Proceedings of the Project Review, Geo-Mathematical Imaging Group (Purdue University, West Lafayette IN), Vol. 1 (2010) pp. 69-92.

3D SEISMIC IMAGING VIA A MASSIVELY PARALLEL STRUCTURED APPROXIMATE DIRECT HELMHOLTZ SOLVER*

SHEN WANG†, JIANLIN XIA‡, AND MAARTEN V. DE HOOP§

Abstract. We consider the discretization and 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 seismic inverse problems in general, and imaging 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 3D nested dissection based domain decomposition, and introduce an approximate direct solver by developing a new parallel Hierarchically SemiSeparable (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 processor-to-processor communications. 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. $\mathcal{O}(rN\log N)$, while the storage is linear as well, $\mathcal{O}(N\log(r\log N))$, if r is the maxium numerical rank of all off-diagonal blocks in the multifrontal procedure.

Key words. massively parallel, Helmholtz solver, multifrontal method, HSS matrix, seismic imaging

1. Introduction. We consider the discretization and 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 seismic inverse problems in general, and imaging in particular. We include solution and least-squares optimization strategies reminiscent of the limiting absorption principle. We develop a massively parallel structured approximate direct solver based on 3D nested dissection domain decomposition, rank revealing QR factorization, parallel Hierarchically SemiSeparable (HSS) matrices compression, factorization and solution.

Solvers of the inhomogeneous Helmholtz equation play a key role in the frequency-domain implementation of reverse-time migration (RTM) and imaging, also as part of so-called (local optimization based) full waveform inversion (FWI). The multi-frequency formulation of the seismic inverse problem furthermore aids in developing multi-scale regularization methods to mitigate its nonlinear nature. The Helmholtz-equation approach to RTM and 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. RTM and FWI have the advantage to, recursively, accommodate and enhance illumination without user interference; they also provide, in principle, a common framework for wave-equation tomography and inverse scattering.

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 not Hermitian. We require that the coefficients are sufficiently regular, viz., continuously differentiable a sufficient number of times. The regularity of coefficients appears naturally in the processes of imaging and velocity inversion. However, the solver we develop here is more generally applicable, for example, to a discretization based on certain interior penalty discontinuous Galerkin methods [1].

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 [2, 3, 4, 5]. In general, iterative solvers lack the efficiency to deal with many forcing terms, and suffer from

*This research was supported in part by the members, BP, ConocoPhillips, ExxonMobil, StatoilHydro and Total, of the Geo-Mathematical Imaging Group.

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

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

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

decreasing convergence rate 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 [6] [7]. Here, we design an approximate direct solver by developing a new parallel HSS matrix compression and factorization, which is the main result. Not only do we obtain estimates for the associated computational complexity almost linear in the size, N say, of the matrix, viz. $\mathcal{O}(rN\log N)$, but also for the storage, $\mathcal{O}(N\log(r\log N))$ if r is the maximum numerical rank of all off-diagonal block compressions.

It has been recognized that direct methods have a fill-in issue, in particular, in the case of 3D problems, which means that LU factors would overflow the memory limit of a typical computational platform. The latest work [6] used a multifrontal massively parallel solver called MUMPS to minimize the fill-in problem by resorting to the multifrontal method [8, 9] in combination with a matrix reordering strategy. However, [10] demonstrated that for truly large problems, even with MUMPS one cannot realistically store all the factors in memory and has to resort to parallel out-of-core storage.

In many problems following the discretization of linear partial differential equations from (geo)physics, the off-diagonal blocks of both the frontal and update matrices, resulting from the multifrontal method, have the low-rank property [11, 12, 13]. We note that our 3D Helmholtz equation yields a typical example of this low-rank phenomenon, which implies that our frontal and update matrices can be compressed subject to a given tolerance. We represent the discrete Helmholtz operator in terms of Hierarchically SemiSeparable (HSS) matrices. Fast algorithms for compression, factorization and solution have been developed for HSS matrices [14, 15, 16, 17, 18, 19]. In this paper, we investigate HSS structure and parallelize its compression, factorization and solution stages in the framework of a massively parallel multifrontal method.

In Section 2, we discuss the Helmholtz equation and the limiting absorption principle. We summarize the processes of RTM and FWI. In Section 3, we follow the work of [6] to develop a 27-point parsimonious mixed-grid stencil to discretize the Helmholtz equation supplemented with a PML boundary condition. In Section 4, we discuss 3D nested dissection based domain decomposition and the general parallel multifrontal method. In Section 5 we introduce and discuss our massively parallel HSS compression, factorization, and solution approaches. In Section 6 we numerically test the performance of our solver using a 3D lens model. We end the paper with some conclusions.

2. 3D seismic imaging with the Helmholtz equation. The Helmholtz equation is given by

$$-[\Delta + n(\mathbf{x},\omega)]\, u(\mathbf{x},\omega) = f(\mathbf{x},\omega), \quad \mathbf{x} \in \mathbb{R}^3, \tag{2.1}$$

where $n(\mathbf{x}) = \omega^2 c(\mathbf{x})^{-2}$ if $c = c(\mathbf{x})$ denotes the wavespeed and ω the angular frequency. Without any restriction we assume that $\omega > 0$; ω is fixed. For the solutions equivalent to causality in the time domain, we impose the Sommerfeld radiation condition.

It is a standard procedure to perturb (2.1) according to

$$(\mathrm{i}\varepsilon + \Delta + n)\, u_\varepsilon = -f, \quad \varepsilon > 0. \tag{2.2}$$

Because $\varepsilon \in \mathbb{R}$, then, for any $f \in L^2$, a solution $u_\varepsilon \in L^2$ of (2.2) satisfies the estimate

$$\|u_\varepsilon\|_{L^2} \leq \frac{1}{\varepsilon}\, \|(\mathrm{i}\varepsilon + \Delta + n)\, u_\varepsilon\|_{L^2}, \quad \text{or} \quad \|u_\varepsilon\|_{L^2} \leq \frac{1}{\varepsilon}\, \|f\|_{L^2}, \tag{2.3}$$

whence the problem is well posed. Clearly, the right-hand side of (2.3) blows up if $\varepsilon \to 0^+$, which necessitates the introduction of norms different from the L^2 norm to analyze the problem of solving the Helmholtz equation. We mention the Morrey-Campanato norm,

$$|||\ u\ |||^2 = \sup_{\mathbf{y}\in\mathbb{R}^3, R>0} \frac{1}{R} \int_{|\mathbf{x}-\mathbf{y}|<R} |u(\mathbf{x})|^2\, \mathrm{d}\mathbf{x}, \tag{2.4}$$

and its dual norm

$$N(f) = \sum_{j\in\mathbb{Z}} 2^{(j+1)/2} \|f\|_{L^2(A_j)}. \tag{2.5}$$

Under certain assumptions,

$$||| \nabla u_\varepsilon |||^2 + ||| n^{1/2} u_\varepsilon |||^2 \lesssim (\varepsilon + \|n\|_{L^\infty})\, N(n^{-1/2} f)^2 \tag{2.6}$$

[20] [21]; applying the limiting absorption principle, that is, letting $\varepsilon \to 0^+$, one obtains the existence of solutions to (2.1).

We will consider the problem of "full waveform inversion" of seismic data. Data, $d(\mathbf{x}_s, ., \omega)$, are modelled as solutions to the Helmholtz equation, generated by a family of sources $f(\mathbf{x}, \omega; \mathbf{x}_s)$ parametrized by $\mathbf{x}_s \in \Sigma_s$, and restricted to $\Sigma_r \subset \Sigma$, where Σ is the boundary of a (bounded) domain in which the function c is unknown. We assume, also, that $\Sigma_s \subset \Sigma$. Typically, one formulates this problem in terms of a minimum norm optimization. The computation of the gradient in such a formulation corresponds with imaging; see, for example, [22]. The image is obtained as

$$\mathcal{I}(\mathbf{x}) = \sum_{\omega} \sum_{\mathbf{x}_s \in \Sigma_s} \overline{w_s(\mathbf{x}, \omega; \mathbf{x}_s)}\, u^*(\mathbf{x}, \omega; \mathbf{x}_s) \tag{2.7}$$

with

$$[\Delta + n(\mathbf{x}, \omega)]\, w_s(\mathbf{x}, \omega; \mathbf{x}_s) = -f(\mathbf{x}, \omega; \mathbf{x}_s) \tag{2.8}$$

and

$$[\Delta + n(\mathbf{x}, \omega)]\, u^*(\mathbf{x}, \omega; \mathbf{x}_s) = \omega^2 \int_{\Sigma_r} d(\mathbf{x}_s, \mathbf{x}_r, \omega) \delta(\mathbf{x} - \mathbf{x}_r)\, \mathrm{d}\mathbf{x}_r, \tag{2.9}$$

in accordance with the adjoint state method; here, we substitute for c a presumed background model. In the gradient, d is replaced by the data residual and c by the current model.

One has developed various strategies, including preconditioning, to yield computational results. We mention "exponential damping" [23] [24] [25] [26]. Here, the angular frequencies are complex:

$$\int d(t) \exp(\tau\, t_0) \exp[\mathrm{i}(\omega + \mathrm{i}\tau)t]\, \mathrm{d}t = \int d(t) \exp[-\tau\, (t - t_0)] \exp(\mathrm{i}\omega t)\, \mathrm{d}t$$

signifying the complex Laplace transform of the data (residuals), if t_0 is an estimate of the first arrival time and τ is a chosen damping term. Thus n is replaced by $(\omega + \mathrm{i}\tau)^2 c(\mathbf{x})^{-2}$ with $0 < \tau < \tau_0 \ll \omega$. We write $n_\tau(\mathbf{x}, \omega) = (\omega^2 - \tau^2)\, c(\mathbf{x})^{-2}$. In the above, we can replace ε by τ upon identifying ε with $2\omega\tau c(\mathbf{x})^{-2}$. Our solution procedure is designed to admit $\varepsilon > 0$ at the cost of loosing self-adjointness of the partial differential operator.

Equations (2.7)-(2.9) describe the process of what exploration seismologists call reverse-time migration (RTM). It requires solving the Helmholtz equation for many right-hand sides. In 2D, it has been, hence, natural to exploit the LU factorization of the discrete Helmholtz operator [3]. In view of the size of realistic seismic imaging problems (of the order of 10^9 grid points ($\mathbf{x}$) and 10^3 source points ($\mathbf{x}_s$)), standard LU factorization is not feasible in 3D. Most developments in 3D indeed are based on iterative Helmholtz-equation solvers [3, 4, 5]. Here, we develop a structured, approximate direct solver based on a rank revealing QR factorization in combination with domain decomposition.

3. Discretization: 27-point parsimonious mixed grid finite differences with PML boundary condition. In this section, we follow the work of [6] and discuss a 27-point compact parsimonious mixed grid finite difference (FD) stencil. The corresponding scheme has been proven to be $4th$ order [27]; in practice, this implies that a sampling rate of 4 grid points per wavelength

yields sufficiently accurate results. This spatially compact stencil also has the advantage that the factorization of the discrete Helmholtz operator below is optimized, because it reduces the bandwidth of the matrix and hence reduces the storage of the factorization significantly compared with the $4th$ order centered finite difference stencil. The mixed grid stencil incorporates coordinate rotations, and hence numerical anisotropy is, to the given order, minimized. The Helmholtz operator can be generalized to include medium anisotropy according to polarized (elastic-)wave equations [7].

We introduce coordinates, $\mathbf{x} = (x, y, z)$. We write (2.1) in the form

$$H(\omega)\, U(x, y, z, \omega) = s(x, y, z, \omega), \tag{3.1}$$

in which $H(\omega)$ is the Helmholtz operator,

$$\begin{aligned} H(\omega) &= \Gamma(\omega) + \frac{\omega^2}{\rho(x,y,z)\ c(x,y,z)^2}, \\ \Gamma(\omega) &= \frac{1}{S_x}\frac{\partial}{\partial x}\left(\frac{1}{\rho(x,y,z)S_x}\frac{\partial}{\partial x}\right) + \frac{1}{S_y}\frac{\partial}{\partial y}\left(\frac{1}{\rho(x,y,z)S_y}\frac{\partial}{\partial y}\right) + \frac{1}{S_z}\frac{\partial}{\partial z}\left(\frac{1}{\rho(x,y,z)S_z}\frac{\partial}{\partial z}\right). \end{aligned} \tag{3.2}$$

Here, Γ is sometimes referred to as the stiffness term, and $\omega^2 c^{-2}$ as the mass term; $\rho(x, y, z)$ is the density. S_x, S_y and S_z are damping functions defined at the Perfect Matched Layer (PML) [28] surrounding the computational domain: $[0, L_x] \times [0, L_y] \times [0, L_z]$. Let $0 < L_{x1} < L_x$, then

$$S_x = \begin{cases} 1 & \text{if } 0 \le x \le L_{x1}, \\ 1 - \mathrm{i}\frac{\sigma_0}{\omega}\left(\frac{x - L_{x1}}{L_x - L_{x1}}\right)^p & \text{if } L_{x1} < x \le L_x, \end{cases} \tag{3.3}$$

and similarly for S_y and S_z. Here, σ_0 is an appropriately chosen constant and p is commonly set to 2.

To form the discretization of $\Gamma(\omega)$, we utilize eight coordinate systems with orientions of their axes connecting all pairs of (outer) nodes of the standard cubic cell; with each coordinate system we associate a $4th$ order finite difference stencil by incorporating the midpoint between two grid points. The coordinate systems are displayed in figure (1): The top diagram shows the standard Cartesian coordinate system, denoted by $B_c(x, y, z)$, as well as the indices of positions of the nodes relative to the center (corresponding with (x, y, z) below) of the cell. We denote the mesh step size by h. The seven nodes used in the component stencil, Γ_{B_c}, are [0 0 0], [1 0 0], [-1 0 0], [0 1 0], [0 -1 0], [0 0 1] and [0 0 -1]. The middle diagram shows the three coordinate systems obtained by rotating B_c over 45° around the $x-$, $y-$, and $z-$axis, respectively. They are denoted by $B_x(x, y'_x, z'_x)$, $B_y(y, x'_y, z'_y)$ and $B_z(z, x'_z, y'_z)$. The largest distance between the center point ([0 0 0]) and the grid points involved in these three stencils is $\sqrt{2}h$. For example,

$$\begin{aligned} \Gamma_{B_y}(\omega)U(x,y,z,\omega) = {} & \frac{1}{h^2 (S_y)_{\,0\ 0\ 0}}\left[\frac{U_{\,0\ 1\ 0} - U_{\,0\ 0\ 0}}{(\rho S_y)_{\,0\ 1/2\ 0}} - \frac{U_{\,0\ 0\ 0} - U_{\,0\ -1\ 0}}{(\rho S_y)_{\,0\ -1/2\ 0}}\right] \\ & + \frac{1}{4h^2 (S_z)_{\,0\ 0\ 0}}\left[\frac{U_{\,1\ 0\ 1} - U_{\,0\ 0\ 0}}{(\rho S_z)_{\,1/2\ 0\ 1/2}} - \frac{U_{\,0\ 0\ 0} - U_{\,-1\ 0\ -1}}{(\rho S_z)_{\,-1/2\ 0\ -1/2}}\right] \\ & - \frac{1}{4h^2 (S_z)_{\,0\ 0\ 0}}\left[\frac{U_{\,1\ 0\ 0} - U_{\,0\ 0\ 1}}{(\rho S_z)_{\,1/2\ 0\ 1/2}} - \frac{U_{\,0\ 0\ -1} - U_{\,-1\ 0\ 0}}{(\rho S_z)_{\,-1/2\ 0\ -1/2}}\right] \\ & - \frac{1}{4h^2 (S_z)_{\,0\ 0\ 0}}\left[\frac{U_{\,1\ 0\ 0} - U_{\,0\ 0\ -1}}{(\rho S_z)_{\,1/2\ 0\ -1/2}} - \frac{U_{\,0\ 0\ 1} - U_{\,-1\ 0\ 0}}{(\rho S_z)_{\,-1/2\ 0\ 1/2}}\right] \end{aligned}$$

$$
\begin{aligned}
&+\frac{1}{4h^2(S_z)_{\,0\,0\,0}}\left[\frac{U_{\,1\,0\,-1}-U_{\,0\,0\,0}}{(\rho S_z)_{\,1/2\,0\,-1/2}}-\frac{U_{\,0\,0\,0}-U_{\,-1\,0\,1}}{(\rho S_z)_{\,-1/2\,0\,1/2}}\right]\\
&+\frac{1}{4h^2(S_x)_{\,0\,0\,0}}\left[\frac{U_{\,1\,0\,1}-U_{\,0\,0\,0}}{(\rho S_x)_{\,1/2\,0\,1/2}}-\frac{U_{\,0\,0\,0}-U_{\,-1\,0\,-1}}{(\rho S_x)_{\,-1/2\,0\,-1/2}}\right]\\
&+\frac{1}{4h^2(S_x)_{\,0\,0\,0}}\left[\frac{U_{\,1\,0\,0}-U_{\,0\,0\,1}}{(\rho S_x)_{\,1/2\,0\,1/2}}-\frac{U_{\,0\,0\,-1}-U_{\,-1\,0\,0}}{(\rho S_x)_{\,-1/2\,0\,-1/2}}\right]\\
&+\frac{1}{4h^2(S_x)_{\,0\,0\,0}}\left[\frac{U_{\,1\,0\,0}-U_{\,0\,0\,-1}}{(\rho S_x)_{\,1/2\,0\,-1/2}}-\frac{U_{\,0\,0\,1}-U_{\,-1\,0\,0}}{(\rho S_x)_{\,-1/2\,0\,1/2}}\right]\\
&+\frac{1}{4h^2(S_x)_{\,0\,0\,0}}\left[\frac{U_{\,1\,0\,-1}-U_{\,0\,0\,0}}{(\rho S_x)_{\,1/2\,0\,-1/2}}-\frac{U_{\,0\,0\,0}-U_{\,-1\,0\,1}}{(\rho S_x)_{\,-1/2\,0\,1/2}}\right]
\end{aligned}
$$

The bottom diagram shows the four coordinate systems, each one of which is obtained by rotation of B_c to three of the four main diagonals. They are denoted by $B_1(d_1,d_2,d_3)$, $B_2(d_1,d_2,d_4)$, $B_3(d_1,d_3,d_4)$ and $B_4(d_2,d_3,d_4)$. The largest distance between the center point ([0 0 0]) and the grid points involved in these four stencils is $\sqrt{3}h$. For example,

$$
\begin{aligned}
\Gamma_{B_2}(\omega)U(x,y,z,\omega) = &\frac{1}{4h^2(S_x)_{\,0\,0\,0}}\left[\frac{U_{\,1\,1\,1}-U_{\,0\,0\,0}}{(\rho S_x)_{\,1/2\,1/2\,1/2}}-\frac{U_{\,0\,0\,0}-U_{\,-1\,-1\,-1}}{(\rho S_x)_{\,-1/2\,-1/2\,-1/2}}\right]\\
&-\frac{1}{4h^2(S_x)_{\,0\,0\,0}}\left[\frac{U_{\,0\,1\,1}-U_{\,1\,0\,0}}{(\rho S_x)_{\,1/2\,1/2\,1/2}}-\frac{U_{\,-1\,0\,0}-U_{\,0\,-1\,-1}}{(\rho S_x)_{\,-1/2\,-1/2\,-1/2}}\right]\\
&-\frac{1}{4h^2(S_x)_{\,0\,0\,0}}\left[\frac{U_{\,0\,1\,1}-U_{\,-1\,0\,0}}{(\rho S_x)_{\,-1/2\,1/2\,1/2}}-\frac{U_{\,1\,0\,0}-U_{\,0\,-1\,-1}}{(\rho S_x)_{\,1/2\,-1/2\,-1/2}}\right]\\
&+\frac{1}{4h^2(S_x)_{\,0\,0\,0}}\left[\frac{U_{\,-1\,1\,1}-U_{\,0\,0\,0}}{(\rho S_x)_{\,-1/2\,1/2\,1/2}}-\frac{U_{\,0\,0\,0}-U_{\,1\,-1\,-1}}{(\rho S_x)_{\,1/2\,-1/2\,-1/2}}\right]\\
&+\frac{1}{4h^2(S_y)_{\,0\,0\,0}}\left[\frac{U_{\,1\,1\,1}-U_{\,0\,0\,0}}{(\rho S_y)_{\,1/2\,1/2\,1/2}}-\frac{U_{\,0\,0\,0}-U_{\,-1\,-1\,-1}}{(\rho S_y)_{\,-1/2\,-1/2\,-1/2}}\right]\\
&+\frac{1}{4h^2(S_y)_{\,0\,0\,0}}\left[\frac{U_{\,0\,1\,0}-U_{\,1\,0\,1}}{(\rho S_y)_{\,1/2\,1/2\,1/2}}-\frac{U_{\,-1\,0\,-1}-U_{\,0\,-1\,0}}{(\rho S_y)_{\,-1/2\,-1/2\,-1/2}}\right]\\
&+\frac{1}{4h^2(S_y)_{\,0\,0\,0}}\left[\frac{U_{\,0\,1\,0}-U_{\,-1\,0\,-1}}{(\rho S_y)_{\,-1/2\,1/2\,-1/2}}-\frac{U_{\,1\,0\,1}-U_{\,0\,-1\,0}}{(\rho S_y)_{\,1/2\,-1/2\,1/2}}\right]\\
&+\frac{1}{4h^2(S_y)_{\,0\,0\,0}}\left[\frac{U_{\,-1\,1\,-1}-U_{\,0\,0\,0}}{(\rho S_y)_{\,-1/2\,1/2\,-1/2}}-\frac{U_{\,0\,0\,0}-U_{\,1\,-1\,1}}{(\rho S_y)_{\,1/2\,-1/2\,1/2}}\right]\\
&+\frac{1}{4h^2(S_z)_{\,0\,0\,0}}\left[\frac{U_{\,-1\,1\,1}-U_{\,0\,0\,0}}{(\rho S_z)_{\,-1/2\,1/2\,1/2}}-\frac{U_{\,0\,0\,0}-U_{\,-1\,-1\,-1}}{(\rho S_z)_{\,1/2\,-1/2\,-1/2}}\right]\\
&-\frac{1}{4h^2(S_z)_{\,0\,0\,0}}\left[\frac{U_{\,-1\,1\,0}-U_{\,0\,0\,1}}{(\rho S_z)_{\,-1/2\,1/2\,1/2}}-\frac{U_{\,0\,0\,-1}-U_{\,1\,-1\,0}}{(\rho S_z)_{\,1/2\,-1/2\,-1/2}}\right]\\
&-\frac{1}{4h^2(S_z)_{\,0\,0\,0}}\left[\frac{U_{\,-1\,1\,0}-U_{\,0\,0\,-1}}{(\rho S_z)_{\,-1/2\,1/2\,-1/2}}-\frac{U_{\,0\,0\,1}-U_{\,1\,-1\,0}}{(\rho S_z)_{\,1/2\,-1/2\,1/2}}\right]\\
&+\frac{1}{4h^2(S_z)_{\,0\,0\,0}}\left[\frac{U_{\,-1\,1\,-1}-U_{\,0\,0\,0}}{(\rho S_z)_{\,-1/2\,1/2\,-1/2}}-\frac{U_{\,0\,0\,0}-U_{\,1\,-1\,1}}{(\rho S_z)_{\,1/2\,-1/2\,1/2}}\right]
\end{aligned}
$$

The 27 nodes used in these four coordinate systems incorporate all nodes.

The stencils associated with the different coordinate systems are weighted according to

$$
\begin{aligned}
\Gamma(\omega) = W_1\,\Gamma_{B_c}(\omega) + \frac{W_2}{3}\left(\Gamma_{B_x}(\omega)+\Gamma_{B_y}(\omega)+\Gamma_{B_z}(\omega)\right)\\
+\frac{W_3}{4}\left(\Gamma_{B_1}(\omega)+\Gamma_{B_2}(\omega)+\Gamma_{B_3}(\omega)+\Gamma_{B_4}(\omega)\right),
\end{aligned}
\tag{3.4}
$$

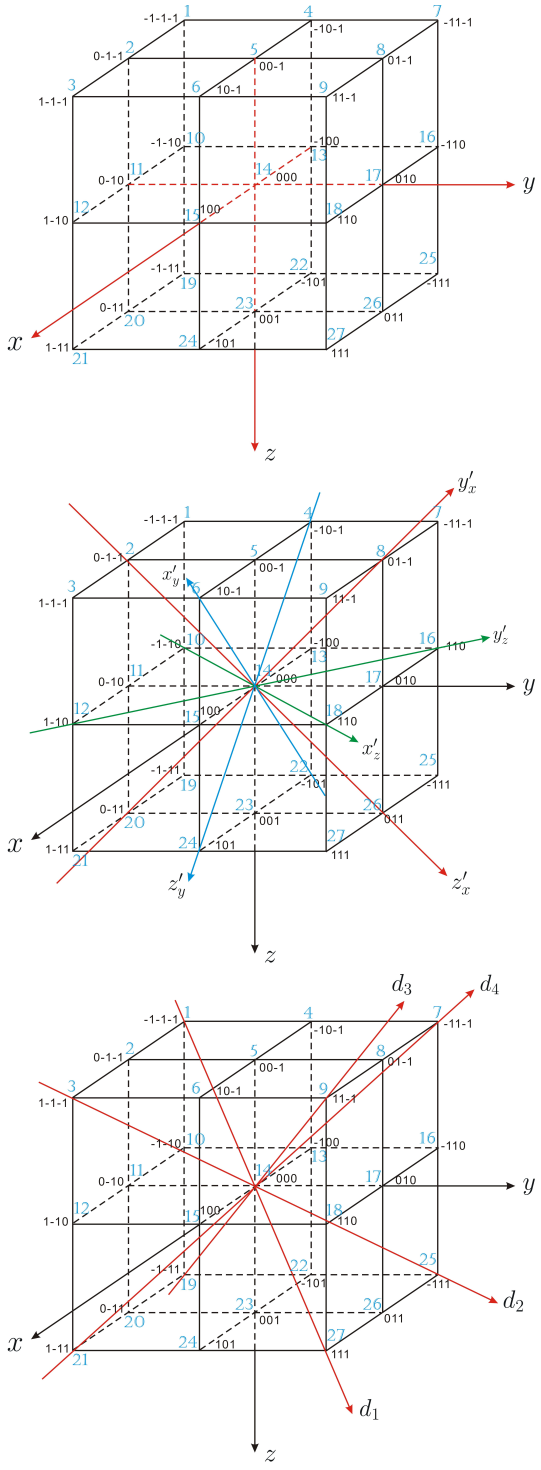

FIG. 1. *Mixed grid finite difference coordinates:* ***top****: classical Cartesian coordinate* $B_c(x, y, z)$*;* ***middle****: three coordinate systems obtained by rotating around* $x, y,$ *and* z *respectively. They are denoted as* $B_x(x, y'_x, z'_x)$, $B_y(y, x'_y, z'_y)$ *and* $B_z(z, x'_z, y'_z)$*;* ***bottom****: four coordinate systems obtained by coordinate transform from Cartesian system to three main diagonals out of four. They are denoted as* $B_1(d_1, d_2, d_3)$, $B_2(d_1, d_2, d_4)$, $B_3(d_1, d_3, d_4)$ *and* $B_4(d_2, d_3, d_4)$.

with $W_1 + W_2 + W_3 = 1$. Similarly,

$$\frac{\omega^2}{c^2}U(x,y,z,\omega) = \omega^2\left(W_{m_1}\left[\frac{U}{c^2}\right]_0 + \frac{W_{m_2}}{6}\left[\frac{U}{c^2}\right]_1 + \frac{W_{m_3}}{12}\left[\frac{U}{c^2}\right]_{\sqrt{2}} + \frac{W_{m_4}}{8}\left[\frac{U}{c^2}\right]_{\sqrt{3}}\right), \tag{3.5}$$

with $W_{m_1} + W_{m_2} + W_{m_3} + W_{m_4} = 1$, and

$$\begin{aligned}
\left[\frac{U}{c^2}\right]_0 &= \frac{U_{\,0\;0\;0}}{c^2_{0\;0\;0}},\\
\left[\frac{U}{c^2}\right]_1 &= \frac{U_{\,1\;0\;0}}{c^2_{1\;0\;0}} + \frac{U_{-1\;0\;0}}{c^2_{-1\;0\;0}} + \frac{U_{\,0\;1\;0}}{c^2_{0\;1\;0}} + \frac{U_{\,0-1\;0}}{c^2_{0-1\;0}} + \frac{U_{\,0\;0\;1}}{c^2_{0\;0\;1}} + \frac{U_{\,0\;0-1}}{c^2_{0\;0-1}},\\
\left[\frac{U}{c^2}\right]_{\sqrt{2}} &= \frac{U_{\,1\;0-1}}{c^2_{1\;0-1}} + \frac{U_{\,1\;0\;1}}{c^2_{1\;0\;1}} + \frac{U_{-1\;0\;1}}{c^2_{-1\;0\;1}} + \frac{U_{-1\;0-1}}{c^2_{-1\;0-1}} + \frac{U_{\,0-1-1}}{c^2_{0-1-1}} + \frac{U_{\,0-1\;1}}{c^2_{0-1\;1}}\\
&\quad + \frac{U_{\,0\;1-1}}{c^2_{0\;1-1}} + \frac{U_{\,0\;1\;1}}{c^2_{0\;1\;1}} + \frac{U_{-1-1\;0}}{c^2_{-1-1\;0}} + \frac{U_{\,1-1\;0}}{c^2_{1-1\;0}} + \frac{U_{-1\;1\;0}}{c^2_{-1\;1\;0}} + \frac{U_{\,1\;1\;0}}{c^2_{1\;1\;0}},\\
\left[\frac{U}{c^2}\right]_{\sqrt{3}} &= \frac{U_{-1-1-1}}{c^2_{-1-1-1}} + \frac{U_{\,1-1-1}}{c^2_{1-1-1}} + \frac{U_{-1\;1-1}}{c^2_{-1\;1-1}} + \frac{U_{\,1\;1-1}}{c^2_{1\;1-1}} + \frac{U_{-1-1\;1}}{c^2_{-1-1\;1}} + \frac{U_{\,1-1\;1}}{c^2_{1-1\;1}} + \frac{U_{-1\;1\;1}}{c^2_{-1\;1\;1}} + \frac{U_{\,1\;1\;1}}{c^2_{1\;1\;1}}.
\end{aligned}$$

The weights are determined by dispersion analysis, viz. by minimizing the phase velocity dispersion. In [6] the authors obtain $W_1 = 1.8395 \times 10^{-5}$, $W_2 = 0.8901$, $W_3 = 0.1099$, $W_{m_1} = 0.4965$, $W_{m_2} = 0.4510$, $W_{m_3} = 0.0525$ and $W_{m_4} = 0.4552 \times 10^{-5}$. Essentially, W_1 and W_{m_4} can be set to zero.

Organizing the discrete points in a vector standardly according to reshaping 3D $u(\omega)$ and $s(\omega)$ column-wise, (3.4)-(3.5) generate a matrix system, $A(\omega)u(\omega) = s(\omega)$. The matrix $A(\omega)$ has a symmetric pattern, but it is not Hermitian in view of the PML damping functions.

In figure (2) we show a comparison of the patterns of different matrices. The left figure shows the matrix generated by a $4th$ order centered finite difference scheme while the right figure shows the matrix associated with the optimal parsimonious mixed finite difference scheme used here (illustrated by figure (1)), which is also proved to be of $4th$ order accuracy [27]. The apparent bandwidth reduction plays an important role in the local matrix factorization developed below, in particular with a view to communication among processors in massive parallelization.

4. 3D massively parallel domain decomposition and multifrontal method. In this section, we briefly summarize the general multifrontal method, and introduce the domain decomposition in connection with a massively parallel structured approximate factorization (section 5), tailored to the 3D Helmholtz operator. The goal is to develop a strategy alternative to the full LU factorization.

4.1. 3D nested dissection based domain decomposition. It is well known that the traditional LU factorization of the matrix $A(\omega)$ will cause a catastrophic fill-in problem during Gaussian elimination. In the 2D case, the corresponding matrix is much smaller, and the fill-in issue is affordable with a large number of processors equipped with a relatively large memory. It has been recognized that the fill-in problem can be alleviated by nested dissection reordering [29, 9] of the original mesh. If N is the size of the matrix, the complexity of the factorization and the storage can thus be reduced to $\mathcal{O}(N^2)$ and $\mathcal{O}(N\log N)$ respectively [30].

Our algorithm for 3D nested dissection is illustrated in figure (3). It recursively divides the 3D mesh into subdomains and separators. At the first level (figure (3) (top)), a z direction separator divides the original mesh into three disjoint parts: two subdomains and the separator itself. The separator here plays a key role not only in data locality, but also in data communication. The grid points associated with the separators are ordered after those associated with subdomains. At the second level (figure (3) (middle)) and the third level (figure (3) (bottom)), y direction and x direction separators further divide the subdomains into lower level subdomains and separators, respectively. Based on this rule, much lower level separations can be pursued.

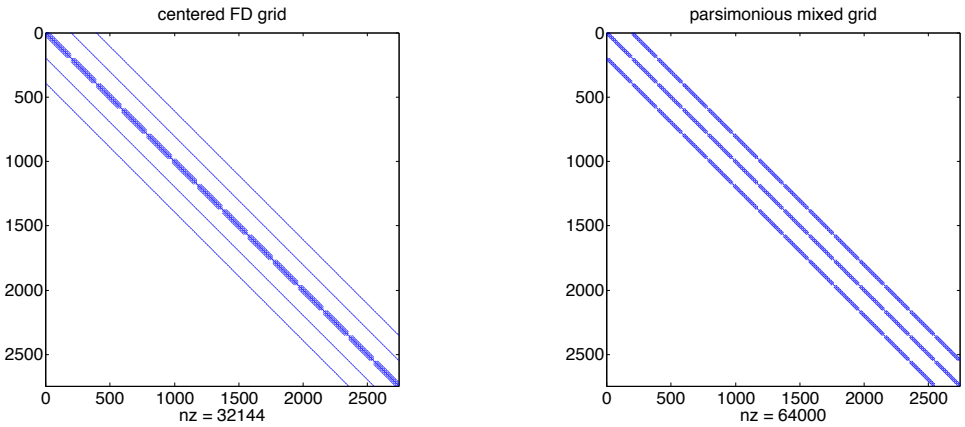

FIG. 2. *patterns of the global matrix $A(\omega)$ on a mesh $N_x = 14, N_y = 14, N_z = 14$, the size of $A(\omega)$ is $N_x \times N_y \times N_z = 2744$.* ***left****: $A(\omega)$ generated by 4th order centered FD;* ***right****: $A(\omega)$ generated by optimal parsimonious mixed FD illustrated in figure (1). Note that the bandwidth of the matrix on the right has been reduced significantly compared with the one on the left.*

We let lower level separators and subdomains be reordered prior to higher level ones. After reordering, we end up with a reordered linear system of equations illustrated in figure (4) (left) and an assembly tree [31] [32] [33] illustrated in figure (4) (right). Each node on the tree has a parent (except the root node separator) and two children (except the bottom level subdomains, which are called *leaves*). In the 3D physical domain, if two regions represented by two nodes in the tree have connections with each other, which means that the 27-point stencil spans across two regions, these two nodes are called *neighbors* of one another. figure (5) (left) shows an example of a lower level reordering of the global matrix $A(\omega)$ displayed in figure (2) for $N_x = N_y = N_z = 14$, while figure (5) (middle) shows a higher level reordering.

In the process of 3D nested dissection, we note that each "early" separator in the 3D domain has its own 2D nested dissection substructure. While reordering a 3D separator, we have to consider this substructure. Figure (5) (right) shows the global matrix pattern after incorporating the 2D substructures of each separator. This consideration plays a crucial role in the introduction of the 3D multifrontal method in that it minimizes the connections between each node and its neighbors and thus improves the efficiency significantly.

We note that the nested dissection algorithm itself automatically provides us a platform upon which 3D massively parallel domain decomposition can be performed. During the process of generating the assembly tree from the highest level to the bottom level, the original domain is decomposed. It appears natural that parallelization comes into play at a certain level, *Lpar* say, and each processor deals with one local tree from such a level downwards. We refer to *Lpar* as the *parallel level*. As a consequence, data communication will happen at levels *Lpar* and upward.

Figure (6) illustrates an example of the assembly tree, which is also displayed in figure (4) (right), and how we partition it into a global tree stored on all processors and local trees stored on each individual processor. The parallel level is indicated by the red dashed line. The black dashed boxes representing local trees are eliminated in parallel first. Then the global tree indicated by the blue dashed box above the parallel level is eliminated via massive data communications. Figure (7) illustrates the basic idea how we distribute the reordered global matrix, which has already been shown in figure (4) (left), into a given number of processors on the working platform. Suppose we have four processors which are denoted as 0, 1, 2, and 3. The submatrix stored on processor 0

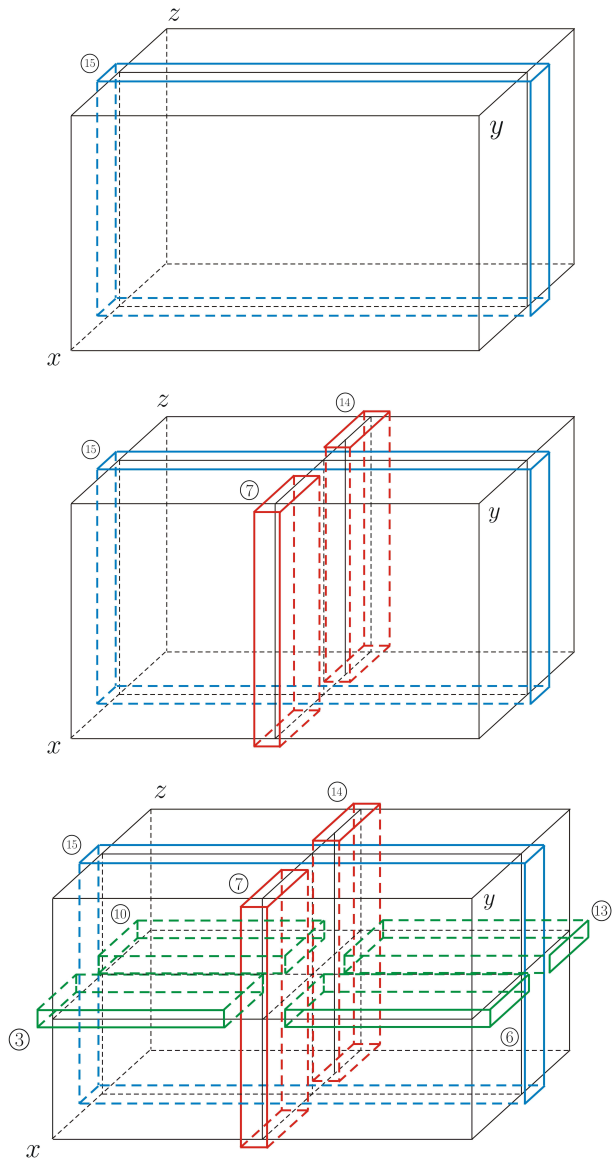

FIG. 3. *3D nested dissection based domain decomposition:* ***top****: level one;* ***middle****: level two;* ***bottom****: level three.*

is indicated by the shaded area in figure (7) (right). The submatrices stored on other processors can also be derived in the same way. We point out that the storage of the global matrix is not disjoint, which makes the communication happen at the overlapping interfaces that are represented as separators in the process of nested dissection.

4.2. Massively parallel multifrontal method. The basic idea of the multifrontal method is to reorganize the overall factorization of the nested dissection reordered matrix, shown in figure (4) (left), into partial updates and factorizations of smaller dense matrices according to the assembly tree ([32], [34], [33], [9]). It converts and accumulates large sparse factorizations into small dense ones. Thus it generally has good data locality and is more efficient than, for example, sparse direct LU factorization algorithms. Besides that, since the global matrix is reordered based on the assembly tree which is typically an example of fractal data structure, the multifrontal method is very well tailored for parallelization.

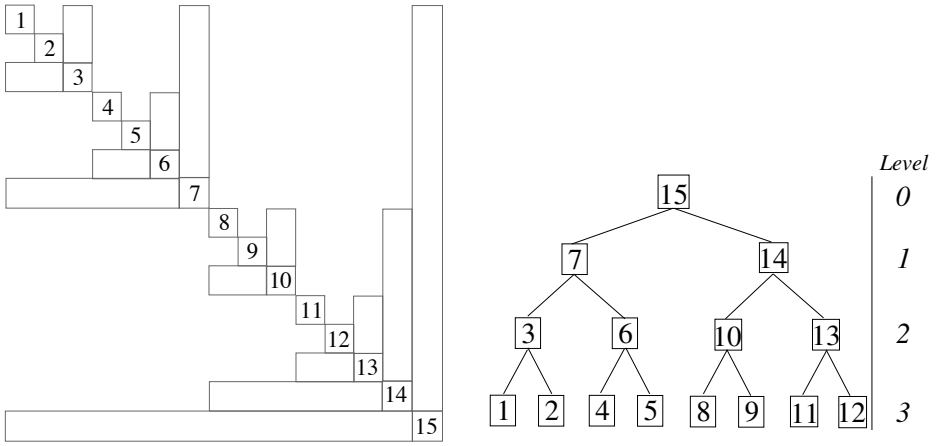

FIG. 4. ***left***: *the pattern of the global matrix $A(\omega)$ displayed on a single processor after nested dissection reordering illustrated in figure (3);* ***right***: *the global assembly tree structure of nested dissection reordering.*

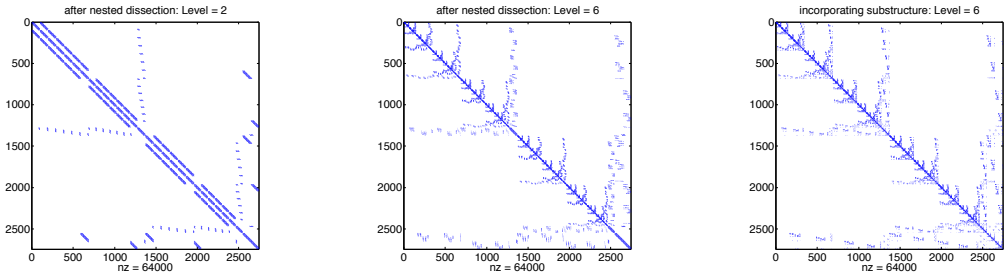

FIG. 5. *the nested dissection reordering of the global matrix $A(\omega)$, which is displayed in figure (2) (right), on a single processor. The mesh is $N_x = 14, N_y = 14, N_z = 14$, the size of $A(\omega)$ is $N_x \times N_y \times N_z = 2744$.* ***left***: *lower order reordering, level two;* ***middle***: *higher order reordering, level six;* ***right***: *higher order reordering incorporating the 2D elaborate substructure, level six.*

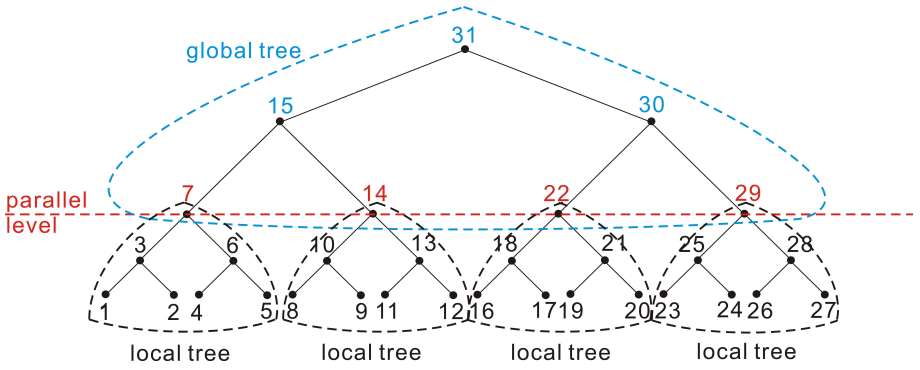

FIG. 6. *illustration of the 3D massively parallel multifrontal method. The red dashed line indicates the level (Lpar) that parallelization kicks in. The entire assembly tree, shown in figure (4) (right), is partitioned into two parts. The* ***first*** *part is the local tree stored in each individual 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. Communications only happen at this stage.*

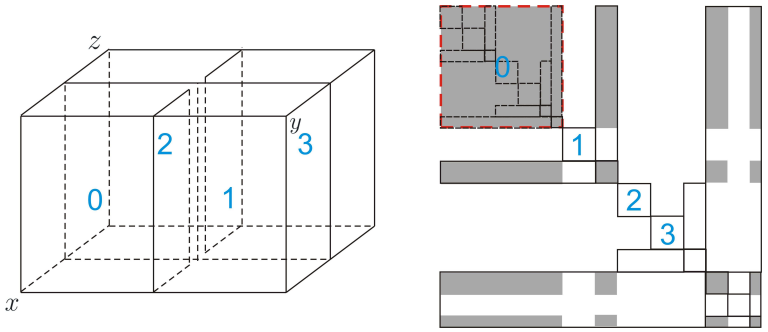

FIG. 7. *3D domain decomposition configuration for four processors which are denoted as 0, 1, 2, and 3.* ***left****: the original physical domain, shown in figure (3);* ***right****: the submatrix (shaded areas) stored locally on processor 0 imbedded in the global nested dissection reordered matrix, which is illustrated by figure (4) (left).*

There are two types of matrices occurring in the multifrontal method: *frontal matrices* and *update matrices*. We denote the current frontal matrix by $\mathcal{F}$, which basically glues and accumulates the submatrix associated with the current node and all submatrices pieces associated with the current neighbors in the global matrix, together into a dense matrix. Suppose we achieve a certain type of factorization, say $A = L_A U_A$, then we obtain the following deduction:

$$\begin{aligned} \mathcal{F} &= \begin{pmatrix} A & E \\ F & B \end{pmatrix} = \begin{pmatrix} L_A U_A & E \\ F & B \end{pmatrix} \\ &= \begin{pmatrix} L_A & 0 \\ F U_A^{-1} & I \end{pmatrix} \begin{pmatrix} U_A & L_A^{-1} E \\ 0 & B - (F U_A^{-1})(L_A^{-1} E) \end{pmatrix} \end{aligned} \tag{4.1}$$

Here, the matrix $\mathcal{U} = B - (F U_A^{-1})(L_A^{-1} E)$ is the Schur complement of the original frontal matrix $\mathcal{F}$, and is called the update matrix. Figure (8) displays the process how frontal matrices and update matrices are formed in the multifrontal method, starting from the global reordered matrix illustrated by figure (4) (left). The submatrix associated with the current node is indicated by the upper left red box. Other red boxes indicate the submatrices associated with all neighbors of the current node. Shaded areas indicate the optimal connections between the current node and its neighbors. Therefore, the frontal matrix is obtained by gluing all shaded areas together, which is illustrated by the lower right figure, and the update matrix is indicated by a dashed box imbedded in the frontal matrix.

Update matrices from children are assembled into the upper level frontal matrices with an operation called *extend-add* [9] [17] via both local stacking within a single processor and parallel stacking across all the processors. Figure (9) shows the process of such a parallel extend-add. S denotes the Schur complement or the update matrix (displayed in figure (8) (right)) of the current frontal matrix; $S = \mathcal{U}$ in this case.

Figure (6) indicates how we implement the multifrontal method in the context of massive parallelization. Below the parallel level, each processor eliminates its own local tree independently, ending up with a local dense update matrix waiting for communications with other processors. Up to this stage, no communications occur. We use the SIMD (Single Instruction Multiple Data) model. Above the parallel level, the frontal dense matrix will be stored and factorized over multiple processors. For example, in figure (6) the frontal matrix and update matrix of node 15 will be stored in two processors whose root nodes of local trees are 7 and 14, respectively. Communications only happen between 7 and 14. However, the frontal matrix of node 31 will be stored in four processors whose root nodes of local trees are 7, 14, 22 and 29, respectively. In this case, four processors labeled by 0, 1, 2 and 3 in figure (7) exactly have the root nodes 7, 14, 22 and 29, respectively. We note that the root node of the entire global tree will involve all the processors in the final stage of the factorization. (Thus, in our code development, we call the MPI subroutine MPI_Comm_create to

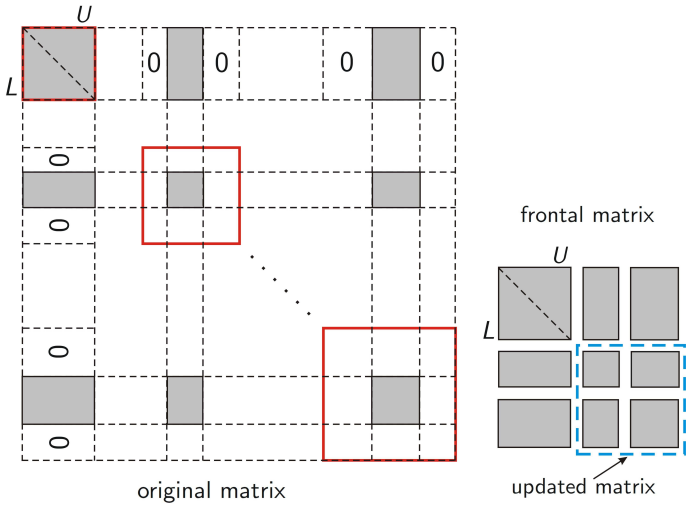

FIG. 8. *the illustration of frontal matrix and update matrix in the multifrontal method.* ***left****: the current node (upper left red box which is to be factorized), all of its corresponding neighbors (the rest red boxes) and their minimum connection areas (shaded boxes) in the original reordered global matrix* $A(\omega)$ *illustrated by figure (4) (left);* ***right****: frontal matrix formed by gluing all shaded areas in the original reordered global matrix together into a dense matrix, update matrix (blue dashed box) computed by the Schur complement of the frontal matrix.*

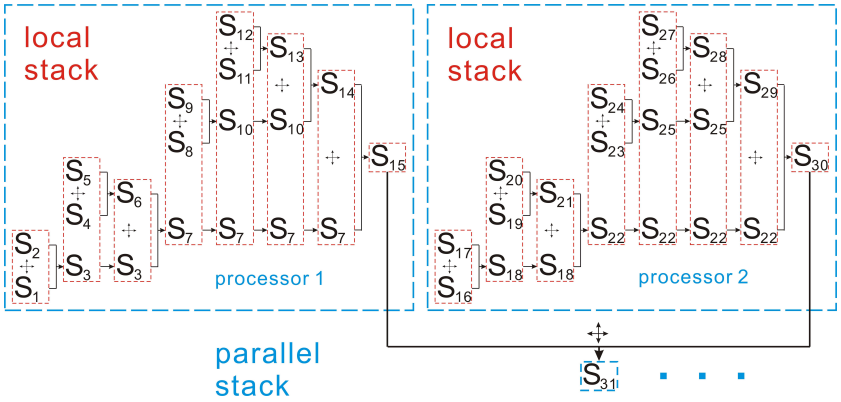

FIG. 9. *illustration of the extend-add procedure via local and parallel stacking: red dashed boxes indicate the local stacking within each processor, while blue boxes indicate parallel stacking over all the processors.* S *denotes the Schur complement or the update matrix of the current frontal matrix, illustrated by figure (8) (right).*

create a new communicator instead of using MPI_COMM_WORLD for each node operation, for the sake of minimizing communications.)

Furthermore, the solution to the global multifrontal matrix is achieved in parallel ([9] [17]) according the assembly tree displayed in figure (6), via forward and backward substitution indicated by the top-down and bottom-up going arrows, respectively, in figure (10).

5. Parallel HSS compression, factorization and solution of the dense frontal matrix $\mathcal{F}$. In section 4, we have already noted that the crucial ingredient in the parallel multifrontal method is how to factorize the frontal matrix $\mathcal{F}$, which is distributed over multiple processors at higher levels, with less storage and complexity. In this section, we present a certain compression strategy tailored to our frontal matrix, for the sake of storage economy and complexity reduction.

In many problems following the discretization of partial differential equations from (geo)physics, the off-diagonal blocks of both the frontal and update matrices have the low-rank property [11, 12,

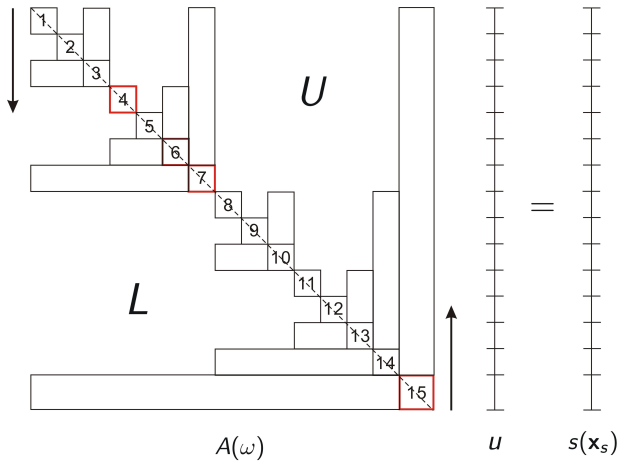

FIG. 10. *illustration of the global massively parallel multifrontal solver via forward and backward substitution indicated by the top-down and bottom-up arrows, respectively.*

13]. We note that our 3D Helmholtz problem is a typical example of this low-rank phenomenon, which implies that our frontal and update matrices in the multifrontal method can be compressed. Typical examples of rank-structured matrices include $\mathcal{H}$ matrices, $\mathcal{H}^2$ matrices, and quasiseparable matrices. Recently, more structured matrices called Hierarchically SemiSeparable (HSS) matrices [14, 15, 16, 17, 18] were introduced and analyzed, and corresponding fast algorithms have been proposed. However, most of the existing HSS algorithms are tailored for sequential implementation [19, 18]. Here, we propose a set of fast, massively parallel HSS compression, factorization and solution algorithms in the context of a parallel realization of the multifrontal method.

5.1. Parallel HSS compression of the dense frontal matrix $\mathcal{F}$. We start this subsection with an introduction of our compression strategy, called the rank revealing QR factorization [18], of a given matrix, say A, which possesses the low rank property. Full SVD is another candidate for compression. However, the complexity of a full SVD is much larger than that of a rank revealing QR factorization. We utilize the modified Gram-Schmidt with column pivoting (*mgsclpv*) method to realize the QR factorization with rank revealing; *mgsclpv* basically selects the column with the largest L_2 norm in the process of Gram-Schmidt and then permutes it, eventually yielding a permutation or column pivoting matrix P, such that $AP = QR$ and $|R_{11}| > |R_{22}| > ... > |R_{kk}| > ...$, in which R_{kk} are the diagonal entries of R. Then Q and R are truncated subject to a predefined tolerance τ (for instance 10^{-6}), such that $|R_{rr}| \geq \tau \times |R_{11}|$ and $|R_{r+1,r+1}| < \tau \times |R_{11}|$, in which r is the numerical rank of A. We obtain $AP = QR \approx Q_1 R_1$ and the eventual compressional form $A = Q_1\ (R_1 P^T)$. Figure (11) illustrates the process of our compression strategy based on rank revealing QR.

We utilize the rank revealing QR compression strategy to compress dense frontal matrix off-diagonal blocks with the low-rank property. Here we follow another binary tree structure called *HSS tree* to realize such compression of different levels, and also later on factorization and solution stages. The fully compressed form of the dense frontal matrix is called the *HSS matrix*.

DEFINITION 5.1. *HSS tree and HSS matrix [15]*
For a dense matrix $\mathcal{F}$, the HSS tree is defined to be a postordering binary tree on which each node is associated with a block in $\mathcal{F}$ such that it can be compressed via rank revealing QR illustrated in figure (11). Then $\mathcal{F}$ can be compressed into its corresponding HSS matrix form, with the following recursion:

$$\begin{pmatrix} D_i & U_i B_i V_j^T \\ U_j B_j V_i^T & D_j \end{pmatrix}, \quad with \quad U_i = \begin{pmatrix} U_{c_1} R_{c_1} \\ U_{c_2} R_{c_2} \end{pmatrix}, V_i = \begin{pmatrix} V_{c_1} W_{c_1} \\ V_{c_2} W_{c_2} \end{pmatrix}, \tag{5.1}$$

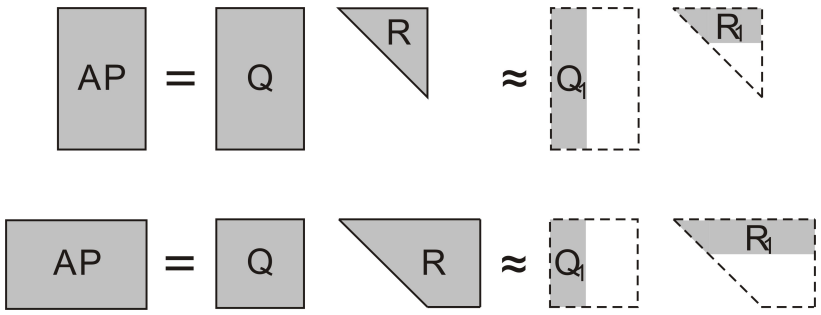

FIG. 11. *illustration of the QR factorization with the rank revealing strategy, which is realized by the modified Gram-Schmidt with column pivoting method. Given an example of a matrix A with low rank property, suppose we have a certain column pivoting matrix P, such that* $AP = QR$ *and* $|R_{11}| > |R_{22}| > ... > |R_{kk}| > ...$*, then we can truncate Q and R subject to a predefined tolerance level, which means* $AP = QR \approx Q_1 R_1$*. The eventual compressional form of A will be* $A = Q_1\ (R_1 P^T)$*. if the size of A is* $m \times n$*,* ***top*** *figure displays the case of* $m \geq n$ *and* ***bottom*** *figure displays the case of* $m < n$*.*

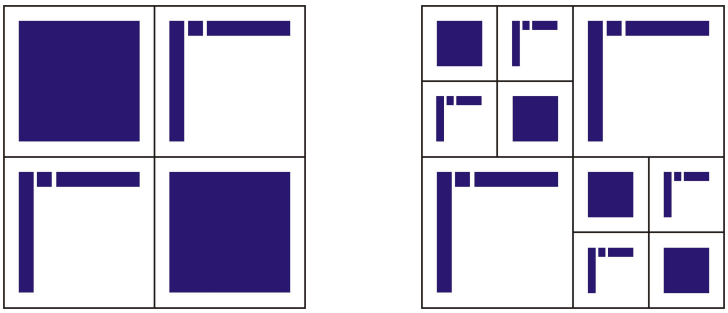

FIG. 12. *the HSS compression of the dense frontal matrix for different levels, resulted from rank revealing QR compression illustrated in figure (11).* ***left****: one level,* ***right****: two levels.*

where D, B, U, V, R, W *are called HSS factors or HSS generators,* c_1 *and* c_2 *indicate two children nodes of the node i if i is not a leaf of the HSS tree.*

Figure (12) shows two examples of HSS matrices resulting from recursive rank revealing QR compression illustrated in figure (11) and defined by eq.(5.1). Figure (12) (left) displays the pattern of an HSS matrix with one partition level.

$$\begin{pmatrix} D_1 & U_1 B_1 V_2^T \\ U_2 B_2 V_1^T & D_2 \end{pmatrix},$$

while figure (12) (right) displays the pattern of an HSS matrix with two partition levels.

$$\begin{pmatrix} D_1 & U_1 B_1 V_2^T & U_1 R_1 B_3 W_4^T V_4^T & U_1 R_1 B_3 W_5^T V_5^T \\ U_2 B_2 V_1^T & D_2 & U_2 R_2 B_3 W_4^T V_4^T & U_2 R_2 B_3 W_5^T V_5^T \\ U_4 R_4 B_6 W_1^T V_1^T & U_4 R_4 B_6 W_2^T V_2^T & D_4 & U_4 B_4 V_5^T \\ U_5 R_5 B_6 W_1^T V_1^T & U_5 R_5 B_6 W_2^T V_2^T & U_5 B_5 V_4^T & D_5 \end{pmatrix}.$$

The level we refer to here is the level of the corresponding HSS tree, not of the assembly tree mentioned in the multifrontal method (section 4). Thus, our algorithm is a hybrid of the outer multifrontal method (assembly tree) and the inner HSS strategy (HSS tree). We discuss our parallel HSS compression algorithm illustrated by an example of a 4×4 dense block matrix denoted by $\mathcal{F}$, presuming here that $\mathcal{F}$ is a frontal matrix generated in the process of the multifrontal method (section 4):

$$\mathcal{F} = \begin{pmatrix} D_1 & T_{12} & T_{14} & T_{15} \\ T_{21} & D_2 & T_{24} & T_{25} \\ T_{41} & T_{42} & D_4 & T_{45} \\ T_{51} & T_{52} & T_{54} & D_5 \end{pmatrix}, \tag{5.2}$$

where D_i are the diagonal blocks, T_{ij} are the off-diagonal blocks, and $\mathcal{F}$ is the initial dense frontal matrix before any compression.

- *parallel row compression 1*
  At this stage, we have the following compressions, which are carried out by the rank revealing QR strategy mentioned earlier:

$$\begin{pmatrix} T_{12} & T_{14} & T_{15} \end{pmatrix} \approx U_1 \begin{pmatrix} \widetilde{T}_{12} & \widetilde{T}_{14} & \widetilde{T}_{15} \end{pmatrix}, \qquad \begin{pmatrix} T_{21} & T_{24} & T_{25} \end{pmatrix} \approx U_2 \begin{pmatrix} \widetilde{T}_{21} & \widetilde{T}_{24} & \widetilde{T}_{25} \end{pmatrix}$$
$$\begin{pmatrix} T_{41} & T_{42} & T_{45} \end{pmatrix} \approx U_4 \begin{pmatrix} \widetilde{T}_{41} & \widetilde{T}_{42} & \widetilde{T}_{45} \end{pmatrix}, \qquad \begin{pmatrix} T_{51} & T_{52} & T_{54} \end{pmatrix} \approx U_5 \begin{pmatrix} \widetilde{T}_{51} & \widetilde{T}_{52} & \widetilde{T}_{54} \end{pmatrix}$$

  Here, the U_i are row factors which have been compressed out and will be stored in memory. $\widetilde{T}_{ij}$ will wait for further compressions at later stages. The partial compression form of $\mathcal{F}$ is:

$$\mathcal{F} \approx \begin{pmatrix} D_1 & U_1\widetilde{T}_{12} & U_1\widetilde{T}_{14} & U_1\widetilde{T}_{15} \\ U_2\widetilde{T}_{21} & D_2 & U_2\widetilde{T}_{24} & U_2\widetilde{T}_{25} \\ U_4\widetilde{T}_{41} & U_4\widetilde{T}_{42} & D_4 & U_4\widetilde{T}_{45} \\ U_5\widetilde{T}_{51} & U_5\widetilde{T}_{52} & U_5\widetilde{T}_{54} & D_5 \end{pmatrix} \tag{5.3}$$

- *parallel row compression 2*
  At this stage, we have the following compressions, which are carried out by the rank revealing QR strategy mentioned earlier:

$$\begin{pmatrix} \widetilde{T}_{14} & \widetilde{T}_{15} \\ \widetilde{T}_{24} & \widetilde{T}_{25} \end{pmatrix} \approx \begin{pmatrix} R_1 \\ R_2 \end{pmatrix} \begin{pmatrix} \widetilde{T}_{34} & \widetilde{T}_{35} \end{pmatrix}, \qquad \begin{pmatrix} \widetilde{T}_{41} & \widetilde{T}_{42} \\ \widetilde{T}_{51} & \widetilde{T}_{52} \end{pmatrix} \approx \begin{pmatrix} R_4 \\ R_5 \end{pmatrix} \begin{pmatrix} \widetilde{T}_{61} & \widetilde{T}_{62} \end{pmatrix}$$

  Here, the R_i are row factors which have been compressed out and will be stored in memory. $\widetilde{T}_{ij}$ will wait for further compressions at later stages. The partial compression form of $\mathcal{F}$ is:

$$\mathcal{F} \approx \begin{pmatrix} D_1 & U_1\widetilde{T}_{12} & U_1R_1\widetilde{T}_{34} & U_1R_1\widetilde{T}_{35} \\ U_2\widetilde{T}_{21} & D_2 & U_2R_2\widetilde{T}_{34} & U_2R_2\widetilde{T}_{35} \\ U_4R_4\widetilde{T}_{61} & U_4R_4\widetilde{T}_{62} & D_4 & U_4\widetilde{T}_{45} \\ U_5R_5\widetilde{T}_{61} & U_5R_5\widetilde{T}_{62} & U_5\widetilde{T}_{54} & D_5 \end{pmatrix} \tag{5.4}$$

- *parallel column compression 1*
  At this stage, we have the following compressions, which are carried out by the rank revealing QR strategy mentioned earlier:

$$\begin{pmatrix} \widetilde{T}_{21} \\ \widetilde{T}_{61} \end{pmatrix} \approx \begin{pmatrix} B_2 \\ \widehat{T}_{61} \end{pmatrix} V_1^T, \quad \begin{pmatrix} \widetilde{T}_{12} \\ \widetilde{T}_{62} \end{pmatrix} \approx \begin{pmatrix} B_1 \\ \widehat{T}_{62} \end{pmatrix} V_2^T,$$
$$\begin{pmatrix} \widetilde{T}_{34} \\ \widetilde{T}_{54} \end{pmatrix} \approx \begin{pmatrix} \widehat{T}_{34} \\ B_5 \end{pmatrix} V_4^T, \quad \begin{pmatrix} \widetilde{T}_{35} \\ \widetilde{T}_{45} \end{pmatrix} \approx \begin{pmatrix} \widehat{T}_{35} \\ B_4 \end{pmatrix} V_5^T$$

Here, the V_i are column factors which have been compressed out at this stage and will be stored in memory. The B_i are pivoting factors which are outputs of column compression. $\widehat{T}_{ij}$ will wait for further compressions at later stages. The partial compression form of $\mathcal{F}$ is:

$$\mathcal{F} \approx \begin{pmatrix} D_1 & U_1B_1V_2^T & U_1R_1\widehat{T}_{34}V_4^T & U_1R_1\widehat{T}_{35}V_5^T \\ U_2B_2V_1^T & D_2 & U_2R_2\widehat{T}_{34}V_4^T & U_2R_2\widehat{T}_{35}V_5^T \\ U_4R_4\widehat{T}_{61}V_1^T & U_4R_4\widehat{T}_{62}V_2^T & D_4 & U_4B_4V_5^T \\ U_5R_5\widehat{T}_{61}V_1^T & U_5R_5\widehat{T}_{62}V_2^T & U_5B_5V_4^T & D_5 \end{pmatrix} \tag{5.5}$$

- *parallel column compression 2*
  At this stage, we have the following compressions, which are carried out by the rank revealing QR strategy mentioned earlier:

$$\begin{pmatrix} \widehat{T}_{34} & \widehat{T}_{35} \end{pmatrix} \approx B_3 \begin{pmatrix} W_4^T & W_5^T \end{pmatrix}, \qquad \begin{pmatrix} \widehat{T}_{61} & \widehat{T}_{62} \end{pmatrix} \approx B_6 \begin{pmatrix} W_1^T & W_2^T \end{pmatrix}$$

  Here, the W_i are column factors which have been compressed out and will be stored in memory. The B_i are pivoting factors which are the outputs of parallel column compression. The final compression form of $\mathcal{F}$ is:

$$\mathcal{F} \approx \begin{pmatrix} D_1 & U_1B_1V_2^T & U_1R_1B_3W_4^TV_4^T & U_1R_1B_3W_5^TV_5^T \\ U_2B_2V_1^T & D_2 & U_2R_2B_3W_4^TV_4^T & U_2R_2B_3W_5^TV_5^T \\ U_4R_4B_6W_1^TV_1^T & U_4R_4B_6W_2^TV_2^T & D_4 & U_4B_4V_5^T \\ U_5R_5B_6W_1^TV_1^T & U_5R_5B_6W_2^TV_2^T & U_5B_5V_4^T & D_5 \end{pmatrix} \tag{5.6}$$

  which can also be rewritten in a recursive form as:

$$\mathcal{F} \approx \begin{pmatrix} \begin{pmatrix} D_1 & U_1B_1V_2^T \\ U_2B_2V_1^T & D_2 \end{pmatrix} & \begin{pmatrix} U_1R_1 \\ U_2R_2 \end{pmatrix} B_3 \begin{pmatrix} W_4^TV_4^T & W_5^TV_5^T \end{pmatrix} \\ \begin{pmatrix} U_4R_4 \\ U_5R_5 \end{pmatrix} B_6 \begin{pmatrix} W_1^TV_1^T & W_2^TV_2^T \end{pmatrix} & \begin{pmatrix} D_4 & U_4B_4V_5^T \\ U_5B_5V_4^T & D_5 \end{pmatrix} \end{pmatrix} \tag{5.7}$$

  The compression is finalized at this stage. There are altogether six types of factors which are compressed out: D, B, U, V, R and W, with the D factors unchanged compared with the initial diagonal blocks.

Up to now, we have obtained the final HSS compressed form, given in eq.(5.6), of the original dense frontal matrix eq.(5.2). The leading structure of the HSS matrices satisfies the recursive relationship defined in eq.(5.1), which can be seen in eq.(5.7). In our implementation, during the HSS compression stage, we call the MPI subroutines MPI_Send and MPI_Recv. Conceptually, most communications occur upon merging two $\widetilde{T}_{ij}$ or $\widehat{T}_{ij}$ blocks stored on different processors while compressing. However, in practice, the entire block of $\widetilde{T}_{ij}$ or $\widehat{T}_{ij}$ should not be sent from one processor to another in that it will slow down the efficiency dramatically. Instead, the object that is transferred from one processor to another by MPI_Send and MPI_Recv is the L_2 column norms of two $\widetilde{T}_{ij}$ or $\widehat{T}_{ij}$ blocks stored on separate processors. The value of the norm controls which column is to be permuted and when to stop subject to a predefined tolerance during the process of rank revealing QR. Thus transferring the norm vector instead of the whole block matrix improves the efficiency significantly.

5.2. Parallel HSS factorization of the HSS compressed form of the dense frontal matrix $\mathcal{F}$. After we obtain an approximate HSS compressed form (eq.(5.6)) of the original dense frontal matrix (eq.(5.2)), the next step is to factorize the HSS matrix (eq.(5.6)) in parallel, motivated

by [14]. Here, we would like to use an explicit parallel ULV factorization instead of LU factorization. Figure (13) illustrates the basic idea of parallel HSS factorization, using a block 2×2 matrix denoted by H as an example. We start from figure (13) (upper left), which is the HSS matrix shown in figure (12) (left).

We have

$$H = \begin{pmatrix} D_1 & U_1 B_1 V_2^T \\ U_2 B_2 V_1^T & D_2 \end{pmatrix}. \tag{5.8}$$

Suppose that the size of U_i $(i = 1, 2)$ is $m_i \times k_i$, and $m_i \geq k_i$, After QL factorization of $U_i = Q_i \begin{pmatrix} 0 \\ \widetilde{U}_i \end{pmatrix}$, we obtain the following equivalent form:

$$H = \begin{pmatrix} D_1 & Q_1 \begin{pmatrix} 0 \\ \widetilde{U}_1 \end{pmatrix} B_1 V_2^T \\ Q_2 \begin{pmatrix} 0 \\ \widetilde{U}_2 \end{pmatrix} B_2 V_1^T & D_2 \end{pmatrix}. \tag{5.9}$$

We multiply by Q_1^T and Q_2^T on the left in parallel, yielding that the first $m_i - k_i$ rows of the off-diagonal blocks are zeroed out,

$$\begin{pmatrix} Q_1^T & 0 \\ 0 & Q_2^T \end{pmatrix} \begin{pmatrix} D_1 & U_1 B_1 V_2^T \\ U_2 B_2 V_1^T & D_2 \end{pmatrix} = \begin{pmatrix} Q_1^T D_1 & \begin{pmatrix} 0 \\ \widetilde{U}_1 B_1 V_2^T \end{pmatrix} \\ \begin{pmatrix} 0 \\ \widetilde{U}_2 B_2 V_1^T \end{pmatrix} & Q_2^T D_2 \end{pmatrix}, \tag{5.10}$$

leaving the size of $\widetilde{U}_i$ to be $k_i \times k_i$. Figure (13) (upper right) displays the consequence of this row multiplication: D_1 and D_2 are updated to $Q_1^T D_1$ and $Q_2^T D_2$ respectively, while U_1 and U_2 are reduced to $\widetilde{U}_1$ and $\widetilde{U}_2$ respectively. We then carry out LQ factorization (transposed QR) in parallel for the upper part of $Q_1^T D_1$ and $Q_2^T D_2$, respectively. The output orthogonal matrices at this step are denoted by q_1 and q_2. We multiply eq.(5.10) on the right by q_1 and q_2, and obtain

$$\begin{pmatrix} Q_1^T & 0 \\ 0 & Q_2^T \end{pmatrix} \begin{pmatrix} D_1 & U_1 B_1 V_2^T \\ U_2 B_2 V_1^T & D_2 \end{pmatrix} \begin{pmatrix} q_1 & 0 \\ 0 & q_2 \end{pmatrix} = \begin{pmatrix} Q_1^T D_1 q_1 & \begin{pmatrix} 0 \\ \widetilde{U}_1 B_1 (q_2^T V_2)^T \end{pmatrix} \\ \begin{pmatrix} 0 \\ \widetilde{U}_2 B_2 (q_1^T V_1)^T \end{pmatrix} & Q_2^T D_2 q_2 \end{pmatrix}. \tag{5.11}$$

Figure (13) (lower left) shows that after this column multiplication, V_1 and V_2 are updated to $q_1^T V_1$ and $q_2^T V_2$, respectively. Additionally, $Q_i^T D_i q_i$ $(i = 1, 2)$ consists of three factors: a lower triangular matrix, a flat block matrix, and a residual diagonal block matrix. Eventually, the residual diagonal block matrices, merged with the residual off-diagonal block matrices, are transferred to the higher level for further recursive factorization, which is depicted in figure (13) (lower right). Upon reaching the highest level of the HSS tree, the LU factorization is carried out for the final residual diagonal matrix.

In summary, Such a ULV factorization scheme is both stable and parallelizable. The use of orthonormal factorizations leads to the stability. The intermediate factorizations are done locally along the nodes of HSS tree.

5.3. Parallel HSS solution imbedded in the parallel global multifrontal solution. We still use a block 2×2 matrix as an example. Our target is solve the HSS compressed frontal matrix system (eq.(5.8)) with HSS factorization (eq.(5.11))

$$\begin{pmatrix} D_1 & U_1 B_1 V_2^T \\ U_2 B_2 V_1^T & D_2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}. \tag{5.12}$$

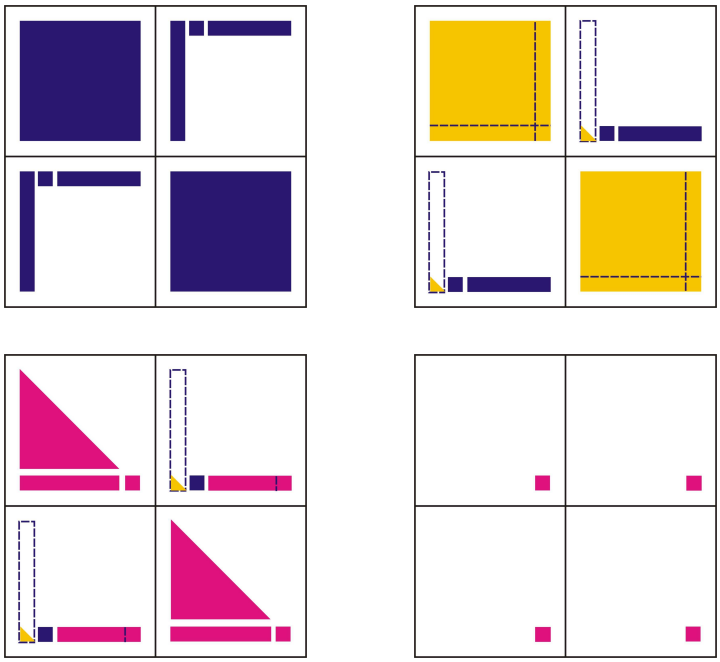

FIG. 13. *an example of parallel factorization of the HSS matrix with only one level, which is also shown in figure (12) (left).* ***upper left****: the original HSS compressed form.* ***upper right****: parallel multiplication by orthogonal matrices Q_1^T and Q_2^T row-wise, respectively.* ***lower left****: parallel multiplication by orthogonal matrices q_1 and q_2 column-wise, respectively.* ***lower right****: the residual blocks are transferred to higher level factorization. If the highest level is reached, LU factorization is carried out.*

With parallel row multiplications by Q_1^T and Q_2^T, and parallel column multiplications by q_1 and q_2, we obtain the following equivalent formulation of eq.(5.12):

$$\underbrace{\begin{pmatrix} Q_1^T & 0 \\ 0 & Q_2^T \end{pmatrix}\begin{pmatrix} D_1 & U_1B_1V_2^T \\ U_2B_2V_1^T & D_2 \end{pmatrix}\begin{pmatrix} q_1 & 0 \\ 0 & q_2 \end{pmatrix}}\begin{pmatrix} q_1^Tx_1 \\ q_2^Tx_2 \end{pmatrix} = \begin{pmatrix} Q_1^Tb_1 \\ Q_2^Tb_2 \end{pmatrix} \tag{5.13}$$

From eq.(5.11) we note that the part indicated by an underbrace in eq.(5.13) can be replaced by the right hand side of eq.(5.11), which results in:

$$\begin{pmatrix} Q_1^TD_1q_1 & \begin{pmatrix} 0 \\ \widetilde{U}_1B_1(q_2^TV_2)^T \end{pmatrix} \\ \begin{pmatrix} 0 \\ \widetilde{U}_2B_2(q_1^TV_1)^T \end{pmatrix} & Q_2^TD_2q_2 \end{pmatrix}\begin{pmatrix} q_1^Tx_{11} \\ q_1^Tx_{12} \\ q_2^Tx_{21} \\ q_2^Tx_{22} \end{pmatrix} = \begin{pmatrix} Q_1^Tb_{11} \\ Q_1^Tb_{12} \\ Q_2^Tb_{21} \\ Q_2^Tb_{22} \end{pmatrix}, \tag{5.14}$$

in which $x_1 = \begin{pmatrix} x_{11} \\ x_{12} \end{pmatrix}$, $x_2 = \begin{pmatrix} x_{21} \\ x_{22} \end{pmatrix}$. We write $\widetilde{x}_{11} = q_1^Tx_{11}, \widetilde{x}_{12} = q_1^Tx_{12}, \widetilde{x}_{21} = q_2^Tx_{21}$ and $\widetilde{x}_{22} = q_2^Tx_{22}$, and $\widetilde{b}_{11} = Q_1^Tb_{11}, \widetilde{b}_{12} = Q_1^Tb_{12}, \widetilde{b}_{21} = Q_2^Tb_{21}$ and $\widetilde{b}_{22} = Q_2^Tb_{22}$, and illustrate the matrix system of eq.(5.14) in figure (14) (right).

From figure (14) (right), we observe that $\widetilde{x}_{11}$ and $\widetilde{x}_{21}$ can be obtained in parallel by solving two lower triangular systems, while $\widetilde{x}_{12}$ and $\widetilde{x}_{22}$ are obtained after updating their corresponding right-hand-sides via $\widetilde{x}_{11}$ and $\widetilde{x}_{21}$, followed by higher level triangular system solvers. Eventually, after obtaining $\widetilde{x}_{11}, \widetilde{x}_{12}, \widetilde{x}_{21}$ and $\widetilde{x}_{22}$, and upon setting $x_1 = q_1\widetilde{x}_1$ and $x_2 = q_2\widetilde{x}_2$ in which $\widetilde{x}_1 = \begin{pmatrix} \widetilde{x}_{11} \\ \widetilde{x}_{12} \end{pmatrix}, \widetilde{x}_2 = \begin{pmatrix} \widetilde{x}_{21} \\ \widetilde{x}_{22} \end{pmatrix}$, we thus obtain an entire parallel solution strategy.

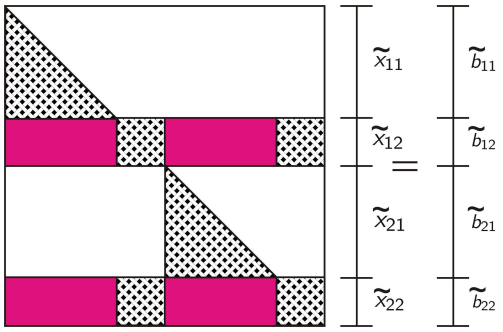

FIG. 14. *illustration of the massively parallel HSS solver, which is resulted from the HSS factorization stage shown in figure (13), and is imbedded in the global multifrontal solver illustrated by figure (10).*

The parallel HSS solution strategy introduced here is recursively imbedded in the global parallel multifrontal solution via forward and backward substitution; this is illustrated in figure (10).

6. Performance tests and numerical examples. In this section, we show the numerical performances of our parallel structured approximate solver, and various numerical examples. Let r be the maximum off-diagonal numerical rank of the dense frontal matrices in the multifrontal procedure, then it is shown in [31] that the total complexity of the structured multifrontal method is $O(rN\log N)$, and the total storage is $O(rN\log(r\log N))$. The cost for solving the linear system with one right-hand-side is $O(rN\log(r\log N))$.

Preliminary numerical tests have been done, and the numerical results are shown in Tables (6.1) and (6.2). Table (6.3) shows how the computational costs and storage scale when the order of the matrix N increases. The scaling factors of the exact factorization are consistent with the complexity and storage counts. The scaling factors of the our structured approximate solver decrease quickly, compared with the classical multifrontal solver (MUMPS). In our future tests with large matrix sizes, we expect the factors to get close to 8.

matrix order (N)	12^3	24^3	48^3	96^3
factorization cost (flops)	$7.49E7$	$4.58E9$	$1.72E11$	$4.74E12$
solution cost (flops)	$1.15E6$	$2.91E7$	$4.63E8$	$5.41E9$
storage (nnz)	$3.45E5$	$4.90E6$	$6.39E7$	$7.64E8$

TABLE 6.1

numerical results (in flops: floating operations per second) and storage (in nnz: number of non-zeros) for the HSS structured direct solver with relative tolerance 10^{-4}.

matrix order (N)	12^3	24^3	48^3	96^3
factorization cost (flops)	$3.36E7$	$2.32E9$	$1.58E11$	$1.05E13$
solution cost (flops)	$5.91E5$	$1.15E7$	$2.13E8$	$4.09E9$
storage (nnz)	$2.95E5$	$5.75E6$	$1.06E8$	$2.04E9$

TABLE 6.2

numerical results (in flops: floating operations per second) and storage (in nnz: number of non-zeros) for the classical LU multifrontal direct solver.

Figure (15) displays the storage comparison between the LU multifrontal solver and the HSS structured multifrontal solver. We notice that the storage for the structured solver becomes much smaller than the LU solver with increasing the order of the matrix N.

	factorization			system solution			storage		
N	12^3	24^3	48^3	12^3	24^3	48^3	12^3	24^3	48^3
classical LU	69.0	68.1	66.5	19.6	18.5	19.2	19.5	18.4	19.2
structured HSS	61.1	37.6	27.6	25.3	15.9	11.7	14.2	13.0	11.9

TABLE 6.3

scaling factors $(scale(8N)/scale(N))$ with doubling mesh size, which means the order of the matrix becomes eight times larger.

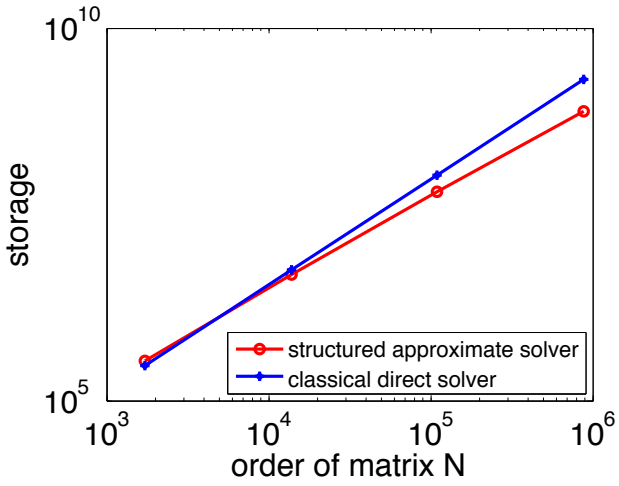

FIG. 15. *storage comparison between the structured approximate solver and the classical multifrontal direct solver (MUMPS). We note that the storage for structured solver becomes much smaller than the one for the classical LU direct solver with increasing the order of the matrix.*

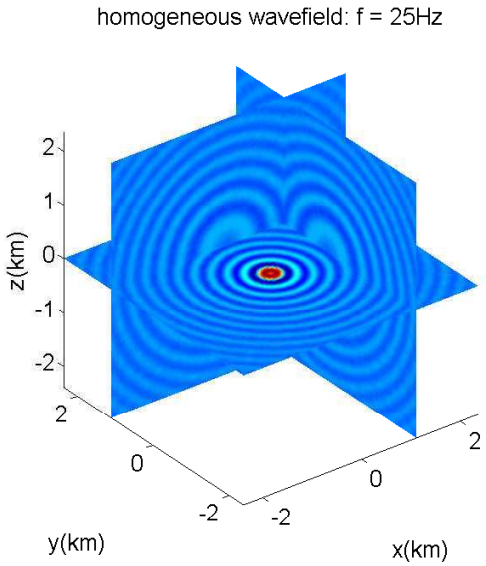

FIG. 16. *time-harmonic $(f = 25Hz)$ wavefield computed on a $96 \times 96 \times 96$ mesh with the homogeneous velocity $c = 2500m/s$, and the source location $x_s = 0km, y_s = 0km, z_s = 0km$.*

Finally, we show some time-harmonic wavefield modeling examples generated by our massively parallel structured approximate solver. Figure (16) displays the time-harmonic ($f = 25Hz$) wavefield computed on a $96 \times 96 \times 96$ mesh with homogeneous velocity $c = 2500m/s$, and the source location $x_s = 0km, y_s = 0km, z_s = 0km$.

Figure (17) (top left) displays a velocity model with a 3D low velocity lens zone in the middle. In this model, the highest velocity is $2500m/s$ and the lowest velocity is $1500m/s$. The source location is $x_s = -1.2km, y_s = -1.2km, z_s = 0km$.

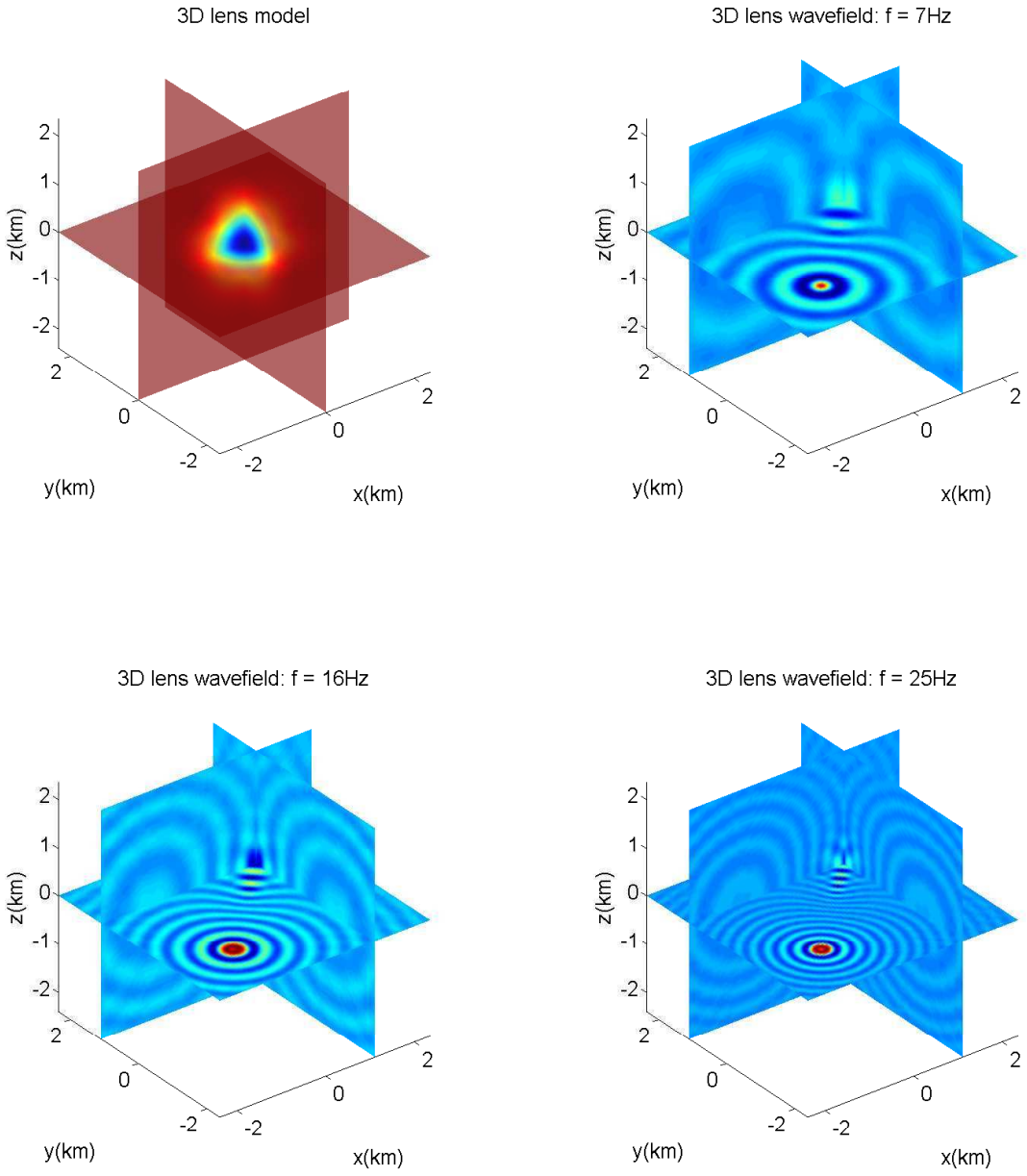

FIG. 17. ***top left:*** *3D lens model with mesh* $96 \times 96 \times 96$*; the highest velocity is* $2500m/s$ *and the lowest velocity is* $1500m/s$*; a source location is* $x_s = -1.2km, y_s = -1.2km, z_s = 0km$. ***top right:*** *time-harmonic* ($f = 7Hz$) *wavefield.* ***bottom left:*** *time-harmonic* ($f = 16Hz$) *wavefield.* ***bottom right:*** *time-harmonic* ($f = 25Hz$) *wavefield.*

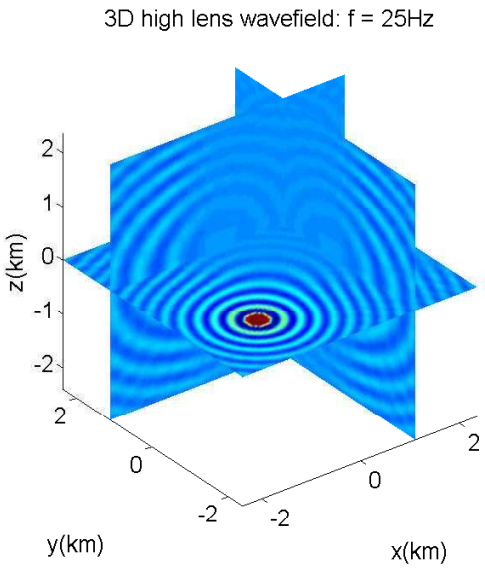

FIG. 18. *time-harmonic ($f = 25Hz$) wavefield computed on a $96 \times 96 \times 96$ mesh with a 3D high lens velocity model whose highest velocity is $4000m/s$ and lowest velocity is $2500m/s$, and a source location $x_s = -1.2km, y_s = -1.2km, z_s = 0km$.*

Figure (17) (top right) displays the time-harmonic ($f = 7Hz$) wavefield. Figure (17) (bottom left) displays the time-harmonic ($f = 16Hz$) wavefield. Figure (17) (bottom right) displays the time-harmonic ($f = 25Hz$) wavefield. The caustic can be seen from these three wavefields.

In contrast with the low velocity lens model, we also compute a time-harmonic ($f = 25Hz$) wavefield, which is displayed in figure (18), on a $96 \times 96 \times 96$ mesh with a 3D high lens velocity model whose highest velocity is $4000m/s$ and the lowest velocity is $2500m/s$, with a source location $x_s = -1.2km, y_s = -1.2km, z_s = 0km$.

7. Conclusions. We presented the discretization and the solution of the inhomogeneous Helmholtz equation in 3D. We resorted to a parsimonious mixed grid finite differences scheme, which is of fourth order accuracy, for discretizing the Helmholtz operator and Perfect Matched Layer boundaries, resulting in a non-Hermitian matrix. We made use of a 3D nested dissection based domain decomposition, and introduced an approximate direct solver by developing a new parallel Hierarchically SemiSeparable (HSS) matrix compression, factorization and solution approaches. We casted our massive parallelization in the framework of the multifrontal method. 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. $\mathcal{O}(rN\log N)$, while the storage is linear as well, $\mathcal{O}(N\log(r\log N))$, if r is the maxium numerical rank of all off-diagonal blocks in the multifrontal procedure.

8. Acknowledgments. We would like to thank the members, BP, ConocoPhillips, ExxonMobil, StatoilHydro and Total, of the Geo-Mathematical Imaging Group for financial support.

REFERENCES

[1] V. Etienne, J. Virieux, S. Operto, A massively parallel time-domain discontinuous galerkin method for 3d elastic wave modeling, SEG Expanded Abstracts 28 (2009) 2657–2661.
[2] Y. Erlangga, C. Oosterlee, C. Vuik, A novel multigrid based preconditioner for heterogeneous helmholtz problems, SIAM J. Scient. Comput. 27 (2006) 1471–1492.
[3] C. Riyanti, Y. Erlangga, R.-E. Plessix, W. Mulder, C. Vuik, C. Oosterlee, A new iterative solver for the time-harmonic wave equation, Geophysics 71 (2006) E57–E63.

[4] R.-E. Plessix, A helmholtz iterative solver for 3d seismic-imaging problems, Geophysics 72 (2007) SM185–SM194.
[5] C. Riyanti, A. Kononov, Y. Erlangga, C. Vuik, C. Oosterlee, R.-E. Plessix, W. Mulder, A parallel multigrid-based preconditioner for the 3d heterogeneous high-frequency helmholtz equation, Journal of Computational Physics 224 (2007) 431–448.
[6] S. Operto, J. Virieux, P. Amestoy, J. L'Excellent, L. Giraud, H. Hadj Ali, 3d finite-difference frequency-domain modeling of visco-acoustic wave propagation using a massively parallel direct solver: A feasibility study, Geophysics 72 (2007) SM195–SM211.
[7] S. Operto, J. Virieux, A. Ribodetti, J. Anderson, Finite-difference frequency-domain modeling of viscoacoustic wave propagation in 2d tilted transversely isotropic (tti) media, Geophysics 74 (2009) T75–T95.
[8] I. Duff, J. Reid, The multifrontal solution of indefinite sparse symmetric linear equations, ACM Trans. Math. Software 9 (1983) 302–325.
[9] J. Liu, The multifrontal method for sparse matrix solution: Theory and practice, SIAM Review 34 (1992) 82–109.
[10] E. Agullo, A. Guermouche, J. L'Excellent, 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 (2008) 296–317.
[11] M. Bebendorf, W. Hackbusch, Existence of $\mathcal{H}$-matrix approximants to the inverse fe-matrix of elliptic operators with l^∞ coefficients, Numer. Math. 95 (2003) 1–28.
[12] M. Bebendorf, Efficient inversion of galerkin matrices of general second-order elliptic differential operators with nonsmooth coefficients, Math. Comp. 74 (2005) 1179–1199.
[13] S. Chandrasekaran, P. Dewilde, M. Gu, On the numerical rank of the off-diagonal blocks of schur complements of discretized elliptic pdes, preprint.
[14] S. Chandrasekaran, M. Gu, T. Pals, A fast *ulv* decomposition solver for hierarchically semiseparable representations, SIAM J. Matrix Anal. Appl. 28 (2006) 603–622.
[15] S. Chandrasekaran, M. Gu, X. Li, J. Xia, Some fast algorithms for hierarchically semiseparable matrices, Technical report LBNL-62897.
[16] S. Chandrasekaran, P. Dewilde, M. Gu, W. Lyons, T. Pals, A fast solver for hss representations via sparse matrices, SIAM J. Matrix Anal. Appl. 29 (2006) 67–81.
[17] J. Xia, S. Chandrasekaran, M. Gu, X. Li, Superfast multifrontal method for large structured linear systems of equations, SIAM J. Matrix Anal. Appl. 31 (2009) 1382–1411.
[18] J. Xia, S. Chandrasekaran, M. Gu, X. Li, Fast algorithms for hierarchically semiseparable matrices, Numer. Linear Algebra Appl.
[19] S. Wang, M. de Hoop, J. Xia, Seismic inverse scattering via helmholtz operator factorization and optimization, Journal of Computational Physics (2010) to appear.
[20] L. Vega, B. Perthame, Morrey-Campanato estimates for Helmholtz equations, Journal of Functional Analysis 164 (1999) 340–355.
[21] E. Fouassier, Morrey-Campanato estimates for Helmholtz equations with two unbounded media, Proceedings of the Royal Society of Edinburgh. Section A. Mathematics 135 (2005) 767–776.
[22] A. Tarantola, Inverse problem theory, Elsevier, New York, 1987.
[23] L. Sirgue, G. Pratt, Waveform inversion under realistic conditions: mitigation of non-linearity, SEG Technical Program Expanded Abstracts 22 (2003) 694.
[24] Y. Cha, C. Shin, 2d laplace-fourier-domain full waveform inversion for both velocity and density models: An experience of the 2004 bp velocity-analysis benchmark dataset, SEG Technical Program Expanded Abstracts 28 (2009) 2258–2262.
[25] N. Koo, C. Shin, Y. Cha, Sequentially ordered single-frequency 2-d acoustic waveform inversion in the laplace-fourier domain, SEG Technical Program Expanded Abstracts 28 (2009) 2248–2252.
[26] C. Shin, W. Ha, A comparison between the behavior of objective functions for waveform inversion in the frequency and laplace domains, Geophysics 73 (2008) VE119–VE133.
[27] B. Hustedt, S. Operto, J. Virieux, Mixed-grid and staggered-grid finite-difference methods for frequency-domain acoustic wave modeling, Geophys. J. Int. 157 (2004) 1269–1296.
[28] E. Turkel, A. Yefet, Absorbing pml boundary layers for wave-like equations, Applied Numerical Mathematics 27 (1998) 533–557.
[29] J. George, Nested dissection of a regular finite element mesh, SIAM J. Numer. Anal. 10 (1973) 345–363.
[30] A. Hoffman, M. Martin, D. Rose, Complexity bounds for regular finite difference and finite element grids, SIAM J. Numer. Anal. 10 (1973) 364–369.
[31] J. Xia, M. Gu, Robust structured multifrontal factorization and preconditioning for discretized pdes, `http://math.purdue.edu/~xiaj/work/mfprec.pdf`, preprint.
[32] J. Liu, The role of elimination trees in sparse factorization, SIAM J. Matrix Anal. Appl. 18 (1990) 134–172.
[33] S. Eisenstat, J. Liu, The theory of elimination trees for sparse unsymmetric matrices, SIAM J. Matrix Anal. Appl. 26 (2005) 686–705.
[34] E. N. J.R. Gilbert, Predicting structure in nonsymmetric sparse matrix factorizations: Graph Theory and Sparse Matrix Computation, Springer-Verlag, 1993.