Source Estimation for Higher Order FDTD

David Ge (dge893@gmail.com)

Thursday, February 9, 2023

Abstract

For a (2N, 2M) order Yee-algorithm and a (2N+1, 2M+1) order Time-Space-Synchronized (TSS) algorithm, if the order is higher than (2, 2) then curls and temporal derivatives of the field source are needed. To satisfy this requirement, algorithms are developed to estimate the curls and the temporal derivatives of the field source, from the discrete values of the field source.

Contents

Introduction 1

Source estimation for (2N, 2M) order Yee-algorithm 2

Yee-digitization 2

Recursive formation of the source 3

Estimations 3

Source estimation for (2N+1, 2M+1) TSS algorithm 4

Recursive formation of the source 4

Estimations 4

Numeric results 5

Reference 5

Introduction

Consider Maxwell's equations in a well-known 3D form:

$\frac{\partial E}{\partial t} = \frac{1}{\varepsilon}\nabla \times H - \frac{1}{\varepsilon}J$	(1.1)
$\frac{\partial H}{\partial t} = -\frac{1}{\mu}\nabla \times E$	(1.2)
$E, H, J \in R^3;\ \varepsilon, \mu \in R$	

Where $J(x, y, z, t)$ is a 3D vector representing the field source.

From a FDTD algorithm point of view, $J(x, y, z, t)$ is an input, and $E(x, y, z, t)$ and $H(x, y, z, t)$ are outputs. All FDTD algorithms in literatures work in such a manner: given discrete input values $J(x_i, y_j, z_k, t_q)$, generate discrete outputs $E(x, y, z, t)$ and $H(x, y, z, t)$.

This situation changes for my arbitrary order FDTD algorithms ([1]).

My (2N, 2M) order Yee-algorithm is

$$H\left(t + \frac{1}{2}\Delta_t\right) \approx H\left(t - \frac{1}{2}\Delta_t\right) + \sum_{n=0}^{N-1} G_{e,n} + S_h(t)$$

$$E(t + \Delta_t) \approx E(t) + \sum_{n=0}^{N-1} G_{h,n} + S_e\left(t + \frac{1}{2}\Delta_t\right)$$

Where Δ_t is a time forward step size, $G_{e,n}$ is a vector formed by constant linear combination of finite differences of electric field, and $G_{h,n}$ is a vector formed by constant linear combination of finite differences of magnetic field. Field source is represented by $S_h(t)$ and $S_e\left(t + \frac{1}{2}\Delta_t\right)$, and given by

$S_h(t) = 2\sum_{n=0}^{N-1} \frac{\left(\frac{1}{2}\Delta_t\right)^{2n+1}}{(2n+1)!} \begin{cases} \vec{0}, n = 0 \\ \sum_{m=1}^{n} \frac{(-1)^{m+1}}{(\varepsilon\mu)^m} \nabla^{2m-1} \times \frac{\partial^{2(n-m)+1} J(t)}{\partial t^{2(n-m)+1}}, n > 0 \end{cases}$	(1.3)

$$S_e\left(t+\frac{1}{2}\Delta_t\right)=\frac{2}{\varepsilon}\sum_{n=0}^{N-1}\frac{\left(\frac{1}{2}\Delta_t\right)^{2n+1}}{(2n+1)!}\sum_{m=0}^{n}\frac{(-1)^{m+1}}{(\varepsilon\mu)^m}\nabla^{2m}\times\frac{\partial^{2(n-m)}J\left(t+\frac{1}{2}\Delta_t\right)}{\partial t^{2(n-m)}}$$	(1.4)

My (2N+1, 2M+1) order Time-Space-Synchronized (TSS) algorithm is

$$H(t+\Delta_t)=2H(t)-H(t-\Delta_t)+\sum_{n=0}^{N-1}F_{h,n}+S_h(t)+O((\Delta_t)^{2N+1})$$

$$E(t+\Delta_t)=2E(t)-E(t-\Delta_t)+\sum_{n=0}^{N-1}F_{e,n}+S_e(t)+O((\Delta_t)^{2N+1})$$

Where $F_{e,n}$ is a vector formed by constant linear combination of finite differences of electric field, and $F_{h,n}$ is a vector formed by constant linear combination of finite differences of magnetic field. Field source is represented by $S_h(t)$ and $S_e(t)$, and given by

$$S_h(t)=2\sum_{n=0}^{N-1}\frac{(\Delta_t)^{2(n+1)}}{(2(n+1)!}\sum_{m=0}^{n}\frac{(-1)^m}{(\varepsilon\mu)^{m+1}}\nabla^{2m+1}\times\frac{\partial^{2(n-m)}J(t)}{\partial t^{2(n-m)}}$$	(1.5)
$$S_e(t)=\frac{2}{\varepsilon}\sum_{n=0}^{N-1}\frac{(\Delta_t)^{2(n+1)}}{(2(n+1)!}\sum_{m=0}^{n}\frac{(-1)^{m+1}}{(\varepsilon\mu)^m}\nabla^{2m}\times\frac{\partial^{2(n-m)+1}J(t)}{\partial t^{2(n-m)+1}}$$	(1.6)

From (1.3), (1.4), (1.5) and (1.6), S_h and S_h are formed by

$$\nabla^u\times\frac{\partial^v J}{\partial t^v};u,v\geq 0$$

$$u=v=0\ for\ (2,2)\ order\ FDTD$$

Comparing to a (2,2)-order FDTD, the higher order FDTD introduces following difficulties.

1. If $J(x,y,z,t)$ is not analytically available or if it is not sufficiently differentiable then it is not possible to get $\nabla^u\times\frac{\partial^v J}{\partial t^v}$ for higher order FDTD.
2. In the cases where $J(x,y,z,t)$ is analytically available and sufficiently differentiable, to deduce $\nabla^u\times\frac{\partial^v J}{\partial t^v}$ from $J(x,y,z,t)$ can be a tedious process and error-prone.
3. Even if it is possible to deduce analytical form of $\nabla^u\times\frac{\partial^v J}{\partial t^v}$ from $J(x,y,z,t)$, most likely it can only be done with fixed integers of u and v. Thus, for a (2N,2M) order or a (2N+1,2M+1) order algorithm, the selections of N and M are limited by fixed integers of u and v, not truly arbitrary.

To overcome these difficulties, we may use discrete values of $J(x,y,z,t)$ to estimate values of $\nabla^u\times\frac{\partial^v J}{\partial t^v}$. Most of the techniques needed for making such estimations are already presented in [1].

Source estimation for (2N, 2M) order Yee-algorithm

Yee-digitization

A 3D simulation domain is represented by integers n_x, n_y, n_z, and real numbers x_{min}, y_{min} and z_{min}. Digitization is represendted by a cell size Δ_s.

$$x_i=x_{min}+i\Delta_s;\ i=0,1,2,\dots,n_x$$ $$y_j=y_{min}+j\Delta_s;\ j=0,1,2,\dots,n_y$$ $$z_k=z_{min}+k\Delta_s;k=0,1,2,\dots,n_z$$ $$n_x>0,n_y>0,n_z>0,\Delta_s>0$$	(2.1)

Define a "Yee-E vector $\mathbb{Y}_e(F[i,j,k])$" and a "Yee-H vector $\mathbb{Y}_h(F[i,j,k])$" to describe the Yee-digitization:

$$\mathbb{Y}_e(F[i,j,k])=\begin{bmatrix}F_x\left(x_i+\frac{1}{2}\Delta_s,y_j,z_k\right)\\F_y\left(x_i,y_j+\frac{1}{2}\Delta_s,z_k\right)\\F_z\left(x_i,y_j,z_k+\frac{1}{2}\Delta_s\right)\end{bmatrix}$$	(2.2)
$$\mathbb{Y}_h(F[i,j,k])=\begin{bmatrix}F_x\left(x_i,y_j+\frac{1}{2}\Delta_s,z_k+\frac{1}{2}\Delta_s\right)\\F_y\left(x_i+\frac{1}{2}\Delta_s,y_j,z_k+\frac{1}{2}\Delta_s\right)\\F_z\left(x_i+\frac{1}{2}\Delta_s,y_j+\frac{1}{2}\Delta_s,z_k\right)\end{bmatrix}$$	(2.3)

According to the space location theorems proved in [1], applying the "Yee-vector" notations to (1.3) and (1.4), the field source becomes (2.4) and (2.5).

$$\mathbb{Y}_h S_h(t)=2\sum_{n=0}^{N-1}\frac{\left(\frac{1}{2}\Delta_t\right)^{2n+1}}{(2n+1)!}\begin{cases}\vec{0},n=0\\\sum_{m=1}^{n}\frac{(-1)^{m+1}}{(\varepsilon\mu)^m}\nabla^{2m-1}\times\frac{\partial^{2(n-m)+1}\mathbb{Y}_e J(t)}{\partial t^{2(n-m)+1}},n>0\end{cases}$$	(2.4)

$$\mathbb{Y}_e S_e\left(t+\frac{1}{2}\Delta_t\right)=\frac{2}{\varepsilon}\sum_{n=0}^{N-1}\frac{\left(\frac{1}{2}\Delta_t\right)^{2n+1}}{(2n+1)!}\sum_{m=0}^{n}\frac{(-1)^{m+1}}{(\varepsilon\mu)^m}\nabla^{2m}\times\frac{\partial^{2(n-m)}\mathbb{Y}_e J\left(t+\frac{1}{2}\Delta_t\right)}{\partial t^{2(n-m)}}$$	(2.5)

Note that only "Yee-E vector" $\mathbb{Y}_e J$ appears in (2.4) and (2.5).

Recursive formation of the source

Define following constants.

$$c=\frac{1}{\sqrt{\varepsilon\mu}}$$	(2.6)
$$c_r=c\frac{\Delta_t}{\Delta_s}$$	(2.7)
$$\eta=\sqrt{\frac{\mu}{\varepsilon}}$$	(2.8)

Define following Yee-algorithm related constants:

$$\delta_t=\frac{1}{2}\Delta_t$$	(2.9)
$$c_{r22}=-\left(\frac{1}{2}c_r\right)^2$$	(2.10)
$$G_{h,n,m}=-c_{r22}\frac{c_r\Delta_s}{(2n+1)!}c_{r22}^m$$	(2.11)
$$G_{e,n,m}=-\eta\frac{c_r\Delta_s}{(2n+1)!}c_{r22}^m$$	(2.12)

(2.4) and (2.5) can be written as (2.13) and (2.14).

$$\mathbb{Y}_h S_h(t)=\sum_{n=1}^{N-1}\sum_{m=0}^{n-1}G_{h,n,m}W_{h,n,m}\,;N>1$$	(2.13)
$$\mathbb{Y}_e S_e\left(t+\frac{1}{2}\Delta_t\right)=\sum_{n=0}^{N-1}\sum_{m=0}^{n}G_{e,n,m}W_{e,n,m}$$	(2.14)

Where

$$W_{h,1,0}=\Delta_s\nabla\times\delta_t\frac{\partial\mathbb{Y}_e J(t)}{\partial t}$$	(2.15)
$$W_{h,n,0}=\delta_t^2\frac{\partial^2}{\partial t^2}W_{h,n-1,0};n>1$$	(2.16)
$$W_{h,n,m}=\Delta_s\nabla^2\times W_{h,n-1,m-1};m>0$$ $$m=0,1,\dots,n-1$$	(2.17)
$$W_{e,0,0}=\mathbb{Y}_e J\left(t+\frac{1}{2}\Delta_t\right)$$	(2.18)
$$W_{e,n,0}=\delta_t^2\frac{\partial^2}{\partial t}W_{e,n-1,0};n>0$$	(2.19)
$$W_{e,n,m}=\Delta_s^2\nabla^2\times W_{e,n-1,m-1};m>0$$ $$m=0,1,\dots,n$$	(2.20)

The values given by (2.15), (2.16), (2.17), (2.19) and (2.20) should be estimated.

Estimations

To estimate (2.15), we need to estimate operator $\delta_t\frac{\partial\mathbb{Y}_e J(t)}{\partial t}$ and $\Delta_s\nabla\times$.

For estimating $\delta_t\frac{\partial\mathbb{Y}_e J(t)}{\partial t}$, use (c.5) in [1], see "**Odd order derivative estimation theorem (even-sampling)**" in [1].

For estimating $\Delta_s\nabla\times$, use (a.7) in [1], see "**Curl estimation theorem (Yee-sampling)**" in [1].

To estimate (2.16), we need to estimate operator $\delta_t^2\frac{\partial^2}{\partial t^2}$, use (g.5) in [1], see "**Even order derivative estimation theorem**" in [1].

To estimate (2.17), we need to estimate operator $\Delta_s\nabla^2\times$. We may use Laplacian to replace double curl because divergence of $W_{h,n-1,m-1}$ is 0. Use (g.7) in [1], see "**Laplacian estimation theorem**" in [1].

To estimate (2.19), we need to estimate operator $\delta_t^2\frac{\partial^2}{\partial t^2}$, use (g.5) in [1], see "**Even order derivative estimation theorem**" in [1].

To estimate (2.20), we need to estimate operator $\Delta_s^2\nabla^2\times W_{e,n-1,m-1}$.

For $m>1$, because $\nabla\cdot W_{e,n,m}=0$, we may use Laplacian to replace double curl. Use (g.7) in [1], see "**Laplacian estimation theorem**" in [1].

For $m=1$, because we cannot guarantee that $\nabla\cdot W_{e,n,0}=0$, we cannot use Laplacian to replace double curl. We may apply curl estimation (c.11) twice. Curl estimation using even-sampling is not presented in [1]. (c.11) is presented below.

Curl estimation theorem (even-sampling). For a sufficiently smooth 3D field $F(x,y,z)$, a 2*M* order estimation of its curl scaled by a sampling size is a *M* -dimentional constant linear combination of nearby finite-differences,

$\Delta_s \nabla \times F = \frac{1}{2}\boldsymbol{c}_0 \cdot \begin{bmatrix} \boldsymbol{p}_{z,y} - \boldsymbol{p}_{y,z} \\ \boldsymbol{p}_{x,z} - \boldsymbol{p}_{z,x} \\ \boldsymbol{p}_{y,x} - \boldsymbol{p}_{x,y} \end{bmatrix} (F) + O((\Delta_s)^{2M})$	(c.11)

Where $\boldsymbol{c}_0$ is a constant vector given by (c.3) and (c.4) in [1], $\boldsymbol{p}_{u,v}$ is a vector of finite differences formed by (c.7) in [1].

Proof. By (c.5) and (c.7) in [1],

$$\Delta_s \nabla \times F = \begin{bmatrix} \Delta_s \frac{\partial F_z}{\partial y} - \Delta_s \frac{\partial F_y}{\partial z} \\ \Delta_s \frac{\partial F_x}{\partial z} - \Delta_s \frac{\partial F_z}{\partial x} \\ \Delta_s \frac{\partial F_y}{\partial x} - \Delta_s \frac{\partial F_x}{\partial y} \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\boldsymbol{c}_0 \cdot \boldsymbol{p}_{z,y} - \frac{1}{2}\boldsymbol{c}_0 \cdot \boldsymbol{p}_{y,z} \\ \frac{1}{2}\boldsymbol{c}_0 \cdot \boldsymbol{p}_{x,z} - \frac{1}{2}\boldsymbol{c}_0 \cdot \boldsymbol{p}_{z,x} \\ \frac{1}{2}\boldsymbol{c}_0 \cdot \boldsymbol{p}_{y,x} - \frac{1}{2}\boldsymbol{c}_0 \cdot \boldsymbol{p}_{x,y} \end{bmatrix} (F) + O((\Delta_s)^{2M})$$

$$= \frac{1}{2}\boldsymbol{c}_0 \cdot \begin{bmatrix} \boldsymbol{p}_{z,y} - \boldsymbol{p}_{y,z} \\ \boldsymbol{p}_{x,z} - \boldsymbol{p}_{z,x} \\ \boldsymbol{p}_{y,x} - \boldsymbol{p}_{x,y} \end{bmatrix} (F) + O((\Delta_s)^{2M})$$

QED

Source estimation for (2N+1, 2M+1) TSS algorithm

Recursive formation of the source

Define following constants.

$c_{r2} = -(c_r)^2$	(3.1)
$G_{h,n,m} = c_r \frac{2c_r\Delta_s}{(2(n+1)!} c_{r2}^m$	(3.2)
$G_{e,n,m} = -\eta \frac{2c_r\Delta_s}{(2(n+1))!} c_{r2}^m$	(3.3)

(1.5) and (1.6) can be written as (3.4) and (3.5):

$S_h(t) = \sum_{n=0}^{N-1} \sum_{m=0}^{n-1} G_{h,n,m} W_{h,n,m}$	(3.4)
$S_e(t) = \sum_{n=0}^{N-1} \sum_{m=0}^{n} G_{e,n,m} W_{e,n,m}$	(3.5)

Where

$W_{h,0,0} = \Delta_s \nabla \times J(t)$	(3.6)
$W_{h,n,0} = \Delta_t^2 \frac{\partial^2}{\partial t^2} W_{h,n-1,0}; n > 0$	(3.7)
$W_{h,n,m} = \Delta_s^2 \nabla^2 \times W_{h,n-1,m-1}; m > 0$ $m = 0,1,\dots,n$	(3.8)
$W_{e,0,0} = \Delta_t \frac{\partial J(t)}{\partial t}$	(3.9)
$W_{e,n,0} = \Delta_t^2 \frac{\partial^2 J(t)}{\partial t^2} W_{e,n-1,0}; n > 0$	(3.10)
$W_{e,n,m} = \Delta_s^2 \nabla^2 \times W_{e,n-1,m-1}; m > 0$ $m = 0,1,\dots,n$	(3.11)

The values given by (3.6) to (3.11) should be estimated.

Estimations

To estimate (3.6), we need to estimate operator $\Delta_s \nabla \times$. Use (c.11) to do it, see "**Curl estimation theorem (even-sampling)**" above.

To estimate (3.7), we need to estimate operator $\Delta_t^2 \frac{\partial^2}{\partial t^2}$. Use (g.5) in [1], see "**Even order derivative estimation theorem**" in [1].

To estimate (3.8), we need to estimate operator $\Delta_s^2 \nabla^2 \times$. Because the divergence of $W_{h,n-1,m-1}$ is 0, we may use Laplacian to replace double curl. Use (g.7) in [1], see "**Laplacian estimation theorem**" in [1].

To estimate (3.9), we need to estimate operator $\Delta_t \frac{\partial}{\partial t}$. Use (c.5) in [1], see "**Odd order derivative estimation theorem (even-sampling)**" in [1].

To estimate (3.10), we need to estimate operator $\Delta_t^2 \frac{\partial^2}{\partial t^2}$. Use (g.5) in [1], see "**Even order derivative estimation theorem**" in [1].

To estimate (3.11), we need to estimate $\Delta_s^2 \nabla^2 \times W_{e,n-1,m-1}$.

For $m > 1$, because $\nabla \cdot W_{e,n-1,m-1} = 0$, we may use Laplacian to replace double curl. Use (g.7) in [1], see "**Laplacian estimation theorem**" in [1].

For $m = 1$, because we cannot guarantee that $\nabla \cdot W_{e,n-1,0} = 0$, we cannot use Laplacian to replace double curl. We may apply curl estimation (c.11) twice. See "**Curl estimation theorem (even-sampling)**" above.

Numeric results

The source code for the above algorithms and numeric results will be uploaded to Github at https://github.com/DavidGeUSA/FDTD

Reference

[1] David Wei Ge, Arbitrary Order FDTD for Maxell's Equations, January 2023,
https://www.researchgate.net/publication/367652640_Arbitrary_Order_FDTD_for_Maxell's_Equations