

Matrix-Free FEM on Tensor Product Bases: Insights into Cache Size, Vectorization and GPU Efficiency

Balint M. Nyakas¹, Dennis Giannacopoulos¹ and Zeljko Zilic¹

¹McGill University, Montreal, H3A 0G4 Canada, balint.nyakas@mail.mcgill.ca

Matrix-free finite element methods (FEM) combined with tensor product basis functions have garnered significant attention as effective strategies to address the memory bandwidth limitations inherent in classical FEM formulations. By avoiding the explicit storage of global matrices and leveraging the efficiency of tensor product structures, these approaches substantially reduce memory requirements and computational overhead. This capability is particularly advantageous in high-dimensional problems and large-scale simulations, including computational electromagnetics, where memory constraints often pose critical challenges. In this study, we aim to explore the effect of modern hardware acceleration in this framework. By benchmarking performance and analyzing the scalability of these methods on contemporary high-performance computing platforms, we seek to uncover insights into their potential for solving increasingly complex problems in computational electromagnetics and beyond.

Index Terms—Finite element analysis, Cache memory, Multicore processing, Parallel architectures

I. INTRODUCTION

THE finite element method (FEM) is a cornerstone computational technique extensively utilized across various engineering disciplines, including computational electromagnetics (CEM). Its versatility and robustness make it indispensable for solving complex problems involving irregular geometries, heterogeneous materials, and intricate boundary conditions, all of which are common challenges in CEM. FEM enables the accurate modelling of electromagnetic phenomena, from wave propagation and scattering to resonance analysis in intricate structures such as in [2] and [3]. The method's adaptability to high-performance computing frameworks further enhances its relevance, enabling efficient solutions to large-scale problems that are increasingly encountered in modern engineering applications such as in [1].

Matrix-free methods in the finite element method have emerged as powerful tools in addressing the computational challenges associated with solving large-scale problems [4]–[5]. These methods bypass the explicit assembly and storage of global stiffness or mass matrices, focusing instead on efficient computation of matrix-vector products, which is around 60–95% of time in either implicit or explicit time stepping schemes. This approach is particularly relevant in high-performance computing, where memory bandwidth and cache efficiency often dominate execution time [6]. By avoiding the storage of large sparse matrices, matrix-free FEM methods significantly reduce memory requirements, allowing simulations of much larger domains on modern hardware [7].

The importance of matrix-free techniques is amplified in applications where high-order elements or tensor-product basis functions are employed, as these formulations benefit greatly from optimized computation [8]. Moreover, such methods align seamlessly with GPU architectures and other parallel computing platforms, which excel in handling vectorized and localized computations. This alignment facilitates the efficient use of modern computational resources, enabling faster and more scalable solutions [9].

In this work, we aim to quantify the benefit of larger cache sizes, vectorization and GPU acceleration within the matrix-free FEM framework with tensor product basis functions in a 2D explicit time-stepping Maxwell equation solver. Our analysis is accompanied by Intel Vtune's Roofline model for CPUs and Nvidia's Nsight. We compare Intel Xeon Scalable processor with 33MB of L3 cache and AMD EPYC 7001/7002 processor with 256MB L3 cache in dual setting. Various GPU backends, including CUDA, Kokkos, and SYCL, will be evaluated and benchmarked for performance comparison.

II. METHODOLOGY

We solve the time-dependent equation:

$$\epsilon \frac{\partial^2 E}{\partial t^2} + \nabla \cdot \left(\frac{1}{\mu} \nabla E \right) = f(x, y, t), \quad (1)$$

where ϵ is the permittivity, $E(x, y, t)$ is the electric field, μ is the permeability. $E(x, y, t)$ is approximated using tensor-product bilinear basis functions on each quadrilateral element as presented,

$$E(x, y, t) \approx \sum_{i=1}^4 E_i(t) \phi_i(x, y), \quad (2)$$

where $\phi_i(x, y)$ are the tensor-product bilinear basis functions defined as:

$$\phi_i(x, y) = \phi_i(x) \phi_i(y). \quad (3)$$

In order to simplify the computation of the stiffness matrix contributions in a matrix-free implementation, we calculate the gradients

$$\frac{\partial \phi_i(x, y)}{\partial x} = \frac{\partial \phi_i(x)}{\partial x} \phi_i(y) \quad (4)$$

and

$$\frac{\partial \phi_i(x, y)}{\partial y} = \frac{\partial \phi_i(y)}{\partial y} \phi_i(x). \quad (5)$$

We multiply the governing equation by a test function, $\phi(x, y)$ and integrating over the domain:

$$\int_{\Omega} \epsilon \frac{\partial^2 E}{\partial t^2} \phi_j dA + \int_{\Omega} \mu^{-1} \nabla E \cdot \nabla \phi_j dA = \int_{\Omega} f \phi_j dA. \quad (6)$$

Substituting the basis functions:

$$\sum_{i=1}^4 \left(\epsilon \int_{\Omega} \frac{\partial^2 E_i}{\partial t^2} \phi_i \phi_j dA + \int_{\Omega} \mu^{-1} \nabla \phi_i \cdot \nabla \phi_j dA \right) = \int_{\Omega} f \phi_j dA. \quad (7)$$

In the matrix-free implementation, the computation of the stiffness matrix (K) and the mass matrix (M) is done directly. The stiffness matrix contribution for a single element is:

$$K_{ij}^{(e)} = \int_e \mu^{-1} (\nabla \phi_i \cdot \nabla \phi_j) dA. \quad (8)$$

Using the tensor-product basis function gradients:

$$K_{ij}^{(e)} = \mu^{-1} \left(\int_e \frac{\partial \phi_i}{\partial x} \frac{\partial \phi_j}{\partial x} dA + \int_e \frac{\partial \phi_i}{\partial y} \frac{\partial \phi_j}{\partial y} dA \right). \quad (9)$$

In a matrix-free implementation, this is computed for each element and applied to the nodal vector E^n . The mass matrix contribution for a single element is:

$$M_{ij}^{(e)} = \int_e \epsilon \phi_i \phi_j dA. \quad (10)$$

Since the basis functions are tensor products, these integrals are separable. Now, we focus on the time-stepping and substituting the spatial discretization into the time-stepping scheme:

$$M \frac{E^{n+1} - 2E^n + E^{n-1}}{\Delta t^2} + KE^n = F^n. \quad (11)$$

This can be rearranged to

$$E^{n+1} = 2E^n - E^{n-1} + \Delta t^2 M^{-1} (F^n - KE^n), \quad (12)$$

where M^{-1} is applied matrix-free and KE^n is also computed matrix-free using the stiffness contributions from each element.

The algorithm is structured as follows: Initially, element-level contributions are computed to address the local discretization of the problem. Subsequently, the field is updated through a time-stepping scheme to progress the solution temporally. This study primarily concentrates on the first step—computing element-level contributions—with a focus on analyzing memory usage, computational intensity, and the impact of vectorization, GPU acceleration and cache size.

III. PRELIMINARY RESULTS

To initiate our investigation, we have developed a matrix-free finite element method implementation utilizing tensor-product basis functions with explicit time-stepping in MATLAB. In the final implementation, the method will be transitioned to C++ for enhanced performance and scalability. We test our code with a time-dependent Gaussian pulse in electromagnetic field.

$$\mathbf{E}(\mathbf{r}, t = 0) = \mathbf{E}_0 e^{-\frac{|\mathbf{r}-\mathbf{r}_0|^2}{2\sigma^2}} \cos(k|\mathbf{r}-\mathbf{r}_0| - \omega t), \quad (13)$$

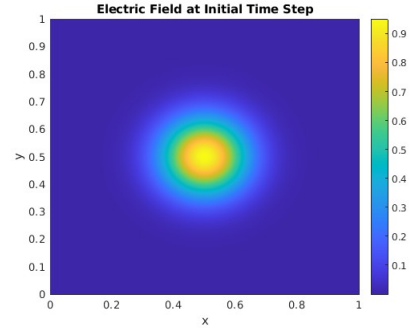


Fig. 1: Electric field at initial time step.

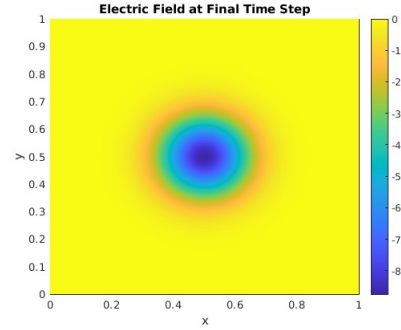


Fig. 2: Electric field at final time step.

where $\mathbf{r} = (x, y)$ are the spatial coordinates, E_0 is the amplitude vector of the electric field, σ is the width of the Gaussian pulse and k is the wave number.

The figures illustrate the propagation of the Gaussian pulse within the domain, validating our time-stepping scheme. Our objective is to examine the underlying architectural mechanisms across various microarchitectures within this framework.

REFERENCES

- [1] Chari, M. Finite element analysis of electrical machinery and devices. *IEEE Transactions On Magnetics*. **16**, 1014-1019 (1980)
- [2] Abraham, D. & Giannacopoulos, D. A parallel finite-element time-domain method for nonlinear dispersive media. *IEEE Transactions On Magnetics*. **56**, 1-4 (2020)
- [3] Abraham, D. & Giannacopoulos, D. A convolution-free finite-element time-domain method for the nonlinear dispersive vector wave equation. *IEEE Transactions On Magnetics*. **55**, 1-4 (2019)
- [4] Ljungkvist, K. Matrix-free finite-element computations on graphics processors with adaptively refined unstructured meshes. *Proceedings Of The 25th High Performance Computing Symposium*. pp. 1-12 (2017)
- [5] Moxey, D., Amici, R. & Kirby, M. Efficient matrix-free high-order finite element evaluation for simplicial elements. *SIAM Journal On Scientific Computing*. **42**, C97-C123 (2020)
- [6] Schussnig, R., Fehn, N., Munch, P. & Kronbichler, M. Matrix-free higher-order finite element methods for hyperelasticity. *Computer Methods In Applied Mechanics And Engineering*. **435** pp. 117600 (2025)
- [7] Brown, J. Efficient nonlinear solvers for nodal high-order finite elements in 3D. *Journal Of Scientific Computing*. **45** pp. 48-63 (2010)
- [8] Muthing, S., Piatkowski, M. & Bastian, P. High-performance implementation of matrix-free high-order discontinuous Galerkin methods. *ArXiv Preprint ArXiv:1711.10885*. (2017)
- [9] Ljungkvist, K. Matrix-free finite-element operator application on graphics processing units. *Euro-Par 2014: Parallel Processing Workshops: Euro-Par 2014 International Workshops, Porto, Portugal, August 25-26, 2014, Revised Selected Papers, Part II* 20. pp. 450-461 (2014)