Arbitrary Order FDTD for Maxell's Equations

David Ge (dge893@gmail.com)

Monday, January 9, 2023

Abstract

The relationship between curl and derivative with respect to time is extended to arbitrary orders. Arbitrary order estimations of curls, Laplacian and gradient of divergence are formed by constant linear combinations. Using these results, various arbitrary order FDTD algorithms can be formed. As examples, a (2N, 2M) order Yee-algorithm and a (2N+1, 2M+1) order Time-Space-Synchronized (TSS) algorithm is presented, where N and M are any positive integers.

Contents

Introduction

Higher order FDTD algorithms are mostly in fixed orders represnedted by (N,M) where N represents the estimation order for time forward and M represents the estimation order for curls. N and M are fixed numbers, and usually N = 2 or 4, and M=2, 4, or 6 ([4],[7],[8]). The estimation order of the standard Yee algorithm ([1],[2]) is (2,2).

Two years ago, Gokhan Aydin, Atef Z Elsherbeni, Ercument Arvas, and Jay K Lee developed a single field algorithm based on vector wave equation ([11]). It does not mention the estimation order. Because its estimation equation does not involve estimation orders (see equation (16) in [11]), I'll assume that it is also a fixed order algorithm.

Shin Jongchul developed an algorithm based on the Green's function. It uses the second order estimation (see equation (2.50) in [12]) and it does not use Yee-digitization. It is a time-space-synchronized (TSS) algorithm.

Some algorithms are for (2,2M) order, where M can be any positive integer (V. A. Bokil AND N. L. Gibson [3]).

The FDTD algorithms usually look quite complex, with the standard Yee algorithm being the most simple one. Due to such complexity, many FDTD algorithms are presented in 1-D or 2-D forms, even though their ideas can be applied to 3-D space ([9],[11]).

Since 2018, I have formed a basic theory of FDTD algorithms of arbitrary order, based on Taylor series. It is presented in ResearchGate ([5], [6]). An open source FDTD software implementing the results was uploaded to Github.

In this paper I am going to present some new results and re-present the old results.

I have also refined the software. I am going to upload a new version of my FDTD software to Github. I am still collecting numerical data and making graphic charts. I will upload the numeric data soon.

Arbitrary Order Temporal Estimation

Time-Space Theorem

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)
$E = \begin{bmatrix} E_x \\ E_y \\ E_z \end{bmatrix}, H = \begin{bmatrix} H_x \\ H_y \\ H_z \end{bmatrix}, J = \begin{bmatrix} J_x \\ J_y \\ J_z \end{bmatrix} \in R^3$ $E_x, E_y, E_z, H_x, H_y, H_z, J_x, J_y, J_z, \varepsilon, \mu \in R$	

Introduce a concept of "curl order" $\nabla^m \times$ similar to the concept of derivative order $\partial^m/\partial t^m$. An m-th order curl is defined for a 3-D vector F by

$\nabla^0 \times F \equiv F$ $\nabla^m \times F \equiv \underbrace{\nabla \times \nabla \times \ldots \times \nabla \times}_{m} F, m \geq 0$	(1.3)

Time-Space Theorem. For an electromagnetic field, if the sequence of derivative and curl is interchangeable: $\frac{\partial}{\partial t}\nabla \times = \nabla \times \frac{\partial}{\partial t}$ then

1. A derivative of any odd order of the magnetic field with respect to time equals a space curl of the electric field of the same order, as described by (1.4);
2. A derivative of any odd order of the electric field with respect to time equals a space curl of the magnetic field of the same order, as described by (1.6);
3. A derivative of any even order of the magnetic field with respect to time equals a space curl of the magnetic field of the same order, as described by (1.5);
4. A derivative of any even order of the electric field with respect to time equals a space curl of the electric field of the same order, as described by (1.7);

$\frac{\partial^{2n+1}H}{\partial t^{2n+1}} = \frac{1}{\mu}\frac{(-1)^{n+1}}{(\varepsilon\mu)^n}\nabla^{2n+1} \times E + \begin{cases} \vec{0}, n = 0 \\ \sum_{m=1}^{n} \frac{(-1)^{m+1}}{(\varepsilon\mu)^m}\nabla^{2m-1} \times \frac{\partial^{2(n-m)+1}J}{\partial t^{2(n-m)+1}}, n > 0 \end{cases}$	(1.4)
$\frac{\partial^{2(n+1)}H}{\partial t^{2(n+1)}} = \frac{(-1)^{n+1}}{(\varepsilon\mu)^{n+1}}\nabla^{2(n+1)} \times H + \sum_{m=0}^{n} \frac{(-1)^m}{(\varepsilon\mu)^{m+1}}\nabla^{2m+1} \times \frac{\partial^{2(n-m)}J}{\partial t^{2(n-m)}}$	(1.5)
$\frac{\partial^{2n+1}E}{\partial t^{2n+1}} = \frac{1}{\varepsilon}\frac{(-1)^n}{(\varepsilon\mu)^n}\nabla^{2n+1} \times H + \frac{1}{\varepsilon}\sum_{m=0}^{n} \frac{(-1)^{m+1}}{(\varepsilon\mu)^m}\nabla^{2m} \times \frac{\partial^{2(n-m)}J}{\partial t^{2(n-m)}}$	(1.6)
$\frac{\partial^{2(n+1)}E}{\partial t^{2(n+1)}} = \frac{(-1)^{n+1}}{(\varepsilon\mu)^{n+1}}\nabla^{2(n+1)} \times E + \frac{1}{\varepsilon}\sum_{m=0}^{n} \frac{(-1)^{m+1}}{(\varepsilon\mu)^m}\nabla^{2m} \times \frac{\partial^{2(n-m)+1}J}{\partial t^{2(n-m)+1}}$	(1.7)
$n = 0,1,2,\ldots$	

Proof. This theorem can be proved by induction based on Maxwell's equations.

Consider the case when $n = 0$.

For $n = 0$ (1.4) becomes (1.2). So (1.4) holds.

By taking temporal derivative of (1.2) and substituting (1.1) into it, we have

$$\frac{\partial^2 H}{\partial t^2} = -\frac{1}{\mu}\nabla \times \left(\frac{1}{\varepsilon}\nabla \times H - \frac{1}{\varepsilon}J\right)$$

$$\frac{\partial^2 H}{\partial t^2} = -\frac{1}{\varepsilon\mu}\nabla^2 \times H + \frac{1}{\varepsilon\mu}\nabla \times J \tag{1.8}$$

The above is (1.5) for $n = 0$. So, (1.5) holds.

For $n = 0$ (1.6) becomes (1.1). So, (1.6) holds.

By taking temporal derivative of (1.1) and substituting (1.2) into it, we have

$$\frac{\partial^2 E}{\partial t^2} = \frac{1}{\varepsilon}\nabla \times \left(-\frac{1}{\mu}\nabla \times E\right) - \frac{1}{\varepsilon}\frac{\partial J}{\partial t}$$

$$\frac{\partial^2 E}{\partial t^2} = -\frac{1}{\varepsilon\mu}\nabla^2 \times E - \frac{1}{\varepsilon}\frac{\partial J}{\partial t} \tag{1.9}$$

The above is (1.7) when $n = 0$. So, (1.7) holds for $n = 0$.

The above proved that (1.4) – (1.7) hold for $n = 0$.

Consider the case when $n = 1$.

Taking the temporal derivative on (1.8) and substituting (1.2) into it, we have

$$\frac{\partial^3 H}{\partial t^3} = -\frac{1}{\varepsilon\mu}\nabla^2 \times \left(-\frac{1}{\mu}\nabla \times E\right) + \frac{1}{\varepsilon\mu}\nabla \times \frac{\partial J}{\partial t}$$

$$\frac{\partial^3 H}{\partial t^3} = \frac{1}{\mu}\frac{1}{\varepsilon\mu}\nabla^3 \times E + \frac{1}{\varepsilon\mu}\nabla \times \frac{\partial J}{\partial t} \tag{1.10}$$

The above is (1.4) when $n = 1$. So, (1.4) holds for $n = 1$.

Taking the temporal derivative on (1.10) and substituting (1.1) into it, we have

$$\frac{\partial^4 H}{\partial t^4} = \frac{1}{\mu}\frac{1}{\varepsilon\mu}\nabla^3 \times \left(\frac{1}{\varepsilon}\nabla \times H - \frac{1}{\varepsilon}J\right) + \frac{1}{\varepsilon\mu}\nabla \times \frac{\partial^2 J}{\partial t^2} = \frac{1}{(\varepsilon\mu)^2}\nabla^4 \times H - \frac{1}{(\varepsilon\mu)^2}\nabla^3 \times J + \frac{1}{\varepsilon\mu}\nabla \times \frac{\partial^2 J}{\partial t^2}$$

The above is (1.5) when $n = 1$. So, (1.5) holds for $n = 1$.

Taking the temporal derivative on (1.9) and substituting (1.1) into it, we have

$$\frac{\partial^3 E}{\partial t^3} = -\frac{1}{\varepsilon\mu}\nabla^2 \times \left(\frac{1}{\varepsilon}\nabla \times H - \frac{1}{\varepsilon}J\right) - \frac{1}{\varepsilon}\frac{\partial^2 J}{\partial t^2}$$

$$\frac{\partial^3 E}{\partial t^3} = -\frac{1}{\varepsilon}\frac{1}{\varepsilon\mu}\nabla^3 \times H + \frac{1}{\varepsilon}\frac{1}{\varepsilon\mu}\nabla^2 \times J - \frac{1}{\varepsilon}\frac{\partial^2 J}{\partial t^2} \tag{1.11}$$

The above is (1.6) when $n = 1$. So, (1.6) holds for $n = 1$.

Taking the temporal derivative on (1.11) and substituting (1.2) into it, we have

$$\frac{\partial^4 E}{\partial t^4} = -\frac{1}{\varepsilon}\frac{1}{\varepsilon\mu}\nabla^3 \times \left(-\frac{1}{\mu}\nabla \times E\right) + \frac{1}{\varepsilon}\frac{1}{\varepsilon\mu}\nabla^2 \times \frac{\partial J}{\partial t} - \frac{1}{\varepsilon}\frac{\partial^3 J}{\partial t^3} = \frac{1}{(\varepsilon\mu)^2}\nabla^4 \times E + \frac{1}{\varepsilon}\frac{1}{\varepsilon\mu}\nabla^2 \times \frac{\partial J}{\partial t} - \frac{1}{\varepsilon}\frac{\partial^3 J}{\partial t^3}$$

The above is (1.7) when $n = 1$. So, (1.7) holds for $n = 1$.

The above proved that (1.4) – (1.7) hold for $n = 0\ and\ n = 1$.

Assume for a $n > 1$, (1.4) – (1.7) hold. Consider the case of $n + 1$.

Taking the temporal derivative on (1.5) and substituting (1.2) into it, we have

$$\frac{\partial^{2(n+1)+1} H}{\partial t^{2(n+1)+1}} = \frac{(-1)^{n+1}}{(\varepsilon\mu)^{n+1}}\nabla^{2(\mathrm{n}+1)} \times \left(-\frac{1}{\mu}\nabla \times E\right) + \sum_{m=0}^{n}\frac{(-1)^m}{(\varepsilon\mu)^{m+1}}\nabla^{2m+1} \times \frac{\partial^{2(n-m)+1} J}{\partial t^{2(n-m)+1}}$$

$$= \frac{1}{\mu}\frac{(-1)^{n+2}}{(\varepsilon\mu)^{n+1}}\nabla^{2(\mathrm{n}+1)+1} \times E + \sum_{m=1}^{n+1}\frac{(-1)^{m-1}}{(\varepsilon\mu)^m}\nabla^{2(m-1)+1} \times \frac{\partial^{2(n-(m-1))+1} J}{\partial t^{2(n-(m-1))+1}}$$

$$\frac{\partial^{2(n+1)+1} H}{\partial t^{2(n+1)+1}} = \frac{1}{\mu}\frac{(-1)^{n+2}}{(\varepsilon\mu)^{n+1}}\nabla^{2(\mathrm{n}+1)+1} \times E + \sum_{m=1}^{n+1}\frac{(-1)^{m+1}}{(\varepsilon\mu)^m}\nabla^{2m-1} \times \frac{\partial^{2(n+1-m)+1} J}{\partial t^{2(n+1-m)+1}} \tag{1.12}$$

The above is (1.4) when substituting n with $n + 1$. So, (1.4) holds for $n + 1$.

Taking the temporal derivative on (1.12) and substituting (1.1) into it, we have

$$\frac{\partial^{2(n+1)+2}H}{\partial t^{2(n+1)+2}}=\frac{1}{\mu}\frac{(-1)^{n+2}}{(\varepsilon\mu)^{n+1}}\nabla^{2n+3}\times\left(\frac{1}{\varepsilon}\nabla\times H-\frac{1}{\varepsilon}J\right)+\sum_{m=1}^{n+1}\frac{(-1)^{m+1}}{(\varepsilon\mu)^{m}}\nabla^{2m-1}\times\frac{\partial^{2(n+1-m)+2}J}{\partial t^{2(n+1-m)+2}}=\frac{(-1)^{n+2}}{(\varepsilon\mu)^{n+2}}\nabla^{2n+4}\times H-\frac{(-1)^{n+2}}{(\varepsilon\mu)^{n+2}}\nabla^{2n+3}\times J+\sum_{m=0}^{n}\frac{(-1)^{m}}{(\varepsilon\mu)^{m+1}}\nabla^{2m+1}\times\frac{\partial^{2(n-m)+2}J}{\partial t^{2(n-m)+2}}$$
$$=\frac{(-1)^{n+2}}{(\varepsilon\mu)^{n+2}}\nabla^{2(n+2)}\times H+\sum_{m=0}^{n+1}\frac{(-1)^{m}}{(\varepsilon\mu)^{m+1}}\nabla^{2m+1}\times\frac{\partial^{2(n+1-m)}J}{\partial t^{2(n+1-m)}}$$

The above is (1.5) when substituting n with $n+1$. So, (1.5) holds for $n+1$.

Taking the temporal derivative on (1.7) and substituting (1.1) into it, we have

$$\frac{\partial^{2(n+1)+1}E}{\partial t^{2(n+1)+1}}=\frac{(-1)^{n+1}}{(\varepsilon\mu)^{n+1}}\nabla^{2(n+1)}\times\left(\frac{1}{\varepsilon}\nabla\times H-\frac{1}{\varepsilon}J\right)+\frac{1}{\varepsilon}\sum_{m=0}^{n}\frac{(-1)^{m+1}}{(\varepsilon\mu)^{m}}\nabla^{2m}\times\frac{\partial^{2(n-m)+2}J}{\partial t^{2(n-m)+2}}=\frac{1}{\varepsilon}\frac{(-1)^{n+1}}{(\varepsilon\mu)^{n+1}}\nabla^{2(n+1)+1}\times H-\frac{1}{\varepsilon}\frac{(-1)^{n+1}}{(\varepsilon\mu)^{n+1}}\nabla^{2(n+1)}\times J+\frac{1}{\varepsilon}\sum_{m=0}^{n}\frac{(-1)^{m+1}}{(\varepsilon\mu)^{m}}\nabla^{2m}\times\frac{\partial^{2(n-m+1)}J}{\partial t^{2(n-m+1)}}$$

$$\boxed{\frac{\partial^{2(n+1)+1}E}{\partial t^{2(n+1)+1}}=\frac{1}{\varepsilon}\frac{(-1)^{n+1}}{(\varepsilon\mu)^{n+1}}\nabla^{2(n+1)+1}\times H+\frac{1}{\varepsilon}\sum_{m=0}^{n+1}\frac{(-1)^{m+1}}{(\varepsilon\mu)^{m}}\nabla^{2m}\times\frac{\partial^{2(n+1-m)}J}{\partial t^{2(n+1-m)}}} \tag{1.13}$$

The above is (1.6) when substiuting n with $n+1$. So, (1.6) holds for $n+1$.

Taking the temporal derivative on (1.13) and substituting (1.2) into it, we have

$$\frac{\partial^{2n+4}E}{\partial t^{2n+4}}=\frac{1}{\varepsilon}\frac{(-1)^{n+1}}{(\varepsilon\mu)^{n+1}}\nabla^{2n+3}\times\left(-\frac{1}{\mu}\nabla\times E\right)+\frac{1}{\varepsilon}\sum_{m=0}^{n+1}\frac{(-1)^{m+1}}{(\varepsilon\mu)^{m}}\nabla^{2m}\times\frac{\partial^{2(n+1-m)+1}J}{\partial t^{2(n+1-m)+1}}=\frac{(-1)^{n+2}}{(\varepsilon\mu)^{n+2}}\nabla^{2(n+2)}\times E+\frac{1}{\varepsilon}\sum_{m=0}^{n+1}\frac{(-1)^{m+1}}{(\varepsilon\mu)^{m}}\nabla^{2m}\times\frac{\partial^{2(n+1-m)+1}J}{\partial t^{2(n+1-m)+1}}$$

The above is (1.7) when substituting n with $n+1$. So, (1.7) holds for $n+1$.

The above proved that (1.4) – (1.7) hold for $n+1$.

Thus, (1.4) – (1.7) hold for $n\geq 0$.

QED.

Maxwell's equations give a relation between the first order spacial curl and the first order temporal derivative. This theorem says that such a relation exists for any orders.

This theorem does not give new information other than what Maxwell's equations already give. However, once used in discretized time and space, all higher order information contained in Maxwell's equations is lost except the first order information. That is why the time forward estimation order can only be 2 by the traditional Yee-algorithm.

This theorem extends Maxwell's equations to arbitrary orders. Thus, time forward estimation of arbitrary order can be made. Two examples of arbitrary order FDTD algorithms are presented below.

2N order time forward estimation

By Taylor's series, we have

$$H\left(t+\frac{1}{2}\Delta_t\right)=H\left(t-\frac{1}{2}\Delta_t\right)+2\sum_{n=0}^{\infty}\frac{\left(\frac{1}{2}\Delta_t\right)^{2n+1}}{(2n+1)!}\frac{\partial^{2n+1}H(t)}{\partial t^{2n+1}}$$

$$E(t+\Delta_t)=E(t)+2\sum_{n=0}^{\infty}\frac{\left(\frac{1}{2}\Delta_t\right)^{2n+1}}{(2n+1)!}\frac{\partial^{2n+1}E(t+\frac{1}{2}\Delta_t)}{\partial t^{2n+1}}$$

Applying (1.4) and (1.6) of the Time-Space Theorem, we have

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

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

We get following $2N$ order time forward estimation.

$$\boxed{H\left(t+\frac{1}{2}\Delta_t\right)=H\left(t-\frac{1}{2}\Delta_t\right)+\frac{2}{\mu}\sum_{n=0}^{N-1}\frac{\left(\frac{1}{2}\Delta_t\right)^{2n+1}}{(2n+1)!}\frac{(-1)^{n+1}}{(\varepsilon\mu)^{n}}\nabla^{2n+1}\times E(t)+S_h(t)+O\left(\left(\frac{1}{2}\Delta_t\right)^{2N}\right)} \tag{2.1}$$

$$\boxed{E(t+\Delta_t)=E(t)+\frac{2}{\varepsilon}\sum_{n=0}^{N-1}\frac{\left(\frac{1}{2}\Delta_t\right)^{2n+1}}{(2n+1)!}\frac{(-1)^{n}}{(\varepsilon\mu)^{n}}\nabla^{2n+1}\times H\left(t+\frac{1}{2}\Delta_t\right)+S_e\left(t+\frac{1}{2}\Delta_t\right)+O\left(\left(\frac{1}{2}\Delta_t\right)^{2N}\right)} \tag{2.2}$$

$S_h(t) = 2\sum_{n=0}^{N-1}\frac{\left(\frac{1}{2}\Delta_t\right)^{2n+1}}{(2n+1)!}\begin{cases}\vec{0}, n = 0 \\ \sum_{m=1}^{n}\frac{(-1)^{m+1}}{(\varepsilon\mu)^m}\nabla^{2m-1}\times\frac{\partial^{2(n-m)+1}J(t)}{\partial t^{2(n-m)+1}}, n > 0\end{cases}$	(2.3)
$S_e\left(t+\frac{1}{2}\Delta_t\right) = \frac{2}{\varepsilon}\sum_{n=0}^{N-1}\frac{\left(\frac{1}{2}\Delta_t\right)^{2n+1}}{(2n+1)!}\sum_{m=0}^{n}\frac{(-1)^{m+1}}{(\varepsilon\mu)^m}\nabla^{2m}\times\frac{\partial^{2(n-m)}J\left(t+\frac{1}{2}\Delta_t\right)}{\partial t^{2(n-m)}}$	(2.4)

2N+1 order time forward estimation

By Taylor's series, we have

$$H(t+\Delta_t) + H(t-\Delta_t) = 2\sum_{n=0}^{\infty}\frac{(\Delta_t)^{2n}}{(2n)!}\frac{\partial^{2n}H(t)}{\partial t^{2n}}$$

$$E(t+\Delta_t) + E(t-\Delta_t) = 2\sum_{n=0}^{\infty}\frac{(\Delta_t)^{2n}}{(2n)!}\frac{\partial^{2n}E(t)}{\partial t^{2n}}$$

Time forward becomes

$$H(t+\Delta_t) = 2H(t) - H(t-\Delta_t) + 2\sum_{n=0}^{\infty}\frac{(\Delta_t)^{2(n+1)}}{(2(n+1)!}\frac{\partial^{2(n+1)}H(t)}{\partial t^{2(n+1)}}$$

$$E(t+\Delta_t) = 2E(t) - E(t-\Delta_t) + 2\sum_{n=0}^{\infty}\frac{(\Delta_t)^{2(n+1)}}{(2(n+1))!}\frac{\partial^{2(n+1)}E(t)}{\partial t^{2(n+1)}}$$

Applying (1.5) and (1.7) of the Time-Space Theorem, we have

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

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

We get following $2N+1$ order time forward estimation.

$H(t+\Delta_t) = 2H(t) - H(t-\Delta_t) + 2\sum_{n=0}^{N-1}\frac{(\Delta_t)^{2(n+1)}}{(2(n+1)!}\frac{(-1)^{n+1}}{(\varepsilon\mu)^{n+1}}\nabla^{2(n+1)}\times H(t) + S_h(t) + O((\Delta_t)^{2N+1})$	(3.1)
$E(t+\Delta_t) = 2E(t) - E(t-\Delta_t) + 2\sum_{n=0}^{N-1}\frac{(\Delta_t)^{2(n+1)}}{(2(n+1))!}\frac{(-1)^{n+1}}{(\varepsilon\mu)^{n+1}}\nabla^{2(n+1)}\times E(t) + S_e(t) + O((\Delta_t)^{2N+1})$	(3.2)
$S_h(t) = 2\sum_{n=0}^{N-1}\frac{(\Delta_t)^{2(n+1)}}{(2(n+1)!}\sum_{m=0}^{n}\frac{(-1)^m}{(\varepsilon\mu)^{m+1}}\nabla^{2m+1}\times\frac{\partial^{2(n-m)}J(t)}{\partial t^{2(n-m)}}$	(3.3)
$S_e(t) = \frac{2}{\varepsilon}\sum_{n=0}^{N-1}\frac{(\Delta_t)^{2(n+1)}}{(2(n+1))!}\sum_{m=0}^{n}\frac{(-1)^{m+1}}{(\varepsilon\mu)^m}\nabla^{2m}\times\frac{\partial^{2(n-m)+1}J(t)}{\partial t^{2(n-m)+1}}$	(3.4)

(3.1) only involves magnetic field, and (3.2) only involves electric field. It is possible to just simulate single field if boundary condition can be handled properly. [10] and [11] demonstrate advantages of single field simulation.

This algorithm does not use Yee-digitization. [12] mentions the advantages of not using Yee-digitization.

Arbitrary Order Spatial Estimations

The arbitrary order time forward equations (2.1) – (2.4) and (3.1) – (3.4) require spacial curls. Before presenting algorithms for arbitrary order estimations of curls, reform equations (2.1) – (2.4) and (3.1) – (3.4) into recursive formations. Define following constants to simplify the notations.

$c_r = c\frac{\Delta_t}{\Delta_s}$	(4.1)
$\eta = \sqrt{\frac{\mu}{\varepsilon}}$	(4.2)

(2N,2M) order Yee-algorithm

2N order recursive time forward

Using (4.1) and (4.2), equations (2.1) and (2.2) are reformed into following recursive formation.

$H\left(t+\frac{1}{2}\Delta_t\right) \approx H\left(t-\frac{1}{2}\Delta_t\right)+\sum_{n=0}^{N-1} G_{e,n}+S_h(t)$	(5.1)
$E(t+\Delta_t) \approx E(t)+\sum_{n=0}^{N-1} G_{h,n}+S_e\left(t+\frac{1}{2}\Delta_t\right)$	(5.2)
$G_{e,0} = -\frac{c_r}{\eta}\Delta_s\nabla\times E(t)$	(5.3)
$G_{e,n} = g_n\Delta_s^2\nabla^2 G_{e,n-1}$	(5.4)
$G_{h,0} = \eta c_r\Delta_s\nabla\times H\left(t+\frac{1}{2}\Delta_t\right)$	(5.5)
$G_{h,n} = g_n\Delta_s^2\nabla^2 G_{h,n-1}$	(5.6)
$g_n = \frac{c_r^2}{8(2n+1)n}; n>0$	(5.7)

In deduction of the above algorithm, it uses the fact that the divergence of a curl is 0, and thus, we have

$$\nabla^2 \times = \nabla(\nabla\cdot) - \nabla^2 = -\nabla^2$$

2M order spatial estimations

We only need to estimate $\Delta_s\nabla\times$ and Laplacian $\Delta_s^2\nabla^2$.

Odd order derivative estimation theorem (Yee-sampling). For a sufficiently smooth scalar function $f(s)$, a $2M$ order estimation of its odd order derivatives scaled by a sampling size is a M -dimentional constant linear combination of nearby finite-differences,

$\frac{\left(\frac{\Delta_s}{2}\right)^{2h+1}}{(2h+1)!}\frac{\partial^{2h+1}f(s)}{\partial s^{2h+1}} = \frac{1}{2}\boldsymbol{a}_h\cdot\boldsymbol{d}+O(\Delta_s^{2M}); h=0,1,\dots,M-1$	(a.1)
$\boldsymbol{d}=\begin{bmatrix} d_0 \\ d_1 \\ \vdots \\ d_{M-1}\end{bmatrix}; d_m = f\left(s+(m+\frac{1}{2})\Delta_s\right)-f\left(s-(m+\frac{1}{2})\Delta_s\right)$ $M>0$ $\Delta_s>0$	(a.2)
$\boldsymbol{A}_M=\begin{bmatrix} 1 & 1 & \cdots & 1 \\ 3 & 3^3 & \cdots & 3^{2M-1} \\ \vdots & \vdots & \ddots & \vdots \\ 2M-1 & (2M-1)^3 & \cdots & (2M-1)^{2M-1}\end{bmatrix}$	(a.3)
$\boldsymbol{A}_M^{-1}=\begin{bmatrix} \boldsymbol{a}_0^T \\ \boldsymbol{a}_1^T \\ \vdots \\ \boldsymbol{a}_{M-1}^T\end{bmatrix}$	(a.4)

Proof. From Taylor's series,

$$f\left(s+(m+\frac{1}{2})\Delta_s\right)=\sum_{h=0}^{\infty}\frac{(m+\frac{1}{2})^h}{h!}\Delta_s^h\frac{\partial^h f(s)}{\partial s^h}$$

$$f\left(s-(m+\frac{1}{2})\Delta_s\right)=\sum_{h=0}^{\infty}(-1)^h\frac{(m+\frac{1}{2})^h}{h!}\Delta_s^h\frac{\partial^h f(s)}{\partial s^h}$$

we have

$$d_m = 2\sum_{h=0}^{\infty}\frac{(m+\frac{1}{2})^{2h+1}}{(2h+1)!}\Delta_s^{2h+1}\frac{\partial^{2h+1}f(s)}{\partial s^{2h+1}}$$

$$m=0,1,2,\dots,M-1; M>0$$

It can be writtern as

$$d_m = 2\sum_{h=0}^{M-1}(2m+1)^{2h+1}u_h + O\left(\left(\frac{1}{2}\Delta_s\right)^{2M}\right)$$

by defining

$$u_h = \frac{\left(\frac{\Delta_s}{2}\right)^{2h+1}}{(2h+1)!}\frac{\partial^{2h+1}f(s)}{\partial s^{2h+1}}$$

Writtern in matrix form using (a.3), we have

$$\boldsymbol{d} = 2\boldsymbol{A}_M\boldsymbol{u} + O\left(\left(\frac{1}{2}\Delta_s\right)^{2M}\right)$$

by defining

$$\boldsymbol{u}=\begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{M-1}\end{bmatrix}$$

We get a $2M$ order estimation of vector $\boldsymbol{u}$ by

$$\boldsymbol{u} = \frac{1}{2}\boldsymbol{A}_M^{-1}\boldsymbol{d} + O\left(\left(\frac{1}{2}\Delta_s\right)^{2M}\right)$$

By (a.4) we get

$$\frac{\left(\frac{\Delta_s}{2}\right)^{2h+1}}{(2h+1)!}\frac{\partial^{2h+1}f(s)}{\partial s^{2h+1}} = \frac{1}{2}\boldsymbol{a}_h \cdot \boldsymbol{d} + O\left(\left(\frac{1}{2}\Delta_s\right)^{2M}\right); h = 0,1,\dots,M-1$$

QED

By this theorem. We have 2M order estimation for operator ∂/∂

$$\Delta_s\frac{\partial f(s)}{\partial s} = \boldsymbol{a}_0 \cdot \boldsymbol{d} + O\left(\left(\frac{1}{2}\Delta_s\right)^{2M}\right) \tag{a.5}$$

With the above result, we have following curl estimation theorem.

For a 3D vector $F(x,y,z)$ the finite deffrences defined in (a.2) are expressed using subscripts.

$$d_{u,v,m}(F)$$

Where the first index u indicates the vector component, the second index v indicates the axis, and the third index m indicates the sampling distance. The finite differences form a M –dimensional vector

$$\boldsymbol{d}_{u,v}(F) = \begin{bmatrix} d_{u,v,0}(F) \\ d_{u,v,1}(F) \\ \vdots \\ d_{u,v,M-1}(F) \end{bmatrix} \tag{a.6}$$

For example,

$$\boldsymbol{d}_{x,y}(F) = \begin{bmatrix} d_{x,y,0}(F) \\ d_{x,y,1}(F) \\ \vdots \\ d_{x,y,M-1}(F) \end{bmatrix}; d_{x,y,m}(F) = F_x\left(x, y + \left(m+\frac{1}{2}\right)\Delta_s, z\right) - F_x\left(x, y - \left(m+\frac{1}{2}\right)\Delta_s, z\right)$$

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

$$\Delta_s\nabla \times F = \boldsymbol{a}_0 \cdot \begin{bmatrix} \boldsymbol{d}_{z,y} - \boldsymbol{d}_{y,z} \\ \boldsymbol{d}_{x,z} - \boldsymbol{d}_{z,x} \\ \boldsymbol{d}_{y,x} - \boldsymbol{d}_{x,y} \end{bmatrix}(F) + O\left(\left(\frac{1}{2}\Delta_s\right)^{2M}\right) \tag{a.7}$$

Where $\boldsymbol{a}_0$ is a constant vector given by (a.3) and (a.4).

Proof. By (a.5),

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

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

QED

To estimate a Laplacian, we need following theorem.

Even order derivative estimation theorem. For a sufficiently smooth scalar function $f(s)$, a $2M+1$ order estimation of its even order derivatives scaled by a sampling size is a M -dimentional constant linear combination of nearby finite-differences,

$$\frac{(\Delta_s)^{2(h+1)}}{(2(h+1))!}\frac{\partial^{2(h+1)}f(s)}{\partial s^{2(h+1)}} = \frac{1}{2}\boldsymbol{b}_h \cdot \boldsymbol{g} + O((\Delta_s)^{2M+1}); h = 0,1,\dots,M-1 \tag{g.1}$$

$$\boldsymbol{g} = \begin{bmatrix} g_0 \\ g_1 \\ \vdots \\ g_{M-1} \end{bmatrix}; \; g_m = f(s+(m+1)\Delta_s) + f(s-(m+1)\Delta_s) - 2f(s) \tag{g.2}$$

$$\boldsymbol{B}_M = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 2^2 & 2^4 & \cdots & 2^{2M} \\ \vdots & \vdots & \ddots & \vdots \\ M^2 & M^4 & \cdots & M^{2M} \end{bmatrix} \tag{g.3}$$

$$\boldsymbol{B}_M^{-1} = \begin{bmatrix} \boldsymbol{b}_0^T \\ \boldsymbol{b}_1^T \\ \vdots \\ \boldsymbol{b}_{M-1}^T \end{bmatrix}$$	(g.4)

Proof. From Taylor's series,

$$f(s + (m+1)\Delta_s) = \sum_{h=0}^{\infty} \frac{(m+1)^h}{h!} \Delta_s^h \frac{\partial^h f(s)}{\partial s^h}$$

$$f(s - (m+1)\Delta_s) = \sum_{h=0}^{\infty} (-1)^h \frac{(m+1)^h}{h!} \Delta_s^h \frac{\partial^h f(s)}{\partial s^h}$$

we have

$$g_m = 2 \sum_{h=0}^{\infty} \frac{(m+1)^{2(h+1)}}{(2(h+1))!} \Delta_s^{2(h+1)} \frac{\partial^{2(h+1)} f(s)}{\partial s^{2(h+1)}}$$

$$m = 0,1,2,\dots,M-1; M > 0$$

It can be writtern as

$$g_m = 2 \sum_{h=0}^{M-1} (m+1)^{2(h+1)} v_h + O((\Delta_s)^{2M+1})$$

by defining

$$v_h = \frac{(\Delta_s)^{2(h+1)}}{(2(h+1))!} \frac{\partial^{2(h+1)} f(s)}{\partial s^{2(h+1)}}$$

Writtern in matrix form using (g.3), we have

$$\boldsymbol{g} = 2\boldsymbol{B}_M \boldsymbol{v} + O((\Delta_s)^{2M+1})$$

by defining

$$\boldsymbol{v} = \begin{bmatrix} v_0 \\ v_1 \\ \vdots \\ v_{M-1} \end{bmatrix}$$

We get a $2M+1$ order estimation of vector $\boldsymbol{v}$ by

$$\boldsymbol{v} = \frac{1}{2} \boldsymbol{B}_M^{-1} \boldsymbol{g} + O((\Delta_s)^{2M+1})$$

By (g.4), we get

$$\frac{(\Delta_s)^{2(h+1)}}{(2(h+1))!} \frac{\partial^{2(h+1)} f(s)}{\partial s^{2(h+1)}} = \frac{1}{2} \boldsymbol{b}_h \cdot \boldsymbol{g} + O((\Delta_s)^{2M+1})$$

QED

By this theorem. We have 2M+1 order estimation for operator ∂^2/∂^2

$$\Delta_s^2 \frac{\partial^2 f(s)}{\partial s^2} = \boldsymbol{b}_0 \cdot \boldsymbol{g} + O((\Delta_s)^{2M+1})$$	(g.5)

With the above result, we have following Laplacian estimation theorem.

To estimate a Laplacian, we need to apply estimator (g.5) to 3D vectors. Use following notation to express finite differences defined in (g.2) for 3D vectors.

$g_{u,v,m}(F) = F_u(v + (m+1)\Delta_s) + F_u(v - (m+1)\Delta_s) - 2F_u(v)$	(g.6)

Where the first index u indicates the vector component, the second index v indicates the axis, and the third index m indicates the sampling distance. For example,

$$\boldsymbol{g}_{x,y}(F) = \begin{bmatrix} g_{x,y,0}(F) \\ g_{x,y,1}(F) \\ \vdots \\ g_{x,y,M-1}(F) \end{bmatrix}$$

$$g_{x,y,m}(F) = F_x(x, y + (m+1)\Delta_s, z) + F_x(x, y - (m+1)\Delta_s, z) - 2F_x(x,y,z)$$

If vector F can be deduced by context without ambiguity then simply write the difference as

$$\boldsymbol{g}_{x,y} = \begin{bmatrix} g_{x,y,0} \\ g_{x,y,1} \\ \vdots \\ g_{x,y,M-1} \end{bmatrix}$$

Laplacian estimation theorem. For a sufficiently smooth 3D field $F(x, y, z)$, a 2*M+1* order estimation of its Laplacian scaled by a sampling size is a M -dimentional constant linear combination of nearby finite-differences,

$$\Delta_s^2 \nabla^2 F = \begin{bmatrix} \boldsymbol{g}_{x,x} + \boldsymbol{g}_{x,y} + \boldsymbol{g}_{x,z} \\ \boldsymbol{g}_{y,x} + \boldsymbol{g}_{y,y} + \boldsymbol{g}_{y,z} \\ \boldsymbol{g}_{z,x} + \boldsymbol{g}_{z,y} + \boldsymbol{g}_{z,z} \end{bmatrix}^T \boldsymbol{b}_0 + \boldsymbol{O}((\Delta_s)^{2M+1}) \quad \text{(g.7)}$$

Where $\boldsymbol{b}_0$ is a constant vector given by (g.4) and (g.3).

Proof. By (g.5), a Laplacian can be expressed as

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

$$= \begin{bmatrix} \boldsymbol{g}_{x,x} + \boldsymbol{g}_{x,y} + \boldsymbol{g}_{x,z} \\ \boldsymbol{g}_{y,x} + \boldsymbol{g}_{y,y} + \boldsymbol{g}_{y,z} \\ \boldsymbol{g}_{z,x} + \boldsymbol{g}_{z,y} + \boldsymbol{g}_{z,z} \end{bmatrix}^T \boldsymbol{b}_0 + O((\Delta_s)^{2M+1})$$

QED

By now, estimations of all operators used in the (2N,2M) order Yee-algorithm are provided by constant linear combinations of finite-differences.

We have one more issue to deal with. It cannot be taken for granted that $\nabla \times E$ and $\nabla \times H$ given by curl estimator (a.7) are at the desired space locations, and that $\nabla^2 \times G_{e,n-1}$ and $\nabla^2 \times G_{h,n-1}$ given by Laplacian estimator (g.7) are at the desired space locations. We need to prove that.

Space locations of estimations

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.

$$\begin{gathered} x_i = x_{min} + i\Delta_s;\ i = 0,1,2,\dots,n_x \\ y_j = y_{min} + j\Delta_s;\ j = 0,1,2,\dots,n_y \\ z_k = z_{min} + k\Delta_s; k = 0,1,2,\dots,n_z \\ \\ n_x > 0, n_y > 0, n_z > 0, \Delta_s > 0 \end{gathered} \quad \text{(s.1)}$$

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:

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

$$\mathbb{Y}_h(F[i,j,k]) = \begin{bmatrix} F_x\left(x_i, y_j + \frac{1}{2}\Delta_s, z_k + \frac{1}{2}\Delta_s\right) \\ F_y\left(x_i + \frac{1}{2}\Delta_s, y_j, z_k + \frac{1}{2}\Delta_s\right) \\ F_z\left(x_i + \frac{1}{2}\Delta_s, y_j + \frac{1}{2}\Delta_s, z_k\right) \end{bmatrix} \quad \text{(s.3)}$$

For a "Yee-E vector", the finite-difference (a.6) is expressed by

$$\begin{gathered} \boldsymbol{d}_{u,v}\big(\mathbb{Y}_e(F)\big) = \begin{bmatrix} d_{u,v,0}(\mathbb{Y}_e(F)) \\ d_{u,v,1}(\mathbb{Y}_e(F)) \\ \vdots \\ d_{u,v,M-1}(\mathbb{Y}_e(F)) \end{bmatrix} \\ d_{u,v,m}\big(\mathbb{Y}_e(F)\big) = \mathbb{Y}_e(F_u)(v + (m+1)\Delta_s) - \mathbb{Y}_e(F_u)(v - m\Delta_s) \end{gathered} \quad \text{(s.4)}$$

For a "Yee-H vector", the finite-difference (a.6) is expressed by

$$\begin{gathered} \boldsymbol{d}_{u,v}\big(\mathbb{Y}_h(F)\big) = \begin{bmatrix} d_{u,v,0}(\mathbb{Y}_h(F)) \\ d_{u,v,1}(\mathbb{Y}_h(F)) \\ \vdots \\ d_{u,v,M-1}(\mathbb{Y}_h(F)) \end{bmatrix} \\ d_{u,v,m}\big(\mathbb{Y}_h(F)\big) = \mathbb{Y}_h(F_u)(v + m\Delta_s) - \mathbb{Y}_h(F_u)(v - (m+1)\Delta_s) \end{gathered} \quad \text{(s.5)}$$

For a "Yee-E vector", the finite-difference (g.6) is expressed by

$$\boldsymbol{g}_{u,v}\big(\mathbb{Y}_e(F)\big)=\begin{bmatrix} g_{u,v,0}(\mathbb{Y}_e(F)) \\ g_{u,v,1}(\mathbb{Y}_e(F)) \\ \vdots \\ g_{u,v,M-1}(\mathbb{Y}_e(F)) \end{bmatrix}$$
$$g_{u,v,m}\big(\mathbb{Y}_e(F)\big)=\mathbb{Y}_e(F_u)(v+(m+1)\Delta_s)+\mathbb{Y}_e(F_u)(v-(m+1)\Delta_s)-2\mathbb{Y}_e(F_u)(v) \tag{s.6}$$

For a "Yee-H vector", the finite-difference (g.6) is expressed by

$$\boldsymbol{g}_{u,v}\big(\mathbb{Y}_h(F)\big)=\begin{bmatrix} g_{u,v,0}(\mathbb{Y}_h(F)) \\ g_{u,v,1}(\mathbb{Y}_h(F)) \\ \vdots \\ g_{u,v,M-1}(\mathbb{Y}_h(F)) \end{bmatrix}$$
$$g_{u,v,m}\big(\mathbb{Y}_h(F)\big)=\mathbb{Y}_h(F_u)(v+(m+1)\Delta_s)+\mathbb{Y}_h(F_u)(v-(m+1)\Delta_s)-2\mathbb{Y}_h(F_u)(v) \tag{s.7}$$

Location theorem of curl estimation. Applying curl estimator (a.7) to a "Yee-E vector" gives a "Yee-H vector"; applying curl estimator (a.7) to a "Yee-H vector" gives a "Yee-E vector".

Proof. Applying (a.7) to a 3D vector gives another 3D vector,

$$V=\begin{bmatrix} V_x \\ V_y \\ V_z \end{bmatrix}=\boldsymbol{a}_0\cdot\begin{bmatrix} \boldsymbol{d}_{z,y}-\boldsymbol{d}_{y,z} \\ \boldsymbol{d}_{x,z}-\boldsymbol{d}_{z,x} \\ \boldsymbol{d}_{y,x}-\boldsymbol{d}_{x,y} \end{bmatrix}(F)$$

Applying (a.7) to a "Yee-E vector",

$$\boldsymbol{d}_{z,y}-\boldsymbol{d}_{y,z}=\begin{bmatrix} d_{z,y,0}\big(\mathbb{Y}_e(F[i,j,k])\big)-d_{y,z,0}(\mathbb{Y}_e(F[i,j,k])) \\ d_{z,y,1}\big(\mathbb{Y}_e(F[i,j,k])\big)-d_{y,z,1}(\mathbb{Y}_e(F[i,j,k])) \\ \vdots \\ d_{z,y,M-1}\big(\mathbb{Y}_e(F[i,j,k])\big)-d_{y,z,M-1}(\mathbb{Y}_e(F[i,j,k])) \end{bmatrix}$$

$$\boldsymbol{d}_{x,z}-\boldsymbol{d}_{z,x}=\begin{bmatrix} d_{x,z,0}\big(\mathbb{Y}_e(F[i,j,k])\big)-d_{z,x,0}(\mathbb{Y}_e(F[i,j,k])) \\ d_{x,z,1}\big(\mathbb{Y}_e(F[i,j,k])\big)-d_{z,x,1}(\mathbb{Y}_e(F[i,j,k])) \\ \vdots \\ d_{x,z,M-1}\big(\mathbb{Y}_e(F[i,j,k])\big)-d_{z,x,M-1}(\mathbb{Y}_e(F[i,j,k])) \end{bmatrix}$$

$$\boldsymbol{d}_{y,x}-\boldsymbol{d}_{x,y}=\begin{bmatrix} d_{y,x,0}\big(\mathbb{Y}_e(F[i,j,k])\big)-d_{x,y,0}(\mathbb{Y}_e(F[i,j,k])) \\ d_{y,x,1}\big(\mathbb{Y}_e(F[i,j,k])\big)-d_{x,y,1}(\mathbb{Y}_e(F[i,j,k])) \\ \vdots \\ d_{y,x,M-1}\big(\mathbb{Y}_e(F[i,j,k])\big)-d_{x,y,M-1}(\mathbb{Y}_e(F[i,j,k])) \end{bmatrix}$$

Because finite-differences are formed by symmetric samplings, the space location of a finite difference is the average of all space points.

By (a.6) and (s.4), each finite-difference in $\boldsymbol{d}_{z,y}-\boldsymbol{d}_{y,z}$ is defined by

$$d_{z,y,m}(\mathbb{Y}_e(F[i,j,k]))=F_z\left(x_i,y_j+(m+1)\Delta_s,z_k+\frac{1}{2}\Delta_s\right)-F_z\left(x_i,y_j-m\Delta_s,z_k+\frac{1}{2}\Delta_s\right)$$

$$d_{y,z,m}(\mathbb{Y}_e(F[i,j,k]))=F_y\left(x_i,y_j+\frac{1}{2}\Delta_s,z_k+(m+1)\Delta_s\right)-F_y\left(x_i,y_j+\frac{1}{2}\Delta_s,z_k-m\Delta_s\right)$$

The space location of the two differences are the same and unrelated to the value of m:

$$\frac{\left(x_i,y_j+(m+1)\Delta_s,z_k+\frac{1}{2}\Delta_s\right)+\left(x_i,y_j-m\Delta_s,z_k+\frac{1}{2}\Delta_s\right)}{2}=\left(x_i,y_j+\frac{1}{2}\Delta_s,z_k+\frac{1}{2}\Delta_s\right)$$

$$\frac{\left(x_i,y_j+\frac{1}{2}\Delta_s,z_k+(m+1)\Delta_s\right)+\left(x_i,y_j+\frac{1}{2}\Delta_s,z_k-m\Delta_s\right)}{2}=\left(x_i,y_j+\frac{1}{2}\Delta_s,z_k+\frac{1}{2}\Delta_s\right)$$

By (s.3), we have

$$(\Delta_s\nabla\times F)_x\approx\boldsymbol{a}_0\cdot\left(\boldsymbol{d}_{z,y}-\boldsymbol{d}_{y,z}\right)=\mathbb{Y}_h(V_x[i,j,k])$$

By (a.6) and (s.4), each finite-difference in $\boldsymbol{d}_{x,z}-\boldsymbol{d}_{z,x}$ is defined by

$$d_{x,z,m}(\mathbb{Y}_e(F[i,j,k]))=F_x\left(x_i+\frac{1}{2}\Delta_s,y_j,z_k+(m+1)\Delta_s\right)-F_x\left(x_i+\frac{1}{2}\Delta_s,y_j,z_k-m\Delta_s\right)$$

$$d_{z,x,m}(\mathbb{Y}_e(F[i,j,k]))=F_z\left(x_i+(m+1)\Delta_s,y_j,z_k+\frac{1}{2}\Delta_s\right)-F_z\left(x_i-m\Delta_s,y_j,z_k+\frac{1}{2}\Delta_s\right)$$

The space location of the two differences are the same and unrelated to the value of m:

$$\frac{\left(x_i+\frac{1}{2}\Delta_s,y_j,z_k+(m+1)\Delta_s\right)+\left(x_i+\frac{1}{2}\Delta_s,y_j,z_k-m\Delta_s\right)}{2}=\left(x_i+\frac{1}{2}\Delta_s,y_j,z_k+\frac{1}{2}\Delta_s\right)$$

$$\frac{\left(x_i+(m+1)\Delta_s,y_j,z_k+\frac{1}{2}\Delta_s\right)+\left(x_i-m\Delta_s,y_j,z_k+\frac{1}{2}\Delta_s\right)}{2}=\left(x_i+\frac{1}{2}\Delta_s,y_j,z_k+\frac{1}{2}\Delta_s\right)$$

By (s.3), we have

$$(\Delta_s\nabla\times F)_y\approx \boldsymbol{a}_0\cdot\left(\boldsymbol{d}_{x,z}-\boldsymbol{d}_{z,x}\right)=\mathbb{Y}_h\left(V_y[i,j,k]\right)$$

By (a.6) and (s.4), each finite-difference in $\boldsymbol{d}_{y,x}-\boldsymbol{d}_{x,y}$ is defined by

$$d_{y,x,m}(\mathbb{Y}_e(F[i,j,k]))=F_y\left(x_i+(m+1)\Delta_s,y_j+\frac{1}{2}\Delta_s,z_k\right)-F_y\left(x_i-m\Delta_s,y_j+\frac{1}{2}\Delta_s,z_k\right)$$

$$d_{x,y,m}(\mathbb{Y}_e(F[i,j,k]))=F_x\left(x_i+\frac{1}{2}\Delta_s,y_j+(m+1)\Delta_s,z_k\right)-F_x\left(x_i+\frac{1}{2}\Delta_s,y_j-m\Delta_s,z_k\right)$$

The space location of the two differences are the same and unrelated to the value of m:

$$\frac{\left(x_i+(m+1)\Delta_s,y_j+\frac{1}{2}\Delta_s,z_k\right)+\left(x_i-m\Delta_s,y_j+\frac{1}{2}\Delta_s,z_k\right)}{2}=\left(x_i+\frac{1}{2}\Delta_s,y_j+\frac{1}{2}\Delta_s,z_k\right)$$

$$\frac{\left(x_i+\frac{1}{2}\Delta_s,y_j+(m+1)\Delta_s,z_k\right)+\left(x_i+\frac{1}{2}\Delta_s,y_j-m\Delta_s,z_k\right)}{2}=\left(x_i+\frac{1}{2}\Delta_s,y_j+\frac{1}{2}\Delta_s,z_k\right)$$

By (s.3), we have

$$(\Delta_s\nabla\times F)_z\approx \boldsymbol{a}_0\cdot\left(\boldsymbol{d}_{y,x}-\boldsymbol{d}_{x,y}\right)=\mathbb{Y}_h(V_z[i,j,k])$$

Combine the above results, we have

$$\boldsymbol{a}_0\cdot\begin{bmatrix}\boldsymbol{d}_{z,y}-\boldsymbol{d}_{y,z}\\ \boldsymbol{d}_{x,z}-\boldsymbol{d}_{z,x}\\ \boldsymbol{d}_{y,x}-\boldsymbol{d}_{x,y}\end{bmatrix}\left(\mathbb{Y}_e(F[i,j,k])\right)=\mathbb{Y}_h(V[i,j,k])$$

The first part of the theorem is proved.

Applying (a.7) to a "Yee-H vector",

$$\boldsymbol{d}_{z,y}-\boldsymbol{d}_{y,z}=\begin{bmatrix}d_{z,y,0}\left(\mathbb{Y}_h(F[i,j,k])\right)-d_{y,z,0}(\mathbb{Y}_h(F[i,j,k]))\\ d_{z,y,1}\left(\mathbb{Y}_h(F[i,j,k])\right)-d_{y,z,1}(\mathbb{Y}_h(F[i,j,k]))\\ \vdots\\ d_{z,y,M-1}\left(\mathbb{Y}_h(F[i,j,k])\right)-d_{y,z,M-1}(\mathbb{Y}_h(F[i,j,k]))\end{bmatrix}$$

$$\boldsymbol{d}_{x,z}-\boldsymbol{d}_{z,x}=\begin{bmatrix}d_{x,z,0}\left(\mathbb{Y}_h(F[i,j,k])\right)-d_{z,x,0}(\mathbb{Y}_h(F[i,j,k]))\\ d_{x,z,1}\left(\mathbb{Y}_h(F[i,j,k])\right)-d_{z,x,1}(\mathbb{Y}_h(F[i,j,k]))\\ \vdots\\ d_{x,z,M-1}\left(\mathbb{Y}_h(F[i,j,k])\right)-d_{z,x,M-1}(\mathbb{Y}_h(F[i,j,k]))\end{bmatrix}$$

$$\boldsymbol{d}_{y,x}-\boldsymbol{d}_{x,y}=\begin{bmatrix}d_{y,x,0}\left(\mathbb{Y}_h(F[i,j,k])\right)-d_{x,y,0}(\mathbb{Y}_h(F[i,j,k]))\\ d_{y,x,1}\left(\mathbb{Y}_h(F[i,j,k])\right)-d_{x,y,1}(\mathbb{Y}_h(F[i,j,k]))\\ \vdots\\ d_{y,x,M-1}\left(\mathbb{Y}_h(F[i,j,k])\right)-d_{x,y,M-1}(\mathbb{Y}_h(F[i,j,k]))\end{bmatrix}$$

By (a.6) and (s.5), each finite-difference in $\boldsymbol{d}_{z,y}-\boldsymbol{d}_{y,z}$ is defined by

$$d_{z,y,m}(\mathbb{Y}_h(F[i,j,k]))=F_z\left(x_i+\frac{1}{2}\Delta_s,y_j+\frac{1}{2}\Delta_s+m\Delta_s,z_k\right)-F_z\left(x_i+\frac{1}{2}\Delta_s,y_j+\frac{1}{2}\Delta_s-(m+1)\Delta_s,z_k\right)$$

$$d_{y,z,m}(\mathbb{Y}_h(F[i,j,k]))=F_y\left(x_i+\frac{1}{2}\Delta_s,y_j,z_k+\frac{1}{2}\Delta_s+m\Delta_s\right)-F_y\left(x_i+\frac{1}{2}\Delta_s,y_j,z_k+\frac{1}{2}\Delta_s-(m+1)\Delta_s\right)$$

The space location of the two differences are the same and unrelated to the value of m:

$$\frac{\left(x_i+\frac{1}{2}\Delta_s,y_j+\frac{1}{2}\Delta_s+m\Delta_s,z_k\right)+\left(x_i+\frac{1}{2}\Delta_s,y_j+\frac{1}{2}\Delta_s-(m+1)\Delta_s,z_k\right)}{2}=\left(x_i+\frac{1}{2}\Delta_s,y_j,z_k\right)$$

$$\frac{\left(x_i+\frac{1}{2}\Delta_s,y_j,z_k+\frac{1}{2}\Delta_s+m\Delta_s\right)+\left(x_i+\frac{1}{2}\Delta_s,y_j,z_k+\frac{1}{2}\Delta_s-(m+1)\Delta_s\right)}{2}=\left(x_i+\frac{1}{2}\Delta_s,y_j,z_k\right)$$

By (s.3), we have

$$(\Delta_s\nabla\times F)_x\approx \boldsymbol{a}_0\cdot\left(\boldsymbol{d}_{z,y}-\boldsymbol{d}_{y,z}\right)=\mathbb{Y}_e(V_x[i,j,k])$$

By (a.6) and (s.5), each finite-difference in $\boldsymbol{d}_{x,z}-\boldsymbol{d}_{z,x}$ is defined by

$$d_{x,z,m}(\mathbb{Y}_h(F[i,j,k]))=F_x\left(x_i,y_j+\frac{1}{2}\Delta_s,z_k+\frac{1}{2}\Delta_s+m\Delta_s\right)-F_x\left(x_i,y_j+\frac{1}{2}\Delta_s,z_k+\frac{1}{2}\Delta_s-(m+1)\Delta_s\right)$$

$$d_{z,x,m}(\mathbb{Y}_h(F[i,j,k])) = F_z\left(x_i + \frac{1}{2}\Delta_s + m\Delta_s, y_j + \frac{1}{2}\Delta_s, z_k\right) - F_z\left(x_i + \frac{1}{2}\Delta_s - (m+1)\Delta_s, y_j + \frac{1}{2}\Delta_s, z_k\right)$$

The space location of the two differences are the same and unrelated to the value of m:

$$\frac{\left(x_i, y_j + \frac{1}{2}\Delta_s, z_k + \frac{1}{2}\Delta_s + m\Delta_s\right) + \left(x_i, y_j + \frac{1}{2}\Delta_s, z_k + \frac{1}{2}\Delta_s - (m+1)\Delta_s\right)}{2} = \left(x_i, y_j + \frac{1}{2}\Delta_s, z_k\right)$$

$$\frac{\left(x_i + \frac{1}{2}\Delta_s + m\Delta_s, y_j + \frac{1}{2}\Delta_s, z_k\right) + \left(x_i + \frac{1}{2}\Delta_s - (m+1)\Delta_s, y_j + \frac{1}{2}\Delta_s, z_k\right)}{2} = \left(x_i, y_j + \frac{1}{2}\Delta_s, z_k\right)$$

By (s.2), we have

$$(\Delta_s \nabla \times F)_y \approx \boldsymbol{a}_0 \cdot (\boldsymbol{d}_{x,z} - \boldsymbol{d}_{z,x}) = \mathbb{Y}_e(V_y[i,j,k])$$

By (a.6) and (s.5), each finite-difference in $\boldsymbol{d}_{y,x} - \boldsymbol{d}_{x,y}$ is defined by

$$d_{y,x,m}(\mathbb{Y}_h(F[i,j,k])) = F_y\left(x_i + \frac{1}{2}\Delta_s + m\Delta_s, y_j, z_k + \frac{1}{2}\Delta_s\right) - F_y\left(x_i + \frac{1}{2}\Delta_s - (m+1)\Delta_s, y_j, z_k + \frac{1}{2}\Delta_s\right)$$

$$d_{x,y,m}(\mathbb{Y}_h(F[i,j,k])) = F_x\left(x_i, y_j + \frac{1}{2}\Delta_s + m\Delta_s, z_k + \frac{1}{2}\Delta_s\right) - F_x\left(x_i, y_j + \frac{1}{2}\Delta_s - (m+1)\Delta_s, z_k + \frac{1}{2}\Delta_s\right)$$

The space location of the two differences are the same and unrelated to the value of m:

$$\frac{\left(x_i + \frac{1}{2}\Delta_s + m\Delta_s, y_j, z_k + \frac{1}{2}\Delta_s\right) + \left(x_i + \frac{1}{2}\Delta_s - (m+1)\Delta_s, y_j, z_k + \frac{1}{2}\Delta_s\right)}{2} = \left(x_i, y_j, z_k + \frac{1}{2}\Delta_s\right)$$

$$\frac{\left(x_i, y_j + \frac{1}{2}\Delta_s + m\Delta_s, z_k + \frac{1}{2}\Delta_s\right) + \left(x_i, y_j + \frac{1}{2}\Delta_s - (m+1)\Delta_s, z_k + \frac{1}{2}\Delta_s\right)}{2} = \left(x_i, y_j, z_k + \frac{1}{2}\Delta_s\right)$$

By (s.2), we have

$$(\Delta_s \nabla \times F)_z \approx \boldsymbol{a}_0 \cdot (\boldsymbol{d}_{y,x} - \boldsymbol{d}_{x,y}) = \mathbb{Y}_e(V_z[i,j,k])$$

Combine the above results, we have

$$\boldsymbol{a}_0 \cdot \begin{bmatrix} \boldsymbol{d}_{z,y} - \boldsymbol{d}_{y,z} \\ \boldsymbol{d}_{x,z} - \boldsymbol{d}_{z,x} \\ \boldsymbol{d}_{y,x} - \boldsymbol{d}_{x,y} \end{bmatrix} (\mathbb{Y}_h(F[i,j,k])) = \mathbb{Y}_e(V[i,j,k])$$

The second part of the theorem is proved.

QED

Location theorem of Laplacian estimation. Applying Laplacia estimator (g.7) to a "Yee-E vector" gives a "Yee-E vector"; applying Laplacian estimator (g.7) to a "Yee-H vector" gives a "Yee-H vector".

Proof. Applying (g.7) to a 3D vector gives another 3D vector F,

$$V = \begin{bmatrix} V_x \\ V_y \\ V_z \end{bmatrix} = \begin{bmatrix} \boldsymbol{g}_{x,x}(F) + \boldsymbol{g}_{x,y}(F) + \boldsymbol{g}_{x,z}(F) \\ \boldsymbol{g}_{y,x}(F) + \boldsymbol{g}_{y,y}(F) + \boldsymbol{g}_{y,z}(F) \\ \boldsymbol{g}_{z,x}(F) + \boldsymbol{g}_{z,y}(F) + \boldsymbol{g}_{z,z}(F) \end{bmatrix}^T \boldsymbol{b}_0$$

Applying (g.7) to a "Yee-E vector",

$$\nabla^2(\mathbb{Y}_e(F)) = \begin{bmatrix} \boldsymbol{g}_{x,x}(\mathbb{Y}_e(F)) + \boldsymbol{g}_{x,y}(\mathbb{Y}_e(F)) + \boldsymbol{g}_{x,z}(\mathbb{Y}_e(F)) \\ \boldsymbol{g}_{y,x}(\mathbb{Y}_e(F)) + \boldsymbol{g}_{y,y}(\mathbb{Y}_e(F)) + \boldsymbol{g}_{y,z}(\mathbb{Y}_e(F)) \\ \boldsymbol{g}_{z,x}(\mathbb{Y}_e(F)) + \boldsymbol{g}_{z,y}(\mathbb{Y}_e(F)) + \boldsymbol{g}_{z,z}(\mathbb{Y}_e(F)) \end{bmatrix}^T \boldsymbol{b}_0$$

By (s.6), each component of the finite-difference vectors $\boldsymbol{g}_{x,x} + \boldsymbol{g}_{x,y} + \boldsymbol{g}_{x,z}$ is formed by 4 symmetric samplings,

$$g_{x,x,m}(\mathbb{Y}_e(F)) = F_x\left(x_i + \frac{1}{2}\Delta_s + (m+1)\Delta_s, y_j, z_k\right) + F_x\left(x_i + \frac{1}{2}\Delta_s - (m+1)\Delta_s, y_j, z_k\right) - 2F_x\left(x_i + \frac{1}{2}\Delta_s, y_j, z_k\right)$$

$$g_{x,y,m}(\mathbb{Y}_e(F)) = F_x\left(x_i + \frac{1}{2}\Delta_s, y_j + (m+1)\Delta_s, z_k\right) + F_x\left(x_i + \frac{1}{2}\Delta_s, y_j - (m+1)\Delta_s, z_k\right) - 2F_x\left(x_i + \frac{1}{2}\Delta_s, y_j, z_k\right)$$

$$g_{x,z,m}(\mathbb{Y}_e(F)) = F_x\left(x_i + \frac{1}{2}\Delta_s, y_j, z_k + (m+1)\Delta_s\right) + F_x\left(x_i + \frac{1}{2}\Delta_s, y_j, z_k - (m+1)\Delta_s\right) - 2F_x\left(x_i + \frac{1}{2}\Delta_s, y_j, z_k\right)$$

Space locations of these components are the average of the related 4 space points:

$$\frac{\left(x_i + \frac{1}{2}\Delta_s + (m+1)\Delta_s, y_j, z_k\right) + \left(x_i + \frac{1}{2}\Delta_s - (m+1)\Delta_s, y_j, z_k\right) + 2\left(x_i + \frac{1}{2}\Delta_s, y_j, z_k\right)}{4} = \left(x_i + \frac{1}{2}\Delta_s, y_j, z_k\right)$$

$$\frac{\left(x_i+\frac{1}{2}\Delta_s, y_j+(m+1)\Delta_s, z_k\right)+\left(x_i+\frac{1}{2}\Delta_s, y_j-(m+1)\Delta_s, z_k\right)+2\left(x_i+\frac{1}{2}\Delta_s, y_j, z_k\right)}{4}=\left(x_i+\frac{1}{2}\Delta_s, y_j, z_k\right)$$

$$\frac{\left(x_i+\frac{1}{2}\Delta_s, y_j, z_k+(m+1)\Delta_s\right)+\left(x_i+\frac{1}{2}\Delta_s, y_j, z_k-(m+1)\Delta_s\right)+2\left(x_i+\frac{1}{2}\Delta_s, y_j, z_k\right)}{4}=\left(x_i+\frac{1}{2}\Delta_s, y_j, z_k\right)$$

Thus, by (s.2), each component is a x-component of a "Yee-E vector",

$$\left(\boldsymbol{g}_{x,x}(\mathbb{Y}_e(F))+\boldsymbol{g}_{x,y}(\mathbb{Y}_e(F))+\boldsymbol{g}_{x,z}(\mathbb{Y}_e(F))\right)^T\boldsymbol{b}_0=\mathbb{Y}_e(V_x)$$

By (s.6), each component of the finite-difference vectors $\boldsymbol{g}_{y,x}+\boldsymbol{g}_{y,y}+\boldsymbol{g}_{y,z}$ is formed by 4 symmetric samplings,

$$g_{y,x,m}(\mathbb{Y}_e(F))=F_y\left(x_i+(m+1)\Delta_s, y_j+\frac{1}{2}\Delta_s, z_k\right)+F_y\left(x_i-(m+1)\Delta_s, y_j+\frac{1}{2}\Delta_s, z_k\right)-2F_y\left(x_i, y_j+\frac{1}{2}\Delta_s, z_k\right)$$

$$g_{y,y,m}(\mathbb{Y}_e(F))=F_y\left(x_i, y_j+\frac{1}{2}\Delta_s+(m+1)\Delta_s, z_k\right)+F_y\left(x_i, y_j+\frac{1}{2}\Delta_s-(m+1)\Delta_s, z_k\right)-2F_y\left(x_i, y_j+\frac{1}{2}\Delta_s, z_k\right)$$

$$g_{y,z,m}(\mathbb{Y}_e(F))=F_y\left(x_i, y_j+\frac{1}{2}\Delta_s, z_k+(m+1)\Delta_s\right)+F_y\left(x_i, y_j+\frac{1}{2}\Delta_s, z_k-(m+1)\Delta_s\right)-2F_y\left(x_i, y_j+\frac{1}{2}\Delta_s, z_k\right)$$

Space locations of these components are the average of the related 4 space points:

$$\frac{\left(x_i+(m+1)\Delta_s, y_j+\frac{1}{2}\Delta_s, z_k\right)+\left(x_i-(m+1)\Delta_s, y_j+\frac{1}{2}\Delta_s, z_k\right)+2\left(x_i, y_j+\frac{1}{2}\Delta_s, z_k\right)}{4}=\left(x_i, y_j+\frac{1}{2}\Delta_s, z_k\right)$$

$$\frac{\left(x_i, y_j+\frac{1}{2}\Delta_s+(m+1)\Delta_s, z_k\right)+\left(x_i, y_j+\frac{1}{2}\Delta_s-(m+1)\Delta_s, z_k\right)+2\left(x_i, y_j+\frac{1}{2}\Delta_s, z_k\right)}{4}=\left(x_i, y_j+\frac{1}{2}\Delta_s, z_k\right)$$

$$\frac{\left(x_i, y_j+\frac{1}{2}\Delta_s, z_k+(m+1)\Delta_s\right)+\left(x_i, y_j+\frac{1}{2}\Delta_s, z_k-(m+1)\Delta_s\right)+2\left(x_i, y_j+\frac{1}{2}\Delta_s, z_k\right)}{4}=\left(x_i, y_j+\frac{1}{2}\Delta_s, z_k\right)$$

Thus, by (s.2), each component is a y-component of a "Yee-E vector",

$$\left(\boldsymbol{g}_{y,x}(\mathbb{Y}_e(F))+\boldsymbol{g}_{y,y}(\mathbb{Y}_e(F))+\boldsymbol{g}_{y,z}(\mathbb{Y}_e(F))\right)^T\boldsymbol{b}_0=\mathbb{Y}_e(V_y)$$

By (s.6), each component of the finite-difference vectors $\boldsymbol{g}_{z,x}+\boldsymbol{g}_{z,y}+\boldsymbol{g}_{z,z}$ is formed by 4 symmetric samplings,

$$g_{z,x,m}(\mathbb{Y}_e(F))=F_z\left(x_i+(m+1)\Delta_s, y_j, z_k+\frac{1}{2}\Delta_s\right)+F_z\left(x_i-(m+1)\Delta_s, y_j, z_k+\frac{1}{2}\Delta_s\right)-2F_z\left(x_i, y_j, z_k+\frac{1}{2}\Delta_s\right)$$

$$g_{z,y,m}(\mathbb{Y}_e(F))=F_z\left(x_i, y_j+(m+1)\Delta_s, z_k+\frac{1}{2}\Delta_s\right)+F_z\left(x_i, y_j-(m+1)\Delta_s, z_k+\frac{1}{2}\Delta_s\right)-2F_z\left(x_i, y_j, z_k+\frac{1}{2}\Delta_s\right)$$

$$g_{z,z,m}(\mathbb{Y}_e(F))=F_z\left(x_i, y_j, z_k+\frac{1}{2}\Delta_s+(m+1)\Delta_s\right)+F_z\left(x_i, y_j, z_k+\frac{1}{2}\Delta_s-(m+1)\Delta_s\right)-2F_z\left(x_i, y_j, z_k+\frac{1}{2}\Delta_s\right)$$

Space locations of these components are the average of the related 4 space points:

$$\frac{\left(x_i+(m+1)\Delta_s, y_j, z_k+\frac{1}{2}\Delta_s\right)+\left(x_i-(m+1)\Delta_s, y_j, z_k+\frac{1}{2}\Delta_s\right)+2\left(x_i, y_j, z_k+\frac{1}{2}\Delta_s\right)}{4}=\left(x_i, y_j, z_k+\frac{1}{2}\Delta_s\right)$$

$$\frac{\left(x_i, y_j+(m+1)\Delta_s, z_k+\frac{1}{2}\Delta_s\right)+\left(x_i, y_j-(m+1)\Delta_s, z_k+\frac{1}{2}\Delta_s\right)+2\left(x_i, y_j, z_k+\frac{1}{2}\Delta_s\right)}{4}=\left(x_i, y_j, z_k+\frac{1}{2}\Delta_s\right)$$

$$\frac{\left(x_i, y_j, z_k+\frac{1}{2}\Delta_s+(m+1)\Delta_s\right)+\left(x_i, y_j, z_k+\frac{1}{2}\Delta_s-(m+1)\Delta_s\right)+2\left(x_i, y_j, z_k+\frac{1}{2}\Delta_s\right)}{4}=\left(x_i, y_j, z_k+\frac{1}{2}\Delta_s\right)$$

Thus, by (s.2), each component is a z-component of a "Yee-E vector",

$$\left(\boldsymbol{g}_{z,x}(\mathbb{Y}_e(F))+\boldsymbol{g}_{z,y}(\mathbb{Y}_e(F))+\boldsymbol{g}_{z,z}(\mathbb{Y}_e(F))\right)^T\boldsymbol{b}_0=\mathbb{Y}_e(V_z)$$

Combine the above results, we have

$$\begin{bmatrix}\boldsymbol{g}_{x,x}(\mathbb{Y}_e(F))+\boldsymbol{g}_{x,y}(\mathbb{Y}_e(F))+\boldsymbol{g}_{x,z}(\mathbb{Y}_e(F))\\ \boldsymbol{g}_{y,x}(\mathbb{Y}_e(F))+\boldsymbol{g}_{y,y}(\mathbb{Y}_e(F))+\boldsymbol{g}_{y,z}(\mathbb{Y}_e(F))\\ \boldsymbol{g}_{z,x}(\mathbb{Y}_e(F))+\boldsymbol{g}_{z,y}(\mathbb{Y}_e(F))+\boldsymbol{g}_{z,z}(\mathbb{Y}_e(F))\end{bmatrix}^T\boldsymbol{b}_0=\mathbb{Y}_e(V)$$

The first part of the theorem is proved.

Using the same reasoning process, the second part of the theorem can be proved as well.

Actually, applying (g.7) to a "Yee-H vector",

$$\nabla^2(\mathbb{Y}_h(F)) \approx \begin{bmatrix} \boldsymbol{g}_{x,x}(\mathbb{Y}_h(F)) + \boldsymbol{g}_{x,y}(\mathbb{Y}_h(F)) + \boldsymbol{g}_{x,z}(\mathbb{Y}_h(F)) \\ \boldsymbol{g}_{y,x}(\mathbb{Y}_h(F)) + \boldsymbol{g}_{y,y}(\mathbb{Y}_h(F)) + \boldsymbol{g}_{y,z}(\mathbb{Y}_h(F)) \\ \boldsymbol{g}_{z,x}(\mathbb{Y}_h(F)) + \boldsymbol{g}_{z,y}(\mathbb{Y}_h(F)) + \boldsymbol{g}_{z,z}(\mathbb{Y}_h(F)) \end{bmatrix}^T \boldsymbol{b}_0$$

By (s.7), each component of the finite-difference vectors is formed by

$$g_{u,v,m}(\mathbb{Y}_h(F)) = \mathbb{Y}_h(F_u)(v + (m+1)\Delta_s) + \mathbb{Y}_h(F_u)(v - (m+1)\Delta_s) - 2\mathbb{Y}_h(F_u)(v)$$

The space location of the component is

$$\frac{v + (m+1)\Delta_s + v - (m+1)\Delta_s + 2v}{4} = v$$

That is, the space location of the component is the same as the location of the component of the vector,

$$\begin{bmatrix} \boldsymbol{g}_{x,x}(\mathbb{Y}_h(F)) + \boldsymbol{g}_{x,y}(\mathbb{Y}_h(F)) + \boldsymbol{g}_{x,z}(\mathbb{Y}_h(F)) \\ \boldsymbol{g}_{y,x}(\mathbb{Y}_h(F)) + \boldsymbol{g}_{y,y}(\mathbb{Y}_h(F)) + \boldsymbol{g}_{y,z}(\mathbb{Y}_h(F)) \\ \boldsymbol{g}_{z,x}(\mathbb{Y}_h(F)) + \boldsymbol{g}_{z,y}(\mathbb{Y}_h(F)) + \boldsymbol{g}_{z,z}(\mathbb{Y}_h(F)) \end{bmatrix}^T \boldsymbol{b}_0 = \mathbb{Y}_h(V)$$

QED

Applying the location theorem of curl estimation to (5.3) and (5.5), and applying the location theorem of Laplacian estimation to (5.4) and (5.6), the simulation algorithm given by (5.1) and (5.2) works for arbitrary orders.

(2N,2M) order Yee-algorithm

Applying the curl estimation theorem to (5.3) and (5.5), and applying the Laplacian estimation theorem to (5.4) and (5.6), we get (2N, 2M) order Yee-algorithm.

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

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

$G_{e,0} = \boldsymbol{g}_{e,0} \cdot \begin{bmatrix} \boldsymbol{d}_{z,y}(E(t)) - \boldsymbol{d}_{y,z}(E(t)) \\ \boldsymbol{d}_{x,z}(E(t)) - \boldsymbol{d}_{z,x}(E(t)) \\ \boldsymbol{d}_{y,x}(E(t)) - \boldsymbol{d}_{x,y}(E(t)) \end{bmatrix}$	(6.1)
$G_{e,n} = \begin{bmatrix} \boldsymbol{g}_{x,x}(G_{e,n-1}) + \boldsymbol{g}_{x,y}(G_{e,n-1}) + \boldsymbol{g}_{x,z}(G_{e,n-1}) \\ \boldsymbol{g}_{y,x}(G_{e,n-1}) + \boldsymbol{g}_{y,y}(G_{e,n-1}) + \boldsymbol{g}_{y,z}(G_{e,n-1}) \\ \boldsymbol{g}_{z,x}(G_{e,n-1}) + \boldsymbol{g}_{z,y}(G_{e,n-1}) + \boldsymbol{g}_{z,z}(G_{e,n-1}) \end{bmatrix}^T \boldsymbol{g}_n$	(6.2)
$G_{h,0} = \boldsymbol{g}_{h,0} \cdot \begin{bmatrix} \boldsymbol{d}_{z,y}\left(H\left(t + \frac{1}{2}\Delta_t\right)\right) - \boldsymbol{d}_{y,z}\left(H\left(t + \frac{1}{2}\Delta_t\right)\right) \\ \boldsymbol{d}_{x,z}\left(H\left(t + \frac{1}{2}\Delta_t\right)\right) - \boldsymbol{d}_{z,x}\left(H\left(t + \frac{1}{2}\Delta_t\right)\right) \\ \boldsymbol{d}_{y,x}\left(H\left(t + \frac{1}{2}\Delta_t\right)\right) - \boldsymbol{d}_{x,y}\left(H\left(t + \frac{1}{2}\Delta_t\right)\right) \end{bmatrix}$	(6.3)
$G_{h,n} = \begin{bmatrix} \boldsymbol{g}_{x,x}(G_{h,n-1}) + \boldsymbol{g}_{x,y}(G_{h,n-1}) + \boldsymbol{g}_{x,z}(G_{h,n-1}) \\ \boldsymbol{g}_{y,x}(G_{h,n-1}) + \boldsymbol{g}_{y,y}(G_{h,n-1}) + \boldsymbol{g}_{y,z}(G_{h,n-1}) \\ \boldsymbol{g}_{z,x}(G_{h,n-1}) + \boldsymbol{g}_{z,y}(G_{h,n-1}) + \boldsymbol{g}_{z,z}(G_{h,n-1}) \end{bmatrix}^T \boldsymbol{g}_n$	(6.4)
$\boldsymbol{g}_{e,0} = -\frac{c_r}{\eta}\boldsymbol{a}_0$	(6.5)
$\boldsymbol{g}_{h,0} = \eta c_r \boldsymbol{a}_0$	(6.6)
$\boldsymbol{g}_n = \frac{c_r^2}{8(2n+1)n}\boldsymbol{b}_0; n > 0$	(6.7)

where $\boldsymbol{g}_{e,0}, \boldsymbol{g}_{h,0}, and\ \boldsymbol{g}_n$ are M-dimensional constant vectors. $\boldsymbol{d}_{u,v}\ and\ \boldsymbol{g}_{u,v}$ are M-dimensional finite-difference vectors.

(2N+1,2M+1) order time-space synchronized algorithm

2N+1 order recursive time forward

Using (4.1), equations (3.1) and (3.2) are reformed into following recursive formation.

$H(t + \Delta_t) = 2H(t) - H(t - \Delta_t) + \sum_{n=0}^{N-1} g_n G_{h,n} + S_h(t) + O((\Delta_t)^{2N+1})$	(7.1)
$G_{h,0} = \Delta_s^2 \nabla^2 \times H(t)$	(7.2)
$G_{h,n} = -\Delta_s^2 \nabla^2 G_{h,n-1}; n > 0$	(7.3)
$E(t + \Delta_t) = 2E(t) - E(t - \Delta_t) + \sum_{n=0}^{N-1} g_n G_{e,n} + S_e(t) + O((\Delta_t)^{2N+1})$	(7.4)
$G_{e,0} = \Delta_s^2 \nabla^2 \times E(t)$	(7.5)
$G_{e,n} = -\Delta_s^2 \nabla^2 G_{e,n-1}; n > 0$	(7.6)

$g_n = 2\frac{(-c_r^2)^{n+1}}{(2(n+1))!}$	(7.7)

In deduction of the above algorithm, it uses the fact that the divergence of a curl is 0, and thus, in source-free regions, we have

$$\nabla^2 \times = \nabla(\nabla\cdot) - \nabla^2 = -\nabla^2$$

2M+1 order spatial estimations

We only need to estimate double curls $\Delta_s^2\nabla^2\times$. For $n > 0$ double curl estimation can be replaced by Laplacian estimation formula (g.7). For $n = 0$ in source-free space, we may still use Laplacian estimator. In space region where a field source presents, we need to estimate $\nabla(\nabla\cdot)$, which needs estimation of the first order derivatives.

Odd order derivative estimation theorem (even-sampling). For a sufficiently smooth scalar function $f(s)$, a 2M order estimation of its odd order derivatives scaled by a sampling size is a M -dimentional constant linear combination of nearby finite-differences,

$\frac{(\Delta_s)^{2h+1}}{(2h+1)!}\frac{\partial^{2h+1}f(s)}{\partial s^{2h+1}} = \frac{1}{2}\boldsymbol{c}_h\cdot\boldsymbol{p} + O(\Delta_s^{2M}); h = 0,1,\dots,M-1$	(c.1)
$\boldsymbol{p} = \begin{bmatrix} p_0 \\ p_1 \\ \vdots \\ p_{M-1} \end{bmatrix}; p_m = f(s+(m+1)\Delta_s) - f(s-(m+1)\Delta_s)$ $M > 0$ $\Delta_s > 0$	(c.2)
$\boldsymbol{C}_M = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 2 & 2^3 & \cdots & 2^{2M-1} \\ \vdots & \vdots & \ddots & \vdots \\ M & M^3 & \dots & M^{2M-1} \end{bmatrix}$	(c.3)
$\boldsymbol{C}_M^{-1} = \begin{bmatrix} \boldsymbol{c}_0^T \\ \boldsymbol{c}_1^T \\ \vdots \\ \boldsymbol{c}_{M-1}^T \end{bmatrix}$	(c.4)

Proof. From Taylor's series,

$$f(s+(m+1)\Delta_s) = \sum_{h=0}^{\infty}\frac{(m+1)^h}{h!}\Delta_s^h\frac{\partial^h f(s)}{\partial s^h}$$

$$f(s-(m+1)\Delta_s) = \sum_{h=0}^{\infty}(-1)^h\frac{(m+1)^h}{h!}\Delta_s^h\frac{\partial^h f(s)}{\partial s^h}$$

we have

$$p_m = 2\sum_{h=0}^{\infty}\frac{(m+1)^{2h+1}}{(2h+1)!}\Delta_s^{2h+1}\frac{\partial^{2h+1}f(s)}{\partial s^{2h+1}}$$

$$m = 0,1,2,\dots,M-1; M > 0$$

It can be writtern as

$$d_m = 2\sum_{h=0}^{M-1}(m+1)^{2h+1}u_h + O((\Delta_s)^{2M})$$

by defining

$$u_h = \frac{(\Delta_s)^{2h+1}}{(2h+1)!}\frac{\partial^{2h+1}f(s)}{\partial s^{2h+1}}$$

Writtern in matrix form using (c.3), we have

$$\boldsymbol{p} = 2\boldsymbol{C}_M\boldsymbol{u} + O((\Delta_s)^{2M})$$

by defining

$$\boldsymbol{u} = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{M-1} \end{bmatrix}$$

We get a $2M$ order estimation of vector $\boldsymbol{u}$ by

$$\boldsymbol{u} = \frac{1}{2}\boldsymbol{C}_M^{-1}\boldsymbol{p} + O((\Delta_s)^{2M})$$

By (c.4) we get

$$\frac{(\Delta_s)^{2h+1}}{(2h+1)!}\frac{\partial^{2h+1}f(s)}{\partial s^{2h+1}} = \frac{1}{2}\boldsymbol{c}_h\cdot\boldsymbol{p} + O((\Delta_s)^{2M}); h = 0,1,\dots,M-1$$

QED

By this theorem. We have 2M order estimation for operator ∂/∂

$$\Delta_s \frac{\partial f(s)}{\partial s} = \frac{1}{2}\boldsymbol{c}_0 \cdot \boldsymbol{p} + O((\Delta_s)^{2M}) \qquad \text{(c.5)}$$

With the above result, we have following divergence estimation theorem and gradient estimation theorem.

For a scalar function on 3D space $f(x, y, z)$, finite-difference (c.2) is modified to use a subscript to indicate the axis,

$$\begin{aligned}
\boldsymbol{p}_x(f) &= \begin{bmatrix} p_{x,0} \\ p_{x,1} \\ \vdots \\ p_{x,M-1} \end{bmatrix}(f); p_{x,m}(f) = f(x + (m+1)\Delta_s, y, z) - f(x - (m+1)\Delta_s, y, z) \\
\boldsymbol{p}_y(f) &= \begin{bmatrix} p_{y,0} \\ p_{y,1} \\ \vdots \\ p_{y,M-1} \end{bmatrix}(f); p_{y,m}(f) = f(x, y + (m+1)\Delta_s, z) - f(x, y - (m+1)\Delta_s, z) \\
\boldsymbol{p}_z(f) &= \begin{bmatrix} p_{z,0} \\ p_{z,1} \\ \vdots \\ p_{z,M-1} \end{bmatrix}(f); p_{z,m}(f) = f(x, y, z + (m+1)\Delta_s) - f(x, y, z - (m+1)\Delta_s)
\end{aligned} \qquad \text{(c.6)}$$

For a 3D vector on 3D space $F(x, y, z)$, finite-difference (c.2) is modified to use a subscript to indicate the component, and another subscript to indicate the axis,

$$\boldsymbol{p}_{u,v}(F) = \begin{bmatrix} p_{u,v,0} \\ p_{u,v,1} \\ \vdots \\ p_{u,v,M-1} \end{bmatrix}(F); p_{u,v,m}(F) = F_u(v + (m+1)\Delta_s) - F_u(v - (m+1)\Delta_s) \qquad \text{(c.7)}$$

For example,

$$\boldsymbol{p}_{x,y}(F) = \begin{bmatrix} p_{x,y,0} \\ p_{x,y,1} \\ \vdots \\ p_{x,y,M-1} \end{bmatrix}(F); p_{x,y,m}(F) = F_x(x, y + (m+1)\Delta_s, z) - F_x(x, y - (m+1)\Delta_s, z)$$

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

$$\Delta_s \nabla \cdot F = \frac{1}{2}\boldsymbol{c}_0 \cdot (\boldsymbol{p}_{x,x} + \boldsymbol{p}_{y,y} + \boldsymbol{p}_{z,z})(F) + O((\Delta_s)^{2M}) \qquad \text{(c.8)}$$

Where $\boldsymbol{c}_0$ is a constant vector given by (c.3) and (c.4).

Proof. By (c.7) and (c.5), we have

$$\Delta_s \nabla \cdot F = \Delta_s \left(\frac{\partial F_x}{\partial x} + \frac{\partial F_y}{\partial y} + \frac{\partial F_z}{\partial z}\right) = \Delta_s \frac{\partial F_x}{\partial x} + \Delta_s \frac{\partial F_y}{\partial y} + \Delta_s \frac{\partial F_z}{\partial z}$$

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

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

QED

Gradient estimation theorem. For a sufficiently smooth function on 3D space $f(x, y, z)$, a 2*M* order estimation of its gradient scaled by a sampling size is a *M* -dimentional constant linear combination of nearby finite-differences,

$$\Delta_s \nabla f = \frac{1}{2}\boldsymbol{c}_0 \cdot \begin{bmatrix} \boldsymbol{p}_x \\ \boldsymbol{p}_y \\ \boldsymbol{p}_z \end{bmatrix}(f) + \boldsymbol{O}((\Delta_s)^{2M}) \qquad \text{(c.9)}$$

Where $\boldsymbol{c}_0$ is a constant vector given by (c.3) and (c.4).

Proof. By (c.6) and (c.5), we have

$$\Delta_s \nabla f = \begin{bmatrix} \Delta_s \frac{\partial f}{\partial x} \\ \Delta_s \frac{\partial f}{\partial y} \\ \Delta_s \frac{\partial f}{\partial z} \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\boldsymbol{c}_0 \cdot \boldsymbol{p}_x + O((\Delta_s)^{2M}) \\ \frac{1}{2}\boldsymbol{c}_0 \cdot \boldsymbol{p}_y + O((\Delta_s)^{2M}) \\ \frac{1}{2}\boldsymbol{c}_0 \cdot \boldsymbol{p}_z + O((\Delta_s)^{2M}) \end{bmatrix}$$

QED

Define an operator to apply estimator (c.8) and then (c.9),

$\mathbb{G}(F) = \frac{1}{2}\boldsymbol{c}_0 \cdot \begin{bmatrix} \boldsymbol{p}_x\left(\frac{1}{2}\boldsymbol{c}_0 \cdot (\boldsymbol{p}_{x,x} + \boldsymbol{p}_{y,y} + \boldsymbol{p}_{z,z})(F)\right) \\ \boldsymbol{p}_y\left(\frac{1}{2}\boldsymbol{c}_0 \cdot (\boldsymbol{p}_{x,x} + \boldsymbol{p}_{y,y} + \boldsymbol{p}_{z,z})(F)\right) \\ \boldsymbol{p}_z\left(\frac{1}{2}\boldsymbol{c}_0 \cdot (\boldsymbol{p}_{x,x} + \boldsymbol{p}_{y,y} + \boldsymbol{p}_{z,z})(F)\right) \end{bmatrix}$	(c.10)

(2N+1,2M+1) order TSS algorithm

With the results presented above, a time-space-synchronized (TSS) algorithm can be formed.

To reduce the amount of calculations, equations (7.1) to (7.7) are reformed to the following.

$H(t+\Delta_t) = 2H(t) - H(t-\Delta_t) + \sum_{n=0}^{N-1} F_{h,n} + S_h(t) + O((\Delta_t)^{2N+1})$	(t.1)
$F_{h,0} = -c_r^2\Delta_s^2\nabla^2 \times H(t)$	(t.2)
$F_{h,n} = \frac{c_r^2}{2n(2n-1)}\Delta_s^2\nabla^2 F_{h,n-1}; n > 0$	(t.3)
$E(t+\Delta_t) = 2E(t) - E(t-\Delta_t) + \sum_{n=0}^{N-1} F_{e,n} + S_e(t) + O((\Delta_t)^{2N+1})$	(t.4)
$F_{e,0} = -c_r^2\Delta_s^2\nabla^2 \times E(t)$	(t.5)
$F_{e,n} = \frac{c_r^2}{2n(2n-1)}\Delta_s^2\nabla^2 F_{h,n-1}; n > 0$	(t.6)

Operator $\mathbb{G}$ is modified to

$\mathbb{G}_0(F) = -\frac{c_r^2}{2}\boldsymbol{c}_0 \cdot \begin{bmatrix} \boldsymbol{p}_x\left(\frac{1}{2}\boldsymbol{c}_0 \cdot (\boldsymbol{p}_{x,x} + \boldsymbol{p}_{y,y} + \boldsymbol{p}_{z,z})(F)\right) \\ \boldsymbol{p}_y\left(\frac{1}{2}\boldsymbol{c}_0 \cdot (\boldsymbol{p}_{x,x} + \boldsymbol{p}_{y,y} + \boldsymbol{p}_{z,z})(F)\right) \\ \boldsymbol{p}_z\left(\frac{1}{2}\boldsymbol{c}_0 \cdot (\boldsymbol{p}_{x,x} + \boldsymbol{p}_{y,y} + \boldsymbol{p}_{z,z})(F)\right) \end{bmatrix}$	(t.7)

Applying the Laplacian estimator (g.7) and the gradient of divergence estimator (t.7) to equations (t.1) to (t.6), we get a (2N+1,2M+1) order TSS algorithm.

$H(t+\Delta_t) = 2H(t) - H(t-\Delta_t) + \sum_{n=0}^{N-1} F_{h,n} + S_h(t) + O((\Delta_t)^{2N+1})$	(8.1)
$F_{h,0} = \begin{bmatrix} \boldsymbol{g}_{x,x}(H(t)) + \boldsymbol{g}_{x,y}(H(t)) + \boldsymbol{g}_{x,z}(H(t)) \\ \boldsymbol{g}_{y,x}(H(t)) + \boldsymbol{g}_{y,y}(H(t)) + \boldsymbol{g}_{y,z}(H(t)) \\ \boldsymbol{g}_{z,x}(H(t)) + \boldsymbol{g}_{z,y}(H(t)) + \boldsymbol{g}_{z,z}(H(t)) \end{bmatrix}^T \boldsymbol{g}_0 + \mathbb{G}_0(H(t))$	(8.2)
$F_{h,n} = \begin{bmatrix} \boldsymbol{g}_{x,x}(F_{h,n-1}) + \boldsymbol{g}_{x,y}(F_{h,n-1}) + \boldsymbol{g}_{x,z}(F_{h,n-1}) \\ \boldsymbol{g}_{y,x}(F_{h,n-1}) + \boldsymbol{g}_{y,y}(F_{h,n-1}) + \boldsymbol{g}_{y,z}(F_{h,n-1}) \\ \boldsymbol{g}_{z,x}(F_{h,n-1}) + \boldsymbol{g}_{z,y}(F_{h,n-1}) + \boldsymbol{g}_{z,z}(F_{h,n-1}) \end{bmatrix}^T \boldsymbol{g}_n; n > 0$	(8.3)
$E(t+\Delta_t) = 2E(t) - E(t-\Delta_t) + \sum_{n=0}^{N-1} F_{e,n} + S_e(t) + O((\Delta_t)^{2N+1})$	(8.4)
$F_{e,0} = \begin{bmatrix} \boldsymbol{g}_{x,x}(E(t)) + \boldsymbol{g}_{x,y}(E(t)) + \boldsymbol{g}_{x,z}(E(t)) \\ \boldsymbol{g}_{y,x}(E(t)) + \boldsymbol{g}_{y,y}(E(t)) + \boldsymbol{g}_{y,z}(E(t)) \\ \boldsymbol{g}_{z,x}(E(t)) + \boldsymbol{g}_{z,y}(E(t)) + \boldsymbol{g}_{z,z}(E(t)) \end{bmatrix}^T \boldsymbol{g}_0 + \mathbb{G}_0(E(t))$	(8.5)
$F_{e,n} = \begin{bmatrix} \boldsymbol{g}_{x,x}(F_{e,n-1}) + \boldsymbol{g}_{x,y}(F_{e,n-1}) + \boldsymbol{g}_{x,z}(F_{e,n-1}) \\ \boldsymbol{g}_{y,x}(F_{e,n-1}) + \boldsymbol{g}_{y,y}(F_{e,n-1}) + \boldsymbol{g}_{y,z}(F_{e,n-1}) \\ \boldsymbol{g}_{z,x}(F_{e,n-1}) + \boldsymbol{g}_{z,y}(F_{e,n-1}) + \boldsymbol{g}_{z,z}(F_{e,n-1}) \end{bmatrix}^T \boldsymbol{g}_n; n > 0$	(8.6)
$\boldsymbol{g}_0 = -c_r^2\boldsymbol{b}_0$	(8.7)
$\boldsymbol{g}_n = \frac{c_r^2}{2n(2n-1)}\boldsymbol{b}_0; n > 0$	(8.8)

Note that equations (8.1), (8.2) and (8.3) calculate the magnetic field; equations (8.4), (8.5) and (8.6) calculate the electric field. These two sets of equations are unrelated. Thus, if only one field is needed then we only need to calculate one field.

Thickness of Boundary

Higher estimation order raises an issue of the thickness of boundary.

Thickness of boundary condition

All the spatial estimation theorems presented previously use *M*-samplings on both sides of a space point. For a space point within a distance of $M\Delta_s$ of the simulation boundary, some samplings locate outside of the boundary. Such unavailable samplings must be determined by boundary conditions.

Therefore, we have following result.

Thickness of boundary condition. For the (2N, 2M) order Yee-algorithm and the (2N+1, 2M+1) order TSS algorithm, boundary conditions must be used in a layer of thickness *M* outside of the boundary.

Thickness of absorbing layer

For a space point within a distance of $M\Delta_s$ of the simulation boundary, the spatial estimations at this space point involve both the samplings determined by the boundary conditions and the samplings not determined by the boundary conditions. Such asymmetric usages of samplings cause accumulation of numeric errors, and lead to instability of the simulation.

An absorbing layer can be used to prevent the accumulation of the numeric errors. For the (2N, 2M) order Yee-algorithm and the (2N+1, 2M+1) order TSS algorithm, there are N terms to make a time domain forward, see equations (5.1), (5.2), (8.1) and (8.4). For each term, there are M space points near the boundary "contaminated" by the boundary conditions. Totally, there are NM space points "contaminated".

Therefore, we have following result.

Thickness of absorbing layer. If an absorbing layer is to be used for the (2N, 2M) order Yee-algorithm or the (2N+1, 2M+1) order TSS algorithm then the minimum thickness of the layer should be NM.

Comparison of Yee and TSS algorithms

Major aspects of a FDTD algorithm are examined below.

- The calculation amount of the Yee-algorithm is determined by equations (6.1) to (6.4). The calculation amount of TSS-algorthm is determined by equations (8.2), (8.3), (8.5) and (8.6). The differences of the calculation amounts are negligible.
- For the same amount of calculations, the estimation order of the TSS algorithm is (2N+1, 2M+1), and the estimation order of the Yee algorithm is (2N, 2M). Therefore, TSS algorithm is more accurate than the Yee-algorithm.
- For the TSS algorithm, all field components are located at the same space point, and the magnetic and electric fields are located at the same time. This is consistent with the continous Maxwell's equations. This is the default assumption by everyone for doing all math operations.
- For the Yee algorithm, all field components are located at different space points, and the magnetic field and the electric field are at different times. This arrangement causes troubles and much complexity to every math operations. It is used exclusively by the Yee-algorithm.
- For the TSS algorithm, the calculation of the magnetic and electric fields are unrelated. It is possible to only simulate one field if the boundary handling is possible by a single field. For the Yee-algorithm, calculation of one field relies on the other field.

It is interesting to note that the TSS algorithm is advantageous in every aspect.

Field Source Estimation

In the (2N, 2M) order Yee-algorithm, field source appears in a format of $S_h(t)$ and $S_e\left(t+\frac{1}{2}\Delta_t\right)$, given by equations (2.3) and (2.4).

In the (2N+1, 2M+1) order TSS-algorithm, field source appears in a format of $S_h(t)$ and $S_e(t)$, given by equations (3.3) and (3.4).

From (2.3), (2.4), (3.3) and (3.4), 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 orders.
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, making the estimation order not truly arbitrary.

To overcome these difficulties, an algorithm is developed to make (2N+1, 2M+1)-order estimations of S_h and S_e from discrete values of $J(x,y,z,t)$. Thus, for a (2N, 2M)-order or a (2N+1, 2M+1) order algorithm, the selections of N and M are only limited by the available computing resources.

The field source estimation algorithm will be presented in another paper.

Numerical Results

Numerical results of the (2N, 2M) order Yee-algorithm and the (2N+1, 2M+1) order TSS algorithm will be presented in another report. The open source software producing the numerical results will also be uploaded to Github at the same time when the report is ready.

Reference

[1] K. S. Yee, “Numerical solution of initial boundary value problems involving Maxwell’s equations in isotropic media,” IEEE Trans. Antennas Propagat., vol. 14, pp. 302–307, 1966 doi:10.1109/TAP.1966.1138693

[2] *Understanding the Finite-Difference Time-Domain Method*, John B. Schneider, www.eecs.wsu.edu/~schneidj/ufdtd, 2010.

[3] V. A. BOKIL AND N. L. GIBSON, Analysis of spatial high-order finite difference methods for Maxwell’s equations in dispersive media, IMA Journal of Numerical Analysis (2012) 32 926–956

[4] Charles W. Manry Jr, Shira L. Broschat, and John B. Schneider, Higher-Order FDTD Methods for Large Problems, https://core.ac.uk/display/24329792

[5] David Wei Ge, A Generic FDTD Form for Maxwell Equations, https://www.researchgate.net/publication/344868091_A_Generic_FDTD_Form_for_Maxwell_Equations
[6] David Wei Ge, Compile and Run TSS FDTD C++ Code for Linux and Windows, https://www.researchgate.net/publication/350269934_Compile_and_Run_TSS_FDTD_C_Code_for_Linux_and_Windows
[7] Qiang Yuan, High Order Energy - Conserved Splitting FDTD Methods for Maxwell's Equations, 28 August 2015 , oai:yorkspace.library.yorku.ca:10315/29838

[8] G Xie, Z Huang, M Fang, WEI Sha, Simulating Maxwell–Schrödinger Equations by High-Order Symplectic FDTD Algorithm, 19 June 2019, oai:eprints.ucl.ac.uk.OAI2:10076916

[9] Maninder Kaur Sarai,Even-Odd Cycled High-Order S-FDTD Method for Maxwell's Equations and Application to Coplanar Waveguides, 27 July 2017, oai:yorkspace.library.yorku.ca:10315/33593

[10] Aydin Gokhan, A Single-Field Finite-Difference Time-Domain Formulations for Electromagnetic Simulations, 01/01/2011

[11] Gokhan Aydin, Atef Z Elsherbeni, Ercument Arvas, Jay K Lee, A Two-Dimensional Single-Field FDTD Formulation for Oblique Incident Electromagnetic Simulations, 4 March 2020, oai:CiteSeerX.psu:10.1.1.1039.4009

[12] Shin Jongchul, Time-Domain Propagator Full-Wave Numerical Method for Electromagnetic Fields,23/01/2019, A Dissertation by JONGCHUL SHIN Submitted to the Office of Graduate and Professional Studies of Texas A&M University in partial fulfillment of the requirements for the degree of DOCTOR OF PHILOSOPH