Notes on the UPML implementation in Meep

Steven G. Johnson

Posted August 17, 2009; updated March 10, 2010

1 Overview

In this note, we go over the implementation of UPML absorbing boundary layers in the free Meep FDTD package. It is assumed that the reader is already familiar with the basic stretched-coordinate derivation of PML (see `http://math.mit.edu/~stevenj/18.369/pml.pdf`), as well as the general technique by which a coordinate transformation can be expressed as a transformation of ε and μ (see `http://math.mit.edu/~stevenj/18.369/coordinate-transform.pdf`), leading to the so-called "UPML" formulation. The focus here is on how these transformed materials are expressed in Meep's FDTD algorithm. The subtlety is that the transformations/materials of PML are frequency-dependent, and so to express them in time domain involves the evolution of appropriate auxiliary differential equations. (Equivalently, multiplication by a frequency-dependent susceptibility corresponds in the time domain to a convolution, leading to so-called "convolutional PML" formulations.) The trick is to keep the number of auxiliary differential equations (and the resulting memory and computational costs) to a minimum, while not making the PML region too complicated (or require too much new code!) compared to the non-PML regions.

Because the treatment of ε and μ are identical except for interchange of $\vec{D}$ with $\vec{B}$ and $\vec{E}$ with $\vec{H}$ (and a sign flip from Ampere's to Faraday's law), we only describe the ε and $d\vec{D}/dt$ equations (Ampere's law) here.

We proceed slightly differently from the UPML as derived in e.g. the Taflove and Hagness FDTD textbook. As reviewed in the appendix, this "standard" UPML performs a matrix factorization that relies on ε commuting with the PML Jacobian, and this is not the case for an arbitrary anisotropic ε. Our factorization, instead, works for anisotropic ε, and turns out to have a nice property: the PML just adds two auxiliary fields $\vec{U}$ and $\vec{W}$ with *diagonal* relationships to $\vec{D}$ and $\vec{E}$, and allow us to use the non-PML timestepping operations *unchanged* except for the addition of these diagonal ODE updates. In particular, we get a true PML for anisotropic, dispersive, media "for free" in the sense that the code for those portions is unchanged (although the case of conducting media necessitates an extra auxiliary field). This is nice because the timestepping for anisotropic media requires special care for stability, and dispersive media may have complicated polarization-update equations, and this way we don't need to

modify any of that in the slightest for the PML. Our approach also makes a clean separation between the $\vec{D}$ update equation (from $\nabla \times \vec{H}$) and terms that affect the $\vec{E}$ update equation (from the constitutive equations), and correspondingly in the Meep code these are handled separately in `step_db` and `update_eh`, respectively.

2 Non-PML materials in Meep

In Meep, we support dispersive media $\varepsilon(\omega)$ of the form:

$$\varepsilon(\omega) = \left(1 + \frac{i\sigma_D}{\omega}\right)\left[\varepsilon_\infty + \chi_E(\omega)\right],$$

where σ_D is a conductivity (which may be anisotropic, but must be diagonal in Cartesian coordinates and hence commutes with $\mathcal{J}$), ε_∞ is a non-dispersive part of the permittivity (which may be non-diagonal anisotropic) and $\chi_E(\omega)$ is some additional dispersive part (possibly anisotropic) implemented by auxiliary ODEs that solve $\vec{P} = \chi(\omega)\vec{E}$ (currently, a sum of Lorentzians). This corresponds, in time-domain, to Ampere's law of the form:

$$\vec{K} = \nabla \times \vec{H} = \frac{\partial \vec{D}}{\partial t} + \sigma_D \vec{D}, \tag{1}$$

denoting $\nabla \times \vec{H}$ by $\vec{K}$ for later convenience, and a constitutive equation

$$\vec{D} = \varepsilon_\infty \vec{E} + \vec{P},$$

where $\vec{P}$ is time-evolved via a system of ODEs derived from $\vec{P} = \chi(\omega)\vec{E}$. (We don't include free currents $\vec{J}$ here, since they have no impact on the PML equations, and indeed one rarely puts free currents inside the PML anyway.)

In FDTD, we discretize these in space and time. Let us denote the time discretization by a superscript: $\vec{D}^n$ denotes $\vec{D}(n\Delta t)$, and similarly for $\vec{P}^n$ and $\vec{E}^n$. The magnetic fields are offset in time by half a time step, giving $\vec{H}^{n+0.5}$ and $\vec{K}^{n+0.5}$. To time-step these fields in FDTD, we first compute $\vec{D}^{n+1}$ from $\vec{D}^n$ and $\vec{K}^{n+0.5}$ by the curl equation, then compute $\vec{E}^{n+1}$ from $\vec{D}^{n+1}$ and $\vec{P}^{n+1}$ by the constitutive equation, and finally compute $\vec{P}^{n+2}$. (Note that, with Lorentzian dispersion, we can compute $\vec{P}^{n+2}$ from $\vec{P}^{n+1}$ and $\vec{P}^n$ given $\vec{E}^{n+1}$.

Using the standard second-order center-difference approximations, the equation for $\vec{D}$ becomes:

$$\vec{K}^{n+0.5} = \frac{\vec{D}^{n+1} - \vec{D}^n}{\Delta t} + \sigma_D \frac{\vec{D}^{n+1} + \vec{D}^n}{2},$$

giving

$$\vec{D}^{n+1} = \left(1 + \frac{\sigma_D \Delta t}{2}\right)^{-1} \left[\left(1 - \frac{\sigma_D \Delta t}{2}\right)\vec{D}^n + \vec{K}^{n+0.5}\right]. \tag{2}$$

The constitutive equation is simply:

$$\vec{E}^{n+1} = \varepsilon_\infty^{-1} \left(\vec{D}^{n+1} - \vec{P}^{n+1} \right). \tag{3}$$

(This is nontrivial to implement correctly for anisotropic media [Werner and Cary, 2009], but the details need not concern us here.)

3 PML formulation in frequency domain

PML is simplest to derive in frequency domain, where the fields all have time-dependence $e^{-i\omega t}$. An ordinary PML in Cartesian coordinates is derived by a complex coordinate stretching, where each coordinate is stretched by a factor

$$s_{x,y,z} = 1 + \frac{i\sigma_{x,y,z}}{\omega},$$

where ω is the frequency and σ is the PML "conductivity." For example, to terminate the cell in the x direction, only σ_x is nonzero. These coordinate stretchings can be absorbed into Maxwell's equations as a change in ε and μ. The original permittivity ε in the PML region is replaced by an effective tensor $\tilde{\varepsilon}$ given by

$$\tilde{\varepsilon} = \frac{\mathcal{J}\varepsilon\mathcal{J}^{\mathrm{T}}}{\det \mathcal{J}},$$

where $\mathcal{J} = \mathrm{diag}(s_x^{-1}, s_y^{-1}, s_z^{-1})$ is the Jacobian matrix of the coordinate stretching.

In frequency domain, replacint $\varepsilon(\omega)$ with $\tilde{\varepsilon}$, eq. (1) becomes:

$$\vec{K} = \nabla \times \vec{H} = -i\omega s_x s_y s_z \left(1 + \frac{i\sigma_D}{\omega}\right) \begin{pmatrix} s_x^{-1} & & \\ & s_y^{-1} & \\ & & s_z^{-1} \end{pmatrix} [\varepsilon_\infty + \chi_E(\omega)] \begin{pmatrix} s_x^{-1} & & \\ & s_y^{-1} & \\ & & s_z^{-1} \end{pmatrix} \vec{E}. \tag{4}$$

Although this is straightforward to implement in frequency domain, where one $\varepsilon(\omega)$ is as good as another, in the time domain a frequency-dependent term requires care. In time-domain, a frequency-dependent term can be thought of as a convolution with a filter (hence the viewpoint of a "convolutional PML" adopted by some authors), where in the language of signal-processing we wish to find a stable recursive filter to implement this convolution with as few taps as possible (to minimize the memory and computational burden). Equivalently, the frequency dependence may be implemented by auxiliary *ordinary* differential equations (discretized ODE = recursive filter). The most convenient ODEs to discretize are first-order ODEs. For example, $a = sb = (1 + i\sigma/\omega)b$ gives $-i\omega a = -i\omega b + \sigma b$, which corresponds to the first-order ODE $\frac{da}{dt} = \frac{db}{dt} + \sigma b$.

So, before we proceed to time domain, we want to *factorize* eq. (16) into terms with only *one* factor of s (or σ/ω) each (or ratios of single s factors). In doing so, we are free to change the definition of $\vec{D}$ and introduce new auxiliary

fields as desired, since the fields in the PML region are not physical. A key trick is the factorization:

$$s_x s_y s_z \begin{pmatrix} s_x^{-1} & & \\ & s_y^{-1} & \\ & & s_z^{-1} \end{pmatrix} = \begin{pmatrix} s_y & & \\ & s_z & \\ & & s_x \end{pmatrix} \begin{pmatrix} s_z & & \\ & s_x & \\ & & s_y \end{pmatrix}.$$

Using this factorization, and defining new auxiliary fields, $\vec{C}$, $\vec{U}$, and $\vec{W}$, we can factorize eq. (16) into the following equivalent form (giving the same relationship between $\vec{H}$ and $\vec{E}$):

$$\vec{K} = \nabla \times \vec{H} = -i\omega \left(1 + \frac{i\sigma_D}{\omega}\right) \vec{C} \tag{5}$$

$$\vec{U} = \begin{pmatrix} s_y^{-1} & & \\ & s_z^{-1} & \\ & & s_x^{-1} \end{pmatrix} \vec{C} \tag{6}$$

$$\vec{D} = \begin{pmatrix} s_z^{-1} & & \\ & s_x^{-1} & \\ & & s_y^{-1} \end{pmatrix} \vec{U} \tag{7}$$

$$\vec{W} = \varepsilon_\infty^{-1} \left(\vec{D} - \vec{P}\right) \tag{8}$$

$$\vec{P} = \chi_E(\omega)\vec{W} \tag{9}$$

$$\vec{E} = \begin{pmatrix} s_x & & \\ & s_y & \\ & & s_z \end{pmatrix} \vec{W} \tag{10}$$

4 PML formulation in time domain

Given eqs. (5–10), a time-domain formulation is now easy because each equation only includes first-order factors in ω (corresponding to first-order time derivatives). Moreover, all of the nontrivial equations, namely (5), (8), and (9), are *exactly* the same as in the non-PML case, meaning that we can use *exactly* the same code except passing new fields $\vec{C}$ and $\vec{W}$ instead of $\vec{D}$ and $\vec{E}$. The other equations are diagonal ODEs that are trivial to implement. It may seem wasteful to have three new auxiliary fields, but in many cases they can be omitted: except in corners of the computational cell, one has PML only in one direction that only one component of $s_{x,y,z}$ is $\neq 1$, and in these cases one need only store one component of $\vec{U}$ and one component of $\vec{W}$ (with the other components replaced by $\vec{D}$ and $\vec{E}$, respectively); $\vec{C}$ can be omitted in non-conducting materials ($\sigma_D = 0$).

For completeness, we write out the equations here, in the order that they would be evaluated. The $\vec{C}$ update from eq. (5) is identical to the $\vec{D}$ timestep

from eq. (11)

$$\vec{C}^{n+1} = \left(1 + \frac{\sigma_D \Delta t}{2}\right)^{-1} \left[\left(1 - \frac{\sigma_D \Delta t}{2}\right) \vec{C}^n + \vec{K}^{n+0.5}\right]. \tag{11}$$

Each component U_k of $\vec{U}$ is updated from the corresponding component of $\vec{C}$ by the ODE $dU_k/dt + \sigma_{k+1} U_k = dC_k/dt$ (where $\sigma_{k\pm1}$ is interpreted as a cyclic shift, e.g. $\sigma_{y+1} = \sigma_z$, $\sigma_{z+1} = \sigma_x$), giving:

$$U_k^{n+1} = \left(1 + \frac{\sigma_{k+1} \Delta t}{2}\right)^{-1} \left[\left(1 - \frac{\sigma_{k+1} \Delta t}{2}\right) U_k^n + C_k^n - C_k^{n-1}\right]. \tag{12}$$

Each component of $\vec{D}$ is then updated from the corresponding component of $\vec{U}$ by the ODE $dD_k/dt + \sigma_{k-1} D_k = dU_k/dt$, giving:

$$D_k^{n+1} = \left(1 + \frac{\sigma_{k-1} \Delta t}{2}\right)^{-1} \left[\left(1 - \frac{\sigma_{k-1} \Delta t}{2}\right) D_k^n + U_k^n - U_k^{n-1}\right]. \tag{13}$$

$\vec{W}$ is then updated from $\vec{D} - \vec{P}$ exactly as in eq. (14):

$$\vec{W}^{n+1} = \varepsilon_\infty^{-1} \left(\vec{D}^{n+1} - \vec{P}^{n+1}\right). \tag{14}$$

Then, each component of $\vec{E}$ is updated from each component of $\vec{W}$ by the ODE $dE_k/dt = dW_k/dt + \sigma_k W_k$, giving:

$$E_k^{n+1} = E_k^n + \left(1 + \frac{\sigma_k \Delta t}{2}\right) W_k^{n+1} - \left(1 - \frac{\sigma_k \Delta t}{2}\right) W_k^n. \tag{15}$$

Finally, $\vec{P}^{n+2}$ is computed using $\vec{P} = \chi_E(\omega)\vec{W}$, exactly as for the non-PML case (but with $\vec{W}$ replacing $\vec{E}$).

Note that, in order to avoid having to save the fields from the previous timestep in yet more auxiliary arrays, the $\vec{C}$, $\vec{U}$, and $\vec{D}$ updates [eqs. (11–13)] have to be performed in a single loop body, while the $\vec{W}$ and $\vec{E}$ updates [eqs. (14–15)] must be performed in another single loop body. [One cannot easily merge the two loops because the offdiagonal anisotropic terms in ε_∞^{-1} combined with the staggered Yee grid mean that eq. (14) is effectively nonlocal in space, requiring $\vec{D}^{n+1}$ components at several spatial points to determine $\vec{W}^{n+1}$. Separating the two loops has the additional advantage that it reduces the combinatorial explosion of the number of material cases that must be handled.]

5 Appendix: Textbook isotropic, nondispersive PML

In this appendix, we review the standard "textbook" UPML for isotropic nondispersive media as found in, for example, Taflove and Hagness. It is important

to distinguish mere differences in notation (the derivation in T&H is far less compact than the one here) from substantive differences, and in particular the usual UPML corresponds to a different factorization of eq. (16) than ours. This textbook formulation only works for isotropic (or at least diagonal) ε, because it assumes that an arbitrary diagonal matrix commutes with ε.

In particular, dropping the σ_D and χ_E terms (assuming nondispersive media), the textbook PML formulation corresponds to the following refactorization of eq. (16):

$$\vec{K} = \nabla \times \vec{H} = -i\omega \begin{pmatrix} s_y & & \\ & s_z & \\ & & s_x \end{pmatrix} \varepsilon_\infty \begin{pmatrix} \frac{s_z}{s_x} & & \\ & \frac{s_x}{s_y} & \\ & & \frac{s_y}{s_z} \end{pmatrix} \vec{E}. \qquad (16)$$

This then factorizes into two equations:

$$\vec{K} = \nabla \times \vec{H} = -i\omega \begin{pmatrix} s_y & & \\ & s_z & \\ & & s_x \end{pmatrix} \vec{D},$$

$$\vec{E} = \varepsilon_\infty^{-1} \begin{pmatrix} \frac{s_x}{s_z} & & \\ & \frac{s_y}{s_x} & \\ & & \frac{s_z}{s_y} \end{pmatrix} \vec{D}.$$

The former discretizes similarly to eq. (2), with the PML $\sigma_{y,z,x}$ taking the place of σ_D, and the latter turns into the ODE $dE_k/dt + \sigma_{k-1} E_k = \varepsilon_\infty^{-1}[dD_k/dt + \sigma_k D_k]$ which is easily discretized.

Only two fields need be stored: $\vec{D}$ and $\vec{E}$. If a conductivity σ_D (or similar) is included, however, one needs an additional auxiliary field $\vec{C}$ (or similar). It turns out that the case of a dispersive $\vec{P} = \chi_E(\omega)\vec{E}$ can also be handled with no additional storage compared to the non-PML case, at least for a Lorentzian χ_E that requires both $\vec{P}^{n+1}$ and $\vec{P}^n$ to be stored anyway. So, the main cost of the Meep formulation is that two additional auxiliary fields $\vec{U}$ and $\vec{W}$ must be stored in the PML; however, this is actually only a slight additional burden in storage since for most PML regions only one component of $\vec{U}$ and $\vec{W}$ actually must be stored (see above). The main problem with the textbook implementation is that it fails for anisotropic media (even if one naively plugs a tensor ε_∞^{-1} into the $\vec{E}$ update, the reflection will not go to zero with increasing resolution). This is especially important given the fact that, even for nominally isotropic media, accurate subpixel averaging requires the discretization to use effective anisotropic media at material boundaries.