Improving Edge Finite Element Assembly For Geophysical Electromagnetic Modelling On Shared-memory Architectures

Octavio Castillo-Reyes, Josep de la Puente and José María Cela
Computer Applications in Science & Engineering
Barcelona Supercomputing Center
Barcelona, Spain
octavio.castillo@bsc.es

***Abstract*—This work presents a set of node-level optimizations to perform the assembly of edge finite element matrices that arise in 3D geophysical electromagnetic modelling on shared-memory architectures. Firstly, we describe the traditional and sequential assembly approach. Secondly, we depict our vectorized and shared-memory strategy which does not require any low level instructions because it is based on an interpreted programming language, namely, Python. As a result, we obtained a simple parallel-vectorized algorithm whose runtime performance is considerably better than sequential version. The set of optimizations have been included to the work-flow of the Parallel Edge-based Tool for Geophysical Electromagnetic Modelling (PETGEM) which is developed as open-source at the Barcelona Supercomputing Center. Finally, we present numerical results for a set of tests in order to illustrate the performance of our strategy.**

Keywords*—*Edge finite element, exploration geophysics, electromagnetic modelling, shared memory, python.

I. Introduction

THE electromagnetic methods (EM) are an established tool in geophysics, finding application in many areas such as hydrocarbon and mineral exploration, reservoir monitoring, CO2 storage characterization, geothermal reservoir imaging and many others. In particular, the marine Controlled-Source Electromagnetic Method (CSEM) has become an important technique for reducing ambiguities in data interpretation in hydrocarbon exploration. In the traditional configuration, the sub-seafloor structure is explored by emitting low-frequency signals from a high-powered electric dipole source towed close to the seafloor. By studying the received signal, the subsurface structures could be detected at scales of a few tens of meters to depths of several kilometers.

On the other hand, in the Finite Element Method for solving electromagnetic field problems, the use of Nédélec elements (edge elements or edge finite element method) has become very popular. In fact, Nédélec elements are often said to be a cure to many difficulties that are encountered (particularly eliminating spurious solutions) and are claimed to yield accurate results.

As summary, main properties of Nédélec elements are the following: degrees of freedom (DOFs) are edge-associated; at interfaces, their tangential component is continuous, while the normal one is, in general, discontinuous; vector basis functions are divergence free and they do not yield conflicting conditions at points where the interface between two different media is not locally flat [8]. As regards the computational burden, only six unknowns are required for each element (Nédélec tetrahedral elements of lower order). It is worth nothing that the linear vectorial Lagrange elements or any other consistently linear 3D-vector functions over a tetrahedral, carry twelve unknowns, three at each of its four nodes [11].

978-1-5090-1496-5/16/$31.00 © 2016 IEEE

Despite the popularity of the EFEM, there are few implementations of it. Furthermore, the 3D modelling of geophysical EM problems can easily overwhelm single core computing resources [3]. To alleviate these issues we present a set of node-level optimizations to perform the assembly of edge finite element matrices that arise in 3D geophysical electromagnetic modelling on shared-memory architectures.

The set of optimizations have been included to the work-flow of the Parallel Edge-based Tool for Geophysical Electromagnetic Modelling (PETGEM) which is developed as open-source at Computer Applications in Science & Engineering of Barcelona Supercomputing Center.

We structured the paper as follows: in section two we describe the physical problem to be solved. In section three we present the edge finite element assembly procedure. We explain the parallel-vectorized algorithm and his implementation in section four. In section five we present the numerical and scalability results for a set of tests. We expose the conclusions and future work in last section.

II. Electromagnetic modelling in geophysics

The electromagnetic methods (EM) are an established tool in geophysics, finding application in many areas such as hydrocarbon and mineral exploration, reservoir monitoring, CO2 storage characterization, geothermal reservoir imaging and many others. In particular, the marine Controlled-Source Electromagnetic Method (CSEM), also referred as seabed logging [7], has become an important technique for reducing ambiguities in data interpretation in hydrocarbon exploration.

In marine CSEM a deep-towed electric dipole transmitter is used to produce a low frequency electromagnetic signal which

interacts with the electrically conductive Earth and induces eddy currents that become sources of a new electromagnetic signal. The two fields add up to a resultant field, which is measured by remote receivers placed on the seabed. Figure 1 depicts the marine CSEM.

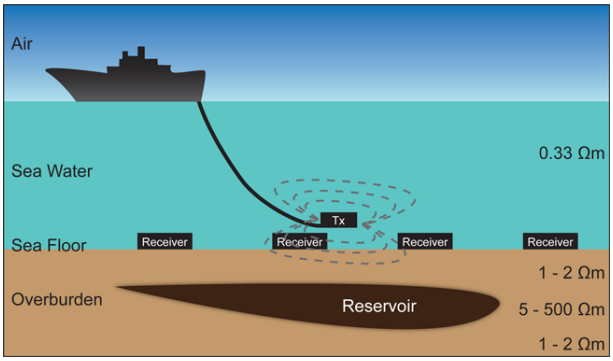

Fig. 1. Marine CSEM [8]

The 3D CSEM modelling is typically solved in frequency domain at low frequency ($\sim$ 1Hz), which involves the numerical solution of Maxwell's equations in stationary regimes in an unbound domain Γ [4].

Following the works by [3] and [4], the edge finite element equation for CSEM modelling can be expressed as follows:

$$[K^e_{jk} + i\omega\tilde{\sigma}_e M^e_{jk}] \cdot \{E_{sk}\} = -i\omega\mu\Delta\tilde{\sigma}_e R^e_k \tag{1}$$

where K^e and M^e are the elemental stiffness and mass matrices R^e_k is the right hand side which requires numerical integration. System of linear equations (1) can be rewritten in a compact form $Ax = b$. Former system is large, sparse, complex and symmetric and his assembly and solution is computationally costly.

Elemental matrices K^e and M^e in system (1) can be defined as:

$$K^e_{ij} = \iiint_{V^e} (\nabla \times N^e_i \cdot S^e_i) \cdot (\nabla \times N^e_j \cdot S^e_j)\, dV \tag{2}$$

$$M^e_{ij} = \iiint_{V^e} (N^e_i \cdot S^e_i) \cdot (N^e_j \cdot S^e_j)\, dV \tag{3}$$

where $\mathbf{N}^e_i$ are the vector basis functions associated to each edge i and $\mathbf{S}^e_i$ are coefficients equal to 1 or -1 depending on the relative local orientation of the *i-th* edge in the element e with respect to the global orientation of the *i-th* edge in the mesh.

Considering the node and edge indexing in figure 2, the vector basis functions $\mathbf{N}^e_i$ can be expressed as follows:

$$\mathbf{N}^e_i = (\lambda^e_{i1}\nabla\lambda^e_{i2} - \lambda^e_{i2}\nabla\lambda^e_{i1})\ell^e_i \tag{4}$$

where subscripts $i1$ and $i2$ are the first and second nodes linked to the *i-th* edge, λ^e_i are the linear nodal basis functions, and ℓ^e_i is the length of the *i-th* edge of the element e. Vector basis functions (4) can be calculated analytically or numerically [8].

Edge i	Node i_1	Node i_2
1	1	2
2	1	3
3	1	4
4	2	3
5	4	2
6	3	4

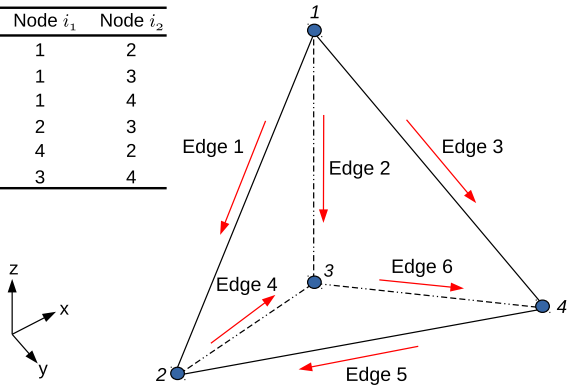

Fig. 2. Tetrahedral Nédélec edge element with node/edge indexing.

III. Edge Finite Element Assembly Procedure

Solve the system (1) using the edge finite element method in frequency domain typically requires the following step:

- Local assembly. For each element e in the computational domain, a NxN local matrix W^e ($W^e = K^e + M^e$), and a N-length vector, b^e, are computed, where N is the number of edges per element. In our case, computing of these local matrices and vectors involves the evaluation of integrals over the element using Gaussian quadrature rules.
- Global assembly. The local matrices, W^e, and vectors, b^e, are used to build a global matrix A and global vector, b, respectively. In this process the contributions of the elements are integrated. The sparsity pattern of the global matrix is function of the mesh connectivity. In this stage, we used the Compressed Sparse Row (CSR) format in order to reduce the storage requirement of the global system and to eliminate redundant computations.
- System solution. Using iterative methods or direct methods, the system of equations $Ax = b$ is solved for x.

Algorithm 1 describe the assembly of matrix A from each associated element matrix W^e whose dimensions are $6x6$ because we used edge elements of lowest order (six edges per element).

On the other hand, the assembly of vector b is depicted in algorithm 2.

As main disadvantages of previous approaches arise the repetition of some operations, indirect addressing and the loop costs when the number of elements is grows.

IV. Parallel Node-Level Optimizations

The global assembly process is a performance bottleneck in real scenarios. Recent investigations of alternative implementations of global assembly techniques show that depending on problem parameters, the point at which it is profitable to switch algorithms for global assembly varies depending on the problem [10].

In our case, the assembly stage for CSEM modelling is a portion of the computation that forms a significant bottleneck.

Algorithm 1: Global assembly of matrix A

```
Data: A=sparse(numEdges, numEdges)
Result: Global matrix A
begin
    for iElem = 1 : numElements do
        edgesEle = get local edge indexes of iElem
        K^e = Compute stiffness matrix of iElem
        M^e = Compute mass matrix of iElem
        W^e = K^e + ω σ̃_e M^e
        for ic = 1 : orderEle do
            for jc = 1 : orderEle do
                A(edgesEle(ic), jc) =
                  A(edgesEle(ic), jc) + W^e(ic, jc)
            end
        end
    end
end
```

Algorithm 2: Global assembly of vector b

```
Data: b=sparse(1, numEdges)
Result: Global vector b
begin
    for iElem = 1 : numElements do
        edgesEle = get local edge indexes of iElem
        gaussPoints = compute Gauss points
        basis = compute basis functions
        Ep = compute electric field
        b^e = −iωμΔσ̃_e ∗ Ep ∗ basis
        for ic = 1 : orderEle do
            b(edgesEle(ic)) =
              b(edgesEle(ic)) + b^e(edgesEle(ic))
        end
    end
end
```

As consequence, we decided to focus in the performance improvement of it through the use of high-level strategies for parallel shared-memory sparse matrix-vector operations. The set of node-level optimizations have been included to the work-flow of the Parallel Edge-based Tool for Geophysical Electromagnetic Modelling (PETGEM) which is a Python code for the scalable solution of geophysics electromagnetic problems on tetrahedral meshes, as these are the easiest to scale-up to very large domains or arbitrary shape. PETGEM is developed as open-source at Computer Applications in Science & Engineering of Barcelona Supercomputing Center. Detailed information about PETGEM, such as software stack, main functionalities and capabilities, are described in [4], [5].

A. Vectorization strategies

Vectorization is a technique that allows speed up algorithms with minimum effort by utilising computational architectures along with some highly optimised vector routines. Since PETGEM is a Python-based code, is possible apply operations at once to an entire set of values (arrays). Therefore, parts of the assembly process eligible for vectorizing are the following: computation of stiffness and mass matrices, b^e computation, Gauss points computation, basis functions computation and electric field computation. Here, our main target was avoid the use of loops in the numerical formulation by [3], [4] and [5]. As consequence, in the new version of the code all operations can be developed in a matrix-vector manner, which significantly reduces the memory access (indirect addressing) and therefore the computation time.

On the other hand, the global assembly process had been improved by the inclusion of sparse matrices functions by scipy [9]. Among of that, we choose the Compressed Sparse Row (CSR) format because is has an efficient row slicing and a fast matrix vector products.

B. Shared-memory strategies

In order to meet the high computational cost of vectorized functions, we used shared-memory strategies for matrix-vector operations in Python. In Python language, the multiprocessing package supports spawning processes using an API similar to a threading OpenMp [14] approach. The multiprocessing package offers both local and remote concurrency, effectively side-stepping the Global Interpreter Lock by using subprocesses instead of threads. Due to this, the multiprocessing module allows the programmer to fully leverage multiple processors on a given machine [15].

The previous technique exploits the use of multi-core architectures, specially when the problem to be solved is computationally intensive as in our case. In our approach, the assembly process is seen as a function across multiple input values (data parallelism), namely, functions are defined as modules so that child processes can successfully import and execute that module.

C. Parallel-vectorized algorithms

Taking into account previous strategies, new version of algorithms for global assembly of matrix A and vector b are depicted in algorithm 3 and algorithm 4, respectively.

In algorithm 3 computation of K^e and M^e is done in a vectorized manner. On the other hand, in algorithm 4 computation of basis functions and primary field has been vectorized because these are the most expensive code region.

V. Numerical and scalability results

In order to verify the performance of the node-level optimizations we used the canonical model by [6] which is deeply described in [4]. Furthermore, we have prepared a set of hierarchically refined meshes in order to investigate the scalability. In 3D computational domains, this technique results in 8 times more tetrahedral elements. Table I depicts main data of our meshes. For the numerical integration we used 8 gauss points per tetrahedral element.

The experiments were performed on the Marenostrum supercomputer with two 8 cores Intel Xeon processors E52670 at 2.6 GHz per node.

Algorithm 3: Global assembly of matrix A in parallel

Data: Computation of constants
Result: Global matrix A (CSR format)
begin
  Start parallel pool(number of processors)
  for *iElem = 1 : numElements* **do** *Parallel loop*
    $edgesEle$ = get local edge indexes of iElem
    K^e = Compute stiffness matrix of iElem
    M^e = Compute mass matrix of iElem
    $W^e = K^e + \omega\tilde{\sigma}_e M^e$
    W^e = convert matrix W^e to array
    return W^e to master as W
  end
  Close parallel pool()
  I = row indexes computation
  J = column indexes computation
  A = CSR-Building(W, I, J)
end

Algorithm 4: Global assembly of vector b in parallel

Data: Computation of constants
Result: Global vector b
begin
  Start parallel pool(number of processors)
  for *iElem = 1 : numElements* **do** *Parallel loop*
    $edgesEle$ = get local edge indexes of iElem
    $gaussPoints$ = compute Gauss points
    $basis$ = compute basis functions
    Ep = compute electric field
    $b^e = \Delta\tilde{\sigma}_e * Ep * basis$
    return b^e to master as V
  end
  Close parallel pool()
  I = row indexes computation
  b = CSR-Building(V, I, 1)
  $b = -i\omega\mu * b$
end

TABLE I. MESHES INFORMATION

Test	$\#\,nodes$	$\#\,elements$	$\#\,edges$	$\#\,dofs$
Mesh 1	729	3,072	4,184	3,032
Mesh 2	4,913	24,576	31,024	26,416
Mesh 3	35,937	196,608	238,688	220,256
Mesh 4	274,625	1,572,864	1,872,064	1,798,336
Mesh 5	2,146,689	12,582,912	1,482,704	14,532,992

TABLE II. SUMMARY OF RUNTIMES (MINUTES)

Test	Processors				
	Sequential	2	4	8	16
1	0.0764	0.0478	0.0300	0.0219	0.0181
2	0.5368	0.2899	0.1608	0.1003	0.0790
3	4.2441	2.2959	1.2247	0.7239	0.5606
4	36.2597	18.3408	9.9575	4.7760	2.6941
5	295.1132	142.1810	77.8087	39.8862	19.4709

TABLE III. SUMMARY OF STRONG SCALING TEST (%)

Test	Processors				
	Sequential	2	4	8	16
1	100	79.91	63.66	43.60	26.38
2	100	92.58	83.47	66.89	42.45
3	100	92.42	86.63	73.28	47.31
4	100	98.84	91.03	94.90	84.11
5	100	99.83	94.82	92.48	94.72

Table II summarizes the run-time for each test. The sequential time and parallel time (both expressed in minutes) for each test are included in function of the number of processors. According to results presented in [5], new version of PETGEM is significantly more efficient than previous one .

Table III show the parallel scalability for the new version of PETGEM. Here, for each mesh we used a strong scaling where the problem size stays fixed but the number of processing elements were increased (cpu-bound test). Following the work by [3] the parallel scalability or parallel efficiency is given by:

$$\chi = \frac{S}{n \cdot Sn} \cdot 100 \tag{5}$$

where S is the amount of time to complete a work unit with 1 processing unit, n is the number of processing units and Sn is the amount of time to complete the same unit of work with n processing units. Results in table III are congruent because the chache miss and non-uniform memory access (NUMA).

Howewer, because our approach is based on shared-memory architectures the communication overhead has no an important negative impact in the parallel scalability for a considerable work-load as show figure 3 for mesh 4 and figure 4 for mesh 5. As consequence, we obtained a quasi linear speed-up, which is plotted in figure 5 and 6 for mesh 4 and mesh 5 respectively.

VI. CONCLUSIONS AND FUTURE WORK

Nowadays, the electromagnetic methods are a fundamental tool in geophysics area, finding an ample range of application such as hydrocarbon and mineral exploration, reservoir monitoring, CO2 storage characterization, geothermal reservoir imaging and many others. In particular, the marine CSEM has become an important technique for reducing ambiguities in data interpretation in hydrocarbon exploration.

Considering the societal value of exploration geophysics, we presented a set of node-level optimizations to perform the assembly of edge finite element matrices that arise in 3D geophysical electromagnetic modelling on shared-memory

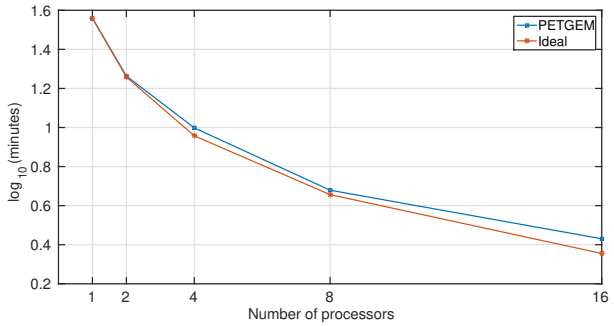

Fig. 3. Parallel scaling for mesh 4

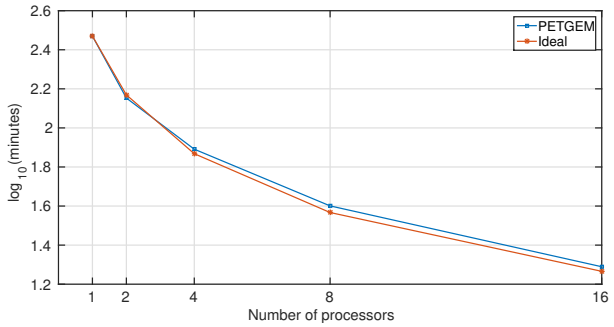

Fig. 4. Parallel scaling for mesh 5

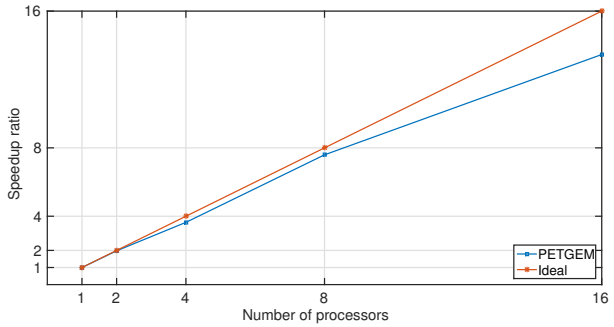

Fig. 5. Parallel scaling for mesh 4

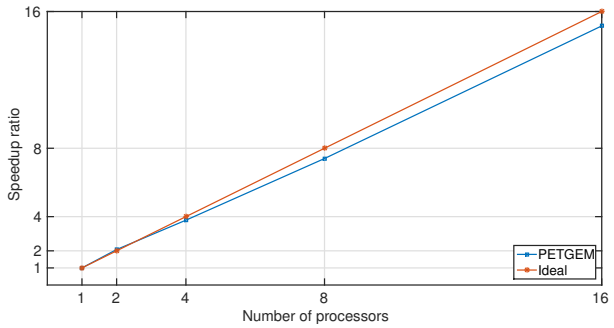

Fig. 6. Parallel scaling for mesh 5

platforms. The set of optimizations have been included to the work-flow of the Parallel Edge-based Tool for Geophysical Electromagnetic Modelling (PETGEM) which is developed as open-source at Computer Applications in Science & Engineering Department of the Barcelona Supercomputing Center.

Using a canonical model of an off-shore hydrocarbon reservoir, we have evaluated the efficiency of PETGEM through scalability tests (strong scaling) for a set of hierarchically refined tetrahedral meshes. Results shows a good parallel efficiency of our code.

Future work will be aimed in two lines. Firstly, at the implementation of an hybrid parallel model (distributed memory-shared memory) to improve the computation time of the whole solution. Secondly, including anisotropy and seafloor bathymetry to the scheme as well as comparing the behaviour of the code to other modelling approaches for CSEM.

ACKNOWLEDGEMENTS

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No. 644202.

The research leading to these results has received funding from the European Union's Horizon 2020 Programme (2014-2020) and from Brazilian Ministry of Science, Technology and Innovation through Rede Nacional de Pesquisa (RNP) under the HPC4E Project (www.hpc4e.eu), grant agreement No. 689772.

Authors gratefully acknowledge the support from the Mexican National Council for Science and Technology (CONACYT).

All numerical tests were performed on the MareNostrum supercomputer of the Barcelona Supercomputing Center - Centro Nacional de Supercomputación (www.bsc.es).

REFERENCES

[1] **Acceleware Ltd.** (**2015**). http://www.acceleware.com/.

[2] **Burnett, D. S.** (**1987**). Finite element analysis: from concepts to applications. Prentice Hall.

[3] **Castillo-Reyes, O., de la Puente, J., Puzyrev, V., Cela, J.M.** (**2015**). Edge-based electric field formulation in 3D CSEM simulations: a parallel approach. IEEE - Proceedings of the 6th International Conference and Workshop on Computing and Communication.

[4] **Castillo-Reyes, O., de la Puente, J., Puzyrev, V., Cela, J.M.** (**2016**). Edge-based parallel framework for the simulation of 3D CSEM surveys. ICE Barcelona - AAPG/SEG International Conference & Exhibition. Society of Exploration Geophysicists.

[5] **Castillo-Reyes, O., de la Puente, Modesto, D., J., Puzyrev, V., Cela, J.M.(2016)**. Parallel tool for numerical approximation of 3D electromagnetic surveys in geophysics. Computacion y Sistemas, Thematic issue: Topic Trends in Computing Research in Catalonia. Vol. 20, No. 1, pp. 29-39. National Polytechnic Institute. Mexico, D.F.

[6] **Constable, S., Weiss, C.J.** (**2006**). Mapping thin resistors and hydrocarbons with marine EM methods: Insights from 1D modeling. Geophysics. G43–G51. Society of Exploration Geophysicists

[7] **Eidesmo, T., Ellingsrud, S., MacGregor, L., Constable, S., Sinha, M., Johansen, S., Kong, F., and Westerdahl, H.** (**2002**). Sea bed logging (SBL), a new method for remote and direct identification of hydrocarbon filled layers in deepwater areas In *First break*. Society of Exploration Geophysicists.

[8] **Jin, J.** (**2002**). The Finite Element Method in Electromagnetics. Wiley, New York, Second edition.

[9] **Jones, E., Oliphant, T., Peterson, P., et. al.** (**2001–**). SciPy: Open source scientific tools for Python. http://www.scipy.org/.

[10] **Markall, GR., Slemmer, A., Ham, DA., Kelly, PHJ., Cantwell, CD., Sherwin, SJ.** (**2011**). Finite element assembly strategies on multi-and many-core architectures. International Journal for Numerical Methods in Fluids.

[11] **Monk, P.** (**2003**). Finite element methods for Maxwell's equations. Clarendon Press Oxford.

[12] **Newman, G. A., Alumbaugh, D. L.** (**2002**) Three-dimensional induction logging problems, Part 2: A finite-difference solution. In Geophysics, 484–491, 67, Society of Exploration Geophysicists.

[13] **Newman, G. A.** (**2014**) A review of high-performance computational strategies for modeling and imaging of electromagnetic induction data. In Surveys in Geophysics, 85–100, 35, Springer.

[14] **OpenMP Architecture Review Board** (**2015**). OpenMP application program interface. Version 4.0.

[15] **Python Software Foundation** (**2016**). Python Language Reference, version 3.4.3. Available at http://www.python.org