Performance Optimisation of Stencil-Based Codes for Shared Memory Architectures

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

***Abstract*—The work presented in this publication proposes a novel view onto the standard form of Maxwell's equations in the FDTD method. The stencil-based equations are cast into a matrix form. Performance of the matrix casting is further improved by means of the OpenMP paradigm. Numerical results for both the sole FDTD and the FDTD with the Huygens Subgridding (HSG) are analysed.**

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

I. Introduction

In the world of Computational Electromagnetics the Finite-Difference Time-Domain (FDTD) method has deserved a lot of attention due to its straightforward formulation and relative ease of the implementation. Nowadays the requirement for higher precision and larger size of the simulated objects produces a strong motivation for ever increasing performance of the simulation method. Luckily, the last decades have witnessed a rapid development in the area of CPU architectures. This work proposes a novel representation of Maxwell's equations in terms of matrices, as well as subsequent parallelisation of the 1D FDTD method by means of OpenMP [1].

II. Mathematical Background

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

Maxwell's curl equations for linear, isotropic, non-dispersive and lossy materials are expressed as:

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

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

where $\boldsymbol{J}$ and $\boldsymbol{K}$ denote the electric and equivalent magnetic current densities, and σ and σ^{*} stand for electric conductivity and equivalent magnetic loss.

Writing out vector components in Cartesian coordinates, assuming all partial derivatives with respect to y and z equal to zero, Maxwell's equations reduce to x-directed y-polarised TEM-mode:

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

$$\frac{\partial H_z}{\partial t} = -\frac{1}{\mu}\left(\frac{\partial E_y}{\partial x} + K_{src,\,z} + \sigma^{*} H_z\right). \tag{4}$$

Using the central difference and central average approximations for space and time partial derivatives Maxwell's equations can be discretised as:

$$E^{n+\frac{1}{2}}_{y\left(i-\frac{1}{2}\right)} = \Gamma_{1\left(i-\frac{1}{2}\right)}\, E^{n-\frac{1}{2}}_{z\left(i-\frac{1}{2}\right)} - \Gamma_{2\left(i-\frac{1}{2}\right)}\left(\frac{H^{n}_{z(i)} - H^{n}_{z(i-1)}}{\Delta x} + J^{n}_{src,\,y\left(i-\frac{1}{2}\right)}\right), \tag{5}$$

$$H^{n+1}_{z(i)} = Z_{1(i)}\, H^{n}_{z(i)} - Z_{2(i)}\left(\frac{E^{n+\frac{1}{2}}_{y\left(i+\frac{1}{2}\right)} - E^{n+\frac{1}{2}}_{y\left(i-\frac{1}{2}\right)}}{\Delta x} + K^{n+\frac{1}{2}}_{src,\,z(i)}\right), \tag{6}$$

where

$$\Gamma_{1(i-1/2)} = \frac{1-\frac{\sigma_{(i-1/2)}\,\Delta t}{2\,\varepsilon_{(i-1/2)}}}{1+\frac{\sigma_{(i-1/2)}\,\Delta t}{2\,\varepsilon_{(i-1/2)}}}, \quad Z_{1(i)} = \frac{1-\frac{\sigma^{*}_{(i)}\Delta t}{2\,\mu_{(i)}}}{1+\frac{\sigma^{*}_{(i)}\Delta t}{2\,\mu_{(i)}}}, \tag{7}$$

$$\Gamma_{2(i-1/2)} = \frac{\Delta t/\varepsilon_{(i-1/2)}}{1+\frac{\sigma_{(i-1/2)}\,\Delta t}{2\,\varepsilon_{(i-1/2)}}}, \quad Z_{2(i)} = \frac{\Delta t/\mu_{(i)}}{1+\frac{\sigma^{*}_{(i)}\,\Delta t}{2\,\mu_{(i)}}}. \tag{8}$$

These equations form a foundation of the Yee's FDTD method [2] in 1D.

B. Casting Stencil-based Equations into Matrix Form

Shifting electromagnetic field components by $-\frac{1}{2}$ in time and $+\frac{1}{2}$ in space, representing electromagnetic field differences with a symbol Δ and omitting the electromagnetic source currents $J^{n}_{src,\,y}, K^{n}_{src,\,z}$ the 1D FDTD equations become:

$$E^{n}_{y(i)} = \Gamma_{1(i)}E^{n-1}_{y(i)} - \Gamma_{2(i)}\Delta H^{n}_{z(i,i-1)}/\Delta x, \tag{9}$$

$$H^{n+1}_{z(i)} = Z_{1(i)}H^{n}_{z(i)} - Z_{2(i)}\Delta E^{n}_{y(i+1,i)}/\Delta x. \tag{10}$$

Careful examination of these equations leads to a realisation that a 1D FDTD stencil is nothing else as a vector dot product:

$$\boldsymbol{u}\cdot\boldsymbol{v} = \sum_{i=1}^{n} u_i v_i = u_1v_1 + u_2v_2 + \cdots + u_nv_n. \tag{11}$$

Since most modern CPUs are highly optimised to perform multiple vector operations at a time, a certain performance benefit is expected from casting stencil equations into a matrix

form. For example, the 1D FDTD equations cast into the matrix form become:

$$E_y^n = \begin{pmatrix} \Gamma_1 & -\Gamma_2/\Delta x \end{pmatrix} \begin{pmatrix} E_y^{n-1} \\ \Delta H_z^n \end{pmatrix}, \tag{12}$$

$$H_z^{n+1} = \begin{pmatrix} Z_1 & -Z_2/\Delta x \end{pmatrix} \begin{pmatrix} H_z^n \\ \Delta E_y^n \end{pmatrix}. \tag{13}$$

Space indices in Eqs. (12) and (13) were omitted for clarity.

III. Parallelising Matrix Form Equations with OpenMP

Once the stencil-based equations have been cast into a matrix form, further performance benefit may be obtained by delving into the pool of multicore resources of modern CPUs. This work focuses on the performance optimisation by means of the OpenMP paradigm.

Fig. 1 gives an example of five different ways to calculate the electric field E_y across a 1D domain stretching from $i \in [i_{\min}, i_{\max}]$. Four approaches are incorporated into the `$omp parallel do` pragma and the last option is contained within the `$omp parallel workshare` pragma. Only one approach is necessary to calculate the field values E_y in the practical FDTD implementation. Each computation approach is denoted by an abbreviation shown in the comment line: "func" uses a standard stencil function, "dotp" uses Fortran intrinsic function `dot_product`, "ddot" relies on the BLAS Level 1 function `ddot` [3]. Words "jit" and "pre" mark, whether the approach uses *just-in-time* calculation or *pre-computed* values of the coefficients $\Gamma_{\{1,2\}}$ and magnetic field difference ΔH_z.

Parallelisation of the magnetic field calculation routines is identical. It is not shown here for the sake of brevity.

```
!$omp parallel do
do i = imin, imax

    ! expl jit
    Ey(i) = G(enc(i),1)*Ey(i) - G(enc(i),2)*(Hz(i+1)-Hz(i))/dx

    ! func jit
    Ey(i) = calc_E( G(enc(i),1), Ey(i),              &
                    G(enc(i),2), Hz(i+1), Hz(i), dx )

    ! dotp jit
    Ey(i) = dot_product( [G(enc(i),1), -G(enc(i),2)/dx], &
                         [Ey(i), Hz(i+1)-Hz(i)] )

    ! ddot jit
    Ey(i) = ddot(2, [G(enc(i),1), -G(enc(i),2)/dx], 1, &
                    [Ey(i), Hz(i+1)-Hz(i)], 1)
end do
!$omp end parallel do
```

Fig. 1. **Fortran code with OpenMP pragmas**

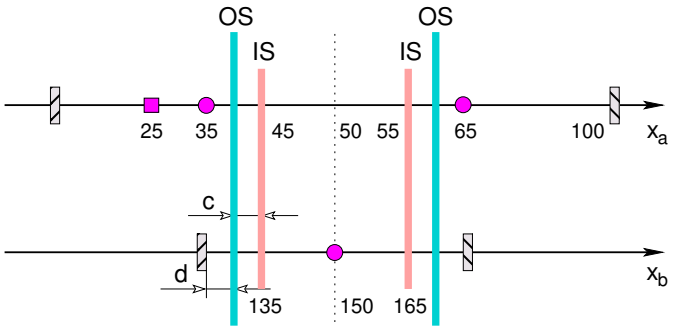

Fig. 2. **HSG scenario setting with** $r = 3$. A unit measurement in the figure equals to 1000 space steps, e.g. 25 is 25000 Δx_a. Excitation source is marked with a filled square and the observation points—with filled circles. Distances c and d are $9\Delta x_b$ and $5\Delta x_b$.

IV. Numerical Results

A. Experiment Setup

To test the performance benefit of equation casting in terms of execution time, CPU load and maximum memory usage two sets of experiments were run: on standard FDTD method and on the FDTD with HSG. All experiments were run in 1D.

Subgridding is an approach of embedding multiple FDTD domains with different spatio-temporal increments $\Delta t, \Delta x$ into each other. This can be extremely useful for large-scale simulations incorporating fine geometrical features such as the human heart response to a defibrillator current in a context of the entire human body [4]. A *subgrid*—domain with small $\Delta t, \Delta x$ will be placed around the fine geometry (the heart), while the *main grid* with large $\Delta t, \Delta x$ will provide the large-scale context (the body). Usually, the interdependence of spatio-temporal increments in a subgridding method is defined by a subgridding ratio. In HSG it is defined as $r = \Delta t_a/\Delta t_b = \Delta x_a/\Delta x_b$. More information on the operation of the HSG can be found in [5].

Fig. 2 presents the experiment setup in detail. Top axis denoted x_a shows the propagation environment setting in the FDTD only experiment or the HSG main grid. The environment stretches from 1 to 100000 space steps Δx_a. Excitation source is placed at $x_a = 25000$. Observation points are located at $x_a = \{35000, 65000\}$. Bottom axis x_b illustrates the setting of the subgrid, where the radio environment occupies 30000 space steps Δx_b, with $x_b \in [135000, 165000]$ and the observation point at $x_b = 150000$. Vertical lines "IS" and "OS" mark locations of the Inner and Outer Huygens Surfaces. They are separated with a minimal distance of $c = 9\Delta x_b$. An additional buffer zone between the OS and the end of the simulation domain is $d = 5\Delta x_b$ long. The simulation domain is terminated with the 1st order Mur's Absorbing Boundary Condition (ABC).

Table I specifies the FDTD simulation settings. Parameter "ni" denotes the size of the simulation domain in a number of space increments, $t_{\max}$ specifies the total number of time step iterations, r is the subgridding ratio, "zone" is the buffer zone

added at the end of the subgrid domain, "ISOS" is the distance between the Inner and Outer Huygens Surfaces, $\Delta x, \Delta t$ are the spatial and temporal increments, N_{CFL} is the Courant–Friedrichs–Lewy (CFL) condition number, χ is the spatial resolution used in the calculation of the FDTD grid space step and $f_{\max}$ is the maximum wave frequency. Indices a, b denote the main and the subgrid respectively.

Gaussian pulse was used as the "soft" excitation source in the experiments:

$$J^n_{src,\,y} = \exp\left[-\left(\{t - 3\,T\}/T\right)^2\right], \text{ with } T = 0.5\,f^{-1}_{\max}. \tag{14}$$

B. Hardware Platform

Numerical experiments were run on a 32 core Intel Xeon machine. The full specification of the system is given in Table II.

The execution time, CPU load and maximum memory usage measurements were taken with the GNU `time` command:

```
time -f "%e %P %Mk"
```

The source code has been compiled with the Intel Fortran compiler `ifort`, version 14.0.0 20130728 and `-O3 -qopenmp` flags.

C. Result Analysis

The plot in Fig. 3 compares five different approaches for parallel computation of electromagnetic fields E_y, H_z in 1D FDTD. The data along the y-axis is depicted with logarithmic scaling. The `$omp parallel workshare` pragma (sky-blue line) is not beneficial for parallelisation of an iterative stencil-based method. The temporal overhead grows with the number of execution threads. Level 1 BLAS function `ddot` (purple line) also did not bring the expected performance boost despite highly optimised Intel MKL implementation. The three most high-performing options are based either on a built-in Fortran intrinsic `dot_product` or apply a simple stencil function. Stencil function and `dot_product` options with just-in-time calculation of all parameters share comparable results, with the stencil function slightly outperforming the `dot_product` alternative. The best computation variant (green line) is based on the `dot_product` function and uses pre-calculated values for the coefficients $\Gamma_{\{1,\,2\}}$.

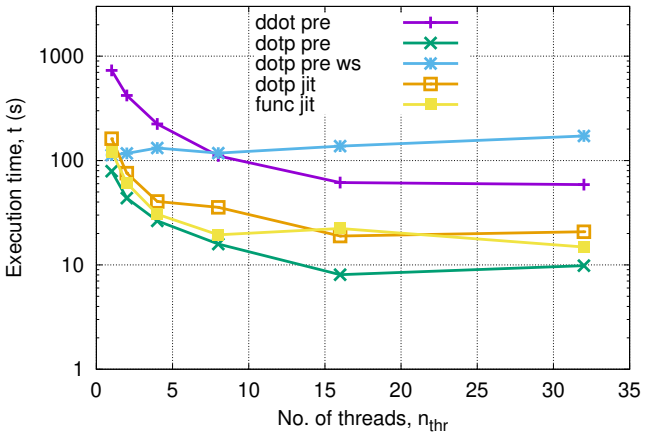

Fig. 3. **No. of OpenMP threads vs execution time**

The higher is the CPU load shown in Fig. 4 the more work each CPU performs and the better is the load-balancing. High CPU load means the processors are occupied and are not idle. The most time-inefficient code (sky-blue line) based on the `$omp parallel workshare` pragma almost fully utilises all of the processors. The most time-efficient variant "dotp pre" (green line) also provides very good load-balancing. On the other hand the "dotp jit" utilises only one half of the CPU power available. This option might be good for sharing the CPU resources with the other tasks or programs.

Second set of experiments dealt with the 1D FDTD with HSG, that is referred to as HSG for convenience. Fig. 5 illustrates the relationship between the subgridding ratio r and the execution time for a given number of OpenMP threads. As expected with increase of the subgridding ratio r size of the subgrid increases and requires more computation time. The 1D HSG code shows good scalability for the given number of threads—execution time decreases with increase of the number of computation threads.

Fig. 6 presents the HSG memory consumption for different subgridding ratios r. Higher subgridding ratio results in a finer grained subgrid with higher number of computation cells. Memory consumption grows linearly with the increase of the parameter r. In this case a higher number of computation

TABLE I
Simulation settings

Parameter	**Main grid** x_a	**Subgrid** x_b
ni	100000	$10000\,r$
$t_{\max}$	100000	$100000\,r$
r		3, 5, 7, 9, 11
zone		$5\,\Delta x_b$
ISOS		$3\,r\,\Delta x_b$
source type	soft	
source location	25000	
Δx	5 cm	$\Delta x_a/r$
Δt	15.51 ns	$\Delta t_a/r$
N_{CFL}	0.93	0.93
χ	20	$20\,r$
$f_{\max}$	2.997 924 58 GHz	

TABLE II
Testing system specification

Parameter	**Value**
Operating System	CentOS Linux release 7.2.1511 (Core)
Kernel	3.10.0
CPU Specification	Intel Xeon CPU E5-2650, 2.00 GHz
L2 Cache Size	20 MB
CPU = Sockets × Cores × Threads	$32 = 2 \times 8 \times 2$
Address sizes	46 bits physical, 48 bits virtual
Memory, total	64 GB

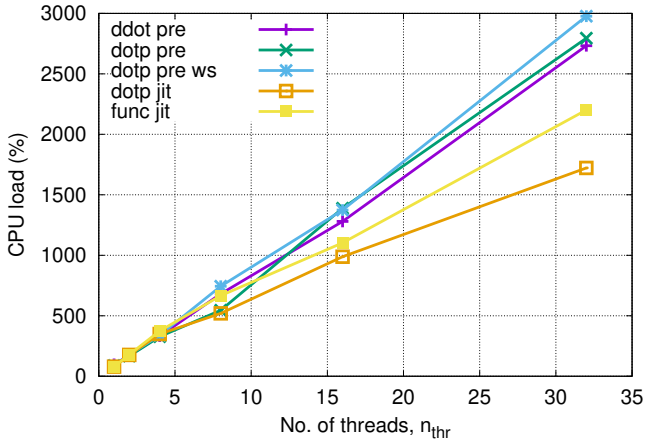

Fig. 4. **No. of OpenMP threads vs CPU load**

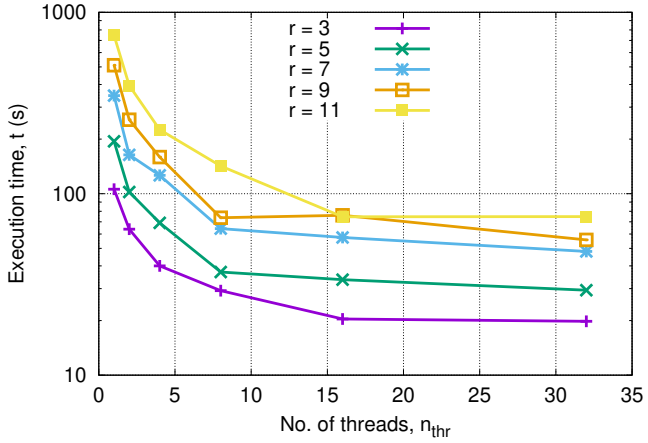

Fig. 5. **No. of OpenMP threads vs execution time in 1D HSG for various** $r \in [3, 11]$

threads creates a slight memory overhead of $\approx 1\,\mathrm{MB}$.

V. Conclusion

Casting of FDTD equations into a matrix form has been proposed. Five different approaches for parallelisation of electromagnetic field calculation in the 1D FDTD method were compared. The casting and OpenMP parallelisation have proven to be beneficial in terms of performance. The best variant "dotp pre" with minimal execution time and high CPU load is based on the `$omp parallel do` pragma, uses Fortran intrinsic function `dot_product` and pre-computed values of the coefficients $\Gamma_{\{1,2\}}, Z_{\{1,2\}}$.

HSG experiments have shown a good scalability of the source code, linear growth of the memory consumption with the growth of the subgridding ratio r and a slight memory overhead due to the OpenMP thread management.

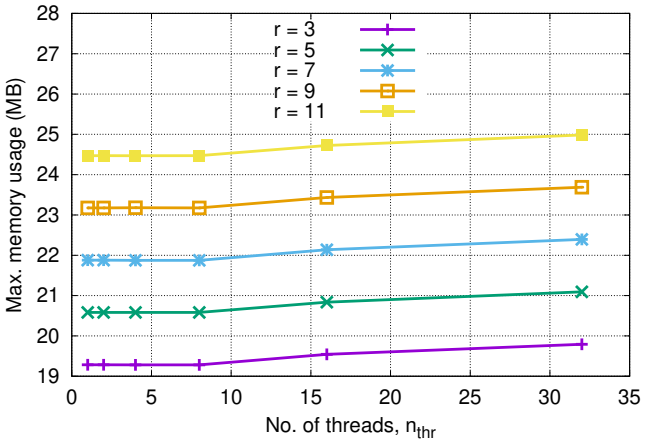

Fig. 6. **No. of OpenMP threads vs max. memory usage in 1D HSG code with** $r \in [3, 11]$

VI. Future work

It is expected that equation casting into matrix form will bring even more performance benefit in 2D and 3D versions of the FDTD method. The work is underway to test this hypothesis. Another expectation is that the casting can be easily expanded to the other stencil-based methods from other scientific domains such as for example Lattice Boltzmann and Crank–Nicolson methods. Other parallelisation platforms such as NVIDIA CUDA and OpenCL can also benefit from the matrix casting. Finally, a multitude of small matrix–matrix operations resulting from equation casting presents a perfect case for testing the Batched BLAS library.

Acknowledgements

The author would like to thank Professor Jack Dongarra for his commentary onto casting of the FDTD equations into matrix form.

This work has been funded by the EPSRC grant for the Scale-free, Energy-aware, Resilient and Transparent Adaptation (SERT) project (reference number EP/M01147X/1).

References

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

[2] 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.

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

[4] M. Abaļenkovs, F. Costen, J.-P. Bérenger, R. Himeno, H. Yokota, and M. Fujii, "Huygens subgridding for 3-D frequency-dependent Finite-Difference Time-Domain method," *IEEE Trans. Antennas Propag.*, vol. 60, no. 9, pp. 4336–4344, Sep. 2012.

[5] J.-P. Bérenger, "The Huygens subgridding for the numerical solution of the Maxwell equations," *Journal of Computational Physics*, vol. 230, pp. 5635–5659, 2011.