I S A V

Journal of Theoretical and Applied

Vibration and Acoustics

journal homepage: http://tava.isav.ir

MPI- and CUDA- implementations of modal finite difference method for P-SV wave propagation modeling

Hossein Samadiyeh, **Reza Khajavi** [*]

Earthquake Research Center, Ferdowsi University of Mashhad, Mashhad, Iran

ARTICLE INFO

Article history:
Received 18 July 2016

Received in revised form 21 August 2016

Accepted 17 October 2016

Available online 25 December 2016

Keywords:
Finite difference method

Graphics Processing Unit (GPU)

Message Passing Interface (MPI)

Modal

Wave propagation

ABSTRACT

Among different discretization approaches, Finite Difference Method (FDM) is widely used for acoustic and elastic full-wave form modeling. An inevitable deficit of the technique, however, is its sever requirement to computational resources. A promising solution is parallelization, where the problem is broken into several segments, and the calculations are distributed over different processors. For the present FD routines, however, such parallelization technique inevitably needs domain-decomposition and inter-core data exchange, due to the coupling of the governing equations. In this study, a new FD-based procedure for seismic wave modeling, named as ‘Modal Finite Difference Method (MFDM)” is introduced, which deals with the simulation in the decoupled modal space; thus, neither domain-decomposition nor inter-core data exchange is anymore required, which greatly simplifies parallelization for both MPI- and CUDA implementations over CPUs and GPUs. With MFDM, it is also possible to simply cut off less-significant modes and run the routine for just the important ones, which will effectively reduce computation and storage costs. The efficiency of the proposed MFDM is shown by some numerical examples.

©2016 Iranian Society of Acoustics and Vibration, All rights reserved.

1. Introduction

The Finite Difference Method (FDM) is supposed to be one of the most efficient numerical procedures for solving differential equations appeared in the science and engineering fields. Among various numerical procedures, the FDM is an old and matured method. The initial attempts for numerical solution of the wave equation involved somehow a FD-based procedure (Alterman and Karal, 1968 [1], Boore, 1970, 1972 [2, 3]). Even with the appearance of new

[*] Corresponding Author.
E-mail address: : rezakhajavi@ferdowsi.um.ac.ir (R. Khajavi)

http://dx.doi.org/

sophisticated numerical methods such as the finite element method (FEM) (Zienkiewicz and Taylor, 2000; Cohen et al., 1994 [4, 5]), pseudo-spectral methods (Kosloff and Baysal, 1982; Kosloff et al., 1984; Fornberg, 1987 [6-8]), the spectral element method (SEM) (Komatitsch and Tromp, 1999 [9]) and the discrete Galerkin Method (DGM) (Grote et al., 2006 [10]), the FDM is still appealing and attractive for many researchers. Such attraction is undoubtedly due to simplicity of the FDM.

Since early employments of FDM in seismic wave-field modeling, different approaches were established to further develop this numerical procedure. Various customized versions of the method for wave-simulation is thus offered and new modified FD-based procedures are still under development. The most recognized among these variations is the staggered finite difference method which solves differential wave equations over staggered grids (Virieux, 1984, 1986, Levander, 1988, Graves, 1996 [11-13]). Eventhough the staggered procedures are proved to be efficient and are vastly used for wave-field modeling (Moczo et al. 2004, 2007 [14, 15]), the FD procedures with simple non-staggered grids are still attractive for their simplicity and ease of implementation. Several researches are recently dedicated to improve the FD solutions over simple grids. Higher-order FD operators (Dablain, 1986 [16]), DSF (Mikhailenko, 2000 [17]) and compact FD procedures (Lele, 1992 [18]) are some extensions to the primary version of the time-domain FD method. They try to improve FD results over simple and coarse meshes (Pirozzoli, 2007 [19]).

The main drawback of the FDTD simulations is that they are drastically data intensive and need a tremendous amount of floating-point calculations which necessitate computational resources. Appropriate FDTD simulations require fine meshes for both spatial and temporal domains to prevent numerical errors such as instability or numerical dispersion. Thus, most realistic simulations inevitably exceed memory limits on a single computer and have to be distributed over a cluster of computers. It is now more than a decade that parallelization on high-performance PC (personal computer) clusters has become a promising approach toward numerical seismic wave-propagation modeling (Bohlen, 2002 [20]).

The FDTD method is naturally appropriate for parallelization; since, for every FD spatial node, the field values of the neighboring cells in the previous - and not the current - time steps are required. However, such parallelization inevitably introduces extra computational costs due to boundary data exchange between adjacent spatial-domain divisions.

Different parallel implementations of FDTD have been proposed in the literature, with a wide range of various parallel architectures and hardware configurations. They get use of different programming models and protocols such as Open MultiProcessing (OpenMP, 2009 [21]) for single computing node or Massage Passing Interface (MPI) (Gropp et al., 1999 [22]) for parallelization across multiple nodes.

There has been a great deal of interest toward employing graphics processing units (GPUs) to accelerate FDTD simulations in recent years. While caching mechanisms on CPUs are not appropriate for FDTD memory access patterns, GPUs provide effective control on data-loading onto a small shared memory which significantly enhances memory access (Shams and Sadeghi, 2011 [23]). Modern GPUs are now installed on desktop computers and offer extremely efficient well-performed computations especially when calculations are duly independent of each other. For an efficient solution with GPU, one may partition the problem into independent subproblems which are distributed over streaming multiprocessors (SMs) through some thread blocks (Mehra et al., 2012 [24]).

Parallelization of FDTD in its conventional form has some fundamental issues which avoid its implementation on any present parallel hardware. Though some architectures are proved to be more efficient than others, there still remains a great deal of details and optimizing considerations such as domain-decomposition techniques, load distribution and memory management to be dealt with. Surely, a time-domain FD technique with no need for domain-decomposition and boundary-data transfer is highly welcomed.

In the current study, a new FD-based method is developed to solve the elastic wave equation numerically. This procedure named as the 'Modal Finite Difference Method (MFDM)' explicitly solves the wave equation in the time domain within the modal space and gets use of the modal orthogonality of the FD matrix operator. Through this procedure, the explicit FD equations are decoupled which facilitates parallel CUDA (Compute Unified Device Architecture)-based computerized implementation without resorting to domain-decomposition techniques. More importantly is the fact that manipulating some few MFD equations might provide responses within acceptable tolerance.

2. Theory

In this study the P-SV differential wave equation is considered for the homogeneous elastic medium:

$$\rho\, u_{tt} = ([\lambda + 2\mu\,]\, u_x)_x + (\mu\, u_z)_z + (\lambda\, w_z)_x + (\mu\, w_x)_z + f_x$$

(1)

$$\rho\, w_{tt} = ([\lambda + 2\mu\,]\, w_z)_z + (\mu\, w_x)_x + (\lambda\, u_z)_x + (\mu\, u_x)_z + f_z$$

where $(.)_{\blacksquare}$ denotes partial derivative with respect to $\blacksquare$ which might be the spatial coordinates *x, z* or the time variable *t*. λ and μ are the Lame` constants, ρ is the mass density and *u* and *w* are the displacement components along *x* and *z* directions respectively. f_x and f_z are the body forces along *x* and *z* that vanish for all medium points other than those associated with the seismic source.

The above differential equations might be recast into the following explicit algebraic equations by the use of appropriate FD stencil for spatial derivatives and second-order central FD stencil for temporal ones:

$$\boldsymbol{u}^{t+\Delta t} = \boldsymbol{A}\boldsymbol{u}^{t} + \boldsymbol{B}\boldsymbol{u}^{t-\Delta t} + \boldsymbol{f}^{\,t} \tag{2}$$

where $\boldsymbol{A}$ and $\boldsymbol{B}$ are some $n \times n$ time-independent FD-based matrices and $\boldsymbol{u}$ is the *n*-dimensional displacement vector (*n* is the number of degrees of freedom). $\boldsymbol{f}^{\,t}$ is the *n*-dimensional source equivalent-force vector which is usually defined as:

$$\boldsymbol{f} = \boldsymbol{f}_x \;\; s(t) \tag{3}$$

where $\boldsymbol{f}_x$ is the vector of source spatial distribution and $s(t)$ is the source temporal function. The superscripts appeared in Eq. (2) represent time steps which are sampled at every Δt. Given $\boldsymbol{\Phi}$ as the eigenvector matrix of $\boldsymbol{A}$, Eq. (2) is rewritten as:

$$\boldsymbol{u}_q^{t+\Delta t} = \boldsymbol{A}_q \boldsymbol{u}_q^t + \boldsymbol{B}_q \boldsymbol{u}_q^{t\text{-}\Delta t} + \boldsymbol{f}_q^{\,t} \tag{4}$$

in which

$$\boldsymbol{A_q} = \Phi^{-1} \boldsymbol{A} \, \Phi \tag{5}$$

$$\boldsymbol{B_q} = \Phi^{-1} \boldsymbol{B} \, \Phi \tag{6}$$

are $\boldsymbol{A}$- and $\boldsymbol{B}$- equivalents in the modal space, and

$$\boldsymbol{u_q} = \Phi^{-1} \boldsymbol{u} \tag{7}$$

and

$$\boldsymbol{f}_q = \Phi^{-1} \boldsymbol{f} = \Phi^{-1} \boldsymbol{f}_x \, s(t) = \boldsymbol{f}_{qx} \, s(t) \tag{8}$$

are the modal displacement and source-force vectors respectively. $\boldsymbol{A}_q$ is clearly a diagonal matrix which entails eigenvalues of $\boldsymbol{A}$. The system of equations (4) is decoupled provided that $\boldsymbol{B}$ is also diagonal; i.e. the eigenvectors of $\boldsymbol{A}$ are also orthogonal with respect to $\boldsymbol{B}$. For the second-order central FD stencil being used for the time-discretization of Eq. (2), $\boldsymbol{B}_\text{q}$=-$\boldsymbol{I}$ and thus Eq. (4) represents the following n decoupled explicit equations:

$$u_{q(i)}^{t+\Delta t} = \lambda_{(i)} \, u_{q(i)}^{t} - u_{q(i)}^{t-\Delta t} + f_{qx(i)} \, s(t), \quad i = 1,2,...,n \tag{9}$$

where $\lambda_{(i)}$ is the i^{th} eigenvector of $\boldsymbol{A}$. Once $\boldsymbol{u}_q$ is found for every time step t, the displacement vector $\boldsymbol{u}$ is calculated by:

$$\boldsymbol{u}^t = \Phi \boldsymbol{u}_q^t \tag{10}$$

$f_{qx(i)}$ in Eq. (9) shows the i^{th} mode contribution in the total response. Based on this, it is possible to contribute the most important modes up to the desired accuracy. For this purpose, the square matrix Φ in Eqs. (5-8) are replaced by a rectangular matrix Φ_m; this matrix involves the m eigenvectors corresponding to the most significant modes. The pseudo-inverse of the rectangular matrix Φ_m must thus be used rather than Φ^{-1}.

3. Implementation

As pointed out in the previous section, MFDM is a two-step procedure. At the first step, denoted as the pre-processing step, matrix $\boldsymbol{A}$ of Eq. (2) as well as its modal characteristics (i.e. eigenvalues and eigenvectors) should be found. The second step, which is the main process, implements the Modal FD method. Since both steps are computationally expensive, they should be implemented on parallel systems.

3.1. Pre-processing (Mode calculations)

Pre-processing (eigenvalues-eigenvectors calculation for matrix $\boldsymbol{A}$) seems to be the major bottleneck and costly step in MFDM. As will be seen in numerical examples (Section 4), point seismic sources in 2D problems inevitably excite a large range of modes of $\boldsymbol{A}$; thus, calculation of a few eigenvalues-eigenvectors for the $\boldsymbol{A}$ matrix is certainly not sufficient to achieve satisfactory results. Because of this, MFDM seems to be especially appropriate for applications where it is required to deal with a large number of different seismic sources in a constant velocity model. Focal search procedures are among such applications. The requirement for several runs of the MFD for different seismic sources may justify costly eigen-system calculations in the pre-processing step. It is however reminded that for underground simulations, since the velocity model might not exactly be determined, an approximation of the unknown fields such as displacement or stress suffices. Moreover, through mode separation, it would become possible to contribute more confident modes (having larger wavelengths).

Since matrix $\boldsymbol{A}$ is a sparse non-symmetric matrix, the well-performed Arnoldi algorithm might simply be adopted for the pre-processing step. This procedure employs an orthogonal projection onto a Krylov subspace (Saad, 1980, 2003 [25, 26]). In order to implement the Arnoldi algorithm on GPU, Compressed Sparse Row (CSR) or Block Compressed Sparse Row (BCSR) format storing is appropriate for the sparse matrix $\boldsymbol{A}$ (Choi et al., 2010 [27]). Other more sophisticated storage formats such as Coordinate List (COO), Ellpack (ELL), Hybrid (HYB) and Diagonal (DIA) formats may also be used (Bell and Gerland, 2009 [28]). Either NVIDIA CUBLAS [29] or CUSPARSE libraries might be employed to compute the eigenvalues and eigenvectors of $\boldsymbol{A}$. It is also notable that for some special structured forms of the sparse matrix $\boldsymbol{A}$, there exist explicit formulae for eigenvalues-eigenvectors calculations (see Chang et al., 2009 [30]).

3.2. MFDM implementation

Different parallel techniques for FDTD implementation have been offered in the literature. The most recognized and applicable ones are those with OpenMP (OpenMP, 2009 [21]) and Message Passing Interface (MPI) (Gropp et al., 1999 [22]) which are customized for parallelization over one single node and multiple nodes, respectively. There has also been a wide interest toward using Graphic Processing Units (GPUs) for parallel FDTD implementation recently. In this section, MPI- and CUDA-based parallel implementations of MFDM are explained.

3.2.1. MPI-based implementation

MPI is one of the most widely used parallel processing standards. Actually, it is a library of functions which are called from C, C++ or Fortran routines. The MPI library is well remarked for its standardization, portability, performance and functionality (Gropp et al., 1999 [22]).

For an MPI-based parallel implementation of MFDM, it is required that each processor is fetched with data related to a portion of system modes according to Eq. (9). The master node reads modal data and sends packages of eigenvalues $\{\lambda_i\}_{i=1}^{k<n}$ and modal forces $\{f_{q(i)}\}_{i=1}^{k}$ to the slaves just once. Each slave then starts up with the received data and updates its own modal displacements $\{u_{q(i)}\}_{i=1}^{k}$ according to Eq. (9) at each time-step *t*. After each update, the slave sends the modal displacements back to the master and starts the new run for the following step.

Figure 1 is the pseudo-code for the proposed MPI-based MFDM parallel implementation. It is notable that the scheme significantly reduces slave-master communications; actually, except at the start of the parallel routine, there remains only a unilateral slave-to-master data stream.

3.2.2. CUDA-based implementation

Nowadays, the programmable Graphics Processing Unit (GPU) is known as an alternative to the traditional CPUs networks for performing massively parallel calculations. In particular, CUDA, the programming model introduced in 2006 by NVIDIA [29], has proven to be a convenient, general purpose framework to exploit the parallel computing engine on NVIDIA GPUs. Not only new programs are routinely implemented in CUDA to solve complex computational problems (Dugan, et al., 2013 [31]), but also many massively parallel scientific packages, developed essentially on the MPI and/or OpenMP models during decades are now being equipped with GPU capabilities.

A GPU consists of an array of a large number of independent execution units and a global memory. The global memory is accessible both by the host CPU and all the execution units of the GPU. Each execution unit naturally has its own set of processors, registers and memory units and is called a multi-threaded Streaming Multiprocessor (SM). The host CPU program distributes the work over the SMs. Each SM can execute a large number of threads simultaneously. Since MFDM skips domain-decomposition and thus no boundary-data is exchanged, each thread can independently execute the FD kernel function. The pseudo-code for CUDA-based MFDM parallel implementation is shown in Fig. 2. The host CPU allocates memory and copies modal data (eigenvalues $\{\lambda_i\}_{i=1}^{k<n}$ and modal forces $\{f_{q(i)}\}_{i=1}^{k}$) onto the device once at the beginning of the application. Thereafter, almost no device-host data transfers are necessary during simulation. This gives rise to an implementation which considerably profits from GPU performance and enables one to competently employ the typical PCI (Peripheral Component Interconnect) express bus data exchange issues. By calling the kernel, the device (GPU) starts up with the data and calculations for each mode (assigned to a single GPU thread) are followed. All operations are done per-mode with a SIMD (single instruction multiple data) approach. Each thread updates its own modal displacements $\{u_{q(i)}\}_{i=1}^{k}$ according to Eq. (9) after which modal displacements are copied from device to host and the displacement vector is retrieved according to Eq. (10). In order to synchronize the received data gathered from different threads, the computed data are first lumped in the shared memory (pertaining to the blocks) and then might be transferred to the global memory.

```
Initialize MPI
Master:
    read eigenvalues and eigenvectors
    Compute $f_q$
    Send  portions to workers
    Receive results ($\mathbf{u_q}$) from slaves
    Calculate $\mathbf{u} = \boldsymbol{\Phi}\ \mathbf{u_q}$
    Print $\mathbf{u}$
Slaves:
    Receive data from master
    Time step starts:
    Apply force $\mathbf{f_q}^t$
    Update $\mathbf{u_q}$ (Eq. 9)
    Send $\mathbf{u_q}$ to master
   Time step ends.
```

Fig. 1. Pseudo-code for MPI-based implementation of MFDM

4. Numerical examples

A 4 km × 4 km 2D spatial domain is considered for the following two cases:

1. Homogeneous domain of Granite Rock (first Lamé constant λ = 23.08 GPa, second Lamé constant μ = 15.38 GPa and density ρ = 2700 kg/m^3)

2. Non-homogenous domain comprised of two layers: a lower layer of Granite rock (depth of the lower layer H_1 = 3 km, λ_1 = 23.08 GPa, μ_1 = 15.38 GPa and ρ_1 = 2700 kg/m^3) and an upper layer of a young sediment deposit (depth of the upper layer H_2 = 1 km, λ_2 = 60.0 MPa, μ_2 = 60.0 MPa, and ρ_2 = 2000 kg/m^3).

```
Host :
        Allocate memory on Device
        Initialize variables
        Read eigenvalues and eigenvectors
        Copy data from Host to Device
        Call kernel
        Copy results from Device to Host
        Calculate $\boldsymbol{u} = \boldsymbol{\Phi}\ \boldsymbol{u_q}$
        Print $\boldsymbol{u}$
Device:
        Identify threads ID
        Time step starts:
            Apply force $\boldsymbol{f_q}^t$
            Update $\boldsymbol{u_q}$ ( Eq. 9)
        Time step ends.
```

Fig. 2. Pseudo-code for CUDA-based implementation of MFDM

A point source with the following form:

$$f^t = g(\boldsymbol{x})\, s(t) \tag{11}$$

is placed at the center of the domain. g(x) is defined as:

$$g(\boldsymbol{x}) = \delta(\boldsymbol{x} - \boldsymbol{x}_s)\,\hat{\boldsymbol{u}}_f \tag{12}$$

in which $\boldsymbol{x}_s$ is the source location, $\delta(.)$ is the Dirac function and $\hat{\boldsymbol{u}}_f$ is the direction of the body force. For the present example, $\hat{\boldsymbol{u}}_f$ is defined for the two cases of explosive source and shearing double-couple. The source time function $s(t)$ is represented by a Ricker wavelet with the maximum amplitude of 1000 Pa and time duration T=1.0 s. The spatial domain is discretized by Δx=Δz=100 m and is bounded with zero displacement for the sake of simplicity. The time step Δt=0.005 s is applied for temporal discretization. Second-order FD stencils are employed for both space and time discretization.

Calculations for both homogeneous and non-homogeneous examples are performed using the following FDTD schemes:

M1. Serial FDTD

M2. Serial modal FDTD

M3. Parallel MPI-based conventional FDTD

M4. Proposed parallel MPI-based modal FDTD

M5. Proposed parallel CUDA-based modal FDTD

4.1. The homogeneous case

For MDFM implementation, the characteristic matrix ***A*** is first developed and its eigenvalues are calculated according to the procedure introduced in Section 3.1. Figures 3(a-b) show diagrams for the eigenvalues $\{\lambda_i\}_{i=1}^{n}$ and modal forces $\{f_{qx(i)}\}_{i=1}^{n}$ for all present modes respectively. As pointed out earlier, modal forces might be supposed as modal contribution factors. From Fig. 3-b, it is concluded that higher modes are contributed (and might be more effective) in the total response. Figure 3-c depicts the sorted absolute modal forces from which it is possible to select an appropriate number of modes according to the required level of accuracy; e.g. 95% of the overall response might be achieved by contribution of at least 2000 modes.

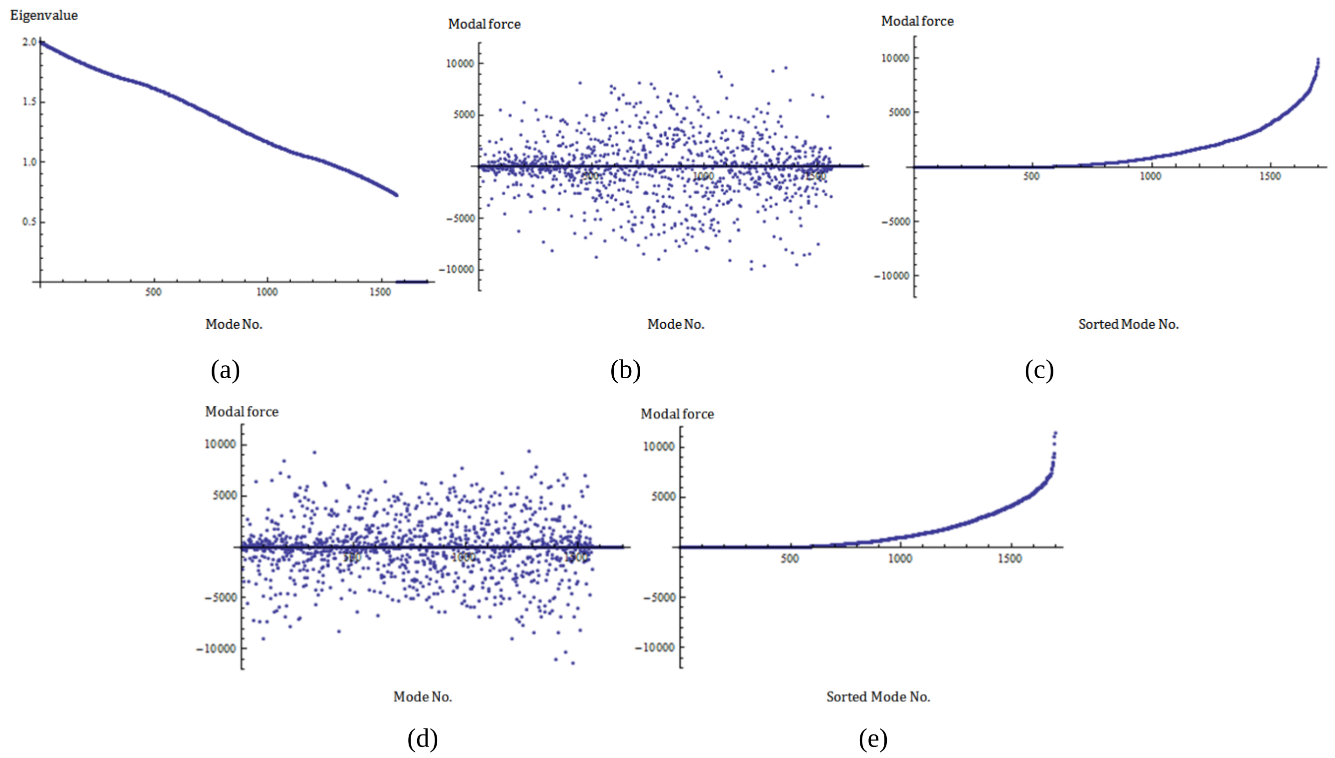


Fig. 3. (a) Eigenvalues, (b) modal force (explosive source) , (c) sorted modal force (explosive source), (d) modal force (for double couple) and (e) sorted modal force (for double couple) in the homogeneous case

Figures 4(a-c) illustrates the first 15 mode shapes as well as the most effective 15 modes for the two cases of explosive and double-couple mid-sources respectively. The mode number associated with each mode shape is reported underneath.

Figure 5 illustrates the convergence of the displacement field at *t*=1.6 s as more modes are contributed for the case of explosive source. It is well observed that an overall demonstration of the wave pattern is appropriately comprised with the contribution of the first 500 modes. More details are determined when higher modes are supplied. Interestingly, mode superposition according to mode numbers (Fig. 5-a) converges slightly better than that the result obtained according to the superposition of most significant modes (Fig. 5-b). Such feature is also observed for the case of double-couple source as shown by Fig. 6.

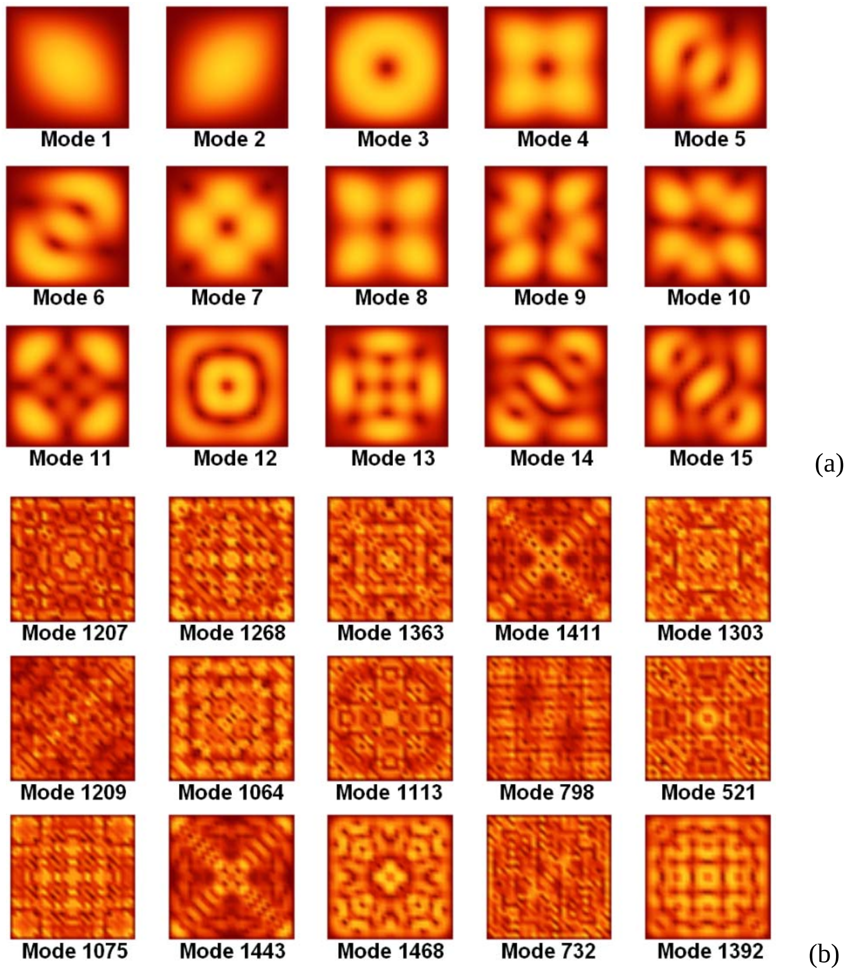
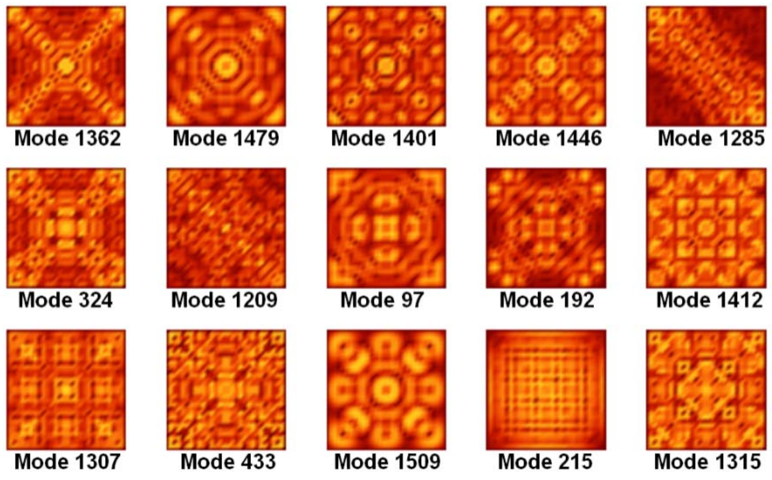
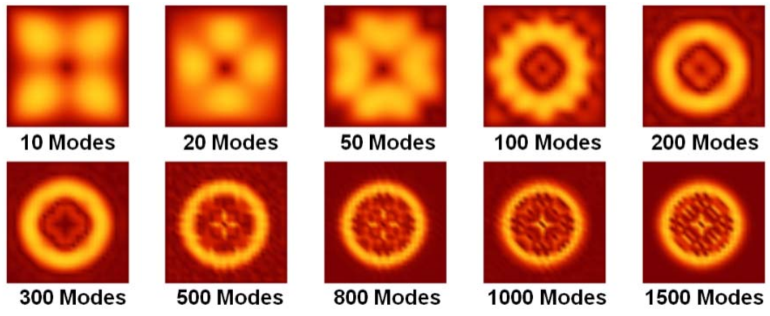


Fig. 4. Displacements of: (a) the first 15 modes , (b) the most effective 15 modes for explosive source, and (c) the most effective 15 modes, for double-couple source (homogeneous case)

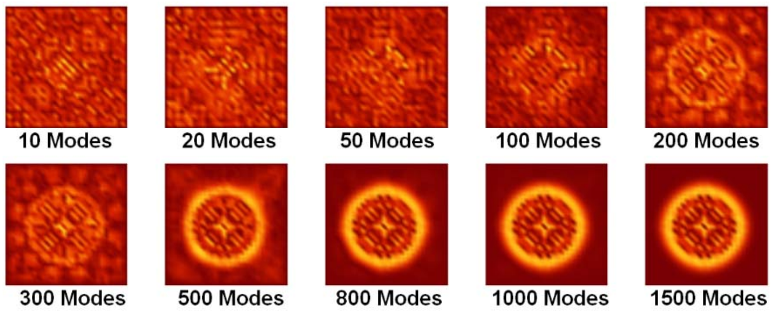


(c)

Fig. 4. (Continued)



(a)



(b)

Fig. 5. Snapshots of the displacement wave at t=1.6 s for increasing number of mode contribution (explosive-source homogeneous case): (a) Modes sorted according to eigenvalues and (b) Modes sorted according to their modal contribution values

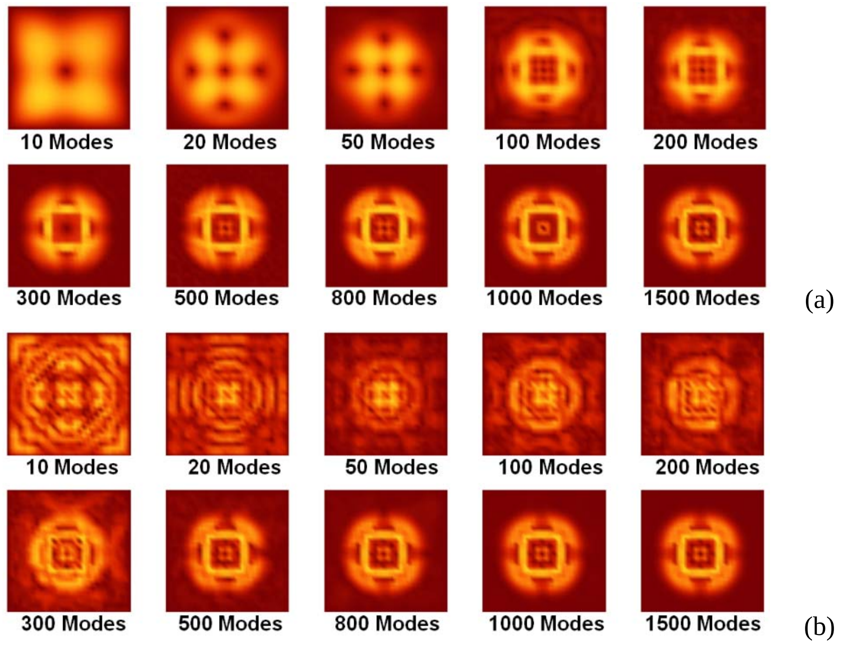


Fig. 6. Snapshots of the displacement wave at t=1.6 s for increasing number of mode contribution (double couple-source homogeneous case): (a) Modes sorted according to eigenvalues and (b) Modes sorted according to their modal contribution values

4.2. The 2-Layer case

Figures 7(a-b) show diagrams for the eigenvalues $\{\lambda_i\}_{i=1}^{n}$ and modal forces $\{f_{qx(i)}\}_{i=1}^{n}$ in all present modes respectively. As pointed out earlier, modal forces might be assumed as modal contribution factors. A comparison of Figs. 3-b and 7-b shows that the contribution of the first modes are effectively reduced in the non-homogeneous case; however, no remarkable difference exists between the diagrams in Figs. 3-c and 7-c which shows that the number of modes required for a specific level of accuracy has not much changed for the two cases. In other words, the contribution to the total response is more uniformly distributed among the modes for the homogeneous case in comparison with the non-homogeneous case.

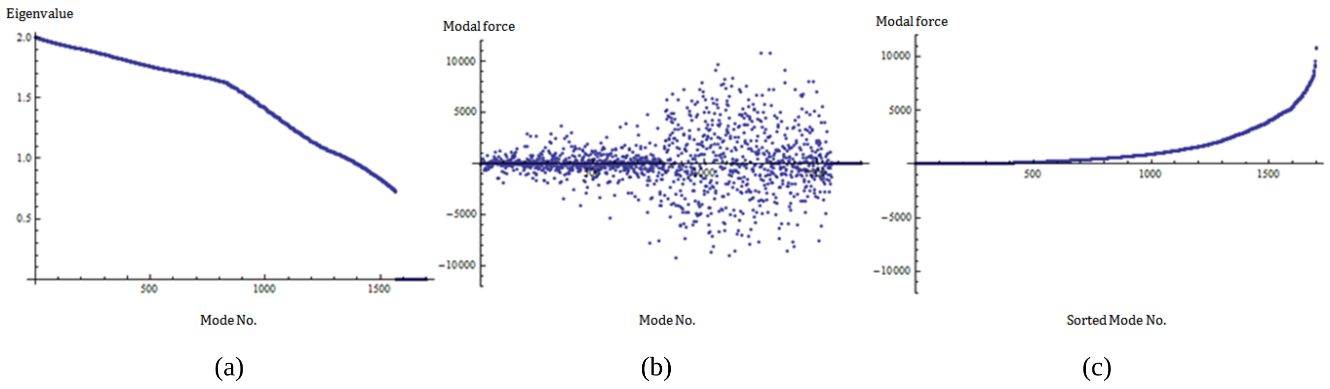
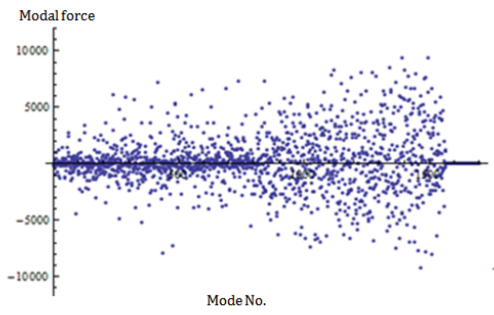
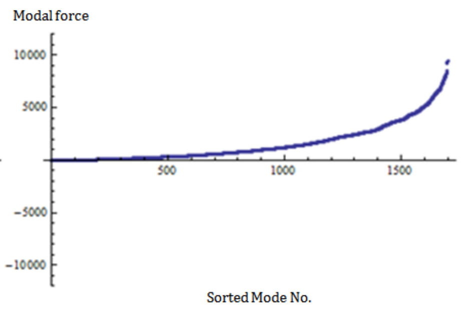


Fig. 7. (a) Eigenvalues, (b) modal force (explosive source), (c) sorted modal force (explosive source), (d) modal force (for double couple) and (e) sorted modal force (for double couple) in the non- homogeneous (2-layer) case



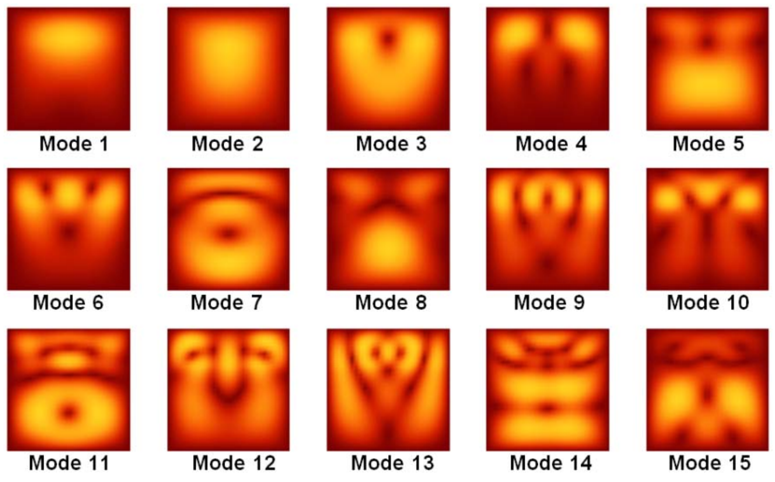
(d)



(e)

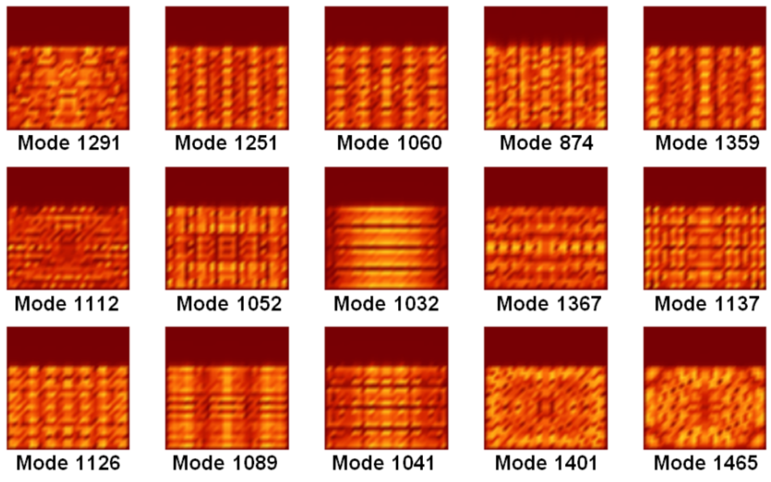
Fig. 7. (Continued)

Figure 8 illustrates the first and the most effective 15 modes for the two cases of explosive and double-couple mid-sources respectively. Figs. 9 and 10 illustrate the convergence of the displacement field at *t*=1.6 s as more modes are contributed for the two cases of explosive and double-couple sources. Again, an overall demonstration of the wave pattern is comprised with the contribution of almost the first 500 modes. As was the case in the previous example, mode superposition according to the mode number (Figs. 9(a) and 10(a)) gives more robust results than the results obtained from superposition based on the most significant modes (Figs. 9(b) and 10(b)).

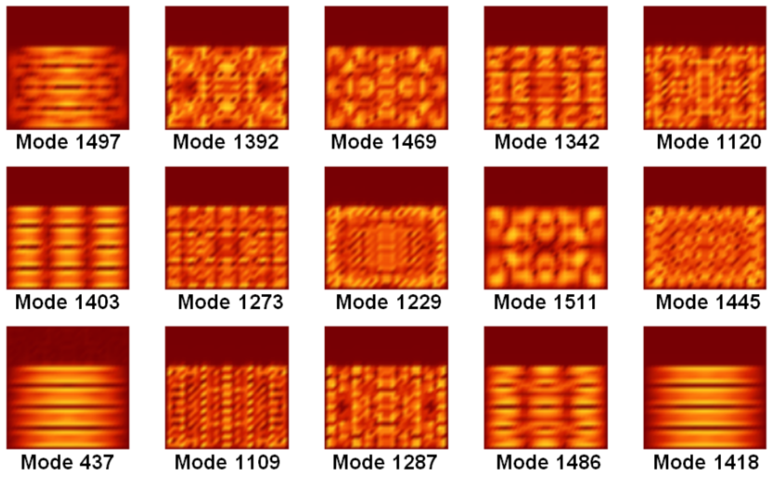


(a)

Fig. 8. Displacements of (a) the first 15 modes, (b) the most effective 15 modes for explosive source and (c) the most effective 15 modes for double-couple source (non-homogeneous 2-layer case)

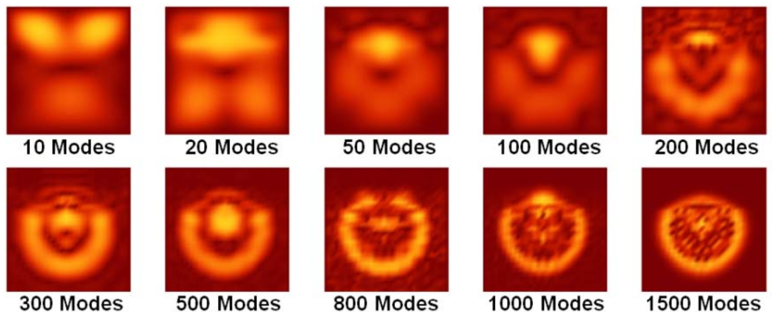


(b)



(c)

Fig. 8. (Continued)



(a)

Fig. 9. Snapshots of the displacement wave at t=0.1 s for increasing number of mode contribution (explosive-source non-homogeneous 2-layer case): (a) Modes sorted according to eigenvalues and (b) Modes sorted according to their modal contribution values

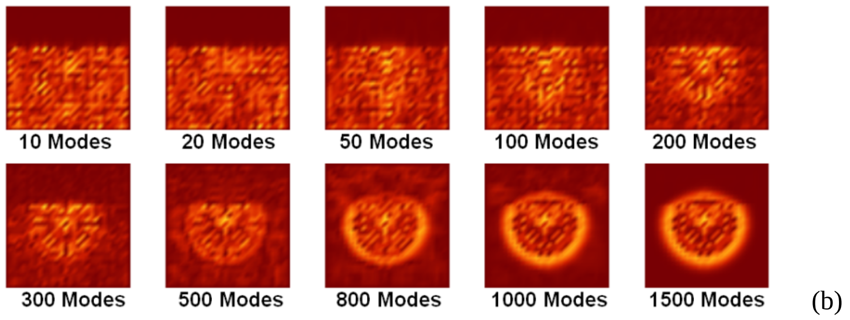


Fig. 9. (Continued)

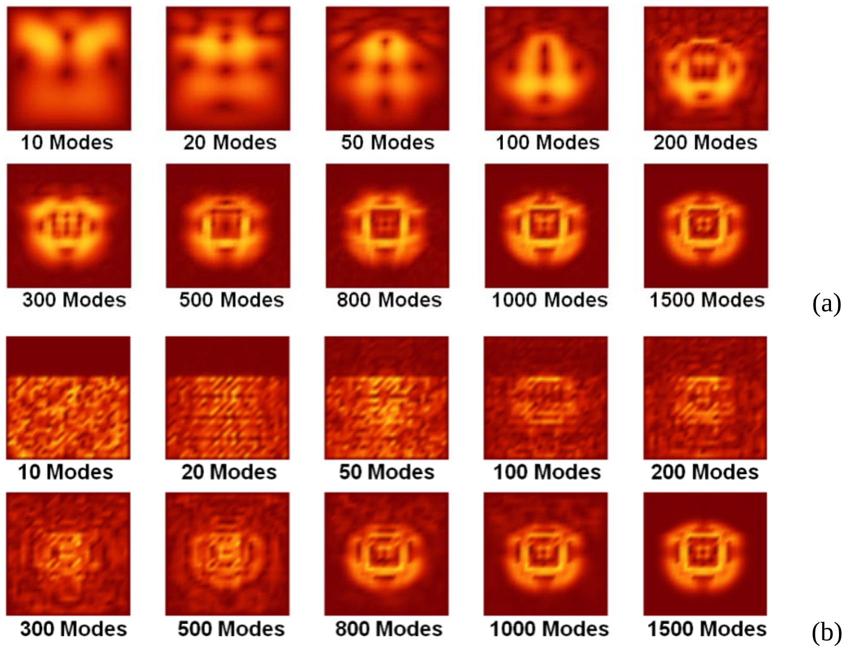


Fig. 10. Snapshots of the displacement wave at t=0.1 s for increasing number of mode contribution (double couple-source non-homogeneous 2-layer case): (a) Modes sorted according to eigenvalues, and (b) Modes sorted according to their modal contribution values

In Fig. 11(a), the elapsed times for serial MFDM (**M2**) runs (on an Intel Pentium Dual Core 3.0 GHz Processor) are illustrated versus the number of contributed modes for the introduced problem (homogeneous case with explosive source) discretized by a 30×30 mesh. The red point on the diagram depicts the elapsed time of 2.168 s obtained by serial FDTD (**M1**). According to the figure, the elapsed time for MFDM (**M2**), with all modes being contributed, is 1.700 s which is notably less than that of **M1**. Specially, when less modes are considered based on the required accuracy, such discrepancy increases (e.g. with the contribution of 500 modes by which a pattern of the wave-field is formed, the elapsed time is 0.281 sec). The discrepancy is surely more remarkable for more refined meshes. However, it is reminded that the pre-processing step is not reckoned for here in the calculation of the elapsed time of **M2** runs; such step is effectively expensive unless an appropriate parallel procedure is employed as explained in Section 3.1. As mentioned earlier, MFDM seems to be appropriate for the applications in which simulations for several source features is required while the velocity model does not change.

In Fig. 11(b), the parallel MPI- and CUDA- based implementations of FD and MFD methods (**M3**, **M4** and **M5**) are compared for the same problem. For MPI-based implementation, a 3-node cluster which contained 8 AMD 2.2GHz processors was employed. The network system used for MPI communications was Gigabit Ethernet with the bandwidth of 0.1 GB/s. The GPU system used for analysis contained two Nvidia Tesla C1060 GPUs with an Intel Quad-Core 2.66 GHz CPU as host.

A simple comparison between Figs. 11(a) and (b) shows that the computation times are considerably reduced by using the parallel procedures. Specially, the discrepancy between FDM and MFDM methods are more remarkable in parallel procedures as compared with serial procedures. It is also notable that CUDA-based implementation is slightly more costly than the MPI- based implementation for MFD runs. This is due to the fact that in MFDM, the communications are efficiently reduced such that the simulation time is attributed to FD temporal updates. Here, MPI implementations get use of a network of 3 CPUs which are computationally more powerful than the GPU system.

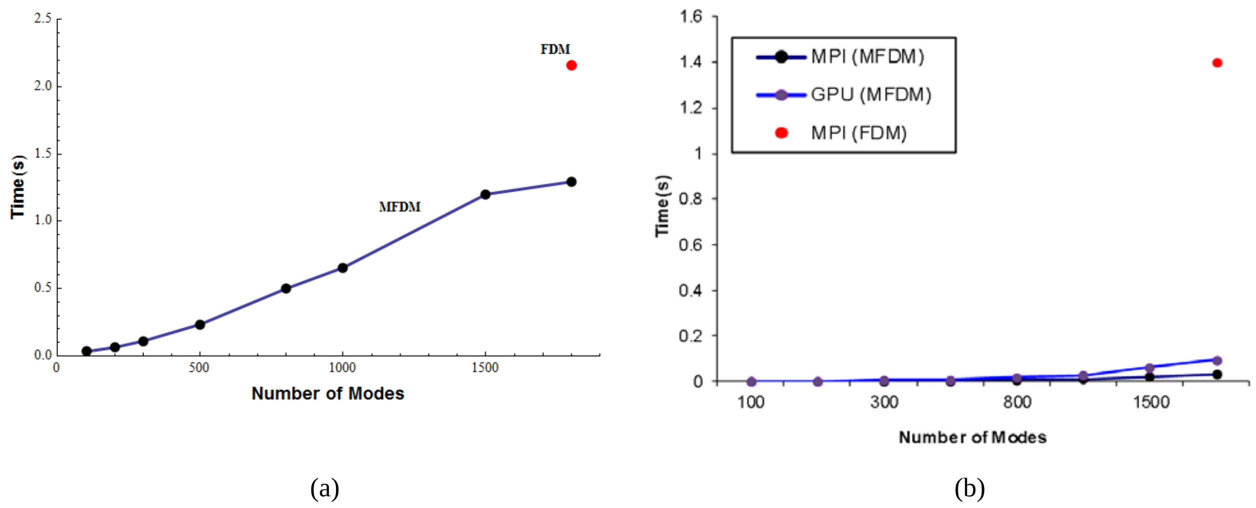


Fig. 11. (a) Serial MFDM (**M2**) performance versus the number of used modes and serial FDM (**M1**) performance (red point); (b) Parallel MPI- and CUDA- based MFDMs (**M4** and **M5**) performance vs. the number of used modes and MPI-based FDM performance (**M3**) (red point)

5. Conclusion

This paper presents the new FD-based procedure ‘Modal FDM’ for wave-field modeling in the elastic medium which explicitly solves wave equations in the time-domain. The proposed method is well-remarked for the following features:

1. Contribution to the overall response is not the same for different modes. In some cases, the significant portion of the response is almost contributed by a few modes and thus, calculations for all eigenvalues might not be required anymore. Such possibility is especially justifiable since the source mechanism and velocity model of the region are unknown or approximately known in most cases. Therefore, a solution procedure adjustable to access any arbitrary accuracy level is mostly demanded.

2. Since the explicit equations appeared in the proposed procedure are decoupled, processing for each mode can be performed independently. Concurrent processing is then simply possible and the method can easily be parallelized by any appropriate parallel standard without the need for domain-decomposition technique and slave-slave inter-core communications. From this point of view, MFDM is even preferred to FDFD procedures. The procedure is especially straightforward for being implemented in GPUs.

3. The parallel implementation of the procedure is dimension independent; i.e. a single parallel MFD algorithm might be used for any 1-, 2- and 3-dimensional wave-modeling problem.

4. MFDM seems to be computationally efficient for applications such as focal search procedures in which simulations for a large number of different sources is required while the velocity model does not change. In such applications, the time-consuming eigen-system calculation is performed once and the MFD runs will greatly reduce the subsequent computational costs in comparison with the conventional FD procedures.

Acknowledgements

The authors would like to thank Mr. Navazandeh for his guidance in MPI programming and Dr. Ali Sadeghi from Swiss Nanoscience Institute and Department of Physics at the University of Basel, Switzerland for his contribution in running the programs.

References

[1] Z. Alterman, F.C. Karal, Propagation of elastic waves in layered media by finite difference methods, Bulletin of the Seismological Society of America, 58 (1968) 367-398.
[2] D.M. Boore, Love waves in nonuniform wave guides: Finite difference calculations, Journal of Geophysical Research, 75 (1970) 1512-1527.
[3] D.M. Boore, Finite difference methods for seismic wave propagation in heterogeneous materials, Methods in computational physics, 11 (1972) 1-37.
[4] O.C. Zienkiewicz, R.L. Taylor, The finite element method: solid mechanics, Butterworth-heinemann, 2000.
[5] G. Cohen, P. Joly, N. Tordjman, Higher-order finite elements with mass-lumping for the 1D wave equation, Finite Elements in Analysis and Design, 16 (1994) 329-336.
[6] D.D. Kosloff, E. Baysal, Forward modeling by a Fourier method, Geophysics, 47 (1982) 1402-1412.
[7] D.D. Kosloff, M. Reshef, D. Loewenthal, Elastic wave calculations by the Fourier method, Bulletin of the Seismological Society of America, 74 (1984) 875-891.
[8] B. Fornberg, The pseudospectral method: Comparisons with finite differences for the elastic wave equation, Geophysics, 52 (1987) 483-501.
[9] D. Komatitsch, J. Tromp, Introduction to the spectral element method for three-dimensional seismic wave propagation, Geophysical Journal International, 139 (1999) 806-822.
[10] M.J. Grote, A. Schneebeli, D. Schötzau, Discontinuous Galerkin finite element method for the wave equation, SIAM Journal on Numerical Analysis, 44 (2006) 2408-2431.
[11] J. Virieux, P-SV wave propagation in heterogeneous media: Velocity-stress finite-difference method, Geophysics, 51 (1986) 889-901.

[12] A.R. Levander, Fourth-order finite-difference P-SV seismograms, Geophysics, 53 (1988) 1425-1436.

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

[14] P. Moczo, J. Kristek, L. Halada, The finite-difference method for seismologists, Comenius University, 2004.

[15] P. Moczo, J.O.A. Robertsson, L. Eisner, The finite-difference time-domain method for modeling of seismic wave propagation, Advances in Geophysics, 48 (2007) 421-516.

[16] M.A. Dablain, The application of high-order differencing to the scalar wave equation, Geophysics, 51 (1986) 54-66.

[17] B.G. Mikhailenko, Seismic modeling by the spectral-finite difference method, Physics of the Earth and Planetary Interiors, 119 (2000) 133-147.

[18] S.K. Lele, Compact finite difference schemes with spectral-like resolution, Journal of Computational Physics, 103 (1992) 16-42.

[19] S. Pirozzoli, Performance analysis and optimization of finite-difference schemes for wave propagation problems, Journal of Computational Physics, 222 (2007) 809-831.

[20] T. Bohlen, Parallel 3-D viscoelastic finite difference seismic modelling, Computers & Geosciences, 28 (2002) 887-899.

[21] OpenMP Application Programming Interface, version 3.0, in, http://openmp.org/wp/openmp-specifications/, 2009.

[22] W. Gropp, E. Lusk, A. Skjellum, Using MPI: portable parallel programming with the message-passing interface, MIT Press, 1999.

[23] R. Shams, P. Sadeghi, On optimization of finite-difference time-domain (FDTD) computation on heterogeneous and GPU clusters, Journal of Parallel and Distributed Computing, 71 (2011) 584-593.

[24] R. Mehra, N. Raghuvanshi, L. Savioja, M.C. Lin, D. Manocha, An efficient GPU-based time domain solver for the acoustic wave equation, Applied Acoustics, 73 (2012) 83-94.

[25] Y. Saad, Variations on Arnoldi's method for computing eigenelements of large unsymmetric matrices, Linear Algebra and its Applications, 34 (1980) 269-295.

[26] Y. Saad, Iterative methods for sparse linear systems, 2003.

[27] J.W. Choi, A. Singh, R.W. Vuduc, Model-driven autotuning of sparse matrix-vector multiply on GPUs, ACM Sigplan Notices, 45 (2010) 115-126.

[28] N. Bell, M. Garland, Implementing sparse matrix-vector multiplication on throughput-oriented processors, in: Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, ACM, Portland, OR, USA, 2009, pp. 18.

[29] CUDA toolkit documentation, in, http://docs.nvidia.com/cuda/index.html.

[30] H.-W. Chang, S.-E. Liu, R. Burridge, Exact eigensystems for some matrices arising from discretizations, Linear Algebra and its Applications, 430 (2009) 999-1006.

[31] N. Dugan, L. Genovese, S. Goedecker, A customized 3D GPU Poisson solver for free boundary conditions, Computer Physics Communications, 184 (2013) 1815-1820.