Efficient Finite Element Assembly of High Order Whitney Forms

N. Marsic, C. Geuzaine**

**University of Liège, Department of Electrical Engineering and Computer Science,*
Montefiore Institute B28, B-4000 Liège, Belgium
E-mail: nicolas.marsic@ulg.ac.be

Keywords: Whitney elements, high order, assembly, finite element method, high performance computing.

Abstract

This paper presents an efficient method for the finite element assembly of high order Whitney elements. We start by reviewing the classical assembly technique and by highlighting the most time consuming part. Then, we show how this classical approach can be reformulated into a computationally efficient matrix – matrix product. We also address the problem of the basis orientation by considering more than one reference space. We conclude by presenting numerical results on a wave guide problem.

1 Introduction

There is a growing consensus that state of the art finite element technology requires, and will continue to require, too extensive computational resources to provide the necessary resolution for complex high-frequency electromagnetic compatibility simulations, even at the rate of computational power increase.

The requirement for precise resolution naturally leads us to consider methods with a higher order of grid convergence than the classical second order provided by most industrial grade codes. This indicates that higher order discretization methods will replace at some point the finite element solvers of today, at least for part of their applications.

2 Classic finite element assembly

Let us star by considering the time harmonic propagation of an electrical wave $\boldsymbol{e}(x,y,z)$ into wave guide $\Omega(x,y,z)$. The solution is interpolated using a curl-conforming [1] basis $\mathbb{B}(\Omega)$, as follow:

$$\boldsymbol{e}(x,y,z) = \sum_{k=1}^{\#\mathbb{B}(\Omega)} c_k \, \boldsymbol{e}_k(x,y,z). \tag{1}$$

By applying the classical Galerkin finite element (FE) scheme [1], the coefficients c_k can be computed using the elementary terms $\mathcal{T}_{i,j}^e$ given in (2) and (3), with μ the magnetic permeability, ε the electric permittivity and k the wavenumber. Each $\mathcal{T}_{i,j}^e$ is giving the contribution of the degrees of freedom (DOF) i and j of the mesh element e:

$$\mathcal{T}_{i,j}^e = \mathcal{I}_{i,j}^e - \mathcal{K}_{i,j}^e, \tag{2}$$

with

$$\begin{cases} \mathcal{I}_{i,j}^e = \displaystyle\int_\omega \mu^{-1} \det{}^{-1} \boldsymbol{J}^e \, (\boldsymbol{J}^e \, \mathbf{curl}\, \boldsymbol{e}_i)^T \big(\boldsymbol{J}^e \, \mathbf{curl}\, \boldsymbol{e}_j\big) \, \mathrm{d}\omega, \\ \mathcal{K}_{i,j}^e = \displaystyle\int_\omega k^2 \varepsilon \det \boldsymbol{J}^e \, \big(\boldsymbol{J}^{e^{-T}} \boldsymbol{e}_i\big)^T \big(\boldsymbol{J}^{e^{-T}} \boldsymbol{e}_j\big) \, \mathrm{d}\omega, \end{cases} \tag{3}$$

where $\boldsymbol{J}^e$ is the Jacobian matrix of the mapping between a reference element ω and the element e of the mesh.

The classical finite element assembly algorithm is quite simple. It consists in iterating on every element. Then, for a given element, the integrals of (2) and (3) are computed for every pair of DOF i and j. The corresponding results are stored in a matrix used to compute the coefficients of (1).

It is worth noticing that increasing the basis order will have two impacts on the computation time:

- each element will have more DOFs, thus increasing the number of $\mathcal{T}_{i,j}^e$ to compute;
- the numerical quadrature will require more points, thus slowing down the computation of each $\mathcal{T}_{i,j}^e$.

These two phenomena will substantially increase the assembly time. To give some orders of magnitude, Table 1 shows the number of $\mathcal{K}_{i,j}^e$ terms in (3), for a given tetrahedron e, and the required number of integration points.

Curl-conforming basis order	1	2	3	4
Integration points	4	11	24	43
$\mathcal{K}_{i,j}^e$ terms in (3)	144	900	3600	11025

Table 1: Number of $\mathcal{K}_{i,j}^e$ terms and the required integration points for a curl-conforming basis defined over a tetrahedron.

3 Efficient assembly

The key idea of a fast assembly procedure is to compute all the integrals of (3) with dense matrix – matrix products, as proposed by [2,3] for H^1 bases. This operation exhibits an excellent cache reuse, and is standardized in the Basic Linear Algebra Subprograms (BLAS). Highly optimized BLAS implementations can be found for almost all architectures.

For simplicity, only the integral $\mathcal{I}_{i,j}^e$ of (3) will be treated. But the same procedure can be applied to $\mathcal{K}_{i,j}^e$. We start by rewriting $\mathcal{I}_{i,j}^e$ in a per-element way, with $F^e = \mu^{-1} \det{}^{-1} \boldsymbol{J}^e$:

$$\mathcal{I}_{i,j}^e = \int_\omega \sum_{a=1}^{3}\sum_{b=1}^{3}\sum_{c=1}^{3} F^e J_{a,b}^e \, J_{a,c}^e \, \mathrm{curl}_b \boldsymbol{e}_i \, \mathrm{curl}_c \boldsymbol{e}_j \, \mathrm{d}\omega. \tag{4}$$

This integral can be computed numerically with a Gaussian quadrature. The set of integration points is denoted by $\mathbb{G}$, and the g^{th} integration weight is written w_g. The notation $\alpha|_g$ means that α is evaluated at the g^{th} point of $\mathbb{G}$. In addition, the operation $\{\cdot,\cdot\}$, resp. $\{\cdot,\cdot,\cdot\}$, is defined as returning a unique value from two, reps. three, others. Thus, (4) becomes:

$$\begin{cases} \mathcal{I}_{i,j}^{e} = \sum_{g=1}^{\#\mathbb{G}} \sum_{b=1}^{3} \sum_{c=1}^{3} \boldsymbol{B}[e,\{g,b,c\}]\, \boldsymbol{C}[\{g,b,c\},\{i,j\}], \\ \boldsymbol{B}[e,\{g,b,c\}] = F^{e}|_g \sum_{a=1}^{3} J_{a,b}^{e}\big|_g\, J_{a,c}^{e}\big|_g, \\ \boldsymbol{C}[\{g,b,c\},\{i,j\}] = w_g\, \mathrm{curl}_b \boldsymbol{e}_i|_g\, \mathrm{curl}_c \boldsymbol{e}_j\big|_g. \end{cases} \tag{5}$$

From (5) it can be immediately seen that $\mathcal{I}_{i,j}^{e}$ is an entry at row e and column $\{i,j\}$ of a matrix, defined as the product of $\boldsymbol{B}$ and $\boldsymbol{C}$ of (5). Matrix $\boldsymbol{B}$ is composed of the Jacobian matrices and potentially non linear terms (μ for instance). Matrix $\boldsymbol{C}$ is composed only of the basis functions defined over the reference element, and is thus geometrically invariant.

4 Orientation problem

As exposed in [4], the functions of a curl-conforming basis are dependent on the orientation of the mesh elements. Usually, this problem is overcome by choosing the right function at runtime, when each element is evaluated during the assembly. However, to apply (5) it is needed to know the basis functions before iterating on the elements.

This situation may be solved by generating every possible basis function, for every possible element orientation. Then, the matrices of (5) are split among these possible orientations.

A tree structure can be used to generate all the possible orientations, and to sort efficiently the elements among their orientation. For a topology composed by V vertices, it is possible to find $V!$ permutations of these vertices. A tree can be constructed where every path from root to leaf is one of these permutations, as shown in Figure 1 for the triangle case.

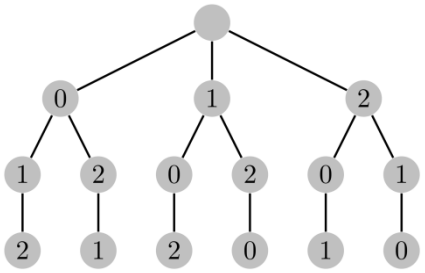

Figure 1: Example of tree structure (triangle case).

To reduce the number of needed bases, the permutations are compared to find rotations that can be exploited. For example, only two bases are needed on a tetrahedron. These rotations will be taken into account using their Jacobian matrix. Then, every leaf is associated to its corresponding oriented basis.

If the elements are represented by an ordered sequence of V vertices, the basis corresponding to an element is found by matching its sequence to a path in the tree. This can be achieved in $\mathcal{O}(V \log_2 V)$. Indeed, it is needed to travel a path of length V. And at a given depth, the next node matching the sequence is found by dichotomic search in $\mathcal{O}(\log_2 V)$.

5 Numerical results

In Figure 2, the assembly time of the classical and efficient matrix – matrix algorithms are presented. The FE matrix is assembled for a propagation problem into a wave guide, meshed with 5585 curved tetrahedra.

The FE system is assembled for an increasing basis order. The tests were done on a Intel Core i7 960 using a serial implementation of ATLAS BLAS.

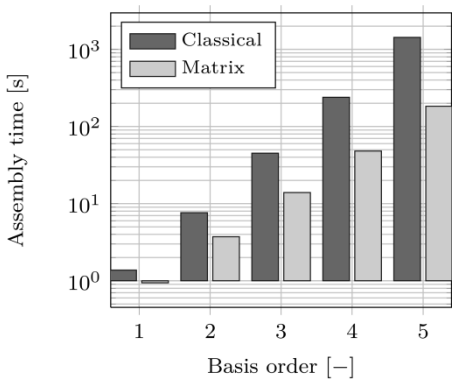

Figure 2: Assembly time for the classical and fast algorithms.

It can be seen from Figure 2 that the matrix algorithm is much faster than the classical one for high order interpolations. For instance, the speedup on an order 5 problem, with around 500000 unknowns, approaches 8.

Acknowledgements

This work was supported in part by the Belgian Science Policy under grant IAP P7/02 and the Walloon Region under grant WIST3 DOMEX. N. Marsic is a fellowship beneficiary with the Belgian Research Training Fund for Industry and Agriculture (FRIA).

References

[1] A. Bossavit. *Computational electromagnetism*, Acedemic Press, (1998).

[2] J. Lambrechts. *Finite Element Methods for Coastal Flows: Application to the Great Barrier Reef*, PhD thesis, Université catholique de Louvain, (2011).

[3] K. Hillewaert. "Exploiting Data Locality in the DGM Discretisation for Optimal Efficiency", *Notes on Numerical Fluid Mechanics and Multidisciplinary Design*, **113**, pp. 11–23, (2010).

[4] M. Ainsworth, J. Coyle. "Hierarchic finite element bases on unstructured tetrahedral meshes", *International Journal for Numerical Methods in Engineering*, **58**, pp. 2103–2130, (2003).