Computers and Geosciences 116 (2018) 23–31

Contents lists available at ScienceDirect

Computers and Geosciences

journal homepage: www.elsevier.com/locate/cageo

An algorithm for fast elastic wave simulation using a vectorized finite difference operator☆,☆☆

Ajay Malkoti [a,*], Nimisha Vedanti [a,b], Ram Krishna Tiwari [a,b]

[a] AcSIR-NGRI, Hyderabad, 500007, India
[b] CSIR-National Geophysical Research Institute, Hyderabad, 500007, India

ARTICLE INFO

Keywords:
Seismic
Elastic wave
Vectorization
Staggered grid

ABSTRACT

Modern geophysical imaging techniques exploit the full wavefield information which can be simulated numerically. These numerical simulations are computationally expensive due to several factors, such as a large number of time steps and nodes, big size of the derivative stencil and huge model size. Besides these constraints, it is also important to reformulate the numerical derivative operator for improved efficiency. In this paper, we have introduced a vectorized derivative operator over the staggered grid with shifted coordinate systems. The operator increases the efficiency of simulation by exploiting the fact that each variable can be represented in the form of a matrix. This operator allows updating all nodes of a variable defined on the staggered grid, in a manner similar to the collocated grid scheme and thereby reducing the computational run-time considerably. Here we demonstrate an application of this operator to simulate the seismic wave propagation in elastic media (Marmousi model), by discretizing the equations on a staggered grid. We have compared the performance of this operator on three programming languages, which reveals that it can increase the execution speed by a factor of at least 2–3 times for FORTRAN and MATLAB; and nearly 100 times for Python. We have further carried out various tests in MATLAB to analyze the effect of model size and the number of time steps on total simulation run-time. We find that there is an additional, though small, computational overhead for each step and it depends on total number of time steps used in the simulation. A MATLAB code package, 'FDwave', for the proposed simulation scheme is available upon request.

1. Introduction

The earth is a heterogeneous, anisotropic and attenuating medium which designates it as a complex model. The interaction of seismic waves with various structures is a complicated process due to inherent heterogeneity in material distribution. Wave phenomena associated with the media properties, shape or location of the layer, thus requires better understanding. Few such examples include-no reflections in a salt diaper, occurrence of low velocity layers, fluid-solid interfaces, free surface, pinching out bed, etc. One way to study the interaction of a wave with the earth is to simulate full seismic wavefield. It would enable us to analyze the wave behavior at all time instances and its response at the surface, as recorded by the receivers. The simulation also has several other significant applications, for example, finding the subsurface structure (Olsen et al., 1995; Yomogida and Etgen, 1993) and optimizing survey parameters and geometries (Moldoveanu and Egan, 2007; Regone, 2007). In fact, seismic simulation is considered as an integral part of seismic interpretation (Ibrahim, 2005) and quantification of results from seismic data (Sayers and Chopra, 2009).

Simulation of the seismic wavefield in earth is a large scale problem and thus requires a considerable memory along with high computational power. Memory demand for a seismic simulation can be mitigated using techniques described in earlier studies, viz. Graves (1996) and Etgen and O'Brien (2007). The number of computations (number of arithmetic operations) can be reduced by using highly accurate stencils with, smaller length for derivative calculation (Holberg, 1987; Lele, 1992; Tam and Webb, 1993; Liu and Sen, 2009). Smaller stencil uses lesser number of points and thereby reduces the computational cost.

☆ The code is available on request from author.
☆☆ AM conceived the idea of vectorization, developed its theoretical framework, implemented it as code and drafted the manuscript. NV and RKT have provided critical feedbacks, supervised the project and helped in manuscript writing. All authors read and approved the final manuscript.

* Corresponding author.
E-mail addresses: ajmalkoti@gmail.com (A. Malkoti), nimisha@ngri.res.in (N. Vedanti), rktiwari54@gmail.com (R.K. Tiwari).

https://doi.org/10.1016/j.cageo.2018.04.002
Received 30 May 2017; Received in revised form 12 March 2018; Accepted 2 April 2018
Available online 5 April 2018

Table 1
List of codes developed by various authors for addressing the wave propagation problems in various media using different techniques.

Language	Rheology	Forward/ Inverse	Author (Program Name)
Parallelization method: MPI			
C	Viscoelastic	FWD	Bohlen (2002) (fdmpi/sofi2D)
C	Elastic	FWI	Köhn et al. (2014) (Denise)
Fortran 2003	Viscoelastic	FWD	Maeda et al. (2017) (OPENSWPC)
C	Poroelastic	FWD	Sheen et al. (2006)
Parallelization method: GPU			
CUDA	Elastic	FWD	Micha and Komatitsch (2010)
CUDA	Elastic	FWD	Okamoto et al. (2010)
CUDA	Acoustic	FWI	Yang et al. (2015)
OpenCL	Viscoelastic	FWI	Fabien-Ouellet et al. (2017) (SeisCL)
Parallelization method: Multiple GPU			
OpenMP + CUDA	Anisotropic elastic	FWD	Weiss and Shragge (2013)
MPI + CUDA	Elastic	FWD	Zhou et al. (2013), Roten et al. (2016)
OpenGL	Elastic	FWD	Rubio et al. (2014)

The computational cost can further be reduced by exploiting the parallelism in computing which can be achieved using mainly two approaches. The first approach is to make many processors work simultaneously so that each processor shares a piece of work. To achieve this, one can use OpenMP or MPI depending upon the memory architecture (shared/distributed). It is also possible to take advantage of thousands of cores available in GPU using CUDA, OpenCL, OpenGL, etc. Few references, where seismic simulation was carried out using such techniques, are listed in Table 1. The second approach is to make each processor deliver more, for which data streaming to processor must be tuned finely. It can be achieved in following ways- (1) Instruction Level Parallelism (ILP), (2) Cache friendly code, (3) Fast algorithms or optimized numerical libraries, and (4) Single-instruction-multiple-data (SIMD) style instruction. ILP technique makes use of efficient sequencing before executing instructions and thus reduces the latency. The cache memory in CPU also plays a major role in determining the execution speed since high cache miss rate can cause high latency. It can be reduced by exploiting the spatial and temporal locality of data in the memory. For example, if matrices are accessed along the major axis, the memory access is faster due to low cache miss rate. Numerical libraries (e.g., BLAS, OPENBLAS, LINPACK, ATLAS, Intel Math Kernel Library, AMD Core Math Library, etc.) reduce the computation run-time by using efficient computational algorithms. For example, Coppersmith-Winograd algorithm is a fast algorithm for matrix multiplication. In SIMD approach (a.k.a. vectorization) a single machine instruction is automatically applied to all the arguments of the same type (a vector) by taking advantage of vector processors present inside each CPU.

Several researchers have studied the vectorization of the wave equation. For example, Vafidis et al. (1992) presented an algorithm based on the 'matrix times vector by diagonals' technique. There are other techniques which modify the pattern to access the data according to cache size for lesser cache miss rate. It involves loop-rearrangement, loop-blocking, and block-striding (Borges and Thierry, 2011; Zhou and Symes, 2014; Titarenko and Hildyard, 2017).

The seismic wave simulation involves BLAS level 1 type operations (e.g., scalar-matrix multiplication, αA; matrix-matrix addition, $A + B$). For such operations no special algorithms are required but SIMD plays a very important role in determining the efficiency of the computations. Programming languages such as FORTRAN, Python and MATLAB can perform better for whole-array operations than serial execution. Executing commands utilizing whole array operations are faster because such operations can be easily vectorized by the compilers. Thus, to achieve computational efficiency and speed, the numerical scheme should be written in such a way that it can take advantage of whole array operations or in other words, the code could be vectorized.

In this paper, we present a methodology to speed up the simulation for the elastic wave propagation in a given medium, by vectorizing the derivatives over the staggered grid. A stress-velocity formulation (Levander, 1988; Virieux, 1986) is solved over staggered grid using finite difference scheme. Problem with such a grid is that the variables cannot be updated using whole matrix operation of derivatives since field variables (i.e., velocities and stresses) are not defined at the same location in the grid. For example, to update the values of σ_{xz}, defined at center in a staggered grid, we need v_x and v_z which are defined at the horizontal edge and the vertical edge of staggered grid, respectively. To overcome this problem, we derived a new finite difference operator by decomposing the original staggered grid into a simpler shifted-grids, which can be easily extended to solve an n-dimensional problem. This method can significantly reduce the simulation run-time.

To show the reduction in computational cost, we have compared simulation run-time of the vectorized derivative operator with the conventional approach (using simple loops) at different platforms viz. FORTRAN, Python, and MATLAB, for different sizes of models and time steps. Further, we carried out a complete seismic simulation using MATLAB for Marmousi model and estimated the simulation run-time for various sizes of the model and time. Our findings indicate that the higher number of time steps can cause a little more computational overhead than the expected simulation run-time. The vectorization strategy helps in avoiding loops and renders the code in a well organized and succinct form which leads to an easy debugging and thus helps in quicker verification of new finite difference schemes.

2. Theory

2.1. Elastic wave formulation

The propagation of an elastic wave in the earth can be described using the momentum conservation equation and the Hook's law (Aki and Richards, 2002). The governing equation (momentum conservation equation) provides a relation between velocities (v_i) and the stresses (σ_{ij}), which can be written as

$$\rho \frac{\partial v_i}{\partial t} = \frac{\partial \sigma_{ij}}{\partial x_j} + \rho f_i \tag{1}$$

where ρ is density of the medium and f_i is the external body force in the i direction and $i, j \in \{x, y\}$. It can be used for estimating the particle velocities in the medium at the given instance $v_i(x, z, t)$. For Eq. (1), the constitutive law is Hook's law. It gives a relationship between the stress (σ_{ij}) and strain ($\varepsilon_{i,j}$), which can be expressed as

$$\frac{\partial \sigma_{ij}}{\partial t} = \lambda \dot{\varepsilon}_{k,k} \delta_{ij} + \mu \left(\dot{\varepsilon}_{i,j} + \dot{\varepsilon}_{j,i} \right) \tag{2}$$

where, λ and μ are the Lame's parameter and the dot over the quantity (i.e. $\dot{\varepsilon}$) implies the time derivative of the quantity. Hence $\dot{\varepsilon}$ represent the strain rate, which can be defined as

$$\dot{\varepsilon}_{i,j} = \frac{1}{2} \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} \right) \tag{3}$$

Using strain rate definition (Eq. (3)), the constitutive relation (Eq. (2)) can be transformed into a stress-velocity relationship. Hence, Eqs. (1) and (2) together form a coupled partial differential equation, known as the velocity-stress formulation. In our study, this equation will be used for simulation of seismic wave propagation in the earth assuming it as an elastic medium.

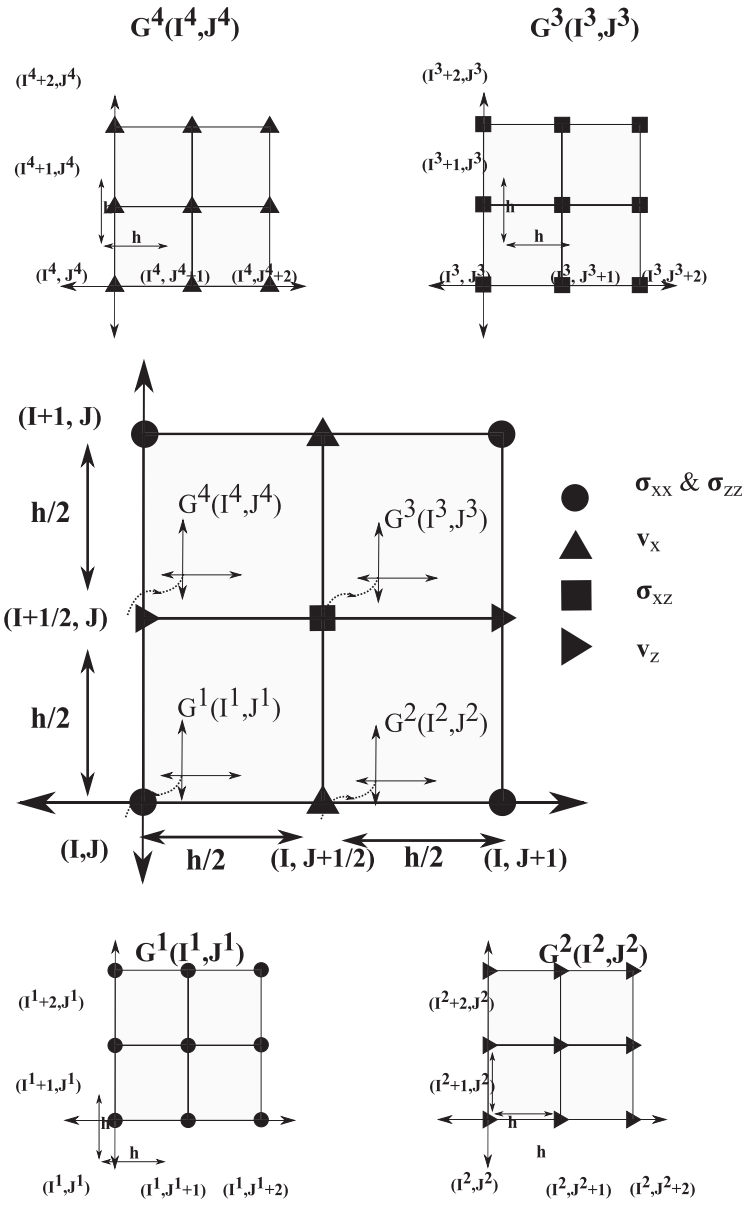

Fig. 1. The grid in the center shows the staggered grid configuration where different field parameters are defined at different locations. To achieve the vectorization, this main grid is decomposed into four simpler equispaced sub-grids, pertaining to respective stress/velocity fields, shown at top and bottom.

2.2. Discretization

Eqs. (1) and (2) can be discretized only after assigning the field variables to appropriate positions over the staggered grid (Fig. 1). Once we get the location for all field variables on the grid, then only we can choose the right type of derivative operators for them. Researchers have experimented with different types of grids (Graves, 1996; Levander, 1988; Saenger and Bohlen, 2004; Virieux, 1986). All of these grids roughly belong to Arakawa grid family and hence, grid selection is subjective to the choice of a researcher. For the grid shown in Fig. 1, Eqs. (1) and (2) can be rewritten as

$$D_t^0 v_i\Big|^n = \frac{1}{\rho}\left[D_j^+ \sigma_{ij}\Big|^n \delta_{ij} + D_j^- \sigma_{ij}\Big|^n (1-\delta_{ij})\right] \tag{4}$$

and

$$D_t^0 \sigma_{ij}\Big|^{\left(n+\frac{1}{2}\right)} = \lambda D_k^- v_k\Big|^{\left(n+\frac{1}{2}\right)} \delta_{ij} + 2\mu D_j^- v_i\Big|^{\left(n+\frac{1}{2}\right)} \delta_{ij} + \mu\left(D_j^+ v_i\Big|^{\left(n+\frac{1}{2}\right)} + D_i^+ v_j\Big|^{\left(n+\frac{1}{2}\right)}\right) \times (1-\delta_{ij}) \tag{5}$$

where D_t^0 is the central time derivative operator, D^+ and D^- are the shifted discrete derivative operators (as explained later in this section). The position of the variable in time and space is shown by their subscript and superscript respectively, viz. $\sigma_{ij}|_{x,z}^t$. These variables are defined over different positions and thus their position can be obtained from the grid. δ_{ij} refers to the Kronecker delta function which is defined as

$$\delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

A standard derivative operator defined for a 1D staggered grid,

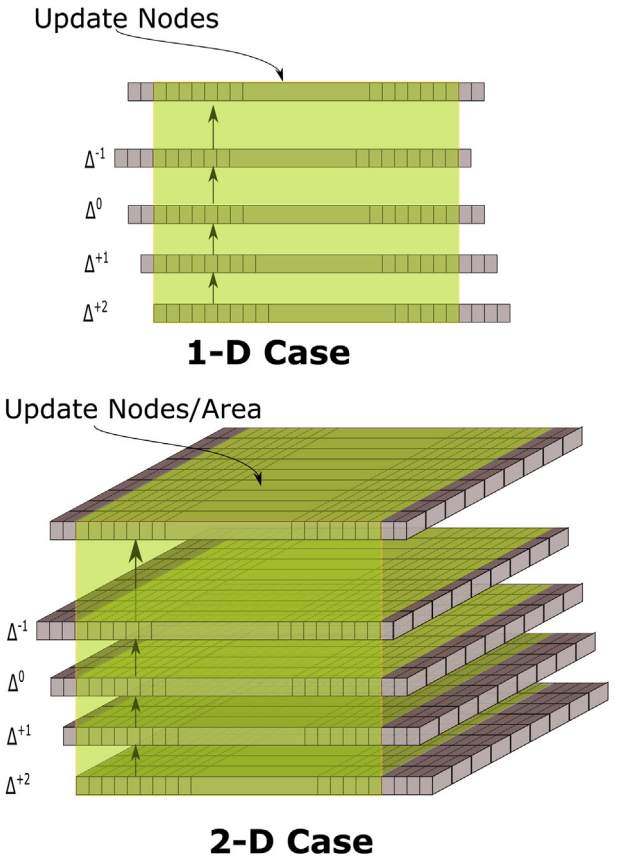

Fig. 2. The functioning of forward derivative operator i.e., D^+, using shift operation. Whole domain is assumed as a vector (1D) (shown at top), or a matrix (2D) (shown at bottom). The updated nodes after derivative operation are shown with the shaded part. For taking vectorized derivative whole domain i.e., vectors/matrices are shifted, according to shifting operator, in the direction of derivative. To complete the operation we sum up all shifted matrices and update the nodes accordingly.

centered at a location, i, can be written as:

$$D_x f(i) = \frac{1}{\Delta x}\left\{9/8\left[f\left(i+\frac{1}{2}\right) - f\left(i-\frac{1}{2}\right)\right] - 1/24\left[f\left(i+\frac{3}{2}\right) - f\left(i-\frac{3}{2}\right)\right]\right\} \tag{6}$$

For the above staggered derivative operator, a simpler form can be achieved by shifting the center (Eq. (6)) by $+\Delta x/2$ and $-\Delta x/2$ which yields D^+ and D^- operators respectively. Now the whole equation can be written using the shift operator (Δ^n), defined as $\Delta^n f(x) = f(x + n\Delta x)$.

$$D^+ f(i) = \frac{1}{24}\left(\Delta^{-1} - 27\Delta^0 + 27\Delta^{+1} - \Delta^{+2}\right) f(i) \tag{7}$$

$$D^- f(i) = \frac{1}{24}\left(\Delta^{-2} - 27\Delta^{-1} + 27\Delta^0 - \Delta^{+1}\right) f(i) \tag{8}$$

We can see that the previous derivative operator (Eq. (6)) is reformulated in the form of forward (D^+) and backward (D^-) derivative (Eqs. (7) and (8)), depending upon the location of center. This new form is appropriate for the vectorization that leads to increase in computation speed.

Fig. 2 shows the application of the forward derivative operator in a graphical form. For 1-D case, the whole domain can be considered as a vector. In this figure, we have shown five vectors, where the top one represents the 'whole domain' and remaining four represent 'one single derivative operation' (e.g. D^+) over complete domain at a time. The

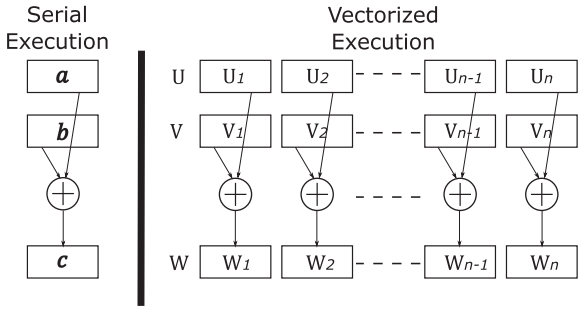

Fig. 3. Difference in serial execution of a command and a vectorized execution of a command at a given instance. Here a, b and c are single values, whereas, U, V, W are vectors such that $U = \{U_1, U_2, U_3, \ldots, U_n\}$, $V = \{V_1, V_2, V_3, \ldots, V_n\}$ and $W = \{W_1, W_2, W_3, \ldots, W_n\}$. For a simple operation like addition the serial code performs one addition operations at one time (i.e. $c = a + b$). In contrast, for a vectorized code, the processor executes n number of addition operations at one time (i.e. $W = U + V$).

vectorized derivative operator is shown as a combination of appropriate shifts. Here, positive and negative shifts are taken in $+x$ and $-x$ directions respectively. Using this operator, whole domain can be updated at once as shown in Fig. 2. This operator can be easily extended to 2-D (or higher-dimensions). The complete domain for 2D can be represented in the form of a matrix (Fig. 2). Similar to the 1D case, we have shown five matrices, where top one represents the complete domain and remaining four show the 2D derivative operator with appropriate shifts. For this particular case, shifts are restricted only in one direction, i.e. the direction in which derivative is taken. The derivative operator updates a certain portion of the domain/nodes, shown in the shaded part (Fig. 2). We are utilizing a complete vector or a matrix to perform this operation which is suitable to out carry out the vectorization (explained in next subsection). Further, as each derivative operation requires its terms to be multiplied with respective coefficients (e.g., Eq. (6)), the vectorized operator also requires its vectors/matrices to be multiplied with the respective coefficients (Eqs. (7) and (8)).

Now the problem that remains is, relating right-hand side derivatives with shifted coordinates to the left-hand side field variables. This problem arises because of the fact that in a staggered grid with grid spacing h, different field variables are located at different nodes which are shifted by $\Delta h/2$ in x or z directions. To resolve this problem, we can assume different coordinate-systems attached to each field variable, f (i.e. velocity, stresses) as G^f. For example, the grid system G^{v_x} is defined with actual coordinate as $(i+1/2, j)$ but its shifted coordinates is (I^{v_x}, J^{v_x}). Hence, this leads to the generation of four independent grid systems which are shifted with respect to each other. These grid systems are defined by $G^f(I^f, J^f)$ where f can be $\{v_i, \sigma_{ij}\}$ and $i \in \{x, y, z\}$. Fig. 1 demonstrates this scheme which has four independent grids namely G^1, G^2, G^3 and G^4 corresponding to field variables σ_{xx} (or σ_{zz}), v_x, v_z and σ_{xz} respectively. Using these independent grids, we can write a vectorized equation as follows:

$$D_t^+ v_i(I^{v_i}, J^{v_i}) = \frac{\gamma}{\rho}\left(D_j^+ \sigma_{ij}(I^{\sigma_{ij}}, J^{\sigma_{ij}})\delta_{ij} + D_j^- \sigma_{ij}(I^{\sigma_{ij}}, J^{\sigma_{ij}})(1-\delta_{ij})\right) \tag{9}$$

$$D_t^+ \sigma_{ij}(I^{\sigma_{ij}}, J^{\sigma_{ij}}) = \gamma\left[\lambda D_k^- v_k(I^{v_i}, J^{v_i})\delta_{ij} + 2\mu D_j^- v_i(I^{v_i}, J^{v_i})\delta_{ij}\right] + \gamma\mu\left[D_j^+ v_i(I^{v_i}, J^{v_i}) + D_i^+ v_j(I^{v_i}, J^{v_i})\right](1-\delta_{ij}) \tag{10}$$

In above equations, the indices for each field variable correspond to its grid system only. Above two equations form the framework to carry

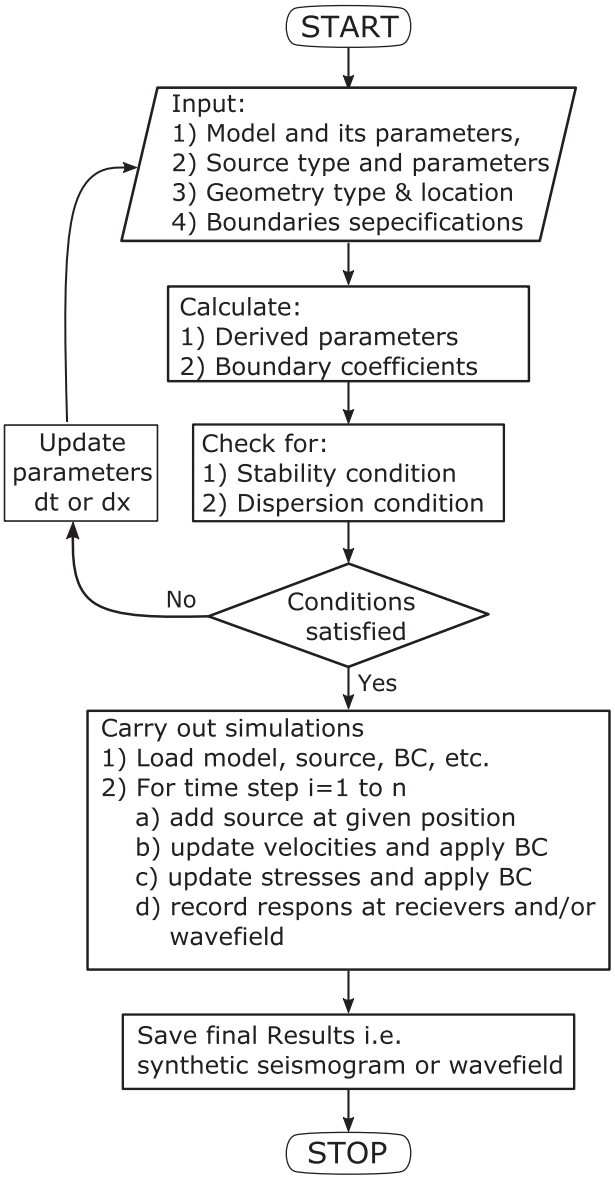

Fig. 4. The flowchart of the steps followed for seismic simulation and synthetic seismogram generation.

out the vectorization. At this stage, it should be noted that the operators are no longer the point operators but now these are vectorized/matricised operators. Thus, using these shifted coordinates, we can carry out the derivative operation directly in an efficient manner.

2.3. Vectorization

In parallel computing, vectorization refers to a process in which a single operation is applied to a set or vector of data simultaneously. Consequently, vectorization of code refers to the process of modifying the code in such a way that it can take advantage of vectored arguments. Fig. 3 shows the serial execution of a command where the single operation is carried out on given operands (a and b) at a given instance, whereas in parallel execution, the same operation is carried out on n-number of operands (i.e. U and V data vectors). Vectorization is available in all modern processors as it consists of one scaler processor unit and one vector processor unit. Each unit has the respective type of registers and operational units. The vector operations utilize the vector registers which can hold a fixed number of arguments depending upon the available number of registers and the argument type. The processor we used for this work, supports the Advanced Vector Extension (AVX-256) Instruction set. Their vector register can hold either eight-single-precision values or four-double-precision floating point values. The vector processor also allows the use of three operands command, i.e., $c = a + b$, which is non destructive as it preserves input operands as well as result. Modern compilers also facilitate auto-vectorization of code, however for this purpose, the commands must be issued in vector friendly manner. The best possible data for vectorization is a matrix, as it provides a freedom to the compiler to rearrange the computation pattern for maximum execution speed. Since, a matrix contain data of the same type and resides in memory in a contiguous manner, we implicitly fulfill the memory alignment requirement which demands memory access pattern to be in multiples of given size of data type. Additionally, compilers also provide some basic optimization for both serial as well as vectorized approach, which further helps in reducing cache miss rate.

2.4. Other considerations

In the elastic wave simulation, we have used a Ricker source wavelet which is defined as

$$s(t) = \left(1 - 2\pi^2 f_0^2 (t - t_0)^2\right) e^{-\pi^2 f_0^2 (t-t_0)^2}$$

Here, f_0 is dominant frequency, t is the time and t_0 is the time shift. To suppress the edge reflections, we have used absorbing/non-reflecting layer at boundaries (Cerjan et al., 1985). It can be implemented by padding the edges of the domain with some extra nodes. These nodes act as an energy absorbing region as it dampens the wave energy traveling within it by reducing the amplitude of the wave. The top surface is treated as a free surface using the imaging method given by Levander (1988). To ensure the stability, size of the time step, Δt was taken according to CFL conditions which makes sure that the speed of the wave is less than the speed of the mesh update. Grid spacing, Δx was chosen appropriately to minimize the grid dispersion. The criteria followed was to ensure that the shortest wavelength is sampled by at least five or more grids to keep the numerical dispersion error below 5% level (Levander, 1988).

3. Flowchart and organization of code

The basic outline of the workflow used for seismic simulation in our code "FDwave" is given in Fig. 4. For ease of exposition, the code can be separated into three parts:

1. *Pre-processing:* Prepare necessary input parameters such as model's elastic parameters, source wavelet, the geometry of receivers and source, boundary condition.
2. *Simulation:* Carry out the numerical seismic wavefield simulation using finite difference method for given number of time steps (N_t).
   a. Load all the input data and parameters
   b. For time step $t = 1 : N_t$
      - Update source amplitude at time t at the given location
      - Update the velocities & apply the boundary condition
      - Update stresses & apply boundary conditions
      - Record the wavefield and/or response at receivers
      - Save final resulsts, i.e., wavefield or synthetic seismogram
3. *Post-processing:* Plotting the synthetic seismogram and wavefield, create animation and other tasks.

Our method differs from the conventional approaches in the way of computing the derivatives for updating velocities or stresses at the nodes. Following is an example of code to demonstrate the difference between the two approaches.

Table 2
List of the parameters used for simulating wave propagation over elastic Marmousi model.

Parameter	Symbol	Value
Time step	dt	0.00025 s
Total time	T	2 s
Grid spacing	dx or dz	4 m
Grid points along the X direction	$N_x T_{sim}$	1250
Grid points along the Z direction	N_z	875
Receiver Spacing	ΔX_{rec}	4 m
Source Type	$S(t)$	Ricker
Source Central freq	f_0	15 Hz
ABC type		Cerjan et al. (1985), Philip Bording (2004)
ABC no of layers	N_{ABC}	50

Conventional approach

for $t = 1 : N_t$;
  for j=1: N_z
    for i=1: N_x

$$v_x = \Sigma_{k=-L}^{L} w(k)\sigma_{xx}(i-k,j) + \sum_{k=-L}^{L} w(k)\sigma_{xz}(i,j-k)$$

      ...
    end
  end
end

Vectorize/matricised operation

for $t = 1 : N_t$;

$$v_x = \Sigma_{k=-L}^{L} w(k)\sigma_{xx}(\Delta_x^k) + \sum_{k=-L}^{L} w(k)\sigma_{xz}(\Delta_z^k)$$

  ...
end

In the conventional approach, additional loops are required which are equal to the dimensions of the model for updating the field variables (e.g. v_i and σ_{ij}) at each and every point in computation grid. Above example shows that the conventional approach requires two extra loops over dimension x and z, represented by i and j respectively. However, in the vectorized approach, we do not require such loops. We can update matrices by straight forward computation of derivative by taking weighted (w_k) sum of vectors or matrices (e.g. σ_{xx}) with a proper amount of shift (Δ^k) in given direction.

4. Application of the method to Marmousi model

To demonstrate the application of our vectorized elastic scheme on a staggered grid, we have chosen the Marmousi model and generated the synthetic shot-gather. The total dimension of this model is 20405 m × 4205 m which is sampled at 1.5 m grid spacing evenly. It has quite a large number of grid nodes; therefore, we have used only a part of this model after interpolation. Various parameters used for the simulation are shown in Table 2. We have carried out a full wave-field simulation using the proposed method. The interaction of the wave with the given model and the response recorded at the surface (synthetic seismogram) are shown in Figs. 5 and 6, respectively.

5. Tests for efficiency

The efficiency of code can be measured by recording the total run-time of a simulation (T_{sim}). To compare on various language platforms, viz. FORTRAN, Python and MATLAB, we carried out a simple test. We generated a random matrix of a given size (N_x,N_z) and took its derivative $10N$ times using serial execution (conventional approach) as well as the whole matrix approach (as suggested in this paper). The number $10N$ represent the total number of spatial derivatives computed in a simulation and it can be obtained by considering total number of steps (N) and total number of spatial finite difference derivatives involved at each step (=10). The computational run-time recorded for each simulation is listed in Table 3. It can be seen that in the test using serial execution, FORTRAN takes the least time followed by MATLAB and then Python. The reason is obvious, as the FORTRAN is a compiled language, whereas the MATLAB and Python are interpreted languages. In all tests carried out using whole

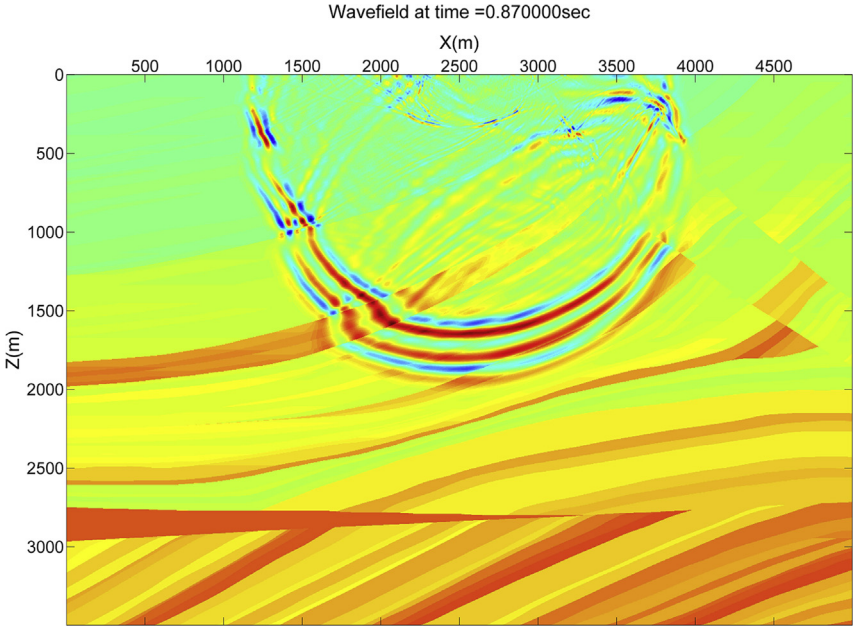

Fig. 5. A snap of the simulation for the wave propagating in the cropped-Marmousi model.

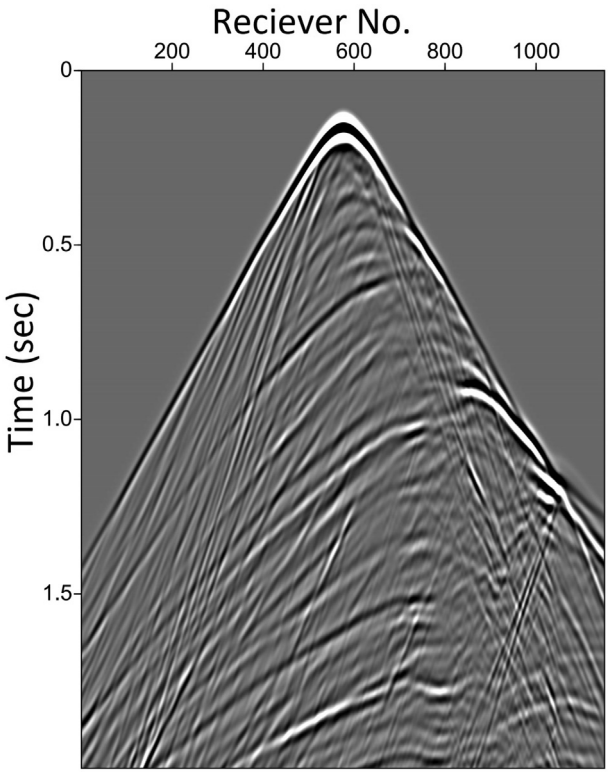

Fig. 6. The seismic response at the surface (synthetic seismogram) recorded during the simulation for Marmousi model.

Table 3
Comparison of computation run-time estimated for serial execution (SE) and vectorized/matricised execution (ME) on different programming languages platforms. The blank space shows that respective test was not carried out due to very high computation run-time.

Time steps	Size	Run-time for the simulation (seconds)					
		FORTRAN		MATLAB		Python	
N_t	N_xN_z	SE	ME	SE	ME	SE	ME
1000	500^2	9.0	2.4	149	53	4757	44
	1000^2	36.5	10.4	621	211	20399	202
	1500^2	82.0	25.5	1396	446	–	424
	2000^2	145.7	45.2	2358	760	–	754
2000	500^2	10.2	4.91	295	106	9511	92
	1000^2	40.7	21.2	1164	424	40554	396
	1500^2	91.1	50.4	2613	899	–	848
	2000^2	161.1	91.0	4659	1518	–	1489
4000	500^2	20.4	9.8	596	212	19044	185
	1000^2	81.1	41.7	2326	835	–	804
	1500^2	182.1	100.8	5294	1802	–	1693
	2000^2	322.1	180.4	10151	3038	–	2977
8000	500^2	40.9	19.5	1200	404	40523	371
	1000^2	162.3	84.2	4649	1667	–	1597
	1500^2	364.1	201.8	10587	3597	–	3404
	2000^2	644.3	362.8	18632	6071	–	5957

array operation, FORTRAN again takes the least time, but in this case Python performs better than MATLAB. The gain in speed for FORTRAN and MATLAB is ~2-3 times and for Python is almost 100 times. It must be mentioned here that our aim is not to compare the simulation run-time among these languages, but to show the improvement in execution speed for each of them. These tests were carried out on HP Z400 workstation with Intel Xeon 3.20 GHz (quad core) processors and 8GB RAM. We have used the Gfortran (ver. 4.4.7), MATLAB (ver. 2016) and Python(ver. 2.7) to test the performance.

Table 4
Computational run-time (in seconds) for various size models and number of time steps (N_t) which were estimated for the seismic simulation.

Model size	Number of time steps (N_t)			
(N_x.N_z) grid points	100	2000	4000	8000
501 × 501	17	156	310	620
1001 × 1001	68	685	1378	2748
1501 × 1501	159	1613	3219	6398
2001 × 2001	281	2840	5691	11373

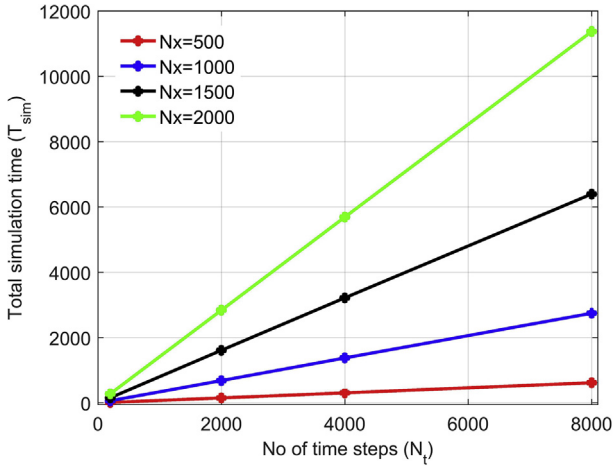

Fig. 7. A plot of the total simulation run-time (T_{sim}) and number of time steps (N_t) is shown for different sizes of model. A linear relationship can be seen between the T_{sim} and N_t.

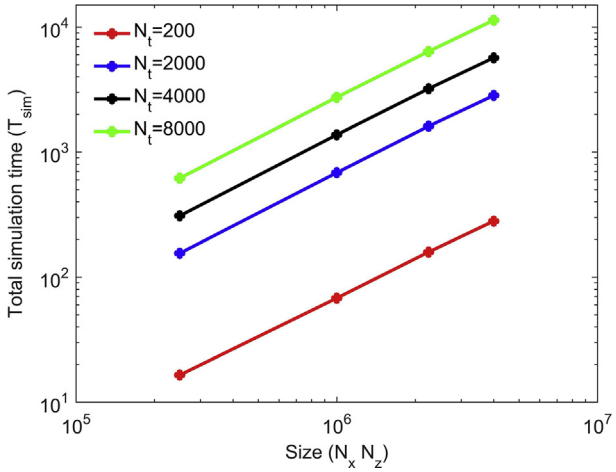

Fig. 8. A plot of the total simulation run-time (T_{sim}) and the size of model (N_xN_z) is shown for different time steps. The relation between T_{sim} and N_t is non-linear hence plotted on log-log scale.

Table 5
Normalized computation run-time, obtained after normalizing the actual simulation run-time T_{sim} estimated in Table 4.

Model size	Number of time steps (N_t)			
(N_x.N_z) grid points	100	2000	4000	8000
501 × 501	1.0000	1.0301	1.0703	1.0633
1001 × 1001	0.9411	1.0370	1.0848	1.0742
1501 × 1501	0.9375	1.0427	1.0821	1.0762
2001 × 2001	0.9376	1.0394	1.0754	1.0754

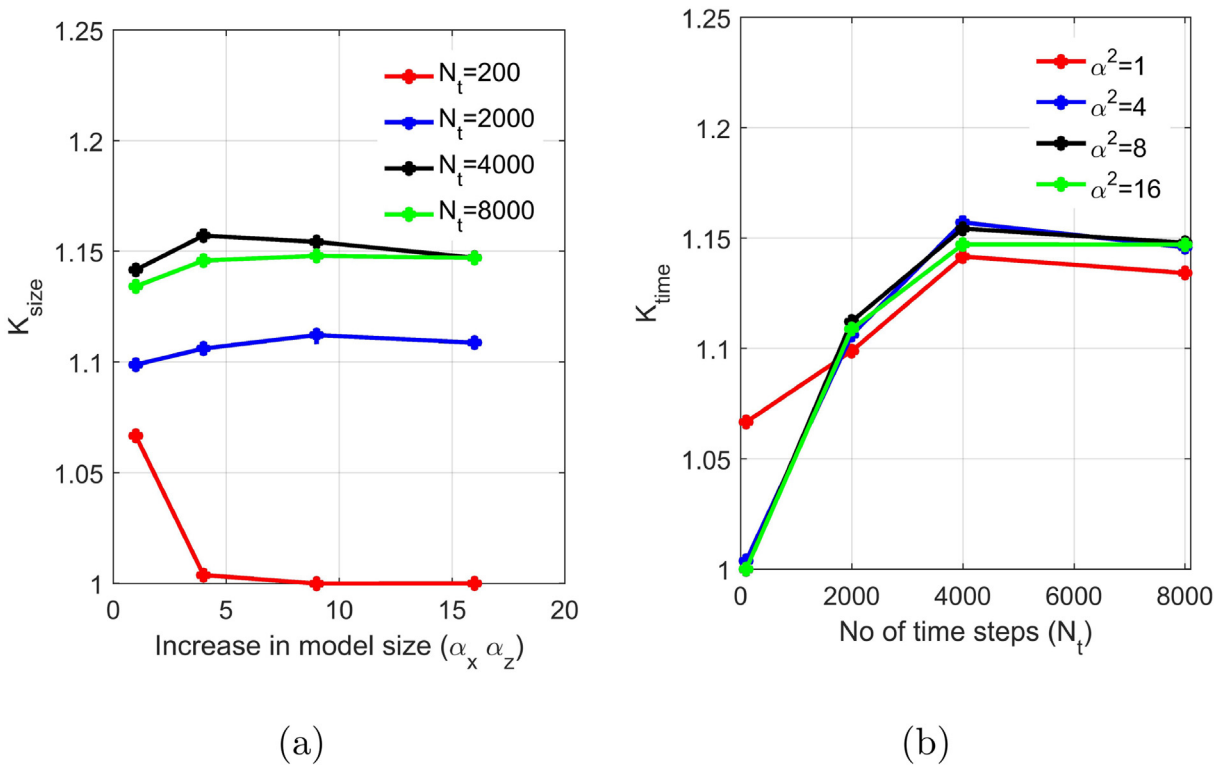

Fig. 9. (a) Variation of computational overhead, K_{size} as a function of model size ($\alpha_x\alpha_z$); (b) Variation of computational overhead K_{time}, as a function of number of the time steps (N_t).

Complete elastic wave simulation is carried out in MATLAB. We have tabulated T_{sim} for different model size ($S_m = N_x.N_z$ nodes) and number of time steps (N_t) without changing other parameters Table 4. We can write an expression for the simulation run-time as $T_{sim} = f(S_m, N_t) \approx f(N_x, N_z, N_t)$. T_{sim} depends linearly on N_t and S_m which can be corroborated from the respective plots shown in Figs. 7 and 8 respectively. Hence we can write the combined relation as

$$T_{sim} = (K_{size}N_xN_z)(K_{time}N_T) \tag{11}$$

$$= K_{size}K_{time}\alpha_x\alpha_z\alpha_t N_x^0 N_z^0 N_T^0 \tag{12}$$

Here K_{size} and K_{time} are respective proportionality factors (counting for computational overhead). Size of each model is shown as a multiple of the smallest model. Thus for each, x, z and t, we can write $N_i = \alpha_i N_i^0$, where, N_i^0 is the number of nodes in smallest model and α_i is the factor required for obtaining equality. To see the effect of K_{size} and K_{time}, we have reduced Table 4 in form of $T_{sim} = K_{size}K_{time}$ by normalization i.e., by removing effect of α_i and N_i^0 (Table 5). A closer inspection of Table 5 reveals that there is more variability along the row than along the column. It can be better understood by plotting the computational overhead with respect to size factor ($\alpha_x\alpha_z$) and simulation run-time factor (α_t), as shown in Fig. 9. For the plot of K_{size} versus $\alpha_x\alpha_z$ (Fig. 9a) the curves are almost constant, which means that the computational overhead due to size is negligible. In the second plot between K_{time} and N_t (Fig. 9b), we can see that the computational overhead increases with increase in the number of steps. Hence, to minimize the computational overhead for run-time, one should first ascertain the grid size and then determine the maximum time step size required for simulation.

6. Conclusion

A new scheme is proposed to improve the efficiency of seismic simulation by introducing vectorization of derivatives over a staggered grid. The proposed derivative scheme was used for the simulation of seismic waves in an elastic media. The scheme is able to circumvent the 'time consuming serial-execution' during the derivative estimation and thus results in faster computation. The vectorized derivative scheme would play an important role in simulation in languages like FORTRAN, Python and MATLAB which adhere to use of vector and matrices for faster execution. It has been shown that by implementing the proposed method, the execution speed increases nearly 2–3 times for MATLAB & FORTRAN, and approximately 100 times for Python language. The cause of this speedup in FORTRAN and MATLAB can be directly attributed to the vectorization, since our processor, with AVX2 instruction sets, could utilize all four track pipes. It is also demonstrated that, in MATLAB, the model size should be fixed first and then optimization for the time step should be done to achieve the maximum efficiency for the simulation. Further, this approach also offers advantages like organized and succinct code, which renders easier reading and debugging. This methodology would help researchers to write faster codes for similar large-scale problems.

Code availability

The wave simulation package "FDwave" and other scripts are available on request from the author.

Acknowledgments

The code made use of Marmousi model[1] and SEGYmat program[2] provided online by Gary Martin, Univ of Houston and Thomas Mejer Hansen, Niels Bohr Institute respectively. For plotting the data, we have used the Seismic Unix.[3] AM is grateful to CSIR-UGC for awarding him the fellowship to carry out his Ph.D. work. RKT is grateful to DAE for

[1] http://www.agl.uh.edu/downloads/downloads.htm.
[2] https://github.com/cultpenguin/segymat.
[3] http://www.cwp.mines.edu/cwpcodes/.

awarding Rajaramanna fellowship. Authors are also thankful to AcSIR Coordinator & Director NGRI for support and permission to publish the results. This work was carried out under CSIR-project, SHORE PSC-0205.

Appendix A. Supplementary data

Supplementary data related to this article can be found at https://doi.org/10.1016/j.cageo.2018.04.002.

References

Aki, K., Richards, P.G., 2002. Quantitative Seismology. University Science Books.
Bohlen, T., 2002. Parallel 3-D viscoelastic finite difference seismic modelling. Comput. Geosci. 28, 887–899. https://doi.org/10.1016/S0098-3004(02)00006-7.
Borges, L., Thierry, P., 2011. 3D finite differences on multi-core processors. available online at: http://software.intel.com/en-us/articles/3d-finite-differences-on-multi-core-processors.
Cerjan, C., Kosloff, D., Kosloff, R., Reshef, M., 1985. A nonreflecting boundary condition for discrete acoustic and elastic wave equations. Geophysics 50, 705–708. https://doi.org/10.1190/1.1441945.
Etgen, J., O'Brien, M., 2007. Computational methods for large-scale 3D acoustic finite-difference modeling: a tutorial. Geophysics 72. https://doi.org/10.1190/1.2753753. SM223–SM230.
Fabien-Ouellet, G., Gloaguen, E., Giroux, B., 2017. Time-domain seismic modeling in viscoelastic media for full waveform inversion on heterogeneous computing platforms with openCL. Comput. Geosci. 100, 142–155. https://doi.org/10.1016/j.cageo.2016.12.004.
Graves, R.W., 1996. Simulating seismic wave propagation in 3D elastic media using staggered-grid finite differences. Bull. Seismol. Soc. Am. 86, 1091–1106.
Holberg, O., 1987. Computational aspects of the choice of operator and sampling interval for numerical differentiation in large-scale simulation of wave phenomena. Geophys. Prospect. 35, 629–655. https://doi.org/10.1111/j.1365-2478.1987.tb00841.x.
Ibrahim, A.A., 2005. 3D ray-trace modeling to assess the effects of overburden and acquisition geometry on illumination of pre-evaporite reservoirs in Karachaganak Field, Kazakhstan. Lead. Edge 24, 940–944. https://doi.org/10.1190/1.2056407.
Köhn, D., Kurzmann, A., De Nil, D., Groos, L., 2014. DENISE User Manual.
Lele, S.K., 1992. Compact finite difference schemes with spectral-like resolution. J. Comput. Phys. 103, 16–42. https://doi.org/10.1016/0021-9991(92)90324-R.
Levander, A.R., 1988. Fourth-order finite-difference P-SV seismograms. Geophysics 53, 1425–1436. https://doi.org/10.1190/1.1442422.
Liu, Y., Sen, M.K., 2009. An implicit staggered-grid finite-difference method for seismic modelling. Geophys. J. Int. 179, 459–474. https://doi.org/10.1111/j.1365-246X.2009.04305.x.
Maeda, T., Takemura, S., Furumura, T., 2017. OpenSWPC: an open-source integrated parallel simulation code for modeling seismic wave propagation in 3D heterogeneous viscoelastic media. Earth Planets Space 69 (102). https://doi.org/10.1186/s40623-017-0687-2.
Micha, D., Komatitsch, D., 2010. Accelerating a three-dimensional finite-difference wave propagation code using GPU graphics cards. Geophys. J. Int. 182, 389–402. https://doi.org/10.1111/j.1365-246X.2010.04616.x.
Moldoveanu, N., Egan, M., 2007. Some aspects of survey design for wide-azimuth towed-streamer acquisition. In: SEG Technical Program Expanded Abstracts. Society of Exploration Geophysicists, pp. 56–60. https://doi.org/10.1190/1.2792381.
Okamoto, T., Takenaka, H., Nakamura, T., Aoki, T., 2010. Accelerating large-scale simulation of seismic wave propagation by multi-GPUs and three-dimensional domain decomposition. Earth Planets Space 62, 939–942. https://doi.org/10.5047/eps.2010.11.009.
Olsen, K.B., Pechmann, J.C., Schuster, G.T., 1995. Simulation of 3D elastic wave propagation in the salt lake basin. Bull. Seismol. Soc. Am. 85, 1688–1710. http://www.bssaonline.org/content/85/6/1688.
Philip Bording, R., 2004. Finite difference modeling-nearly optimal sponge boundary conditions. In: SEG Technical Program Expanded Abstracts 2004. Society of Exploration Geophysicists, pp. 1921–1924. https://doi.org/10.1190/1.1845189.
Regone, C., 2007. Using 3D finite-difference modeling to design wide-azimuth surveys for improved subsalt imaging. Geophysics 72. https://doi.org/10.1190/1.2668602. SM231–SM239.
Roten, D., Cui, Y., Olsen, K.B., Day, S.M., Withers, K., Savran, W.H., Wang, P., Mu, D., 2016. High-frequency nonlinear earthquake simulations on petascale heterogeneous supercomputers. In: High Performance Computing, Networking, Storage and Analysis, SC16: International Conference for. IEEE, pp. 957–968. https://doi.org/10.1109/SC.2016.81.
Rubio, F., Hanzich, M., Farrés, A., De La Puente, J., Cela, J.M., 2014. Finite-difference staggered grids in GPUs for anisotropic elastic wave propagation simulation. Comput. Geosci. 70, 181–189. https://doi.org/10.1016/j.cageo.2014.06.003.
Saenger, E., Bohlen, T., 2004. Finite difference modeling of viscoelastic and anisotropic wave propagation using the rotated staggered grid. Geophysics 69, 583–591. https://doi.org/10.1190/1.1707078.
Sayers, C., Chopra, S., 2009. Introduction to this special sectionRock physics. Lead. Edge 28, 15–16. https://doi.org/10.1190/1.3064140.
Sheen, D.H., Tuncay, K., Baag, C.E., Ortoleva, P.J., 2006. Parallel implementation of a velocity-stress staggered-grid finite-difference method for 2-D poroelastic wave propagation. Comput. Geosci. 32, 1182–1191. https://doi.org/10.1016/j.cageo.2005.11.001.
Tam, C.K.W., Webb, J.C., 1993. Dispersion-relation-preserving finite difference schemes for computational acoustics. J. Comput. Phys. 107, 262–281. https://doi.org/10.1006/jcph.1993.1142.
Titarenko, S., Hildyard, M., 2017. Hybrid multicore/vectorisation technique applied to the elastic wave equation on a staggered grid. Comput. Phys. Commun. 216, 53–62. https://doi.org/10.1016/j.cpc.2017.02.022.
Vafidis, A., Abramovici, F., Kanasewich, E.R., 1992. Elastic wave propagation using fully vectorized high order finite-difference algorithms. Geophysics 57, 218–232. https://doi.org/10.1190/1.1443235.
Virieux, J., 1986. P-SV wave propagation in heterogeneous media: velocity-stress finite-difference method. Geophysics 51, 889–901. https://doi.org/10.1190/1.1442147.
Weiss, R.M., Shragge, J., 2013. Solving 3D anisotropic elastic wave equations on parallel GPU devices. Geophysics 78, F7–F15. https://doi.org/10.1190/geo2012-0063.1.
Yang, P., Gao, J., Wang, B., 2015. A graphics processing unit implementation of time-domain full-waveform inversion. Geophysics 80. https://doi.org/10.1190/geo2014-0283.1.
Yomogida, K., Etgen, J.T., 1993. 3-D wave propagation in the Los Angeles basin for the Whittier-Narrows earthquake. Bull. Seismol. Soc. Am. 83, 1325–1344. http://www.bssaonline.org/content/83/5/1325.
Zhou, J., Cui, Y., Poyraz, E., Choi, D.J., Guest, C.C., 2013. Multi-GPU implementation of a 3D finite difference time domain earthquake code on heterogeneous supercomputers. Proc. Comp. Sci. 18, 1255–1264. https://doi.org/10.1016/j.procs.2013.05.292.
Zhou, M., Symes, W.W., 2014. Wave equation based stencil optimizations on multi-core CPU. In: SEG Technical Program Expanded Abstracts 2014. Society of Exploration Geophysicists, pp. 3551–3555. https://doi.org/10.1190/segam2014-0537.1.