Proceedings of the 2013

International Conference on

Computational and Mathematical

Methods in Science and Engineering

Almería, Spain

June 24-27, 2013

CMMSE

VOLUMES I,II,III,IV & V

Editors: Ian Hamilton & Jesús Vigo-Aguiar

Associate Editors:

H. Adeli, P. Alonso, M.T. De Bustos, M. Demiralp, J.A. Ferreira, A. Q. M. Khaliq,

J.A. López-Ramos, P. Oliveira, J.C. Reboredo, M. Van Daele,

E. Venturino, J. Whiteman, B. Wade

Proceedings of the 2013 International Conference on Computational and Mathematical Methods in Science and Engineering

Cabo de Gata, Almería, Spain

June 24-27, 2013

CMMSE
Computational and Mathematical
Methods in Science and Engineering

Editors

I. P. Hamilton & J. Vigo-Aguiar

Associate Editors

H. Adeli, P. Alonso, M.T. De Bustos,
M. Demiralp, J.A. Ferreira, A. Q. M. Khaliq,
J.A. López-Ramos, P. Oliveira, J.C. Reboredo,
M. Van Daele, E. Venturino, J. Whiteman, B. Wade

ISBN 978-84-616-2723-3

Printed on acid-free paper

Proceedings of the 13th International Conference on Computational and Mathematical Methods in Science and Engineering, CMMSE 2013 24–27 June, 2013.

Performance analysis of SSE instructions in multi-core CPUs and GPU computing on FDTD scheme for solid and fluid vibration problems

Jorge Francés[1], Sergio Bleda[1], Andrés Márquez[1], Cristian Neipp[1], Sergi Gallego[1], Beatriz Otero[2] and Augusto Beléndez[1]

[1] *Dpto. de Física, Ingeniería de Sistemas y Teoría de la Señal, Universidad de Alicante, E-03080, Alicante (Spain)*

[2] *Dpt. d'Arquitectura de Computadors, Universitat Politènica de Catalunya, ES-08034, Barcelona (Spain)*

emails: `jfmonllor@ua.es`, `sergio.bleda@ua.es`, `andres.marquez@ua.es`, `cristian@ua.es`, `sergi.gallego@ua.es`, `botero@ac.upc.edu`, `a.belendez@ua.es`

Abstract

In this work a unified treatment of solid and fluid vibration problems is developed by means of the Finite-Difference Time-Domain (FDTD). The scheme here proposed introduces a scaling factor in the velocity fields that improves the performance of the method and the vibration analysis in heterogenous media. In order to accurately reproduce the interaction of fluids and solids in FDTD both time and spatial resolutions must be reduced compared with the set up used in acoustic FDTD problems. This aspect implies the use of bigger grids and hence more time and memory resources. For reducing the time simulation costs, FDTD code has been adapted in order to exploit the resources available in modern parallel architectures. For CPUs the implicit usage of the streaming SIMD (Singe Instruction Multiple Data) extensions in multi-core CPUs has been considered. In addition, the computation has been distributed along the different cores available by means of OpenMP directives. Graphic Processing Units (GPU) have been also considered and the degree of improvement achieved by means of this parallel architecture has been compared with the highly-tuned CPU scheme by means of the relative speed up. The speed up obtained by the parallel versions implemented were up to 7 and 30 times faster than the best sequential version for CPU and GPU respectively. Results obtained with both parallel approaches demonstrate that parallel programming techniques are mandatory in solid-vibration problems with FDTD.

Key words: FDTD, GPU, CPU, OpenMP, SSE, vibration
MSC 2000: 68U20, 65L12, 68W10

1 Introduction

Kane S. Yee published in 1966 the initial FDTD scheme [1]. During the next two decades the scientific community do not show too much interest in this method due to its high computational requirements. However, the growing of the computer power in the last three decades has permitted that FDTD became a reference in different fields of science such as electromagnetism [2], optics and acoustics. This new scenario has allowed to develop new applications and also new formulations that cover a wider range of problems. The first attempts of FDTD in acoustics were published by Botteldooren [3], LoVetri [4], Wang [5] and its application to solid mechanics was performed by Virieux [6] and Cao [7] in the field of seismology. The formulation of FDTD for solids can be separated in two different schemes. The former is based on the discretization of the Newton's second law and the Hook's law; the latter is based on the vectorial and scalar potentials derived from the two general laws of solid-mechanics mentioned below. Regarding this second approach there are several contributions [8–10]. In this work, the first scheme is developed in order to model solid and fluid vibrations. The formulation based on the scalar and vectorial potentials only has been developed efficiently for homogeneous media, due to the difficulties derived from the boundary conditions in the interfaces between solid and fluid [11].

On the other hand, the direct application of the finite-difference approach to the Newton's second law and the Hook's law allows to model the interaction between fluids and solids, due to the fact that the derivation of the initial FDTD scheme for acoustics is a particular case from these laws. The vibration analysis in fluids and solids require reduced values for time and spatial resolutions, since the propagation in solids use to be faster than in fluids such as air for instance. In addition, FDTD scheme computes the field distribution as a function of time, thus sometimes a big number of time steps is required in order to ensure steady state. FDTD also requires a discretization of the simulation region. The size of this grid affects directly the time simulation costs and also the memory requirements of the method. In order to reduce the time simulation costs in seismology some works related with GPU computing have been developed [12].

In this work, a unified treatment of fluid and solid FDTD analysis is performed. The formulation has been slightly modified from [13] in order to model efficiently the vibration fields for fluid-solid interaction problems. Moreover, a rigorous analysis of the performance of the 2D FDTD in both parallel architectures multi-core CPU and GPU is performed. Both parallel implementations of the method are compared with a sequential code that takes advantage of the auto-vectorization available in modern compilers being faster that the usual sequential codes. Moreover, the parallel CPU version takes advantage of the Streaming SIMD Extensions (SSE) available in modern microprocessors. The SSE set instructions has been directly used in order to exploit the potential of modern CPUs. In addition, parallel strategies have been considered in order to use all the available cores in the CPU, thus OpenMP directives have been also applied. This fine-tuned CPU version of the FDTD scheme has been compared with a massively parallel CUDA code for GPU computing. The idea is to accurately estimate how fast is a GPU against a CPU that exploits all its available resources. This analysis has been also performed for the standard FDTD scheme applied to optics [14] but it has not been done yet for the solid-fluid scheme to the best of our knowledge.

2 Finite-Difference Time-Domain method for the analysis of vibration problems

The fundamental constitutional equations for the propagation of elastic waves in solids can be derived in vectorial notation from the Newton's second law and the Hook's law obtaining the following well known identities:

$$\frac{\partial \boldsymbol{\tau}}{\partial t} = \lambda \mathbf{I}\left(\nabla\cdot\mathbf{v}\right) + \mu\left(\nabla\mathbf{v} + \mathbf{v}\nabla\right), \tag{1}$$

$$\rho\frac{\partial \mathbf{v}}{\partial t} = \nabla\cdot\mathbf{v}, \tag{2}$$

where $\boldsymbol{\tau}$ is the stress tensor, $\mathbf{I}$ is the identity matrix, λ, μ and ρ are the two Lamé constants and the density respectively. The velocity vector is denoted by $\mathbf{v}$. The Eqs. (1)-(2) describes the propagation in linear, homogeneous and non-lossy solid media. Basically, in solid media there are two types of waves, the p-waves or longitudinal waves and the s-waves also known as the transversal waves. The velocity of propagation of both waves is different and defined as follows:

$$c_p = \sqrt{\tfrac{\lambda+2\mu}{\rho}}, \quad c_s = \sqrt{\tfrac{\mu}{\rho}}. \tag{3}$$

where c_p and c_s are the p-wave and s-wave velocities, respectively. It is important to note that those materials with null second Lamé parameter μ (shear modulus) behave as fluids with a compressibility modulus defined as $k = -\lambda$. The presence of heterogeneities and the p and s waves deal to different ondulatory phenomena such as the Rayleigh's and Love's waves.

In this work a modified set of equations are considered based on a scaling of the velocity components. For the specific case of 2D simulation (x, y) the normal stresses from Eq. (1) can be expressed as follows

$$\frac{\partial \tau_{xx}}{\partial t} = c_p\frac{\partial V_x}{\partial x} + \frac{\lambda}{Z_0}\frac{\partial V_y}{\partial y}, \tag{4}$$

$$\frac{\partial \tau_{yy}}{\partial t} = c_p\frac{\partial V_y}{\partial y} + \frac{\lambda}{Z_0}\frac{\partial V_x}{\partial x}, \tag{5}$$

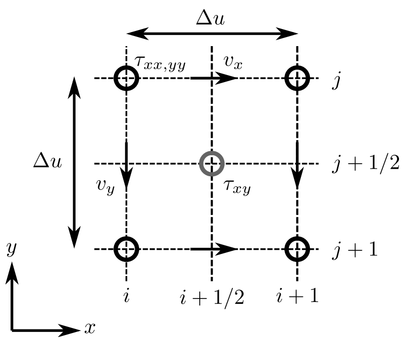

Figure 1: FDTD lattice for solids.

whereas the shear stress can be defined as:

$$\frac{\partial \tau_{xy}}{\partial t} = \frac{c_s^2}{c_p}\left(\frac{\partial v_x}{\partial y} + \frac{\partial v_y}{\partial x}\right). \tag{6}$$

The velocity components can be split from Eq. (2) as follows:

$$\frac{\partial V_x}{\partial t} = c_p \frac{\partial \tau_{xx}}{\partial x} + c_p \frac{\partial \tau_{xy}}{\partial y}, \quad \frac{\partial V_y}{\partial t} = c_p \frac{\partial \tau_{yy}}{\partial y} + c_p \frac{\partial \tau_{yx}}{\partial x}, \tag{7}$$

where $V_i = v_i Z_0$ with $i = x, y$ and $Z_0 = \rho c_p = \sqrt{\rho(\lambda + 2\mu)}$. The scaling of the velocity components has many advantages such as the possibility of handle closer values for the velocities in both solids and fluids. Note that usually the movement of the particles in solids tend to be smaller than in a fluid. On the other hand, this normalization improves the finite error precision in FDTD equations since avoids the round-off errors committed by the processor due to handling numbers with huge differences in their modulus.

This formulation can be easily related with the pressure-velocity scheme for fluid media considering $k = -\lambda$ and $p = -1/2(\tau_{xx} + \tau_{yy})$.

Figure 1 shows the lattice configurations of the FDTD method for analyzing the velocities and stress components. Reformulating Eqs. (4)-(7) by the FDTD method gives

$$\begin{aligned} \tau_{xx}|_{i,j}^{n+1} &= \tau_{xx}|_{i,j}^{n+1} + VPL\left(V_x|_{i+1/2,j}^{n} - V_x|_{i-1/2,j}^{n}\right) + \\ &\quad + VCL\left(V_y|_{i,j+1/2}^{n} - V_y|_{i,j-1/2}^{n}\right), \end{aligned} \tag{8}$$

$$\begin{aligned} \tau_{yy}|_{i,j}^{n+1} &= \tau_{yy}|_{i,j}^{n+1} + VPL\left(V_y|_{i,j+1/2}^{n} - V_y|_{i,j-1/2}^{n}\right) + \\ &\quad + VCL\left(V_x|_{i+1/2,j}^{n} - V_x|_{i-1/2,j}^{n}\right), \end{aligned} \tag{9}$$

$$\tau_{xy}|_{i+1/2,j+1/2}^{n+1} = \tau_{xy}|_{i+1/2,j+1/2}^{n+1} + VSL\left(V_x|_{i+1/2,j+1}^{n} - V_x|_{i+1/2,j}^{n} + \right.$$

$$
\begin{aligned}
&\quad + V_y|^n_{i+1,j+1/2} - V_y|^n_{i,j+1/2}\Big), && (10)\\
V_x|^{n+1/2}_{i+1/2,j} &= V_x|^{n-1/2}_{i+1/2,j} + VPL\Big(\tau_{xx}|^n_{i+1/2,j} - \tau_{xx}|^n_{i-1/2,j}+ \\
&\quad + \tau_{xy}|^n_{i+1/2,j+1/2} - \tau_{xy}|^n_{i+1/2,j-1/2}\Big), && (11)\\
V_y|^{n+1/2}_{i,j+1/2} &= V_y|^{n-1/2}_{i,j+1/2} + VPL\big(\tau_{yy}|^n_{i,j+1} - \tau_{yy}|^n_{i,j}+ \\
&\quad + \tau_{yx}|^n_{i+1/2,j+1/2} - \tau_{yx}|^n_{i-1/2,j-1/2}\Big), && (12)
\end{aligned}
$$

where

$$VPL = \frac{\Delta t c_p}{\Delta u}, \quad VSL = VPL\left(\frac{c_s}{c_p}\right)^2, \quad VCL = \frac{\Delta t\lambda}{\Delta u Z_0}. \tag{13}$$

The spatial and time resolution of FDTD are denoted by Δu and Δt, where square cells are considered ($\Delta u = \Delta x = \Delta y$) (see Fig. 1). Regarding stability and dispersion, FDTD schemes must ensures the Courant-Friedrichs-Lewy (CFL) that for the specific case of solid-fluid interaction is defined as:

$$S = \frac{c_{l,max}\Delta t}{\sqrt{2}\Delta u} \leq 1, \quad R = \frac{c_{s,min}}{f_{max}\sqrt{2}\Delta u} \geq 10 \tag{14}$$

where $c_{l,max}$ is the maximum value of the longitudinal velocity in the domain and $c_{s,min}$ the lowest tangential velocity [13, 15]. The boundaries of the domain have been truncated by means of a Perfectly Matched Layer (PML) that solves the problems due to artifacts and reflections on the boundaries which are very common in finte-difference schemes [13, 16].

3 Computational optimization

In this section the strategies considered for accelerating FDTD for solid-fluid vibration analysis are detailed. In this work an Intel i7-950 processor with 8MB of cache, a clock speed of 3.06 GHz and the possibility of handle efficiently eight threads has been used. Regarding GPU computing, a GTX470 GPU with Fermi architecture is considered for the parallel implementation of the FDTD for solid-fluid vibration analysis. The basic computing unit in this type of hardware consists of 32 threads and this arrangement is defined as a warp. The GPU is capable of swapping warps into and out of context without any performance overhead. This functionality provides an important method of hiding memory and instruction latency on the GPU hardware. The different warps invoked are also arranged into blocks of threads.

3.1 Multi-core CPU and SSE instructions

The SIMD instructions follow the parallel computation model and is the most cost-effective way of accelerating floating- or double-point performance in modern processors. Here, only

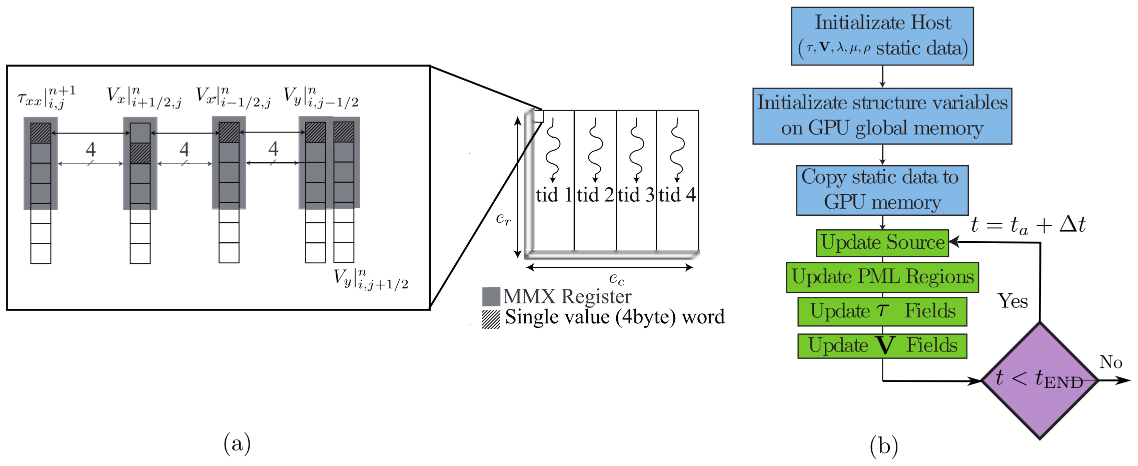

Figure 2: a) Illustration showing the scheme for solving Eq. (8) in CPU. b) Flowchart of GPU parallel programming for FDTD for fluid-solid vibration analysis.

single precison has been considered for FDTD simulations, since single precision is accurate enough for FDTD applications. In order to successfully apply the SIMD instructions, load operations must be done under a set of aligned bytes [17]. For that reason, the allocation of the memory for a matrix with e_r rows and e_c columns is done as a single aligned column vector by means of a new array class fully implemented in C++ [18]. Thus each position is reached taking into account that the storage order is by columns. For simplicity, the scheme of how Eq. (8) is computed by means of the MMX registers and OpenMP is shown in Fig. 2a. The updating process for the rest of the components follow the same scheme. For simplifying the notation, also the PML notation has been supressed but it is also included in the optimization process. For instance, the update of τ_{xx} requires several aligned loads that store 4 consecutive values of the terms involved: $V_x|^n_{i+1/2,j}$, $V_x|^n_{i-1/2,j}$, $V_y|^n_{i,j-1/2}$,$V_y|^n_{i,j+1/2}$. Note that the physical parameters that define the media are also stored in the MMX registers. The next step is to perform the arithmetic operations by means of the MMX registers using the intrinsics functions (`_mm_sub_ps(__m128 a, __m128 b)`, `_mm_add_ps(__m128 a, __m128 b)` or `_mm_mul_ps(__m128 a, __m128 b)` for instace). In addition, OpenMP has been considered in order to parallelize the updating of each field component. Modern CPUs contain several cores that can be used by means of shared memory schemes in order to split the computational load amongst the different cores. By means of OpenMP directives, each component has been parallelized distributing the whole computation by columns. As can be seen in Fig. 2a, each thread is in charge of computing a set of columns of each field component, whereas each thread uses extensively the vectorial SSE instructions along the rows direction.

3.2 Graphic Processing Units (GPUs)

Regarding the SF-FDTD implementation and GPU computing, a number of blocks related with the number of rows and columns are invoked by means of the kernel functions and an array of 192×2 threads are launched per block [19].

Besides the potential of the CUDA kernel, it is necessary to divide the whole computation process in several kernels focused on computing each component of the vibration field. Fig. 2b summarizes the invocation path of the kernels related with the FDTD implementation. Firstly, an initialization in host of the fields to be computed is perfomed. Secondly, the allocation of these componentes is done inside the GPU, those fields that must be filled such as the physical parameters that models the media are copied to the GPU memory. Thirdly, the FDTD computation is performed by a set of kernels that updates each component of the vibration field (stress and velocities). Finally, the fields are downloaded to the host. In this flow chart the post-process is omitted, but mandatory downloads of the τ and $\mathbf{V}$ components must be considered in order to compute the specific desired outputs. The time costs of this process has been also considered in order to compute the speed up in the results section.

4 Results

Firstly, the computational results are summarized in Fig. 3. The simulation grid is modified as a function of the number of rows (e_r) and columns (e_c). More specifically, Fig. 3a-b shows the time simulation cost and the speed up respectively for a set of simulations with $e_r = 500$ varying the number of columns. The speed up has been computed considering as sequential version an auto-vectorized code. This auto-vectorized code is expected to be the fastest sequential code achievable by an programmer without advanced knowledge on parallelization techniques. The same results for $e_c = 1000$ and $e_c = 2000$ are shown in Fig. 3c-d and Fig. 3e-f respectively. As can be seen the time simulation costs grows exponentially with the simulation size and in all cases the GPU and the fine-tuned CPU version are faster than the sequential version. The speed up obtained from the CPU parallel code optimized with SSE instructions and OpenMP behaves quite constant as a function of e_c and a slight maximum can be identified for the specific case of $e_c = 500$ and for $e_r \approx 0.3$ kcells. This local maximum disappears in Fig. 3d and Fig. 3f since it belongs to an optimal usage of the cache memory available in the microprocessor. As the simulation size becomes greater the simulation does not fit in the cache and the speed up remains constant. The overall speed up obtained by this version is closer to 7 compared with the sequential auto-vectorized version. On the other hand, the GPU CUDA version remains more homogeneous as a function of the simulation size and the speed ups are quite near to 30 respect of the auto-vectorized sequential version.

In order to accurately compare the GPU and the SSE+OpenMP CPU version, the rel-

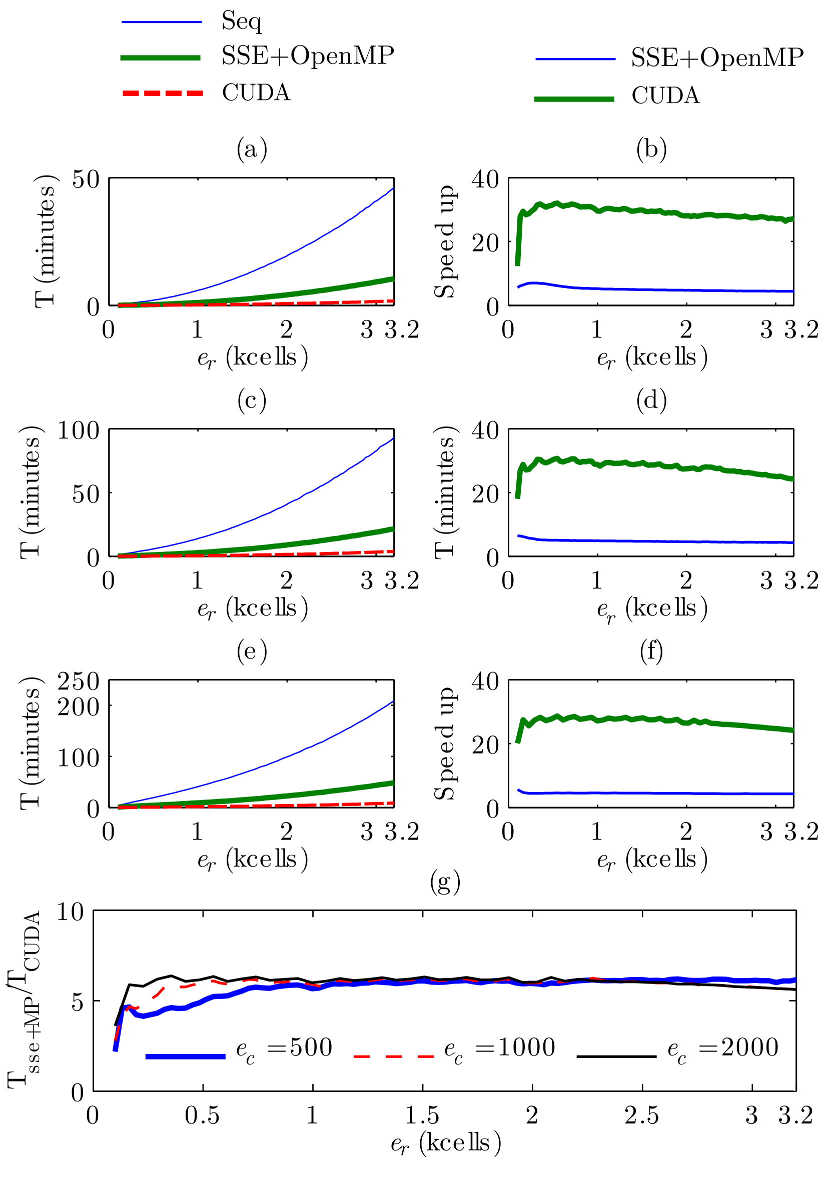

Figure 3: Computational results: (a), (c) and (e) show the time simulation as a function of the number of cells for the specific case of $e_c = 500$, $e_c = 1500$ and $e_c = 2000$ respectively. (b), (d) and (f) represents the speed up for the same situation. (g) Relative speed up between the fine-tuned CPU and GPU codes.

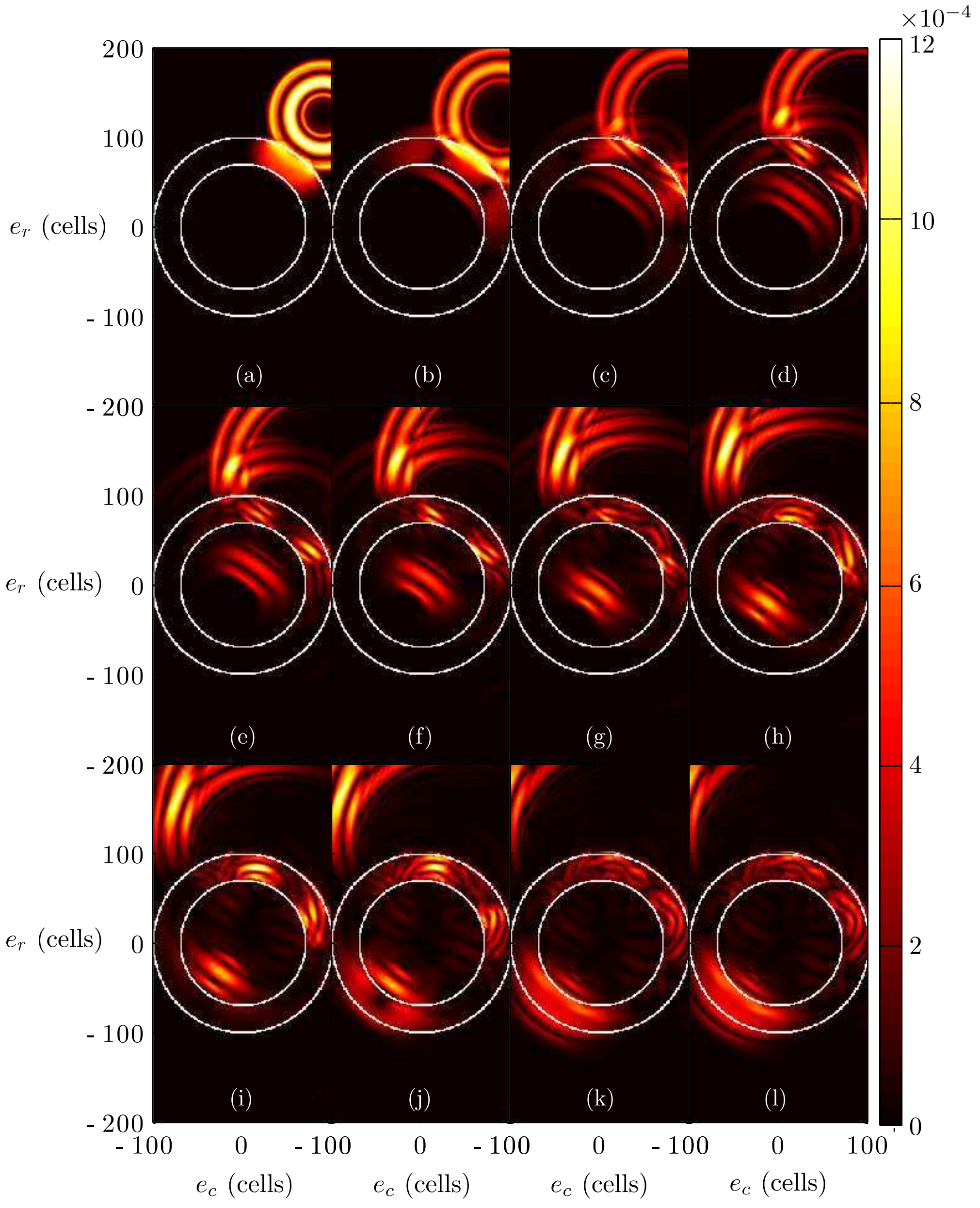

Figure 4: Modulus of the the normalized velocity as a function of the space and time: (a) $n_{\text{step}} = 400$, (b) $n_{\text{step}} = 500$, (c) $n_{\text{step}} = 600$, (d) $n_{\text{step}} = 700$, (e) $n_{\text{step}} = 800$, (f) $n_{\text{step}} = 900$, (g) $n_{\text{step}} = 1000$, (h) $n_{\text{step}} = 1100$, (i) $n_{\text{step}} = 1200$, (j) $n_{\text{step}} = 1300$, (k) $n_{\text{step}} = 1400$, (l) $n_{\text{step}} = 1500$

ISBN: 978-84-616-2723-3

ative speed up between them has been computed and shown in Fig. 3g. As can be seen the GPU version is near to 7 times faster than the fine-tuned CPU code and this behavior remains constant as a function of the simulation size. The effect of the cache size in the SSE+OpenMP CPU version can be also seen in Fig. 3g for $e_r \approx 0.3$. The slight differences in the relative speed up curves reveals that the GPU version is more competitive for bigger simulation sizes, reaching values of near to 7. This value is significantly different from the speed up values obtained in Fig. 3b, 3d and 3f. These results illustrates that a fine-tuned CPU version can be competitive against a massively computational architecture such as GPU in which the number of cores is hundred times greater than in a single CPU. Nevertheless, the authors consider that GPU codes are mandatory in this type of applications in which the requirements in terms of grid size and time steps can be unaffordable for sequential codes and even to multi-core processors.

Fig. 4 shows a sequence of the modulus of the scaled velocity as a function of the grid size and the time-simulation steps. The pressure source is located near the upper right corner. In the center of the domain there is a ring filled with a solid material ($\lambda = 1.95$ GPa, $\mu = 36$ MPa and ρ=900 kg/m^3). It can be easily seen that the propagation inside the solid is faster rather in the fluid and also that different pheonomena can be identified inside the ring. More specifically, Fig. 4d-f shows in the upper right side of the ring a set of transversal waves that are reflected along the ring boundaries. The shear waves cannot travel in fluid media thus, only longitudinal waves travels in the fluid hole inside the ring. This sequence reveals in a qualitatively way the potential of this scheme and are consistent with those obtained in [13]. The set-up of the FDTD method is the following: $e_r = 650$, $e_c = 300$, $\Delta u = 3.7$ mm, $\Delta t = 1.77$ μs and $f_{\max} = 40$ kHz.

5 Conclusions

In this work a unified scheme for FDTD analysis of vibrations on fluid and solid media is considered. A scaling factor has been introduced in the formulation in order to improve its implementation and also for optimizing the vibration analysis in heterogenous media. The formulation has been implemented in parallel hardware architectures such as multi-core CPU and GPU. The CPU optimized version takes advantage of the SSE instructions and also of the multiple cores available by means of OpenMP directives. The results reveal that a fine-tuned CPU version can be competitive compared to GPU codes, since it reaches speed ups closer to 7 compared to the auto-vectorized sequential version. Nevertheless, GPU computing is mandatory for massively computations due to the fact that the speed up obtained is up to 30. It's worth to note that the speed up obtained from GPU codes can vary dramatically as a function of the sequential code selected. In this work a comparison between the multi-core CPU code acclerated with SSE and OpenMP is compared with the GPU CUDA based code in order to establish accurately the degree of improvement achieved revealing that GPU is slightly more than 5 times faster than the fine-tuned CPU version. Finally, FDTD applied to the analysis of elastic waves in solids and fluids has been demonstrated to have a low computational intensity, thus needing parallel strategies in order to reduce the time simulation costs. The authors are considering to extend the current work to 3-D and also to include the new AVX instructions available in modern multi-core CPUs.

Acknowledgements

The work is partially supported by the "Ministerio de Economía y Competitividad" of Spain under project FIS2011-29803-C02-01, by the Ministry of Science and Technology of Spain number contract TIN2007-60625, and by the "Generalitat Valenciana" of Spain under projects PROMETEO/2011/021 and ISIC/2012/013

References

[1] K. S. Yee, *Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media*, IEEE Trans. Antennas Propag. **14** (1966) 302–307.

[2] A. Taflove and S. C. Hagness, *Computational electrodynamics: The Finite-Difference Time-Domain method*, Artech House, Norwood, MA, 2004.

[3] D. Botteldooren, *Finite-difference time-domain simulation of low-frequency room acoustic problems*, J. Acoust. Soc. Am. **98** (1995) 3302–3308.

[4] J. LoVetri, D. Mardare, G. Soulodre, *Modeling of the seat dip effect using the finite-difference time-domaim method*, J. Acoust. Soc. Am. **100** (1996) 2204–2212.

[5] S. Z. Wang, *Finite-difference time-domain approach to underwater acoustic scattering problems*, J. Acoust. Soc. Am. **99** (1996) 1924–1931.

[6] J. Virieux, *P-SV wave propagation in heterogeneous media; velocity-stress finite-difference method*, Geophysics **51** (1986) 889-901.

[7] S. H. Cao and S. Greenhalgh, *Finitedifference simulation of P-SV-wave propagation: a displacement-potential approach*, Geophysical Journal International **109** (1992) 525–535.

[8] M. Sato, Y. Takahata, M. Tahara and I. Sakagami, *Expression of contour vibration modes of a square plate by scalar and vector velocity potentials*, Acoustical Science and Technology **23** (2002) 346–349.

[9] M. SATO, *Formulation of the FDTD method for separating the particle velocity vectors of an elastic wave field into longitudinal and shear wave components*, Acoustical Science and Technology **25** (2004) 382–385.

[10] M. SATO, *Comparing three methods of free boundary implementation for analyzing elastodynamics using the finite-difference time-domain formulation*, Acoustical Science and Technology **28** (2007) 1346–3969.

[11] J. FRANCÉS, J. RAMIS AND J. VERA, *A 3D FDTD scheme for analysis of the elastic wave fields in solids*, In Proceedings of the ICSV16, 5–9 July, Kraków, Poland (2009) 1–8.

[12] T. OKAMOTO AND H. TAKENAKA, *Large-scale simulation of seismic-wave propagation of the 2011 Tohoku-Oki M9 earthquake* Proceedings of the International Symposium on Engineering Lessons Learned from the 2011 Great East Japan Earthquake (2011) 349–360.

[13] N. JIMÉNEZ GONZÁLEZ, *Simulación de tejidos vegetales mediante diferencias finitas*, Tesis de Máster, EPSG-UPV, 2009

[14] J. FRANCÉS, S. BLEDA, C. NEIPP, A. MÁRQUEZ, I. PASCUAL AND A. BELÉNDEZ, *Performance analysis of the FDTD method applied to holographic volume gratings: Multi-core CPU versus GPU computing*, Comput. Phys. Comm. **184** (2013) 469–479.

[15] C. T. SCHROEDER, W. R. SCOTT, *On the stability of the FDTD algorithm for elastic media at a material interface*, IEEE Transactions on Geoscience and Remote Sensing **40** (2002) 474–481.

[16] C. J. RANDALL, *Absorbing boundary condition for the elastic wave equation: Velocitystress formulation*, Geophysics **53** (1989) 1141–1152.

[17] T. SHREEKANT, T. HUFF, *Internet streaming simd extensions*, IEEE Computer Society Press **32** (1999) 26–34.

[18] J. FRANCÉS, S. BLEDA, S. GALLEGO, C. NEIPP, A. MÁRQUEZ, I. PASCUAL AND A. BELÉNDEZ, *Development of a unified FDTD-FEM library for electromagnetic analysis with CPU and GPU computing*, J. of Supercomputing **164** (2013) 28–37.

[19] J. FRANCÉS, S. BLEDA, M. L. ÁLVAREZ, F. J. MARTÍNEZ, A. MÁRQUEZ, C. NEIPP, A. BELÉNDEZ, *Analysis of periodic anisotropic media by means of split-field FDTD method and GPU computing*, In Proc. of SPIE **84980** (2012) 84980K-84980K-9.