physica status solidi b
www.pss-b.com
basic solid state physics

Feature Article

Unconditionally stable perfectly matched layer boundary conditions

H. De Raedt[1] and **K. Michielsen**[2]

[1] Department of Applied Physics, Zernike Institute for Advanced Materials, University of Groningen, Nijenborgh 4, 9747 AG Groningen, The Netherlands
[2] European Marketing and Business Development (EMBD), Vlasakker 21, 2160 Wommelgem, Belgium

Received 5 April 2007, revised 4 June 2007, accepted 4 July 2007
Published online 15 August 2007

PACS 02.60.Cb, 03.50.De, 41.20.Jb

A brief review is given of a systematic, product-formula based approach to construct unconditionally stable algorithms for solving the time-dependent Maxwell equations. The fundamental difficulties that arise when we want to incorporate uniaxial perfectly matched layer boundary conditions into this scheme is discussed. We construct an algorithm that circumvents these difficulties and is unconditionally stable. Results of simulations for a point source and a plane current source inside a three-dimensional volume illustrate that in practice, the algorithm performs as theoretically expected.

phys. stat. sol. (b) **244**, No. 10, 3497–3505 (2007) / **DOI** 10.1002/pssb.200743148

phys. stat. sol. (b) **244**, No. 10, 3497–3505 (2007) / **DOI** 10.1002/pssb.200743148

Feature Article

Unconditionally stable perfectly matched layer boundary conditions

H. De Raedt[*, 1] and **K. Michielsen**[**, 2]

[1] Department of Applied Physics, Zernike Institute for Advanced Materials, University of Groningen, Nijenborgh 4, 9747 AG Groningen, The Netherlands
[2] European Marketing and Business Development (EMBD), Vlasakker 21, 2160 Wommelgem, Belgium

Received 5 April 2007, revised 4 June 2007, accepted 4 July 2007
Published online 15 August 2007

PACS 02.60.Cb, 03.50.De, 41.20.Jb

A brief review is given of a systematic, product-formula based approach to construct unconditionally stable algorithms for solving the time-dependent Maxwell equations. The fundamental difficulties that arise when we want to incorporate uniaxial perfectly matched layer boundary conditions into this scheme is discussed. We construct an algorithm that circumvents these difficulties and is unconditionally stable. Results of simulations for a point source and a plane current source inside a three-dimensional volume illustrate that in practice, the algorithm performs as theoretically expected.

1 Introduction

Finite-difference time-domain (FDTD) algorithms solve the time-dependent Maxwell equations [1] by replacing continuum space by a spatial grid [2]. In general, the discretized curl equations can be written in the compact form [3, 4]

$$\frac{\partial}{\partial t}\boldsymbol{Y}(t) = L\boldsymbol{Y}(t) - \boldsymbol{F}(t), \tag{1}$$

where the vector $\boldsymbol{Y}(t)$ is a shorthand for all the field variables on the grid, including the auxiliary field variables used to incorporate linear dispersive media characterized by e.g. Debye, Drude or Lorentz models [1, 2]. The matrix L is the discrete analogue of the operator that governs the time evolution of the fields and the vector $\boldsymbol{F}(t)$ describes the current source $\boldsymbol{J}(t)$. For instance, if the electromagnetic fields $\boldsymbol{E}(t)$ and $\boldsymbol{H}(t)$ interact with a material that is described by a Drude model with polarization vector $\boldsymbol{P}(t)$, the discretized Maxwell equations can be written in the form [4]

$$\frac{\partial}{\partial t}\begin{pmatrix}\boldsymbol{E}(t)\\ \boldsymbol{H}(t)\\ \boldsymbol{P}(t)\end{pmatrix} = \begin{pmatrix}0 & -c\hat{\nabla}\times & \boldsymbol{W}\\ c\hat{\nabla}\times & 0 & 0\\ -\boldsymbol{W} & 0 & \Gamma\end{pmatrix}\begin{pmatrix}\boldsymbol{E}(t)\\ \boldsymbol{H}(t)\\ \boldsymbol{P}(t)\end{pmatrix} - \begin{pmatrix}\boldsymbol{J}(t)\\ 0\\ 0\end{pmatrix}. \tag{2}$$

[*] Corresponding author: e-mail: h.a.de.raedt@rug.nl, url: www.compphys.net, Phone: +31 50 363 4852, Fax: +31 50 363 4947
[**] e-mail: kristel.michielsen@embd.be, url: www.embd.be

The matrices $\boldsymbol{W}$ and $\boldsymbol{G}$ in Eq. (2) are real and diagonal, with non-negative values that may depend on the position in the lattice and, if nonzero, define the frequency and inverse relaxation time of the Drude-pole model, respectively [4]. The symbol $\widehat{\nabla}\times$ denotes the discretized form of the curl operator. Assuming that the discretization procedure does not change the basic symmetries of the Maxwell equations, the precise form of $\widehat{\nabla}\times$, although very important for applications, is not essential for what follows. Hence, we will not discuss the important technicalities of the spatial discretization any further but refer the reader to Ref. [2].

The formal solution of Eq. (1) is given by

$$\boldsymbol{Y}(t) = U(t)\,\boldsymbol{Y}(0) - \int_0^t U(t-u)\,\boldsymbol{F}(u)\,\mathrm{d}u\,, \tag{3}$$

where the matrix exponential

$$U(t) = \mathrm{e}^{tL} \tag{4}$$

denotes the time-evolution matrix.

As can be seen from the example in Eq. (2), the matrix L is the sum of a skew-symmetric and a negative semi-definite matrix. For any skew-symmetric and negative semi-definite matrix L, we have [4]

$$\|\mathrm{e}^{tL}\| \le 1\,, \tag{5}$$

where $\|.\|$ denotes the 2-norm of a vector or matrix. Eq. (5) is the mathematical equivalent of the statement that the physical system is stable (does not explode) in time. In the absence of dissipation, L is skew symmetric, hence e^{tL} is an orthogonal matrix, and $\|\mathrm{e}^{tL}\| = 1$ expresses the fact that the total energy of the system, $\boldsymbol{Y}(t)^{\mathrm{T}} \cdot \boldsymbol{Y}(t)$, is a conserved quantity [4].

There are two, closely related, strategies to construct an algorithm for performing the time integration of equations such as Eq. (1) [5]. The traditional approach is to discretize (with increasing level of sophistication) the derivative with respect to time [5]. The other is to replace the matrix exponential $U(t) = \mathrm{e}^{tL}$ that appears in the formal solution Eq. (3) which, for real-life applications cannot be computed directly because of prohibitive memory requirements, by an approximate time evolution matrix $\tilde{U}(t)$ that can be handled numerically [5]. In this paper, we adopt the latter approach because it is well-suited to construct algorithms that, by construction, preserve specific symmetries.

2 Product formula approach

A systematic approach to construct approximations to matrix exponentials is to make use of the Lie–Trotter–Suzuki product formula [6–8],

$$\mathrm{e}^{tL} = \mathrm{e}^{t(L_1+\ldots+L_p)} = \lim_{m\to\infty}\left(\prod_{i=1}^{p}\mathrm{e}^{tL_i/m}\right)^m\,, \tag{6}$$

and generalizations thereof [9–11]. Expression Eq. (6) suggests that

$$U_1(\tau) = \mathrm{e}^{\tau L_1}\ldots\mathrm{e}^{\tau L_p} \tag{7}$$

might be a good approximation to $U(\tau)$ if τ is sufficiently small. The Taylor series expansion of $U(\tau)$ and $U_1(\tau)$ shows that $U(\tau)$ and $U_1(\tau)$ are identical up to first order in τ, hence we call $U_1(\tau)$ a first-order approximation to $U(\tau)$.

The product-formula framework provides simple, systematic procedures to improve the accuracy of the approximation to $U(\tau)$ without changing its fundamental symmetries. For example, the matrix

$$U_2(\tau) = U_1(-\tau/2)^{\mathrm{T}}\,U_1(\tau/2) = \mathrm{e}^{\tau L_p/2}\ldots\mathrm{e}^{\tau L_1/2}\,\mathrm{e}^{\tau L_1/2}\ldots\mathrm{e}^{\tau L_p/2} \tag{8}$$

is a second-order approximation to $U(\tau)$ that, by construction, is unconditionally stable whenever $U_1(\tau)$ is unconditionally stable. For arbitrary matrices L_i, it can be shown that [4]

$$||U(t=m\tau)-[U_2(\tau)]^m|| \le c_2\tau^2 t\, \mathrm{e}^{t(\rho(L_1)+\ldots+\rho(L_p))}, \tag{9}$$

where c_2 is a positive number and $\rho(X)$ denotes the largest eigenvalue of $(X+X^T)/2$.

By definition, a numerical scheme that uses a matrix $V(\tau)$ to advance the vector $\boldsymbol{Y}(t)$ in time according to $\boldsymbol{Y}(t+\tau)=V(\tau)\boldsymbol{Y}(t)$ is unconditionally stable if $||V(\tau)|| \le 1$ [5]. As $||U_1(\tau)|| \le ||\mathrm{e}^{\tau L_1}|| \ldots ||\mathrm{e}^{\tau L_p}|| \le \mathrm{e}^{\tau(\rho(L_1)+\ldots+\rho(L_p))} \le 1$, the algorithm defined by Eq. (7) will be unconditionally stable by construction if $\rho(L_i) \le 0$ for $i=1,\ldots,p$. Note that for a numerical algorithm to be unconditionally stable, it is necessary but not sufficient to satisfy the von Neumann stability condition [5, 12]. If each L_i can be written as the sum of a skew-symmetric and a negative semi-definite matrix, then $\rho(L_i)=0$ for $i=1,\ldots,p$ and the last factor in Eq. (9) is equal to one.

Suzuki's fractal decomposition approach [10, 11] gives a general method to construct higher-order approximations based on $U_1(\tau)$. A particularly useful fourth-order approximation is given by [10, 11]

$$U_4(\tau)=U_2(a\tau)\,U_2(a\tau)\,U_2((1-4a)\tau)\,U_2(a\tau)\,U_2(a\tau), \tag{10}$$

where $a=1/(4-4^{1/3})$. The approximations Eqs. (7), (8) and (10) have proven to be very useful in many applications, see the references in Ref. [13].

In FDTD applications, the approximation $U_2(\tau)$ is used to advance the vector $\boldsymbol{Y}(t)$ in time that is, we use $\boldsymbol{Y}(t+\tau)=U_2(\tau)\boldsymbol{Y}(t)$ to update the fields. Note that up to this point, we have not made any specific choice for the number p and for the matrices that appear in the decomposition

$$L=L_1+\ldots+L_p. \tag{11}$$

Equation (9) sets a limit on how the error due to the time-integration scheme (that is, disregarding the error due to the spatial discretization) increases with τ and t. As usual with this kind of theoretical bounds, in practice they are often very pessimistic because c_2 has to be large enough to include the worst case. However, our experience shows that the functional dependence on τ and t is often useful to detect programming mistakes.

In practice, an efficient implementation of the first-order scheme $U_1(\tau)$ is all that is needed to construct efficient higher-order algorithms based on Eqs. (8) and (10). For these time-stepping algorithms to be computationally efficient, all the matrix exponentials that appear in Eq. (7) should be sparse and it should be easy to calculate (analytically) the expressions for the matrix elements of the matrix exponentials $\mathrm{e}^{\tau L_1},\ldots,\mathrm{e}^{\tau L_p}$. Elsewhere, [4, 13] we have shown that the workhorse of FDTD algorithms, the Yee algorithm [2], the alternating-direction-implicit (ADI) time-stepping FDTD methods [2], the Chebyshev polynomial based integrators [14], and a family of unconditionally stable algorithms [3] can all be derived from specific approximations to the matrix exponential e^{tL}. An extension to linear dispersive media was given in Ref. [2]. A first attempt to include in this approach, uniaxial perfectly media layers for 2D systems was reported in Ref. [2]. In the remainder of this paper, we show that in general, uniaxial perfectly media layers can be incorporated in the unified product-formula framework, allowing us to derive algorithms that are unconditionally stable by construction.

3 Uniaxial perfectly matched layers

In practice, any FDTD application faces the problem that the simulation volume is finite. Moreover, for reasons of economy, one would like to use a simulation volume that is as small as possible. Ideally, the computation domain should be large enough to contain the material objects but should not be much larger. This can, in principle, be realized if one can define boundary conditions on the faces of the simula-

tion box such that waves that impinge on these faces are not reflected into the simulation volume. In view of its importance to essentially all real-life applications, there exists a large body of work on this particular aspect of the FDTD method [2].

Of particular interest to us are the perfectly-matched-layer absorbing boundary conditions, implemented by means of a uniaxial material [15]. That the latter has the potential to be incorporated into an unconditionally stable algorithm follows directly from our earlier discussion: If the medium is a physical, linear dissipative material, the time evolution matrix satisfies Eq. (5), suggesting that it should be possible to construct an unconditionally stable algorithm. For detailed mathematical analyses of the stability of other implementations of perfectly-matched-layer absorbing boundary conditions see Refs. [12, 16].

In the remainder of this paper, we consider the case of an empty simulation box surrounded by uniaxial perfectly matched layer (UPML) material only. Due to the modular structure of the product-formula approach, Debye, Drude or Lorentz models for the material object can easily be incorporated into the algorithm. For simplicity of notation, we adopt units such that the permittivity and permeability of empty space are both equal to one.

Following Ref. [15], in this particular case, the Maxwell curl equation in the frequency domain can be written as [15]

$$\begin{aligned} j\omega \boldsymbol{S}\boldsymbol{E} &= \nabla \times \boldsymbol{H}, \\ j\omega \boldsymbol{S}\boldsymbol{H} &= -\nabla \times \boldsymbol{E}, \end{aligned} \tag{12}$$

where $j \equiv \sqrt{-1}$ and $\boldsymbol{S}$ is the material tensor defined by

$$\boldsymbol{S} = \begin{pmatrix} s_x^{-1} s_y s_z & 0 & 0 \\ 0 & s_x s_y^{-1} s_z & 0 \\ 0 & 0 & s_x s_y s_z^{-1} \end{pmatrix}, \tag{13}$$

where, again for simplicity of presentation, we take $s_x = 1 + \sigma_x/j\omega$, $s_y = 1 + \sigma_y/j\omega$, and $s_z = 1 + \sigma_z/j\omega$ [15]. To treat the UPML problem in full generality, we consider the case of a trihedral corner region for which $\sigma_x > 0$, $\sigma_y > 0$ and $\sigma_z > 0$ [15]. The other, simpler cases can be obtained by setting one, two, or all three σ's equal to zero [15].

We start by noting that

$$j\omega s_x^{-1} s_y s_z = j\omega + (\sigma_y + \sigma_z - \sigma_x) + \frac{\sigma_x^2 + \sigma_y \sigma_z}{j\omega + \sigma_x} - \frac{\sigma_x \sigma_y + \sigma_x \sigma_z}{j\omega + \sigma_x}. \tag{14}$$

Using Eq. (14) and the standard trick to introduce auxiliary variables [2], we find from Eq. (12) that the equation for E_x in the time-domain can be written as

$$\begin{aligned} \frac{\partial E_x}{\partial t} &= \frac{\partial H_z}{\partial y} - \frac{\partial H_y}{\partial z} + (\sigma_x - \sigma_y - \sigma_z) E_x - \sqrt{\sigma_x^2 + \sigma_y \sigma_z} U_x + \sqrt{\sigma_x \sigma_y + \sigma_x \sigma_z} V_x, \\ \frac{\partial U_x}{\partial t} &= -\sigma_x U_x + \sqrt{\sigma_x^2 + \sigma_y \sigma_z} E_x, \\ \frac{\partial V_x}{\partial t} &= -\sigma_x V_x + \sqrt{\sigma_x \sigma_y + \sigma_x \sigma_z} E_x, \end{aligned} \tag{15}$$

where it is understood that the fields $\{E, H, U, V\}$ and $\{\sigma_x, \sigma_y, \sigma_z\}$ are functions of (x, y, z, t) and (x, y, z), respectively. The equations for the other E-components and the H-fields have the same structure and can be obtained by interchanges of the x-, y-, z-labels, and/or the E- and H-components. Of

course, for each E- or H-component, we have to introduce two auxiliary U and V fields. From Eq. (15), we see that (after proper discretization of the curl) the matrix L that describes the time evolution is a sum of the usual skew-symmetric matrix representing the matrix for the wave propagation in vacuum, another skew-symmetric matrix that couples E_x and U_x, a symmetric matrix that couples E_x and V_x, and a diagonal matrix, one element of which can take both negative and positive values.

Let us now naively apply the product-formula philosophy and decompose L, as defined by Eq. (15) and similar equations for the other two E-fields and three H-fields. We split L into the skew-symmetric matrix L_0 that describes the wave propagation in vacuum and a matrix L_1 that describes the coupling between the $\{E, H\}$-fields and the auxiliary variables that represent the UPML. L_1 is a block diagonal matrix, consisting of 3×3 matrices that act on field components that are located on the same vertex of the (Yee) grid. For instance, for the fields in Eq. (15), the 3×3 matrix for the grid point (x, y, z) is given by

$$L_1(x, y, z) = \begin{pmatrix} \sigma_x - \sigma_y - \sigma_z & -\sqrt{\sigma_x^2 + \sigma_y \sigma_z} & +\sqrt{\sigma_x \sigma_y + \sigma_x \sigma_z} \\ +\sqrt{\sigma_x^2 + \sigma_y \sigma_z} & -\sigma_x & 0 \\ +\sqrt{\sigma_x \sigma_y + \sigma_x \sigma_z} & 0 & -\sigma_x \end{pmatrix}. \tag{16}$$

The matrix exponential of a block diagonal matrix of submatrices is itself a block diagonal matrix of the matrix exponentials of the submatrices. Hence it can be written as a product of (commuting) matrix exponentials of a direct product of a unit matrix of the full problem dimension and a 3×3 matrix (this is essential for the simulation algorithm to be computationally efficient). Therefore, to analyze the stability of the scheme, it suffices to concentrate on the matrix exponential of $L_1(x,y,z)$. A simple calculation shows that the largest eigenvalue of $(L_1(x,y,z) + L_1^T(x,y,z))/2$ is given by

$$\rho(L_1(x,y,z)) = -\frac{\sigma_y + \sigma_z}{2} + \sqrt{\sigma_x^2 + \left(\frac{\sigma_y + \sigma_z}{2}\right)^2} \geq 0, \tag{17}$$

indicating that this decomposition does not satisfy the condition for an unconditionally stable algorithm if $\sigma_x > 0$. Of course, as this conclusion follows from a specific choice of the decomposition for L, we cannot rule out that there may exist other decompositions that yield unconditionally stable algorithms. We now argue that in a finite system, such decompositions do not exist.

To see this, we return to the original derivation of the conditions for a reflectionless interface between an empty half space ($\times < 0$) and a UPML filling the other half space ($x > 0$) [15]. From the Maxwell equations Eq. (12), it follows immediately that the dispersion relation for the waves in the UPML region are given by

$$\omega^2 = k_y^2 + k_z^2 + \frac{k_x^2}{1 + \sigma_x / j\omega}. \tag{18}$$

Restricting our attention to a wave that propagates in the x-direction, we set $k_y = k_z = 0$. We find that Eq. (18) admits two solutions: $k_x = q_x(1 + \sigma_x/j\omega)$ and $k_x = -q_x(1 + \sigma_x/j\omega)$. The former solution describes a wave ($e^{-jk_x x}$, we adopt the convention of Ref. [2]) that decays as the wave moves away from the interface at $x = 0$, further into the UPML. The latter solution describes a wave that grows exponentially as it moves towards the vacuum region. In the theoretical treatment, the latter solution is eliminated through the obviously correct physical argument that initially, there were no waves moving from the UPML region towards the vacuum region [15, 16]. However, the time evolution matrix L does not know about this boundary condition, and it allows for exponentially growing waves to propagate. Moreover, in a numerical simulation, the UPMLs are finite. Rounding errors and waves that reach the boundary of the simulation box, even though their amplitude may be very small, both act as tiny sources of noise. The numerical algorithm may and, as a rule of thumb it will, pick up this noise and will amplify it, simply be-

cause the eigenvalue problem of the time evolution matrix admits exponentially growing solutions. Fortunately, knowing the origin of the instability, it is a simple matter to eliminate the numerical instability.

Introducing $\sigma = \sigma(x,y,z) = \sigma_x + \sigma_y + \sigma_z$ and the new fields $\tilde{E}_x = \mathrm{e}^{-\sigma t} E_x$, $\tilde{U}_x = \mathrm{e}^{-\sigma t} U_x$ and $\tilde{V}_x = \mathrm{e}^{-\sigma t} V_x$ (and similarly for all other $\{E, H, U, V\}$ components), Eq. (15) transforms into

$$\begin{aligned}
\frac{\partial \tilde{E}_x}{\partial t} &= \frac{\partial \tilde{H}_z}{\partial y} - \frac{\partial \tilde{H}_y}{\partial z} + t\,\frac{\partial \sigma}{\partial y}\,\tilde{H}_z - t\,\frac{\partial \sigma}{\partial z}\,\tilde{H}_y - 2(\sigma_y + \sigma_z)\,\tilde{E}_x \\
&\quad -\sqrt{\sigma_x^2 + \sigma_y\sigma_z}\,\tilde{U}_x + \sqrt{\sigma_x\sigma_y + \sigma_x\sigma_z}\,\tilde{V}_x\,, \\
\frac{\partial \tilde{U}_x}{\partial t} &= -(2\sigma_x + \sigma_y + \sigma_z)\,\tilde{U}_x + \sqrt{\sigma_x^2 + \sigma_y\sigma_z}\,\tilde{E}_x\,, \\
\frac{\partial \tilde{V}_x}{\partial t} &= -(2\sigma_x + \sigma_y + \sigma_z)\,\tilde{V}_x + \sqrt{\sigma_x\sigma_y + \sigma_x\sigma_z}\,\tilde{E}_x\,.
\end{aligned} \tag{19}$$

The 3 × 3 matrices that appear in the decomposition that we introduced earlier read

$$\tilde{L}_1(x,y,z) = \begin{pmatrix} -2(\sigma_y + \sigma_z) & -\sqrt{\sigma_x^2 + \sigma_y\sigma_z} & +\sqrt{\sigma_x\sigma_y + \sigma_x\sigma_z} \\ +\sqrt{\sigma_x^2 + \sigma_y\sigma_z} & -(2\sigma_x + \sigma_y + \sigma_z) & 0 \\ +\sqrt{\sigma_x\sigma_y + \sigma_x\sigma_z} & 0 & -(2\sigma_x + \sigma_y + \sigma_z) \end{pmatrix}. \tag{20}$$

The largest eigenvalue of $(\tilde{L}_1(x, y, z) + \tilde{L}_1^T(x, y, z))/2$ is given by

$$\rho(\tilde{L}_1(x, y, z)) = -\frac{4(\sigma_y + \sigma_z)^2 + 6\sigma_x(\sigma_y + \sigma_z)}{2\sigma_x + 3\sigma_y + 3\sigma_z + \sqrt{4\sigma_x^2 + (\sigma_y + \sigma_z)^2}} \le 0\,, \tag{21}$$

showing that the matrix exponential of these matrices cannot give rise to numerical instabilities.

Note that $\sigma > 0$ in the UPML region only. Hence in the relevant simulation region, the fields are not affected by the transformation, that is $\tilde{E}_x = E_x$ for all points outside the UPML region. However, the transformation to the tilde-fields introduced two explicit time-dependent terms in the equation for E_x (see the first equation of Eq. (19)). Explicit time-dependencies such as the one appearing in Eq. (19) are easily dealt with in the product-formula approach [17]. However, in the present case, we may *define* a new system of equations by removing the terms that contain the derivative of σ with respect to the coordinates. In fact, in the derivation of the conditions of a reflectionless UPML, it is assumed that σ is piecewise constant, hence except at the interface itself, the terms with the explicit time dependence vanish. Evidently, removing the terms in Eq. (19) that contain the derivative of σ with respect to the coordinates does not change the equations for the fields in the region of interest and the resulting system of equations will be free of instabilities.

In practice, the implementation of the UPML boundary condition as defined by Eq. (19) and similar equations for all the other $\{E, H, U, V\}$ components requires repeated multiplications of a vector of three elements with the matrix exponential of $\tilde{L}_1$. Obviously, we have $\mathrm{e}^{-\tau\sigma}\mathrm{e}^{\tau L_1} = \mathrm{e}^{\tau \tilde{L}_1}$ and the general, closed-form expressions of the matrix elements of the latter are given by

$$(\mathrm{e}^{\tau\tilde{L}_1})_{1,1} = \frac{\mathrm{e}^{-\tau\sigma_y}(\sigma_y - \sigma_x) + \mathrm{e}^{-\tau\sigma_z}(\sigma_x - \sigma_z)}{\sigma_y - \sigma_z}\,,$$

$$(\mathrm{e}^{\tau\tilde{L}_1})_{2,1} = -(\mathrm{e}^{\tau\tilde{L}_1})_{1,2} = \frac{\mathrm{e}^{-\tau\sigma_z} - \mathrm{e}^{-\tau\sigma_y}}{\sigma_y - \sigma_z}\sqrt{\sigma_x^2 + \sigma_y\sigma_z}\,,$$

$$(e^{\tau\tilde{L}_1})_{3,1} = +(e^{\tau\tilde{L}_1})_{1,3} = \frac{e^{-\tau\sigma_z} - e^{-\tau\sigma_y}}{\sigma_y - \sigma_z}\sqrt{\sigma_x(\sigma_y + \sigma_z)},$$

$$(e^{\tau\tilde{L}_1})_{2,2} = \frac{e^{-\tau\sigma_x}(\sigma_z^2 - \sigma_y^2)\,\sigma_x + [e^{-\tau\sigma_z}(\sigma_y - \sigma_x) + e^{-\tau\sigma_y}(\sigma_x - \sigma_z)](\sigma_x^2 + \sigma_y\sigma_z)}{(\sigma_x - \sigma_y)(\sigma_x - \sigma_z)(\sigma_y - \sigma_z)},$$

$$(e^{\tau\tilde{L}_1})_{3,2} = -(e^{\tau\tilde{L}_1})_{3,2}$$

$$= \frac{e^{-\tau\sigma_z}(\sigma_y - \sigma_x) + e^{-\tau\sigma_y}(\sigma_x - \sigma_z) + e^{-\tau\sigma_x}(\sigma_z - \sigma_y)}{(\sigma_x - \sigma_y)(\sigma_x - \sigma_z)(\sigma_y - \sigma_z)}\sqrt{\sigma_x(\sigma_y + \sigma_z)(\sigma_x^2 + \sigma_y\sigma_z)},$$

$$(e^{\tau\tilde{L}_1})_{3,3} = \frac{e^{-\tau\sigma_y}\sigma_x(\sigma_y + \sigma_z)}{(\sigma_y - \sigma_x)(\sigma_y - \sigma_z)} + \frac{e^{-\tau\sigma_z}\sigma_x(\sigma_y + \sigma_z)}{(\sigma_z - \sigma_x)(\sigma_z - \sigma_y)} + \frac{e^{-\tau\sigma_x}(\sigma_x^2 + \sigma_y\sigma_z)}{(\sigma_x - \sigma_y)(\sigma_x - \sigma_z)}. \quad (22)$$

4 Simulation results

The stabilized UPML equations derived in Section 3 have been implemented in the massively parallel FDTD Maxwell solver developed by EMBD. In this section, we use EMBD's Maxwell solver to demonstrate that the stabilized UPML performs as expected on theoretical grounds.

As with all implementations of perfectly matched layer absorbing boundaries, there are a number of important technicalities that have to be taken into account, the main reason being that the dispersion relation for the waves on the Yee grid are different from the one in the continuum [15]. One trick to mitigate the effects of the spatial grid is to let the loss parameters σ_x, σ_y, or σ_z increase gradually as we move further into the UPML region [15]. For maximum performance of the UPML, the thickness of the UPML layer as well as the values of σ_x, σ_y, and σ_z have to be chosen judiciously [15]. The results presented in this paper have been obtained by adopting the fourth-order polynomial grading [15]. The maximum value of σ_x, σ_y and σ_z was chosen to be five (in units of $c\lambda$).

In Fig. 1, we show simulation results for the total electromagnetic energy

$$W(t) = \sum_{\{i,j,k\}\in\mathcal{E}} \sum_{\alpha=x,y,z} (E_\alpha^2(i,j,k,t) + H_\alpha^2(i,j,k,t)), \quad (23)$$

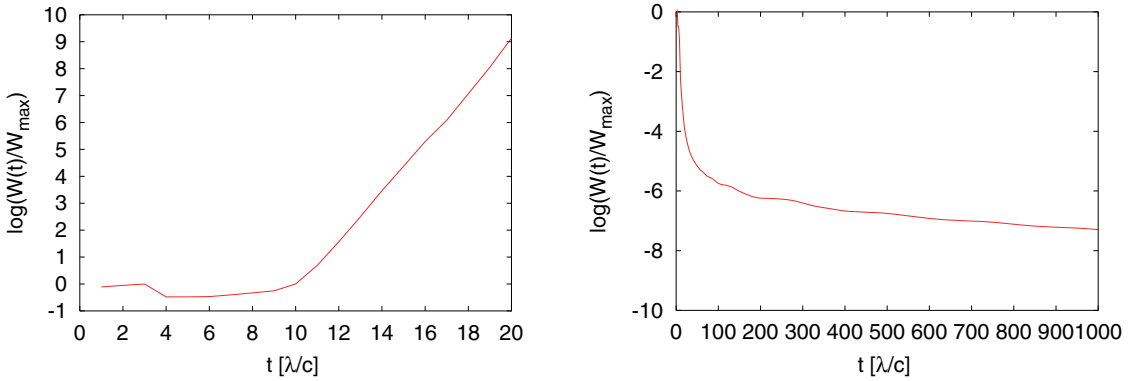

Fig. 1 (online colour at: www.pss-b.com) Normalized total electromagnetic energy $W(t)/W_{\max}$ in the vacuum region as a function of time for a point source, located in the center of the simulation box, that is turned off at $t = T = 4\lambda/c$. The number of mesh points per wavelength λ is 10, the size of the simulation box is 94 × 94 × 94 mesh points, the time step is $0.045\lambda/c$, and the number of mesh points of the UPML (for each of the three directions) is 13. Left: Algorithm that implements the UPML through the use of the matrices L_1. The value of $W_{\max}$ is the same as the one used in Fig. 1(right). Right: Stable algorithm that implements the UPML through the use of the matrices $\tilde{L}_1$.

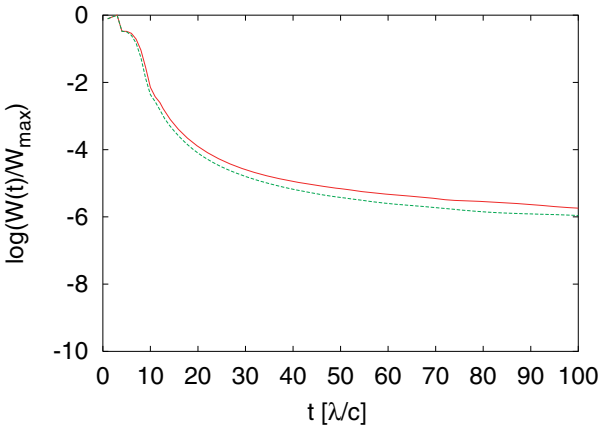

Fig. 2 (online colour at: www.pss-b.com) Normalized total electromagnetic energy $W(t)/W_{\max}$ in the vacuum region as a function of time for a point source, located in the center of the simulation box, that is turned off at $t = T = 4\lambda/c$. The number of mesh points per wavelength λ is 10, the size of the simulation box is $94 \times 94 \times 94$ mesh points, the time step is $0.045\lambda/c$, and the number of mesh points of the UPML (for each of the three directions) is 13 (solid line) and 20 (dashed line), respectively.

in the empty space $\mathcal{E}$, surrounded by UPMLs. Initially, $W(t=0)=0$ and the electromagnetic energy is injected into the system through a current (point) source

$$\boldsymbol{J}(\boldsymbol{r},t) = \delta(\boldsymbol{r}-\boldsymbol{r}_0)\,\hat{\boldsymbol{x}}\,\Theta(t)\,\Theta(T-t)\sin\frac{2\pi tc}{\lambda}, \tag{24}$$

where $\boldsymbol{r}_0$ is the vector to the center of the simulation volume, $\hat{\boldsymbol{x}}$ is the unit vector in the x-direction, $\Theta(t)$ is the Heaviside step function, and the wavelength λ fixes the length scale. In our simulations, the current source is switched on at $t=0$ and is turned off at $T = 4\lambda/c$ and we monitor the energy $W(t)$ as a function of time. If the UMPL works well, we expect that $W(t)$ decreases monotonically for $t > T$. From

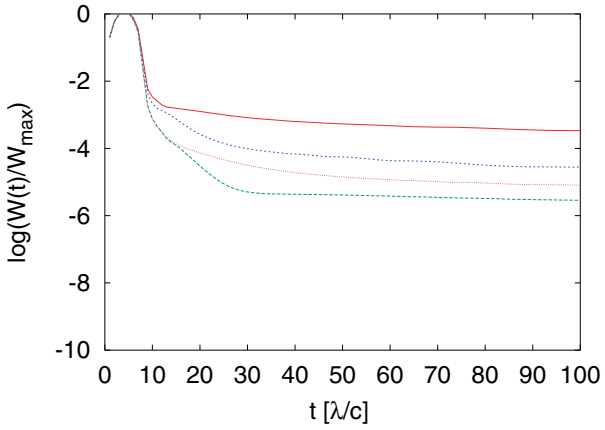

Fig. 3 (online colour at: www.pss-b.com) Normalized total electromagnetic energy $W(t)/W_{\max}$ in the vacuum region as a function of time for a plane current source, located in the middle of the simulation box, that is turned off at $t = T = 4\lambda/c$. Solid line (red): The number of mesh points per wavelength λ is 10, the size of the simulation box is $94 \times 94 \times 94$ mesh points, the time step is $0.0058\lambda/c$, and the number of mesh points of the UPML (for each of the three directions) is 13. Long dashes (green): The number of mesh points per wavelength λ is 20, the size of the simulation box is $187 \times 187 \times 187$ mesh points, the time step is $0.0029\lambda/c$, and the number of mesh points of the UPML (for each of the three directions) is 25. Short dashes (blue): The number of mesh points per wavelength λ is 10, the size of the simulation box is $94 \times 94 \times 94$ mesh points, the time step is $0.045\lambda/c$, and the number of mesh points of the UPML (for each of the three directions) is 13. Dots (magenta): The number of mesh points per wavelength λ is 20, the size of the simulation box is $187 \times 187 \times 187$ mesh points, the time step is $0.023\lambda/c$, and the number of mesh points of the UPML (for each of the three directions) is 25.

Fig. 1 (right), we conclude that the unconditionally stable UPML works as expected on theoretical grounds: There is no indication of an instability (compare to Fig. 1 (left)).

In all other respects, the stabilized UPML equations derived in Section 3 behave in the same manner as the other UPML implementations [15]. For instance, one can reduce the reflection at the UPML interface by increasing the UPML layer thickness (see Fig. 2), or by reducing the mesh size (see Fig. 3). For the latter, we used a current source given by

$$\boldsymbol{J}(\boldsymbol{r},t) = \delta(z - z_0)\, \hat{\boldsymbol{x}}\, \Theta(t)\, \Theta(T - t) \sin \frac{2\pi t c}{\lambda}, \tag{25}$$

that is, there is a current on each point of the plane defined by $z = z_0$, z_0 denoting the middle of the simulation box in the z-direction. Of course, improving the performance of the UPML absorbing boundary condition comes at the cost of increased memory and CPU time. In the test results that we have presented here, no effort has been made to achieve the best performance.

References

[1] M. Born and E. Wolf, Principles of Optics (Pergamon, Oxford, 1964).
[2] A. Taflove and S. C. Hagness, Computational Electrodynamics – The Finite-Difference Time-Domain Method (Artech House, Boston, 2005).
[3] J. S. Kole, M. T. Figge, and H. De Raedt, Phys. Rev. E **64**, 066705 (2001).
[4] Ref. [2], Chapter 18.
[5] G. D. Smith, Numerical solution of partial differential equations (Clarendon Press, Oxford, 1985).
[6] H. F. Trotter, Proc. Am. Math. Soc. **10**, 545 (1959).
[7] M. Suzuki, S. Miyashita, and A. Kuroda, Prog. Theor. Phys. **58**, 1377 (1977).
[8] H. De Raedt, Comput. Phys. Rep. **7**, 1 (1987).
[9] H. De Raedt and B. De Raedt, Phys. Rev. A **28**, 3575 (1983).
[10] M. Suzuki, J. Math. Phys. **26**, 601 (1985).
[11] M. Suzuki, J. Math. Phys. **32**, 400 (1991).
[12] S. Abarbnel and D. Gottlieb, J. Comput. Phys. **134**, 357 (1997).
[13] H. De Raedt, J. S. Kole, K. F. L. Michielsen, and M. T. Figge, Comput. Phys. Commun. **156**, 43 (2003).
[14] H. De Raedt, K. F. L. Michielsen, J. S. Kole, and M. T. Figge, IEEE Trans. Antennas Propag. **51**, 3155 (2003).
[15] Ref. [2], Chapter 7.
[16] E. Béache, P. G. Petropoulos, and S. D. Gedney, IEEE Trans. Antennas Propag. **52**, 1335 (2004).
[17] H. De Raedt and K. Michielsen, in: Handbook of Theoretical and Computational Nanotechnology, Vol. 3: Quantum and molecular computing, quantum simulations, edited by M. Rieth and W. Schommers (American Scientific Publishers, Los Angeles, 2006), Chapter 1, pp. 2–48.

