SYMPLECTIC LOCAL TIME-STEPPING IN NON-DISSIPATIVE DGTD METHODS APPLIED TO WAVE PROPAGATION PROBLEMS

Serge Piperno

Cermics, projet Caiman
Ecole des Ponts (ParisTech) et INRIA
Cité Descartes, Champs-sur-Marne, 77455 Marne-la-Valle cedex 2, France
e-mail: Serge.Piperno@cermics.enpc.fr, web page: http://cermics.enpc.fr/~piper/

Key words: Waves, Discontinuous Galerkin, Symplectic schemes, Energy conservation

Abstract. The Discontinuous Galerkin Time Domain (DGTD) methods are now popular for the solution of wave propagation problems required for the accurate transient modeling of systems involving electromagnetic waves in many emerging technologies. While Yee’s explicit, energy-conserving FDTD method is still prominent but restricted to structured grids and second-order accuracy, DGTD methods 1) are able to deal with unstructured, possibly locally-refined meshes around complex geometries, 2) remain fully explicit with easy parallelization, and 3) have extensions to high orders of accuracy. Non-dissipative versions exist, where some discrete electromagnetic energy is exactly conserved. However, the stability limit of the methods, related to the smallest elements in the mesh, calls for the construction of local-time stepping algorithms. These schemes have already been developed for N-body mechanical problems and are known as symplectic schemes. In this paper, they are transformed and applied to DGTD methods on wave propagation problems in order to obtain stable and accurate local time-stepping algorithms.

1 INTRODUCTION

The accurate modeling of systems involving electromagnetic waves, in particular through the resolution of the time-domain Maxwell equations on space grids, remains of strategic interest for many emerging technologies (optical waveguides, furtivity, weapon technologies, etc). Although the explicit, energy-conserving Finite Difference Time-Domain (FDTD) method proposed by Yee is still prominent, it lacks two important features to be easily applicable in industrial contexts: the use of structured or block-structured grids and the limited accuracy.

Many different types of methods have been proposed in order to handle complex geometries and heterogeneous configurations by dealing with unstructured tetrahedral meshes, including Finite Element Time-Domain methods, Finite Volume Time-Domain...
methods\textsuperscript{3, 4, 5}, which are not easily extendible to high orders of accuracy. At the same
time, the global conservation of the electromagnetic energy, which is one particular aspect
of Yee’s original method, can also achieved by FETD methods or FVTD methods based
on totally centered numerical fluxes\textsuperscript{5}, coupled with a centered implicit time-scheme or an
explicit leap-frog time-scheme.

The Discontinuous Galerkin methods enjoy an impressive favor nowadays and are now
used in many and various applications\textsuperscript{6}, taking advantage of their ability to achieve a
high order of accuracy by simply choosing suitable basis functions (spectral elements\textsuperscript{7},
Lagrange high-order polynomials on tetrahedra\textsuperscript{8, 9, 10}) or to handle complicated geometries
and meshes (including locally-refined\textsuperscript{11} and non-conformal grids\textsuperscript{12}). The existing software
are mostly based on upwind fluxes and multi-step low-storage Runge-Kutta time-schemes
\textsuperscript{4, 13, 7}, which lead to robust and stable, but slightly dissipative Discontinuous Galerkin
Time-Domain (DGTD) methods. However, centered fluxes coupled with an explicit leap-
frog time-scheme lead to a convergent, stable, and energy-conserving DGTD method\textsuperscript{14}.
However, like all other time-explicit methods used for computational electromagnetics,
the DGTD methods cannot deal very easily with configurations involving small devices
or details in the geometry, because explicit time-schemes have a stability constraint on
the time-step directly related to the smallest elements in the mesh, while implicit time-
schemes are expensive.

In this paper, we propose two original strategies to overcome these difficulties by intro-
ducing locally implicit time-integration (the time scheme is implicit in some parts of the
domain and explicit everywhere else) or explicit local time-stepping. Similar algorithmic
solutions have already been proposed in the literature in several contexts, with recently
developed additive Runge-Kutta methods\textsuperscript{15, 16, 17, 18}. However, the particular aim of this
work is to preserve the energy conservation property of DGTD methods with centered
fluxes. Solutions have been proposed in the past, including Lagrange multipliers requir-
ing the solution of relatively small linear systems\textsuperscript{19, 20} or deriving from so-called energy-
accurate fluid-structure interaction solution procedures\textsuperscript{21}. The algorithm proposed here
is inspired from the theory of symplectic integrators developed for the numerical time in-
tegration of dynamical Hamiltonian systems. Such methods have been successfully used
in the fields of astronomy and molecular dynamics where numerical accuracy and energy
conservation are very important over large time integration periods\textsuperscript{22}. As the Maxwell’s
equations can be written as an infinite-dimensional Hamiltonian system of PDEs, people
are now considering the use of symplectic schemes for the time discretization in time-
domain simulations\textsuperscript{23, 24, 25}. In this paper, we propose the derivation of a totally explicit
local time-stepping integration scheme (and of a locally implicit scheme) from the one
proposed by Hardy et al. for symplectic systems\textsuperscript{26}. It is globally second-order accurate
and symplectic, thus leading to the conservation of some approximate energy.
2 DISCONTINUOUS GALERKIN METHOD FOR MAXWELL’S SYSTEM

We consider the Maxwell’s equations in three space dimensions for heterogeneous anisotropic linear media with no source. The electric permittivity tensor \( \varepsilon(x) \) and the magnetic permeability tensor \( \mu(x) \) are both assumed symmetric positive definite with uniform strictly positive lower and upper bounds. The electric field \( \vec{E} \) and the magnetic field \( \vec{H} \) verify Maxwell’s system, i.e.

\[
\varepsilon \partial_t \vec{E} = \text{curl} \vec{H}, \quad \mu \partial_t \vec{H} = -\text{curl} \vec{E},
\]

where the symbol \( \partial_t \) denotes a time derivative. These equations are set and solved on a bounded polyhedral domain \( \Omega \). For the sake of simplicity, a metallic boundary condition is set everywhere on the domain boundary \( \partial \Omega \), i.e. \( \vec{n} \times \vec{E} = \vec{0} \) (where \( \vec{n} \) is the unitary outwards normal). We assume we dispose of a partition of a polyhedral domain \( \Omega_h \) (approximating the regular or Lipschitz-continuous domain of interest \( \Omega \)) into a finite number of polyhedra (each one having a finite number of faces). For each polyhedral element \( T_i \), \( V_i \) denotes its volume, and \( \varepsilon_i \) and \( \mu_i \) are respectively the local electric permittivity and magnetic permeability tensors of the medium inside \( T_i \). For each internal polyhedral face \( a_{ik} = T_i \cap T_k \), we denote by \( S_{ik} \) the measure of \( a_{ik} \) and by \( \vec{n}_{ik} \) the unitary normal, oriented from \( T_i \) towards \( T_k \). The same definitions are extended to metallic boundary faces (in the intersection of the domain boundary \( \partial \Omega_h \) with a finite element), the index \( k \) corresponding to a fictitious element outside the domain. Finally, we denote by \( V_i \) the set of indices of the neighboring elements of the \( T_i \) (having a face in common). We also define the perimeter \( P_i \) of \( T_i \) by \( P_i = \sum_{k \in V_i} S_{ik} \). We recall the following geometrical property for all elements: \( \sum_{k \in V_i} S_{ik} \vec{n}_{ik} = 0 \).

Following the Discontinuous Galerkin approach, the electric and magnetic fields inside each finite element \( T_i \) are sought for as linear combinations \( (\vec{E}_i, \vec{H}_i) \) of linearly independent basis vector fields \( \varphi_{ij} \), \( 1 \leq j \leq d_i \), where \( d_i \) denotes the local number of scalar degrees of freedom inside \( T_i \). We denote by \( \mathcal{P}_i = \text{Span}(\varphi_{ij}, 1 \leq j \leq d_i) \). The approximate fields \( (\vec{E}_h, \vec{H}_h) \), defined by \( (\forall i, \vec{E}_h|_{T_i} = \vec{E}_i, \vec{H}_h|_{T_i} = \vec{H}_i) \) are allowed to be completely discontinuous across element boundaries. Because of this complete discontinuity, a global variational formulation cannot be obtained. However, dot-multiplying (1) by any given vector field \( \varphi \in \mathcal{P}_i \), integrating over each single element \( T_i \) and integrating by parts, yields

\[
\begin{align*}
\int_{T_i} \varphi \cdot \varepsilon_i \partial_t \vec{E} &= - \int_{\partial T_i} \varphi \cdot F_{\vec{E}}(\vec{E}, \vec{H}, \vec{n}) + \int_{T_i} \text{curl} \varphi \cdot \vec{H}, \\
\int_{T_i} \varphi \cdot \mu_i \partial_t \vec{H} &= \int_{\partial T_i} \varphi \cdot F_{\vec{H}}(\vec{E}, \vec{H}, \vec{n}) - \int_{T_i} \text{curl} \varphi \cdot \vec{E},
\end{align*}
\]

where \( F_{\vec{E}} \) and \( F_{\vec{H}} \) stand for numerical fluxes approximating respectively \( \vec{H} \times \vec{n} \) and \( \vec{E} \times \vec{n} \) at the interface. We choose to use totally centered fluxes, i.e. \( F_{\vec{H}} \simeq (\vec{H}_i + \vec{H}_k)/2 \times \vec{n} \) and \( F_{\vec{E}} \simeq (\vec{E}_i + \vec{E}_k)/2 \times \vec{n} \), but upwind fluxes (leading to numerical dissipation) could also be
used. For boundaries, the metallic boundary condition on a boundary face \( a_{ik} \) (\( k \) in the element index of the fictitious neighboring element) is dealt with \textit{weakly}, in the sense that traces of fictitious fields \( \mathbf{E}_k \) and \( \mathbf{H}_k \) are used for the computation of numerical fluxes for the boundary element \( T_i \). We simply take \( \mathbf{E}_{ik|a_{ik}} = -\mathbf{E}_{i|a_{ik}} \) and \( \mathbf{H}_{ik|a_{ik}} = \mathbf{H}_{i|a_{ik}} \). Replacing surface integrals using centered fluxes in (2) and re-integrating by parts yields

\[
\left\{ \begin{array}{l}
\int_{T_i} \varphi \cdot \tilde{\varphi}_i \partial_t \mathbf{E} = \frac{1}{2} \int_{T_i} (\text{curl} \varphi \cdot \mathbf{H} + \text{curl} \tilde{\mathbf{H}} \cdot \varphi) - \frac{1}{2} \sum_{k \in \mathcal{V}_i} \int_{a_{ik}} \varphi \cdot (\mathbf{H}_k \times \bar{n}_{ik}), \\
\int_{T_i} \varphi \cdot \tilde{\varphi}_i \partial_t \mathbf{H} = -\frac{1}{2} \int_{T_i} (\text{curl} \varphi \cdot \tilde{\mathbf{E}} + \text{curl} \tilde{\mathbf{E}} \cdot \varphi) + \frac{1}{2} \sum_{k \in \mathcal{V}_i} \int_{a_{ik}} \varphi \cdot (\mathbf{E}_k \times \bar{n}_{ik}).
\end{array} \right.
\] (3)

We can rewrite this formulation in terms of scalar unknowns. Inside each element, the fields are recomposed according to \( \mathbf{E}_i = \sum_{1 \leq j \leq d_i} E_{ij} \varphi_{ij} \), \( \mathbf{H}_i = \sum_{1 \leq j \leq d_i} H_{ij} \varphi_{ij} \). Let us denote by \( \mathbf{E}_i \) and \( \mathbf{H}_i \) respectively the columns \((E_{ud})_{1 \leq l \leq d_i}\) and \((H_{ud})_{1 \leq l \leq d_i}\). The equations (3) can be rewritten as:

\[
M'_i \partial_t \mathbf{E}_i = K_i \mathbf{H}_i - \sum_{k \in \mathcal{V}_i} S_{ik} \mathbf{H}_k, \quad M''_i \partial_t \mathbf{H}_i = -K_i \mathbf{E}_i + \sum_{k \in \mathcal{V}_i} S_{ik} \mathbf{E}_k,
\] (4)

where \( M'_i \), \( M''_i \), and \( K_i \) are three square symmetric matrices of size \( d_i \) (\( M'_i \) and \( M''_i \) being positive definite). They are given by \( (M'_i)_{jl} = \int_{T_i} \tilde{\varphi}_{ij} \tilde{\varphi}_i \mathbf{H}_k \cdot \varphi_{il} \), \( (M''_i)_{jl} = \int_{T_i} \tilde{\varphi}_{ij} \tilde{\varphi}_i \varphi_{il} \), and \( (K_i)_{jl} = \frac{1}{2} \int_{T_i} (\text{curl} \varphi_{ij} \mathbf{H}_k + \tilde{\varphi}_{ij} \mathbf{E}_k) \cdot \tilde{\varphi}_{kl} \). For any interface \( a_{ik} \), the \( d_i \times d_k \) rectangular matrix \( S_{ik} \) is given by \( (S_{ik})_{jl} = \frac{1}{2} \int_{a_{ik}} \tilde{\varphi}_{ij} \cdot (\tilde{\varphi}_{kl} \times \bar{n}_{ik}) \).

Finally, if all electric (resp. magnetic) unknowns are regrouped inside column vectors \( \mathbf{E} \) (resp. \( \mathbf{H} \)) of size \( d = \sum_i d_i \), then the space discretized system (4) can be rewritten as

\[
\left\{ \begin{array}{l}
M'_i \partial_t \mathbf{E} = K \mathbf{H} - \mathbf{A} \mathbf{H} - \mathbf{B} \mathbf{E}, \\
M''_i \partial_t \mathbf{H} = -K \mathbf{E} + \mathbf{A} \mathbf{E} - \mathbf{B} \mathbf{E}.
\end{array} \right.
\]

In the system above, \( M'_i \), \( M''_i \) and \( K \) are \( d \times d \) block diagonal matrices with diagonal blocks equal to \( M'_i \), \( M''_i \), and \( K_i \) respectively. Therefore \( M'_i \) and \( M''_i \) are symmetric positive definite, and \( K \) is symmetric. Also, \( \mathbf{A} \) also is a \( d \times d \) block sparse matrix, whose non-zero blocks are equal to \( S_{ik} \) when \( a_{ik} \) is an internal face. It can be checked that \( S_{ki} = ^tS_{ik} \), and then \( \mathbf{A} \) is symmetric. Finally, \( \mathbf{B} \) is a \( d \times d \) block diagonal matrix, whose non-zero diagonal blocks are equal to \( S_{ik} \) when \( a_{ik} \) is a metallic boundary face of the grid. In that case, \( S_{ik} = -^tS_{ik} \), and then \( \mathbf{B} \) is skew-symmetric (\( ^t\mathbf{B} = -\mathbf{B} \)).

Finally, the Maxwell’s equations, discretized using discontinuous Galerkin finite-elements with centered fluxes and arbitrary local accuracy and basis functions can be written, in function of the matrix \( \mathbf{S} = K - \mathbf{A} - \mathbf{B} \), in the general form:

\[
\left\{ \begin{array}{l}
M'_i \partial_t \mathbf{E} = \mathbf{SH}, \\
M''_i \partial_t \mathbf{H} = -^t\mathbf{SE},
\end{array} \right. \quad (M'_i, M''_i \text{ symmetric positive definite}).
\] (5)
The general form of the system of ordinary differential equations obtained preserves an energy. Indeed, for any solution of (5), the quantity \( \mathcal{E} \equiv \frac{1}{2} (t^\prime \mathbf{E} \mathbf{E}^\prime + t^\prime \mathbf{H} \mathbf{H}^\prime) \) is exactly conserved.

3 SYMPLECTIC SCHEMES FOR LOCALLY REFINED MESHES

3.1 Symplectic schemes for Hamiltonian systems

Symplectic integrators include a variety of different time-discretization schemes designed to preserve the global symplectic structure of the phase space for a Hamiltonian system. These integrators are well established for finite-dimensional Hamiltonian systems\(^{27}\) and the number of applications of symplectic schemes in the context of computational electromagnetics is currently growing\(^{24, 25}\). Indeed, the Maxwell’s equations can be written as an infinite-dimensional Hamiltonian system of PDEs and the discretized Maxwell equations (via FDTD\(^{24}\), FVTD, FETD\(^{25}\), or DGTD) as a finite-dimensional system of ODEs as well if no dissipation is introduced. This lead to time accuracies up to fourth order. In the computational electromagnetics community, the leapfrog time-scheme is widely used and would take the following form for the time-integration of (5):

\[
\begin{align*}
M^e \mathbf{E}^{n+1} &= M^e \mathbf{E}^n + \Delta t S H^{n+\frac{1}{2}}, \\
M^\mu \mathbf{H}^{n+\frac{3}{2}} &= M^\mu \mathbf{H}^{n+\frac{1}{2}} - \Delta t \mathbf{S} E^{n+1}.
\end{align*}
\]

It is well-known that the leapfrog scheme exactly preserves the quadratic form \( \mathcal{E}^n = t^\prime E^n M^e E^n + t^\prime H^{n+\frac{1}{2}} M^\mu H^{n+\frac{1}{2}} \) of numerical unknowns \( \mathbf{E}^n \) and \( \mathbf{H}^{n+\frac{1}{2}} \). The quadratic form \( \mathcal{E}^n \) is positive definite if \( \Delta t \) is small enough, which proves the scheme is stable.

The leapfrog is not used under the same form for the transient solution of N-body systems. The Verlet method, an equivalent but slightly more expensive version with time-collocated unknowns is rather used. It writes:

\[
\begin{align*}
M^e \mathbf{E}^{n+1} &= M^e \mathbf{E}^n + \frac{\Delta t}{2} t^\prime \mathbf{S} \mathbf{E}^n, \\
M^\mu \mathbf{H}^{n+\frac{3}{2}} &= M^\mu \mathbf{H}^n + \frac{\Delta t}{2} t^\prime \mathbf{S} \mathbf{E}^n, \\
M^\mu \mathbf{H}^{n+1} &= M^\mu \mathbf{H}^{n+\frac{1}{2}} - \frac{\Delta t}{2} t^\prime \mathbf{S} \mathbf{E}^{n+1}.
\end{align*}
\]

Although equivalent, the reversible writing of the Verlet method seems to lead more easily to enhancements: an adaptive Verlet method allows for a stable, energy-conserving, leapfrog-type integration with a varying time-step\(^{28}\). Higher-order accurate extensions are available as generalizations of the Verlet method\(^{29}\) and fast, multi-scale, regularized integrators are available for Kepler motion or atomic dynamics\(^{30}\). Finally, the energy-preserving coupling with the implicit midpoint-rule seems much easier and the totally explicit integration of symplectic systems with different time-steps (i.e. local time-stepping) is already available\(^{26}\), from which the algorithms proposed in this paper are derived.

3.2 Symplectic-like schemes for locally refined meshes

In this section, we present two particular symplectic-like algorithms designed for the particular case where a DGTD method is used for the time-domain solution of Maxwell’s
equations on unstructured meshes where some geometrical details or flaws in the mesh generator lead to locally refined grids. In both case, the computational domain is partitioned and the discretized Maxwell’s system is solved as coupled systems of ODEs, this system (coupling the electromagnetic fields in the two subdomains) not being Hamiltonian anymore. However, the ideas of the time-adaptive scheme of\textsuperscript{26} can be adapted to obtain stable energy-preserving algorithms.

3.2.1 A locally-implicit symplectic scheme

We first consider a case where the set of elements has been partitioned into two classes: one made of particularly small elements and the other one gathering all other elements. This partition has been done once and for all, before the beginning of the time-domain simulation and is based for example on geometrical and physical criteria. At this stage, there is no need of a particular assumption on the connectivity of the set of “small” or “large” elements. The “small” elements will be handled using an implicit midpoint rule, while all other elements will be time-advanced using a Verlet method. Using notations inspired from domain decomposition algorithms, we denote with an “e” (resp. “i”) subscript unknowns and matrices related to the explicit (resp. implicit) subdomain. Unknowns are reordered such that explicit elements and unknowns are numbered first and the block-diagonal matrices $M_e^\epsilon$, $M_i^\epsilon$, $K$ and $B$ are decomposed using block-diagonal sub-blocks $M_e^\epsilon/i$, $M_i^\epsilon/i$, $K_{e/i}$, and $B_{e/i}$ respectively. Clearly, $M_e^\epsilon/i$ and $M_i^\epsilon/i$ are symmetric positive definite, $K_{e/i}$ are symmetric, and $B_{e/i}$ are skew-symmetric. The non-block diagonal matrix $A$, corresponding to interfaces fluxes is decomposed into $A = \begin{pmatrix} A_{ee} & A_{ei} \\ A_{ie} & A_{ii} \end{pmatrix}$, where $A_{ee}$ and $A_{ii}$ are symmetric and $A_{ei} = A_{ie}$.

Finally, defining the two symmetric matrices $S_e = K_e - A_{ee} + B_e$ and $S_i = K_i - A_{ii} + B_i$, the system of ordinary differential equations (5) can be rewritten as

$$\begin{cases} M_e^\epsilon \partial_t E_e = S_e H_e - A_{ei} H_i, \\ M_i^\epsilon \partial_t H_i = -S_e E_e + A_{ei} E_i, \end{cases} \quad \begin{cases} M_i^\epsilon \partial_t E_i = S_i H_i - A_{ie} H_e, \\ M_e^\epsilon \partial_t H_e = -S_i E_i + A_{ie} E_e. \end{cases}$$

We propose the following implicit-explicit algorithm: starting from unknowns at time $t^n = n\Delta t$, we perform the three following sub-steps:

Explicit subdomain pseudo-Euler with $\Delta t/2$

$$\begin{align*}
M_e^\epsilon H_e^{n+1/2} &= M_e^\epsilon H_e^n - \frac{\Delta t}{2} S_e E_e^n + A_{ei} E_i^n, \\
M_i^\epsilon E_i^{n+1/2} &= M_i^\epsilon E_i^n + \frac{\Delta t}{2} S_e H_e^n + A_{ei} H_i^n.
\end{align*}$$

Implicit subdomain midpoint rule with $\Delta t$

$$\begin{align*}
M_e^\epsilon E_i^{n+1} &= M_e^\epsilon E_i^n + \Delta t \left( S_e H_e^n + A_{ei} E_i^n \right), \\
M_e^\epsilon H_i^{n+1} &= M_e^\epsilon H_i^n - \Delta t \left( S_e E_e^n + A_{ei} H_i^n \right),
\end{align*}$$

 Explicit subdomain reversed pseudo-Euler with $\Delta t/2$

$$\begin{align*}
M_e^\epsilon E_i^{n+1/2} &= M_e^\epsilon E_i^{n+1/2} + \frac{\Delta t}{2} S_e H_e^{n+1/2} - A_{ei} H_i^{n+1/2}, \\
M_i^\epsilon H_i^{n+1/2} &= M_i^\epsilon H_i^{n+1/2} - \frac{\Delta t}{2} S_e E_e^{n+1/2} + A_{ei} E_i^{n+1/2}.
\end{align*}$$
This algorithm is obviously reversible. One can verify that, if the two subdomains are disconnected (i.e. $A_{ei} = \emptyset_d$), this algorithm reduces to the juxtaposition of the Verlet-method for the “explicit” subdomain and the midpoint-rule for the “implicit” subdomain. The stability of the algorithm (8) can be shown using an energy approach. It can be shown\(^{31}\) that the following quadratic form $E_n$ of numerical unknowns $E^n_e$, $E^n_i$, $H^n_e$, and $H^n_i$ is exactly conserved (i.e. $E^{n+1} = E^n$) through a time step of Algorithm (8):

$$E^n = E^n_e + E^n_i + E^n_c$$

with

$$E^n_e = \frac{t}{2} E^n_i M^e_e E^n_e + \frac{t}{2} H^n_i M^e_e H^n_e$$

$$E^n_i = \frac{t}{2} E^n_i M^i_i E^n_i + \frac{t}{2} H^n_i M^i_i H^n_i$$

$$E^n_c = -\frac{t}{4} H^n_i M^c_{ei}(M^e_{ei})^{-1} A_{ei} H^n_i.$$

One can easily show that the explicit-implicit coupled algorithm (8) is stable for $\Delta t$ small enough (for $\Delta t$ small enough, the total energy $E^n_e$ is a positive definite quadratic form of unknowns). A more closer investigation is required to determine if a sufficient condition on $\Delta t$ for having a stable coupled scheme is that the explicit Verlet scheme alone is stable.

### 3.2.2 A multi-scale fully-explicit symplectic scheme

The fully explicit algorithm proposed in this section is directly inspired from the one introduced by Hardy et al.\(^{26}\). In this paper, the authors propose a second-order accurate symplectic integration scheme for N-body problems with multiple time stepping, i.e. the atoms or bodies are time-advanced simultaneously with different time steps. In their papers, the authors consider the general case where successive classes of bodies have corresponding time steps being multiple of the next one, the choice of powers of 2 being probably the most efficient in general. We present here a less general version, with time steps given as $\Delta t/2^k$ where $\Delta t$ is the global time step of the algorithm. To make things clear, we assume that 1) the set of elements has been partitioned into N classes (this partition has been done once and for all, before the beginning of the time-domain simulation and is based for example on geometrical or physical criteria), and 2) the global time step of the algorithm is $\Delta t$ and for $1 \leq k \leq N$, elements of the class $k$ will be time-advanced using the Verlet method with the local time step $\Delta t/2^{N-k}$ (thus the larger elements should lie in class $N$ and the smallest in class 1).

Let us define recursively the algorithm $R^N(\tau)$ for time-advancing N classes over the time interval $\tau > 0$. We decide that the algorithm $R^1(\tau)$ with only one class is exactly the Verlet method (7) with $\Delta t = \tau$. For any $N \geq 1$, if $R^N(\tau)$ is well defined, we define $R^{N+1}(\tau)$ by:

1. start with all unknowns at time $t^n = n\Delta t$;

2. advance all elements with class $k \leq N$ with $R^N(\Delta t/2)$; if required, use values at time $t^n$ for unknowns in elements of class $N + 1$;
3. advance all elements with class $k = N + 1$ with the Verlet method (i.e. $R^1(\Delta t)$); if required, use values at time $t^n + \Delta t/2$ for unknowns in elements of class $k \leq N$;

4. advance all elements with class $k \leq N$ with $R^N(\Delta t/2)$; if required, use values at time $t^{n+1}$ for unknowns in elements of class $N + 1$;

5. all unknowns at time $t^{n+1} = t^n + \Delta t$ have been computed.

The reader can check that this algorithm does not require any additional storage and remains completely explicit. It is reversible. The algorithms $R^2(\Delta t)$ and $R^3(\Delta t)$ are depicted in Figure 1. Concerning the most simple algorithm $R^2(\Delta t)$, we propose to use the subscripts $k \in \{1, 2\}$ denoting the class of the elements (thus playing the role of subscripts $e$ and $i$ in the implicit-explicit algorithm). Elements are reordered and matrices are substructured as in the explicit-implicit coupling case. Algorithm $R^2(\Delta t)$ then reads:

$$
\begin{align*}
\text{Step 1.} & \quad M_1^n H_1^{n+\frac{1}{2}} = M_1^n H_1^n + \frac{\Delta t}{4} \left( -S_1 E_1^n + A_{12} E_2^n \right), \\
\text{Step 2.} & \quad M_1^n E_1^{n+\frac{1}{2}} = M_1^n E_1^n + \frac{\Delta t}{4} \left( S_1 H_1^{n+\frac{1}{2}} - A_{12} H_2^n \right), \\
\text{Step 3.} & \quad M_1^n H_1^{n+\frac{1}{2}} = M_1^n H_1^n + \frac{\Delta t}{4} \left( -S_1 E_1^{n+\frac{1}{2}} + A_{12} E_2^n \right), \\
\text{Step 4.} & \quad M_2^n H_2^{n+\frac{1}{2}} = M_2^n H_2^n + \frac{\Delta t}{4} \left( -S_2 E_2^n + A_{21} E_1^{n+\frac{1}{2}} \right), \\
\text{Step 5.} & \quad M_2^n E_2^{n+1} = M_2^n E_2^n + \Delta t \left( S_2 H_2^{n+\frac{1}{2}} - A_{21} H_1^{n+\frac{1}{2}} \right), \quad (10) \\
\text{Step 6.} & \quad M_2^n H_2^{n+1} = M_2^n H_2^{n+\frac{1}{2}} + \frac{\Delta t}{2} \left( -S_2 E_2^{n+1} + A_{21} E_1^{n+1} \right), \\
\text{Step 7.} & \quad M_1^n H_1^{n+\frac{1}{2}} = M_1^n H_1^n + \frac{\Delta t}{4} \left( -S_1 E_1^{n+\frac{1}{2}} + A_{12} E_2^{n+1} \right), \\
\text{Step 8.} & \quad M_1^n E_1^{n+1} = M_1^n E_1^n + \frac{\Delta t}{4} \left( S_1 H_1^{n+\frac{1}{2}} - A_{12} H_2^{n+1} \right), \\
\text{Step 9.} & \quad M_1^n H_1^{n+1} = M_1^n H_1^{n+\frac{1}{2}} + \frac{\Delta t}{4} \left( -S_1 E_1^{n+1} + A_{12} E_2^{n+1} \right).
\end{align*}
$$

**Energy conservation and stability.** The stability of the algorithm (10) can be shown using the theory of symplectic schemes. Hence, it does not yield in general an explicit
expression of the energy which is conserved. Such an expression can be obtained using a not so classical energy approach. However, the computations are tedious and the generalizations to more complex versions \( R^N(\Delta t) \) with \( N > 2 \) seems a difficult task. We refer to [31] for the explicit expression on the energy conserved and for a proof of stability (for small \( \Delta t \)).

4 NUMERICAL RESULTS

This section is devoted to numerical results obtained with the local time-stepping algorithms proposed. We consider here the homogeneous Maxwell equations in two space dimensions and in the TE case. The unknown fields are \( E_x, E_y, \) and \( H_z \). The first part of this section is devoted to the numerical investigation of the impact of local time-stepping on the accuracy. The following sections report numerical results obtained on toy problems where detailed structures are involved.

4.1 Impact of explicit local time-stepping on the accuracy

In this section, we investigate the impact of local-time stepping on the global accuracy of the numerical solution. Let us recall the known results on the accuracy of the DGTD method used in this paper. Let is introduce the largest element diameter \( h \) in the mesh. Using the method of lines (i.e. before time discretization and after Discontinuous Galerkin space discretization), the semi-discretized method based on \( \mathbb{P}_k \) polynomials inside all elements yields an error in \( h^k \) if totally centered fluxes are used\(^{14}\) and in \( h^{k+1} \) if upwind fluxes are used\(^{9}\). Although upwind fluxes lead to slightly dissipative but more robust schemes, it is well known they do not cope well with leap-frog-type time schemes (the use of RKDG-scheme is a possible way to get high accuracy in both time and space\(^{32}\), but the global stability of algorithms using locally adapted time steps is still to be proved).

The aim of this paper being to concentrate on local time-stepping, we intend to assess the accuracy of the time-discretization algorithm, assuming the “fixed-k” \( \mathbb{P}_k \) Discontinuous Galerkin spatial discretization is used.

We consider a totally reflecting unit square cavity for which acoustic eigenmodes are well-known. We consider here the \((1, 1)\)-mode. For all computations, the initial approximate solution is obtained via element-wise \( L^2 \)-projection on \( \mathbb{P}_k \) polynomials over each element, a sufficiently accurate quadrature formula being used (we have chosen a quadrature exact for polynomials of degree up to 15). At the end of each computation (final time \( T = 1 \)), we compute the \( L^2 \)-norm of the difference between the discontinuous approximate solution and the element-wise \( L^2 \)-projection on \( \mathbb{P}_k \) of the exact solution (the difference between this projection and the very smooth exact solution is in \( O(h^{k+1}) \)).

In order to discriminate space and time discretization errors, we have chosen to use unstructured quasi-uniform grids. We have built four unstructured but regular grids M4, M3, M2, and M1 (each one being twice finer than the previous one). The characteristics of these meshes are given on Table 1.
We first verified the convergence of the $\mathbb{P}_k$-DGTD method based on totally centered fluxes and the Verlet time-integration scheme. We obtained the errors (in $L^2$-norm) reported in Table 2. The time step $\Delta t$ used is derived from the Courant number $\nu \equiv \Delta t/h_{\text{min}}$, and the $\mathbb{P}_k$-DGTD is numerically proved to be stable for $\nu \leq \nu_k$ (with $\nu_0 \simeq 0.768$, $\nu_1 \simeq 0.252$, $\nu_2 \simeq 0.133$, $\nu_3 \simeq 0.085$, $\nu_4 \simeq 0.0584$, etc). For all computations, the Courant number was chosen as a fraction of $\nu_k = 0.9 \nu_k$. In a general setting, the $L^2$-norm of the error should be bound by $e \leq c_k h_k^{\max} + K_t \Delta t^2$, where $c_k$ and $K_t$ should be mesh- and time-step-independent constants. In our context where $h_{\text{max}}/h_{\text{min}}$ is bound, we get $e \leq c_k h^{\infty} + c_t h^2 \nu^2$ (where we have simply used $h = h_{\text{max}}$ and with $c_t$ is also a mesh- and time-step-independent constant). In view of the results in Table 2, many remarks can be made. First, for $k = 1$, the spatial errors prevail ($\forall \nu \leq 1$) and the convergence according to $e \sim c_1 h$ is confirmed. Also, for $k = 2$, the spatial and time errors have the same order but the space errors seem (in this particular case) dominant. The convergence according to $e \sim O(h^2)$ is also confirmed. Finally, for $k = 3$ and $k = 4$, the observed limit for $e$ when $\nu \rightarrow 0$ is close to the expected $O(h^k)$ and the time errors prevail when $\nu \sim \nu_k$. The value of $e(\nu) - e(\nu \rightarrow 0)$ actually behaves like $O(\nu^2)$, i.e. $O(\Delta t^2)$ (it appears more clearly for $k = 4$). We have given in Table 3 the successive values for $e(\nu) - e(\nu \rightarrow 0)$ for $k = 4$ and different $\nu$.

In a second series of computations, we have used the $R^3(\Delta t)$ algorithm on the same meshes in order to compare easily the errors obtained to those obtained with the classical $R^1(\Delta t)$ Verlet method. In order to actually use different classes of elements, we have artificially reduced the admissible time step in given zones: the global time step used $\Delta t$ is the same as for the corresponding computation using the Verlet method, but the time step is $\Delta t/2$ if the distance to the center is less than 0.1333 and $\Delta t/4$ if it is smaller than 0.2. The time step classes are shown on the regular mesh of the square on Figure 2 (left). The $L^2$-norms of the error at time $T = 1$ are given in Table 4 for the meshes M1 and M2. The results are similar to those obtained with the standard Verlet method. The algorithm is second-order accurate in time: the errors due to time discretization are masked for $k = 1$, and the variation of these errors is clearly in $O(\Delta t^2)$, with a larger constant than for the Verlet method though.

It is interesting to note that, in general, the error obtained with the $R^3(\Delta t)$ algorithm is larger than the one obtained with the Verlet method with $\Delta t$. This means that the local time-stepping should not be used as a way to obtain a better accuracy (although more
<table>
<thead>
<tr>
<th></th>
<th>$k = 1$</th>
<th>$k = 2$</th>
<th>$k = 3$</th>
<th>$k = 4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\nu = \nu_k/1$</td>
<td>$4.71 \times 10^{-2}$</td>
<td>$4.64 \times 10^{-5}$</td>
<td>$5.24 \times 10^{-5}$</td>
<td>$5.66 \times 10^{-5}$</td>
</tr>
<tr>
<td>$\nu = \nu_k/2$</td>
<td>$4.69 \times 10^{-2}$</td>
<td>$4.64 \times 10^{-5}$</td>
<td>$5.21 \times 10^{-5}$</td>
<td>$4.06 \times 10^{-5}$</td>
</tr>
<tr>
<td>$\nu = \nu_k/4$</td>
<td>$4.69 \times 10^{-2}$</td>
<td>$4.63 \times 10^{-5}$</td>
<td>$5.20 \times 10^{-5}$</td>
<td>$4.00 \times 10^{-5}$</td>
</tr>
<tr>
<td>$\nu = \nu_k/8$</td>
<td>$4.69 \times 10^{-2}$</td>
<td>$4.63 \times 10^{-5}$</td>
<td>$5.21 \times 10^{-5}$</td>
<td>$3.99 \times 10^{-5}$</td>
</tr>
<tr>
<td>$\nu = \nu_k/16$</td>
<td>$4.69 \times 10^{-2}$</td>
<td>$4.64 \times 10^{-5}$</td>
<td>$5.21 \times 10^{-5}$</td>
<td>$3.99 \times 10^{-5}$</td>
</tr>
<tr>
<td>$\nu = \nu_k/32$</td>
<td>$4.69 \times 10^{-2}$</td>
<td>$4.64 \times 10^{-5}$</td>
<td>$5.20 \times 10^{-5}$</td>
<td>$3.99 \times 10^{-5}$</td>
</tr>
<tr>
<td>$\nu = \nu_k/64$</td>
<td>$4.69 \times 10^{-2}$</td>
<td>$4.64 \times 10^{-5}$</td>
<td>$5.20 \times 10^{-5}$</td>
<td>$3.99 \times 10^{-5}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$k = 1$</th>
<th>$k = 2$</th>
<th>$k = 3$</th>
<th>$k = 4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\nu = \nu_k/1$</td>
<td>$2.30 \times 10^{-2}$</td>
<td>$1.11 \times 10^{-5}$</td>
<td>$7.78 \times 10^{-5}$</td>
<td>$1.04 \times 10^{-5}$</td>
</tr>
<tr>
<td>$\nu = \nu_k/2$</td>
<td>$2.30 \times 10^{-2}$</td>
<td>$1.11 \times 10^{-5}$</td>
<td>$7.62 \times 10^{-5}$</td>
<td>$3.36 \times 10^{-6}$</td>
</tr>
<tr>
<td>$\nu = \nu_k/4$</td>
<td>$2.30 \times 10^{-2}$</td>
<td>$1.11 \times 10^{-5}$</td>
<td>$7.62 \times 10^{-5}$</td>
<td>$2.32 \times 10^{-6}$</td>
</tr>
<tr>
<td>$\nu = \nu_k/8$</td>
<td>$2.30 \times 10^{-2}$</td>
<td>$1.11 \times 10^{-5}$</td>
<td>$7.62 \times 10^{-5}$</td>
<td>$2.27 \times 10^{-6}$</td>
</tr>
<tr>
<td>$\nu = \nu_k/16$</td>
<td>$2.30 \times 10^{-2}$</td>
<td>$1.11 \times 10^{-5}$</td>
<td>$7.62 \times 10^{-5}$</td>
<td>$2.27 \times 10^{-6}$</td>
</tr>
<tr>
<td>$\nu = \nu_k/32$</td>
<td>$2.30 \times 10^{-2}$</td>
<td>$1.11 \times 10^{-5}$</td>
<td>$7.62 \times 10^{-5}$</td>
<td>$2.27 \times 10^{-6}$</td>
</tr>
<tr>
<td>$\nu = \nu_k/64$</td>
<td>$2.30 \times 10^{-2}$</td>
<td>$1.11 \times 10^{-5}$</td>
<td>$7.62 \times 10^{-5}$</td>
<td>$2.27 \times 10^{-6}$</td>
</tr>
</tbody>
</table>

Table 2: $L^2$-norm of the error at time $T = 1$, using the Verlet method.
computations are performed than with the Verlet method for the same macro time step $\Delta t$). It should rather be seen as a way to accelerate the computation. This conclusion should also be moderated by the fact that the local-time stepping zone is quite large in the test cases considered.

4.2 Test-case with a detailed structure (explicit local time-stepping)

We consider a test-case with a complex geometry. An elliptic domain ($2m \times 1.6m$) encloses again a reflecting inclusion centered at the right focus of the ellipse. The device is a circular array of 0.2mm square, set at a distance equal to 1mm from the focus. The mesh obtained contains 1176 vertices and 2254 elements. The mesh partitioning leads to eleven classes of elements, i.e. the smallest elements are time-advanced 1024 times more often than the largest elements. A zoom of the mesh near the square is shown on Figure 3. Contours of the fields obtained with the algorithm $R_{11}^{11}(2.6ms)$ are shown on Figure 4.

We have also used in this section the $P_5$-DGTD (the fields are described with polynomials of degree at most 5 inside elements). The CPU times obtained with the different time schemes considered are given on Table 5. For this particular case, the computational time is reduced by a factor near 5. This reduction is due to the fact that 80% of elements

<table>
<thead>
<tr>
<th>$k=4$</th>
<th>M1</th>
<th>M2</th>
<th>M3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e(\nu_k/1) - e(\nu \to 0)$</td>
<td>$5.95 \times 10^{-7}$</td>
<td>$2.83 \times 10^{-6}$</td>
<td>$8.13 \times 10^{-7}$</td>
</tr>
<tr>
<td>$e(\nu_k/2) - e(\nu \to 0)$</td>
<td>$1.44 \times 10^{-7}$</td>
<td>$6.19 \times 10^{-7}$</td>
<td>$1.09 \times 10^{-7}$</td>
</tr>
<tr>
<td>$e(\nu_k/4) - e(\nu \to 0)$</td>
<td>$3.15 \times 10^{-8}$</td>
<td>$9.30 \times 10^{-8}$</td>
<td>$5.00 \times 10^{-8}$</td>
</tr>
</tbody>
</table>

Table 3: Study of the time accuracy of Verlet method (based on the $L^2$-norm of the error at time $T = 1$); for a given mesh, the error behaves like $O(\Delta t^2)$. 

Figure 2: Time-step classes on the regular mesh (left) and the irregular mesh (right) of the unit square.
Table 4: $L^2$-norm of the error at time $T = 1$, using the $R^3(\Delta t)$ algorithm.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>$R^{11}(2.6,ms)$</th>
<th>$R^4(3.54,\mu s)$</th>
<th>leap-frog (6, 3.54,$\mu s$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU time</td>
<td>7958</td>
<td>58212</td>
<td>38808</td>
</tr>
<tr>
<td>Gain (vs. leap-frog)</td>
<td>4.88</td>
<td>0.67</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 5: Comparison of CPU times and gain between algorithms $R^{11}$, $R^4$, and a classical leap-frog implementation (6).
Figure 4: Square inclusion: $p/H_z$ (top), $u/E_y$ (middle), and $v/E_x$ (bottom) near the inclusion, obtained with algorithm $R^{11}(2.6ms)$ at $t = 4s$ (extremal values for contours on the zooms have been adapted).
are time-advanced at most 4 times per global time-step, but 11% of elements are time-advanced at most 512 times per global time-step (the refined zone of the mesh is more significant).

4.3 An illustration of the locally implicit algorithm

We give here an illustration for the locally implicit algorithm used jointly with a local time-stepping algorithm. A toy example has been composed by deforming the mesh shown on Figure 3. It is presented on Figure 5, where we have shown the implicit elements (each connected component must be isolated and we have contoured the number of this connected component for implicit elements) and more generally the class number of all elements. One can see that isolated implicit elements are present, which is not a concern at all. Implicit elements correspond to the class #1, and the explicit local time step increases with the class number for explicit elements. This sample mesh was not sufficient to produce a significant difference in CPU time. We then proposed a specially designed case: in the mesh of Figure 3, we have moved two neighboring vertices towards each other, such that their mutual distance has been divided by 100. We then have two very thin elements. The mesh obtained could be seen as a poor result of an automatic mesh generator. We then compared the behaviors of the fully explicit multi-scale algorithm and the implicit-explicit multi-scale algorithm. In both cases, the global time step, for an overall $P_4$-DGTD discretization, is $\Delta t = 1.19\, ms$. However:

- in the fully explicit multi-scale algorithm, the two thin elements are advanced in time with $\Delta t_i = 36.3\, ns = 1.19/2^{-15}\, ms$ (there are 16 classes in the computation);
- in the implicit/explicit multi-scale algorithm (where only these two-elements are dealt with implicitly), only ten classes are necessary; this computation required a very small storage and a CPU time with a reduction of 36%.

Figure 5: Deformed triangular mesh near the circular array: (number of the implicit connected zone (left) and number of class (right).
In the present case, the CPU time reduction is related to one defect in the mesh. This shows the implicit/explicit multi-scale algorithm can lead to important computational time reduction, especially in cases where defects in the mesh are present (small number of ridiculously small elements). Such small elements are not necessarily easy to get rid of, in particular slivers in unstructured tetrahedral meshes.

5 CONCLUSION

In this paper, we have presented two symplectic algorithms which are able to perform a reversible, energy-conserving, second-order accurate, stable, and adaptive time-integration of the Maxwell’s equations after discretization on unstructured meshes using the Discontinuous Galerkin method. The main conclusion is that, if totally centered numerical fluxes are to be used, in order to have no numerical dissipation at all, local time-stepping can overcome the stability limit set by the leapfrog time-scheme.

This kind of algorithm can be particularly valuable if the mesh is distorted or locally refined, i.e. the mesh is refined in a very limited area, for example around a geometrical detail. Two solutions have been proposed. The first one relies on a simple implicit/explicit coupled algorithm. The second one is a totally explicit algorithm, with no additional storage and leads to very efficient implementations, at least in two space dimensions.

Further works will deal with the implementation in three space dimensions, the latter being quite straightforward because the algorithms can be seen as time-step reorganizations only. The main difficult task will certainly consist in obtaining an efficient parallel implementation of these local time-stepping algorithm. In particular, mesh partitioning and message passing have to be optimized. Another promising research direction concerns the use of more than second-order accurate symplectic time schemes.

REFERENCES


