Towards Performance Optimisation of the FDTD Method via Matrix Casting Approach

Maksims Abaļenkovs
School of Mathematics, The University of Manchester, Manchester M13 9PL, UK, m.abalenkovs@manchester.ac.uk

***Abstract*—This publication presents a matrix casting approach for the FDTD method. Applied in 2D and 3D, dimensions it is subsequently parallelised with OpenMP and tested in basic air-filled and realistic landmine-detection scenarios.**

***Index Terms*—electromagnetic propagation, numerical simulation, performance analysis, finite difference methods.**

I. Introduction

A high-precision simulation of electromagnetic behaviour of large-scale devices with fine geometric features requires advanced numerical techniques. Apart from high geometric resolution, the method should provide high computational performance. One popular choice in the world of Computational Electromagnetics is the Finite-Difference Time-Domain (FDTD) method [1]. Following the 1D prototype presented in [2], this project explores the idea of matrix casting further. Here it is applied to 2D and 3D electromagnetic field updates, that are represented in terms of matrix–vector products.

New CPU architectures feature Fused Multiply-Add (FMA) instruction sets that allow floating point multiplication followed by addition to be performed as a single floating point operation, *i.e. Fused Multiply-Add* operation will require the same number of CPU cycles as a single multiplication or addition [3]. Advanced Vector Extensions (AVX) constitute another important development in the area of SIMD instruction sets. AVX allows the CPU vector units to process multiple numbers simultaneously. For example, a modern Haswell CPU with FMA and AVX2 instruction sets can perform 16 double precision FLoating-point Operations (FLOPs) per cycle [4].

Unexploded Ordnance (UXO) presents a humanitarian and environmental problem worldwide. The land contaminated by explosives equates to $\approx\ 300\,\mathrm{km}^2$. The number of severe injuries and causalities rises to above 4000 people per year, of whom 42% are children [5]. Demining is an expensive and time consuming task. The demining process also involves the recognition and rejection of metallic clutter. There are approximately 100 to 1000 inert objects for every mine. Usually UXO detection relies on a metal detector, that exploits the Electromagnetic Induction (EMI) principle. The EMI system consists of an excitation coil T_{x}, emitting primary magnetic field and a receiver coil R_{x}, measuring the induced field, scattering from a metallic object. The primary magnetic field creates Eddy currents in a metallic object. The object demagnetises and emits a secondary field, that will be detected at the receiver. The information about the size, shape, material, location and orientation of the target can be represented by means of a magnetic polarisability tensor [5], [6]. To address the vital problem of the UXO detection, the 3D FDTD solver with matrix casting will be applied to simulate the EMI scenario.

This publication is arranged as follows: Section II outlines the mathematical idea of matrix casting, Section III presents possible ways to parallelise FDTD for shared memory architectures, Section IV describes the testing scenarios (including the problem of EMI) and analyses the numerical results, while Section V concludes the narrative and drafts the plan for future work.

II. Mathematical Background

A. Solution of Maxwell's Equations in the FDTD Method

Eqs. (1) and (2) define Maxwell's curl equations for linear, isotropic, non-dispersive and lossy materials:

$$\varepsilon\frac{\partial \boldsymbol{E}}{\partial t} = \quad \nabla\times\boldsymbol{H} - \boldsymbol{J} - \sigma\boldsymbol{E}, \tag{1}$$

$$\mu\frac{\partial \boldsymbol{H}}{\partial t} = -\nabla\times\boldsymbol{E} - \boldsymbol{K} - \sigma^{*}\boldsymbol{H}, \tag{2}$$

where $\boldsymbol{E}, \boldsymbol{H}$ stand for electric and magnetic fields, $\boldsymbol{J}, \boldsymbol{K}$ are electric and magnetic source currents, ε, μ denote the electric permittivity and magnetic permeability, and σ, σ^{*} symbolise electric conductivity and the equivalent magnetic loss.

Expanding vector components in Cartesian coordinates and eliminating partial derivatives with respect to z-axis, Maxwell's equations reduce to 2D. This form is known as the *transverse-electric mode with respect to* z (TEz):

$$\frac{\partial E_x}{\partial t} = \frac{1}{\varepsilon}\left[\ \frac{\partial H_z}{\partial y} - (J_{src,\,x} + \sigma E_x)\right], \tag{3}$$

$$\frac{\partial E_y}{\partial t} = \frac{1}{\varepsilon}\left[-\frac{\partial H_z}{\partial x} - (J_{src,\,y} + \sigma E_y)\right], \tag{4}$$

$$\frac{\partial H_z}{\partial t} = \frac{1}{\mu}\left[\ \frac{\partial E_x}{\partial y} - \frac{\partial E_y}{\partial x} - (K_{src,\,z} + \sigma^{*} H_z)\right]. \tag{5}$$

B. Casting Stencil-based Equations into Matrix Form

In classical FDTD method partial derivatives representing time and space are approximated with central-differences [7]. Applying approximations to Eqs. (3) to (5) results in a discretised form of the equations. To simplify the notation and increase the computational efficiency of the method, discrete

equations can be cast into a matrix form. Electric field updates will be governed by the following matrix–vector product:

$$\boldsymbol{E}^T = \boldsymbol{\Gamma}^T \boldsymbol{P}, \quad \text{where} \tag{6}$$

$$\boldsymbol{E}^T = \begin{pmatrix} E^{n+\frac{1}{2}}_{x\left(i,j+\frac{1}{2},k+\frac{1}{2}\right)} & E^{n+\frac{1}{2}}_{y\left(i-\frac{1}{2},j+1,k+\frac{1}{2}\right)} \end{pmatrix}, \tag{7}$$

$$\boldsymbol{\Gamma}^T = \begin{pmatrix} \Gamma_1 & \Gamma_2 \end{pmatrix}, \tag{8}$$

$$\boldsymbol{P} = \begin{pmatrix} E^{n-\frac{1}{2}}_x & E^{n-\frac{1}{2}}_y \\ \Delta H^n_{z(j+1,j)}/\Delta y & -\Delta H^n_{z(i,i-1)}/\Delta x \end{pmatrix}, \tag{9}$$

with

$$\Gamma_1 = \frac{1-\sigma\Delta t/2\varepsilon}{1+\sigma\Delta t/2\varepsilon}, \quad \Gamma_2 = \frac{\Delta t/\varepsilon}{1+\sigma\Delta t/2\varepsilon}. \tag{10}$$

Magnetic field component H_z will be updated by the inner product of two vectors:

$$H^{n+1}_{z\left(i,j+1,k+\frac{1}{2}\right)} = \begin{pmatrix} Z_1 & Z_2/\Delta y & -Z_2/\Delta x \end{pmatrix} \begin{pmatrix} H^n_z \\ \Delta E^{n+\frac{1}{2}}_{x\left(j+\frac{3}{2},j+\frac{1}{2}\right)} \\ \Delta E^{n+\frac{1}{2}}_{y\left(i+\frac{1}{2},i-\frac{1}{2}\right)} \end{pmatrix}, \tag{11}$$

with

$$Z_1 = \frac{1-\sigma^*\Delta t/2\mu}{1+\sigma^*\Delta t/2\mu}, \quad Z_2 = \frac{\Delta t/\mu}{1+\sigma^*\Delta t/2\mu}. \tag{12}$$

Parameters $\sigma, \varepsilon, \sigma^*, \mu$ are location-dependent, but the spatial indices were omitted from Eqs. (10) and (12) for clarity.

In case of a full-wave FDTD solver in 3D discretised Maxwell's equations can be cast into two matrix–vector products. For the sake of brevity, these equations are omitted in this publication.

III. Parallelising Matrix Form Equations with OpenMP

The majority of modern CPUs support FMA and AVX instructions, *i.e.* they are capable of scheduling a series of algebraic operations and performing them in one cycle. Apart from casting the equations into a matrix form, further reduction of the execution time may be expected from the vectorisation offered by the FMA and AVX instructions [3], [4]. Another dimension improving the performance of the stencil-based code is provided by multiple cores of contemporary CPU. OpenMP framework [8] is one of the most popular tools to exploit the shared memory parallelism of the FDTD solver.

The following ways to calculate the electromagnetic field values were implemented and tested in this project:

- **expl** uses explicit finite-difference formulae
- **func** applies custom, user-defined function to compute the stencil
- **dotp** relies on Fortran intrinsic function `dot_product()`

```
!$omp parallel do
do i = imin, imax
  do j = jmin, jmax
    ! expl pre
    E(i,j,1) = Gm(i,j,1)* E(i,j,1) +          &
               Gm(i,j,2)*(Hz(i,j)-Hz(i,j-1))/dy
    E(i,j,2) = Gm(i,j,1)* E(i,j,2) +          &
               Gm(i,j,2)*(Hz(i-1,j)-Hz(i,j))/dx
    ! func pre
    E(i,j,1) = calc_E( Gm(i,j,:), E(i,j,1),     &
                       Hz(i,j),   Hz(i,j-1), dy )
    E(i,j,2) = calc_E( Gm(i,j,:), E(i,j,2),   &
                       Hz(i-1,j), Hz(i,j), dx )
    ! matm pre
    E(i,j,:) = matmul(Gm(i,j,:),                                  &
              reshape([E(i,j,1), (Hz(i,j)-Hz(i,j-1))/dy,          &
                       E(i,j,2), (Hz(i-1,j)-Hz(i,j))/dx], [2, 2]))
    ! dotp pre
    E(i,j,1) = dot_product(Gm(i,j,:),                             &
                           [E(i,j,1), (Hz(i,j)-Hz(i,j-1))/dy])
    E(i,j,2) = dot_product(Gm(i,j,:),                             &
                           [E(i,j,2), (Hz(i-1,j)-Hz(i,j))/dx])
  end do
end do
!$omp end parallel do
```

Fig. 1. **Fortran code with OpenMP pragmas (2D case)**

- **ddot** utilises BLAS function `ddot()` to perform the vector dot product
- **dgmv** adopts BLAS routine `dgemv()` to calculate a matrix–vector product
- **matm** employs Fortran intrinsic function `matmul()`
- **dgmm** computes a matrix–matrix product by means of a BLAS function `dgemm()`

Fig. 1 shows four different approaches to updating electromagnetic field values. Only one of the alternatives is required in a practical FDTD solver. The loops over space coordinates i, j were parallelised with `$omp parallel do` pragmas. For the sake of simplicity, only electric field updates with precomputed coefficients $\Gamma_{\{1,2\}}$ are illustrated in 2D. Magnetic field updates with dynamically "just-in-time" calculated coefficients $Z_{\{1,2\}}$ in 3D are obtained by analogy.

IV. Numerical Results

A. Experiment Setup

Three sets of experiments were run in order to analyse performance of the parallel matrix casting approach in the 2D and 3D FDTD methods. Fig. 2 illustrates the scenario setting in 2D case. The simulation domain occupied a square filled with air. In all of the experiments the computation space has been terminated by 1st order Mur's Absorbing Boundary Condition (ABC) [9]. Excitation source and the observation point were placed on the horizontal axis of symmetry. A Gaussian

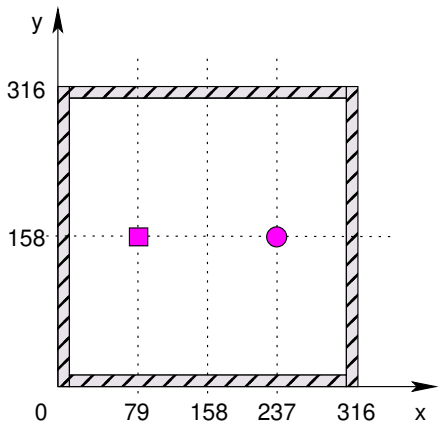

Fig. 2. **Air-filled 2D FDTD scenario setting.** Excitation source is marked with a square and the observation point—with a circle (both are shown in magenta colour). The scale is not observed.

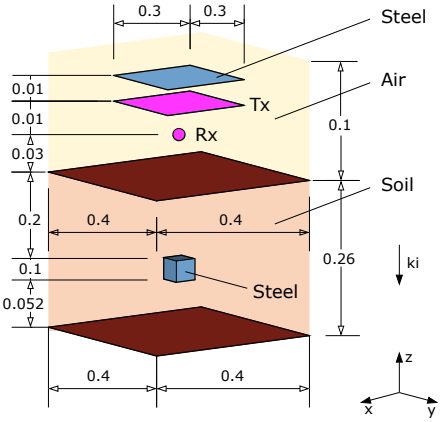

Fig. 3. **EMI 3D FDTD scenario setting in isometric projection.** All sizes are shown in metres. Transmitter coil T_{x} with excitation sources is denoted by a rectangle and the receiver coil R_{x} with an observation point—by a sphere (both are shown in magenta colour). The scale is not observed.

pulse has been used as a "soft" source to excite the y (2D) or z (3D) component of the electric field:

$$J^n_{src} = \exp\left[-\left(\{t - 3\,T\}/T\right)^2\right], \text{ with } T = m\,f^{-1}_{\max}. \quad (13)$$

Table II lists all FDTD simulation parameters in detail. Each column of Table II corresponds to a particular set of experiments: (i) 2D, (ii) 3D and (iii) EMI. Symbols $\mathrm{n}\{\mathrm{i},\mathrm{j},\mathrm{k},\mathrm{t}\}$ denote the spatio-temporal dimensions of the grid, $T_{\mathrm{x}}, R_{\mathrm{x}}$ mark locations of the excitation and observation points, $\Delta\{x, y, z, t\}$ set the spatio-temporal increments of the numerical method, N_{CFL} is the Courant–Friedrichs–Lewy (CFL) condition number, χ is the numerical resolution (number of points per wavelength), $f_{\max}$ describes the maximum wave frequency, while m is the width factor of the Gaussian pulse (See Eq. (13) for details). If a parameter value in the 3D case coincided with the value in the 2D case, it was not duplicated in the table. To make the computation workload in the 2D case equivalent to 3D, the spatial domains in experiments (i) and (ii) were set to $316^2 \approx 46^3$. Each experiment in air-filled 2D and 3D scenarios was run 10 times, and Figs. 4 and 5 present averaged results.

The third set of experiments addresses a practical application of the FDTD method to a UXO detection [10]. Only the forward part of the detection problem is considered in this project. The testing scenario depicted in Fig. 3 emulates low-frequency EMI—a popular method used for UXO detection. Transmitter coil T_{x} is modelled by the two plates of $0.3\,\mathrm{m} \times 0.3\,\mathrm{m}$, where the top plate is made of steel. The bottom plate serves as a plane wave emitter. It injects magnetic current $\boldsymbol{K}$ into the z component of the magnetic field. Receiver coil R_{x} is modelled by an observation point offset by $0.01\,\mathrm{m}$ down the y-axis. The lower part of the spatial domain is occupied by a $0.4\,\mathrm{m} \times 0.4\,\mathrm{m} \times 0.26\,\mathrm{m}$ soil medium. Finally, the test UXO object of size $0.001\,\mathrm{m}^3$ was buried $0.2\,\mathrm{m}$ under the ground. The material specifications used in this scenario are given in Table I. The transmitter plate and the landmine were made of steel. The upper part of the domain was occupied by air and the lower part—by the soil medium. The signal was transmitted at a typical EMI frequency $f_{\max} = 1.58\,\mathrm{kHz}$. To properly accommodate the landmine in the simulation, the spatial increments were set to $\Delta\{x, y, z\} = 2\,\mathrm{mm}$. The simulation lasted for $t = 1\,\mathrm{ms}$.

TABLE I
Material specifications

Propagation medium	σ [S/m]	ε_r	σ^* [Ω/m]	μ_r
Air	0.0	1.0	0.0	1.0
Soil	1.0	1.0	0.0	1.0
Stainless steel (grade 316)	1.3×10^6	1.0	0.0	1.0

B. Hardware Platform

Numerical experiments were performed on the Saturn Alembert cluster at the Innovative Computing Laboratory (ICL) at University of Tennessee in Knoxville (UTK). A detailed hardware platform specification is given in Table III.

The software has been compiled with the Intel Fortran compiler `ifort`, version 17.0.4 (20170411) using `-O3 -ipo -xCORE-AVX2 -funroll-loops -fma -qopenmp -w -vec-report0 -opt-report0` flags.

TABLE II
Simulation settings

Parameter	**2D case**	**3D case**	**EMI case**
$\mathrm{n}\{\mathrm{i},\mathrm{j},\mathrm{k}\}$	316^2	46^3	$200^2 \times 180$
nt	10^5		279170
source type	soft		
T_{x} location	$(79, 158)$	$(12, 23, 23)$	$x, y \in [25, 175], z = 150$
R_{x} location	$(237, 158)$	$(34, 23, 23)$	$(100, 100, 145)$
$\Delta\{x, y, z\}$	5 cm		2 mm
Δt	109.667 ps	89.551 ps	3.582 ps
N_{CFL}	0.93		
χ	20		9.487×10^7
$f_{\max}$	2.998 GHz		1.58 kHz
m	0.5		2.75×10^6

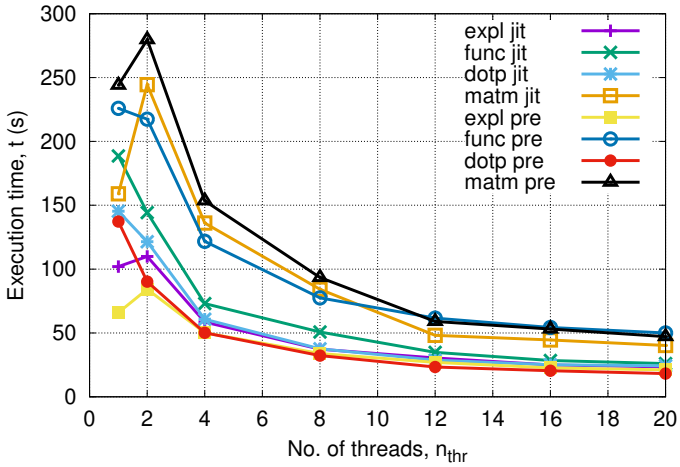

Fig. 4. **No. of OpenMP threads vs execution time in 2D FDTD**

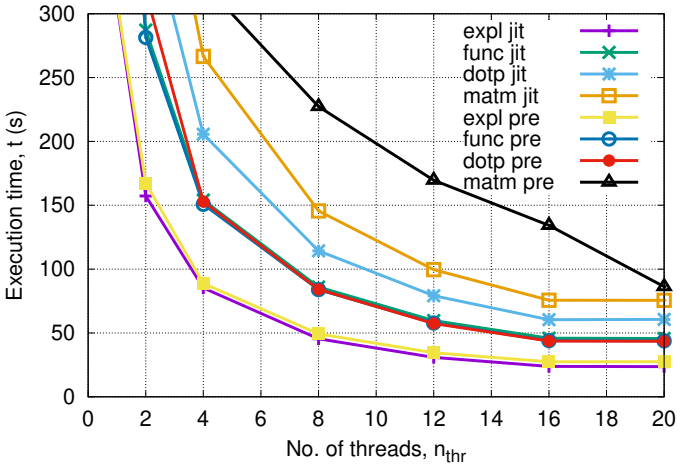

Fig. 5. **No. of OpenMP threads vs execution time in 3D FDTD**

Execution time has been measured by means of the `cpu_time()` Fortran subroutine.

C. Result Analysis

Fig. 4 presents performance results for various stencil updates in 2D FDTD. Abbreviations `jit` and `pre` denote the FDTD update equations with "just-in-time" calculated and pre-computed coefficients $\Gamma_{\{1,2\}}, Z_{\{1,2\}}$. The most high-performing alternative for the electromagnetic field update is `expl pre`, which provides 22.53% change over the second-best curve achieved by the matrix cast formulation with pre-computed coefficients `dotp pre`. Variants `expl jit`, `dotp jit` and `func jit` approach the best results for a high number of OpenMP threads ($n_{\text{thr}} \geq 12$). On the other hand implementations based on the matrix–matrix product `matm jit`, `matm pre` and the variant using user-defined update function `func pre` did not bring the expected performance benefit.

Fig. 5 compares performance of standard stencil updates with the matrix casting approach. In the 3D case, the explicit update curves `expl jit` and `expl pre` almost overlap with each other. The percentage change between `expl jit` and `expl pre` constitutes 9.09%. There is a considerable performance gap between `expl jit` and the second-best alternatives `dotp pre`, `func jit` and `func pre`. The percentage change between `expl jit` and `dotp pre` equals 89.45%. Another interesting observation is the overlapping of `jit` and `pre` variants of the `expl` and `func`-based approaches. Contrary to the 2D case, pre-computing FDTD coefficients $\Gamma_{\{1,2\}}, Z_{\{1,2\}}$ results in a negligible performance boost. The shortest computation time is obtained with $n_{\text{thr}} = 20$ for both 2D and 3D cases. However, for these small spatial domain sizes (316^2 and 46^3) the performance curves are likely to saturate and exhibit constant or larger computation time behaviour for $n_{\text{thr}} \geq 20$.

Alternative formulations based on BLAS routines [11] `ddot()`, `dgemv()` and `dgemm()` resulted in the lowest performance overall. Therefore, their performance curves were omitted from the plots in Figs. 4 and 5. Potentially, the low performance of the BLAS-based variants is due to the time overhead caused by calling the routines. The small stencil size of the standard 2nd order Yee scheme [7] results in a short computation time. Most likely the computation time of an FDTD stencil is much shorter, than the calling time of a BLAS routine. Hence, the total performance of the BLAS-based versions is low.

The matrix casting approach is beneficial in 2D FDTD using the `dotp pre` update alternative with $n_{\text{thr}} \geq 4$. But the performance benefit is negligible compared to the standard explicit update variation with pre-computed coefficients `expl pre`. The performance pattern is slightly different in the 3D case. Again explicit update variants outperform the matrix casting approach, but with a much larger difference. Potential reasons hindering the performance of matrix casting in 3D might be (i) the small size of the spatial domain providing little parallelism potential, (ii) the loop unrolling effect created by a compiler option `-funroll-loops` and (iii) the high amount of L2 cache misses caused by the unsuitable memory layout of neighbouring points used in the stencil calculation.

Fig. 6 shows performance curves for the two fastest alternatives (`expl` and `dotp`) to calculate electromagnetic field update in the EMI scenario (See Fig. 3 for details). In the EMI test case, the matrix casting approach `dotp pre` matches the explicit stencil calculation `expl jit` for $n_{\text{thr}} \leq 12$. For higher numbers of OpenMP threads `expl jit` outperforms `dotp pre`. The overall percentage change between these two

TABLE III
Testing system specification

Parameter	Value
Operating System	Scientific Linux 7.3 (Nitrogen)
Kernel	3.10.0
CPU Specification	Intel Xeon CPU E5-2650 v3, 2.30 GHz
L2 Cache Size	25 MB
CPU = Sockets × Cores	20 = 2 × 10
Address sizes	46 bits physical, 48 bits virtual
Memory, total	64 GB
Interconnect	Mellanox, InfiniBand (EDR 100 Gb/s)

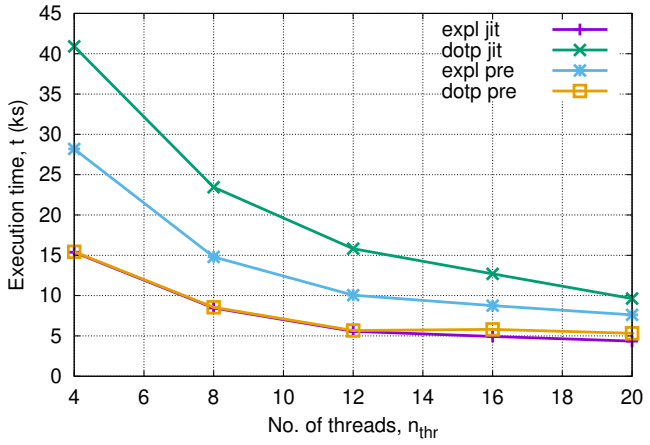

Fig. 6. **No. of OpenMP threads vs execution time in EMI scenario**

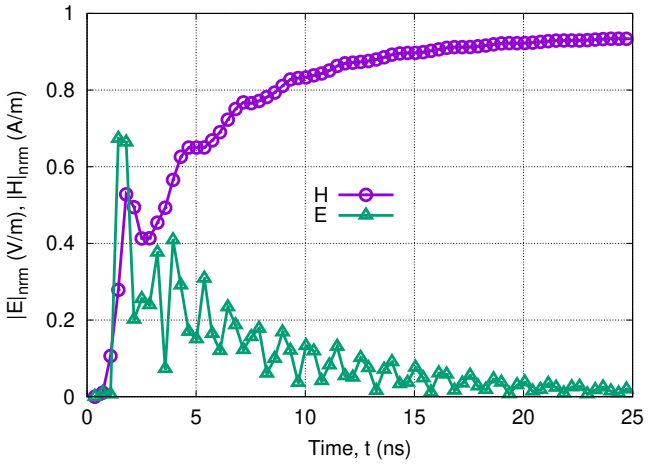

Fig. 7. **Magnitude of electromagnetic fields over time in EMI scenario**

curves mounts to 8.61%. This proves the hypothesis of little parallelism in small spatial domains. The peak performance is reached with the `expl jit` variation for $n_{\text{thr}} = 20$. The application of `expl pre` and `dotp jit` calculation alternatives produces considerable performance drawbacks.

Finally, Fig. 7 shows the normalised magnitude of electromagnetic fields at the observation point. No numerical instabilities are noticeable after 279170 time steps. This proves numerical stability of the matrix casting approach and its equivalence to the standard explicit update experimentally.

V. Conclusion and Future work

The matrix casting approach for the calculation of electromagnetic field updates in 2D and 3D FDTD method has been presented. This technique has been parallelised by means of the OpenMP framework and tested in air-filled scenarios. Matrix casting has also been applied in a practical UXO detection scenario. Current formulation of matrix casting is beneficial in a 2D FDTD method for $n_{\text{thr}} \geq 4$, but the performance benefit is minor. The most high-performing update variation in 2D and 3D is the explicit formulation `expl jit` with FDTD coefficients $\Gamma_{\{1,2\}}, Z_{\{1,2\}}$ calculated "just-in-time". For large spatial domains in 3D, matrix casting reaches the same or lower performance compared to the standard update formulation.

The reasons for lower performance of matrix casting in 3D are being investigated at the moment. This includes the study of L2 cache misses, potential reordering of the neighbouring grid elements in memory and the application of Batched BLAS [12] for the calculation of FDTD stencils in large batches.

Acknowledgements

This research has been enabled by the EPSRC grant for the Scale-free, Energy-aware, Resilient and Transparent Adaptation (SERT) project (reference number EP/M01147X/1). The author would like to thank the ICL at the UTK for access to the Saturn Alembert HPC cluster, Prof. Jack Dongarra for his valuable commentary on the matrix casting approach, Vadims Kožins for the creation of Fig. 3 in isometric projection and Eoin Ward for his English language expertise.

References

[1] K. S. Yee, "Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media," *IEEE Trans. Antennas Propag.*, vol. 14, no. 3, pp. 302–307, May 1966.

[2] M. Abaļenkovs, "Performance optimisation of stencil-based codes for shared memory architectures," in *Proc. 11th European Conference on Antennas and Propagation EuCAP'17*, Paris, France, Mar. 19–24 2017.

[3] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd ed. 3600 University City Science Center, Philadelphia, PA, 19104-2688: Society for Industrial and Applied Mathematics (SIAM), 2002.

[4] V. Codreanu, J. Hertzer, C. Morales, J. Rodriguez, O. W. Saastad, and M. Stachon, *Best Practice Guide Haswell/Broadwell*, PRACE, Jan. 27, 2017, Accessed on Oct 16, 2017. [Online]. Available: http://www.prace-ri.eu/best-practice-guide-haswellbroadwell-january-2017

[5] O. A. Abdel-Rehim, J. L. Davidson, L. A. Marsh, M. D. O'Toole, and A. J. Peyton, "Magnetic polarizability tensor spectroscopy for low metal anti-personnel mine surrogates," *IEEE Sensors J.*, vol. 16, no. 10, pp. 3775–3783, 2016.

[6] O. A. Abdel-Rehim, J. L. Davidson, L. A. Marsh, M. D O'Toole, D. W. Armitage, and A. J. Peyton, "Measurement system for determining the magnetic polarizability tensor of small metal targets," in *Sensors Applications Symposium (SAS)*. IEEE, 2015, pp. 1–5.

[7] A. Taflove and S. C. Hagness, *Computational Electrodynamics: The Finite-Difference Time-Domain Method*, 3rd ed. 685 Canton Street, Norwood, MA 02062: Artech House, 2005.

[8] *OpenMP 4.5 Specifications*, Std., Accessed on Oct 13, 2017. [Online]. Available: http://www.openmp.org/mp-documents/openmp-4.5.pdf

[9] G. Mur, "Absorbing boundary conditions for the finite-difference approximation of the time-domain electromagnetic-field equations," *IEEE Trans. Electromagn. Compat.*, vol. 23, no. 4, pp. 377–382, Nov. 1981.

[10] B. Dekdouk, L. A. Marsh, D. W. Armitage, and A. J. Peyton, "Estimating magnetic polarizability tensor of buried metallic targets for land mine clearance," in *Ultra-Wideband, Short-Pulse Electromagnetics*, F. Sabath and E. Mokole, Eds., vol. 10. New York, NY: Springer, 2014, pp. 425–432.

[11] BLAS (Basic Linear Algebra Subprograms). Accessed on Oct 13, 2017. [Online]. Available: http://www.netlib.org/blas

[12] S. D. Relton, P. Valero-Lara, and M. Zounon, "A comparison of potential interfaces for batched BLAS computations," The University of Manchester, MIMS, UK, Tech. Rep. 2016.42, 2016.