out-of-core active memory. *Parallel Computing* **34**, 296–317.

Bai Z., Demmel J., Dongarra J., Ruhe A. and Van den Vorst H. 2000. *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*. SIAM.

Bebendorf M. 2005. Efficient inversion of galerkin matrices of general second-order elliptic differential operators with nonsmooth coefficients. *Mathematics of Computation* **74**, 1179–1199.

Bebendorf M. and Hackbusch W. 2003. Existence of $\mathcal{H}$-matrix approximants to the inverse FE-matrix of elliptic operators with L^∞ coefficients. *Numerical Mathematics* **95**, 1–28.

Börm S., Grasedyck L. and Hackbusch W. 2003. Introduction to hierarchical matrices with applications. *Engineering Analysis with Boundary Elements* **27**, 405–422.

Börm S. and Hackbusch W., Data-sparse approximation by adaptive $\mathcal{H}^2$-matrices. Technical Report, Max Planck Institute for Mathematics.

Chandrasekaran S., Dewilde P., Gu M. and Somasunderam N. 2010. On the numerical rank of the off-diagonal blocks of schur complements of discretized elliptic pdes. *SIAM Journal on Matrix Analysis and Applications* **31**, 2261–2290.

Chandrasekaran S., Dewilde P., Gu M., Lyons W. and Pals T. 2006. A fast solver for HSS representations via sparse matrices. *SIAM Journal on Matrix Analysis and Applications* **29**, 67–81.

Chandrasekaran S., Dewilde P., Gu M., Pals T., Sun X., van der Veen A. and White D. 2005. Some fast algorithms for sequentially semiseparable representations. *SIAM Journal on Matrix Analysis and Applications* **27**, 341–364.

Chandrasekaran S., Gu M. and Pals T. 2006. A fast *ULV* decomposition solver for hierarchically semiseparable representations. *SIAM Journal on Matrix Analysis and Applications* **28**, 603–622.

Duff I. and Reid J. 1983. The multifrontal solution of indefinite sparse symmetric linear equations. *ACM Transactions on Mathematical Software* **9**, 302–325.

Eidelman Y. and Gohberg I. 1999. On a new class of structured matrices. *Integral Equations Operator Theory* **34**, 293–324.

Eisenstat S. and Liu J. 2005. The theory of elimination trees for sparse unsymmetric matrices. *SIAM Journal on Matrix Analysis and Applications* **26**, 686–705.

Engquist B. and Ying L. Sweeping preconditioner for the helmholtz equation: hierarchical matrix representation. *Communications in Pure and Applied Mathematics*, to appear. http://www.math.utexas.edu/users/lexing/publications/sweephmf.pdf.

Erlangga Y., Oosterlee C. and Vuik C. 2006. A novel multigrid based preconditioner for heterogeneous Helmholtz problems. *SIAM Journal on Scientific Computing* **27**, 1471–1492.

Futterman W. 1962. Dispersive body waves. *Journal of Geophysical Research* **67**, 5279–5291.

George J. 1973. Nested dissection of a regular finite element mesh. *SIAM Journal on Numerical Analysis* **10**, 345–363.

Gilbert E.N.J.R. 1993. *Predicting Structure in Nonsymmetric Sparse Matrix Factorizations: Graph Theory and Sparse Matrix Computation*. Springer-Verlag.

Hackbusch W. 2011. A sparse matrix arithmetic based on $\mathcal{H}$-matrices. Part I: Introduction to $\mathcal{H}$-matrices. *Computing* **62**, 89–108.

Hackbusch L.G.W. and Börm S. 2002. An introduction to hierarchical matrices. *Math. Bohem.* **127**, 229–241.

Hackbusch W. and Khoromskij B.N. 2000. A sparse $\mathcal{H}$-matrix arithmetic. Part-II: Application to multi-dimensional problems. *Computing* **64**, 21–47.

Hackbusch B.K.W. and Sauter S.A. 2003. On $\mathcal{H}^2$-matrices, Lectures on Applied Mathematics. Springer.

Hoffman A., Martin M. and Rose D. 1973. Complexity bounds for regular finite difference and finite element grids. *SIAM Journal on Numerical Analysis* **10**, 364–369.

Hustedt B., Operto S. and Virieux J. 2004. Mixed-grid and staggered-grid finite-difference methods for frequency-domain acoustic wave modeling. *Geophysical Journal International* **157**, 1269–1296.

Kjartansson E. 1979. Constant q-wave propagation and attenuation. *Journal of Geophysical Research* **84**, 4737–4748.

Lipton D.J.R.R.J. and Tarjan R.E. 1979. Generalized nested dissection, *SIAM Journal on Numerical Analysis* **16**, 346–358.

Liu J. 1990. The role of elimination trees in sparse factorization. *SIAM Journal on Matrix Analysis and Applications* **18**, 134–172.

Liu J. 1992. The multifrontal method for sparse matrix solution: Theory and practice. *SIAM Review* **34**, 82–109.

Operto S., Virieux J., Amestoy P., L'Excellent J., Giraud L. and Hadj H. 2007. Ali, 3D finite-difference frequency-domain modeling of visco-acoustic wave propagation using a massively parallel direct solver: A feasibility study. *Geophysics* **72**, SM195–SM211.

Operto S., Virieux J., Ribodetti A. and Anderson J. 2009. Finite-difference frequency-domain modeling of viscoacoustic wave propagation in 2D tilted transversely isotropic (TTI) media. *Geophysics* **74**, T75–T95.

Plessix R.-E. 2007. A Helmholtz iterative solver for 3D seismic imaging problems. *Geophysics* **72**, SM185–SM194.

Riyanti C., Erlangga Y., Plessix R.-E., Mulder W., Vuik C. and Oosterlee C. 2006. A new iterative solver for the time-harmonic wave equation. *Geophysics* **71**, E57–E63.

Riyanti C., Kononov A., Erlangga Y., Vuik C., Oosterlee C., Plessix R.-E. and Mulder W. 2007. A parallel multigrid-based preconditioner for the 3D heterogeneous high-frequency Helmholtz equation. *Journal of Computational Physics* **224**, 431–448.

Teng S. 1997. Fast nested dissection for finite element meshes. *SIAM Journal on Matrix Analysis and Applications* **18**, 552–565.

Turkel E. and Yefet A. 1998. Absorbing PML boundary layers for wave-like equations. *Applied Numerical Mathematics* **27**, 533–557.

Ursin B. and Toverud T. 2002. Comments on comparison of seismic dispersion and attenuation models by ursin b. and toverud, t. *Studia Geophysica et Geodetica* **46**, 293–320.

Wang S., de Hoop M. and Xia J. 2010. Seismic inverse scattering via Helmholtz operator factorization and optimization. *Journal of Computational Physics* **229**, 8445–8462.

Xia J., Chandrasekaran S., Gu M. and Li X. 2010. Superfast multifrontal method for large structured linear systems of equations. *SIAM Journal on Matrix Analysis and Applications* **31**, 1382–1411.

Xia J., Chandrasekaran S., Gu M. and Li X. 2010. Fast algorithms for hierarchically semiseparable matrices. *Numerical Linear Algebra Applications* **17**, 953–976.

Xia J. Efficient structured multifrontal factorization for large sparse matrices, preprint, http://www.math.purdue.edu/∼xiaj/work/mfhss.pdf.