Efficient Algorithms for Crank-Nicolson-Based Finite-Difference Time-Domain Methods

Eng Leong Tan, *Senior Member, IEEE*

***Abstract*— This paper presents new efficient algorithms for implementing three-dimensional Crank-Nicolson-based finite-difference time-domain (FDTD) methods. Two recent methods are considered, namely Crank-Nicolson direct-splitting and Crank-Nicolson cycle-sweep-uniform FDTD methods. The algorithms involve update equations whose right-hand sides are much simpler and more concise than the original ones. Analytical proof is provided to show the equivalence of original and present methods. Comparison of their implementations signifies substantial reductions of the floating-point operations count in the new algorithms. Other computational aspects are also optimized, particularly in regard to the for-looping overhead and the memory space requirement. Through numerical simulation and Fourier stability analysis, it is found that while the Crank-Nicolson direct-splitting FDTD is unconditionally stable, the Crank-Nicolson cycle-sweep-uniform FDTD may actually become unstable.**

***Index Terms*— Crank-Nicolson methods, finite-difference time-domain methods, unconditionally stable methods, computational electromagnetics.**

10.1109/TMTT.2007.914641

I. Introduction

The alternating direction implicit finite-difference time-domain (FDTD) method is a celebrated unconditionally stable method with its time step size not constrained by Courant-Friedrichs-Lewy condition [1]. Recently, there has been considerable interest to develop other unconditionally stable FDTD methods such as those based on split-step approach [2], locally one-dimensional scheme [3], and various Crank-Nicolson-based approximation or factorization-splitting [4]-[6]. The Crank-Nicolson-based FDTD methods have the potential of featuring higher numerical accuracy, smaller anisotropy or greater flexibility in incorporating current source terms (e.g. in merely one sub-step). However, from the implementation point of view, most Crank-Nicolson-based methods presented so far involve complicated update equations with considerable arithmetic operations. To make these methods more attractive, their update equations need to be simplified and their computation efficiencies should be improved. Despite more challenges due to the greater complexity (and the need of stability reassessment) for Crank-Nicolson-based methods, it remains our goal to make them simpler and more efficient. We previously reported the efficient three-dimensional alternating direction implicit and locally one-dimensional FDTD methods [7], [8].

In this paper, new efficient algorithms are devised for implementing three-dimensional Crank-Nicolson-based FDTD methods. Sections II and III present efficient algorithms for two recent methods [6], namely Crank-Nicolson direct-splitting (CNDS) and Crank-Nicolson cycle-sweep-uniform (CNCSU) FDTD methods, respectively. The algorithms involve update equations whose right-hand sides are much simpler and more concise than the original ones in [6]. In Section IV, analytical proof is provided to show the equivalence of original and present methods. Comparison of their implementations in Section V signifies substantial reductions of the floating-point operations (flops) count in the new algorithms. Other computational aspects are also optimized, particularly in regard to the for-looping overhead and the memory space requirement. Through numerical simulation and Fourier stability analysis, it is found that while the CNDS-FDTD is unconditionally stable, the CNCSU-FDTD of [6] may actually become unstable.

II. Efficient CNDS Algorithm

In this section, the update equations are presented for the new efficient CNDS algorithm. Henceforth, we shall adopt the following notations:

$$a_1 = \frac{\Delta t}{2\epsilon}, \qquad a_2 = \frac{\Delta t}{2\mu}; \tag{1}$$

D_x, D_y, D_z are the difference operators for the first derivatives; $s_E^{n+1/2}$ and $s_H^{n+1/2}$ are the electric and magnetic source terms respectively; E^n, H^n and E^{n+1}, H^{n+1} are the (physical) electromagnetic fields at time step n and $n+1$ respectively. While the (nonphysical) intermediate fields are denoted by E^*, H^* in [6], our intermediate fields will be signified by (distinct) $E^{n+1/2}$ along with some auxiliary field variables in small letters e and h. Exploitation of these variables leads to the implementation as follows:

A. First procedure from n to $n + 1/2$

(i) Auxiliary updating for e^n:

$$e_\xi^n = E_\xi^n - e_\xi^{n-1/2}, \quad \xi = x, y, z. \tag{2}$$

(ii) Implicit updating for $E^{n+1/2}$:

$$\frac{1}{2}E_x^{n+1/2} - \frac{a_1 a_2}{2} D_z^2 E_x^{n+1/2} = e_x^n - a_1 D_z h_y^n + \frac{1}{4} s_x^{n+1/2} \tag{3a}$$

The author is with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798 (e-mail: eeltan@ntu.edu.sg).

E. L. Tan, "Efficient Algorithms for Crank-Nicolson-Based Finite-Difference Time-Domain Methods," IEEE Transactions on Microwave Theory and Techniques, Vol. 56, No. 2, pp. 408-413, February 2008. DOI: 10.1109/TMTT.2007.914641

$$\frac{1}{2}E_y^{n+1/2} - \frac{a_1a_2}{2}D_x^2E_y^{n+1/2} = e_y^n - a_1D_xh_z^n + \frac{1}{4}s_y^{n+1/2} \quad (3b)$$

$$\frac{1}{2}E_z^{n+1/2} - \frac{a_1a_2}{2}D_y^2E_z^{n+1/2} = e_z^n - a_1D_yh_x^n + \frac{1}{4}s_z^{n+1/2} \quad (3c)$$

where

$$s_x^{n+1/2} = s_{Ex}^{n+1/2} - a_1D_zs_{Hy}^{n+1/2} \quad (4a)$$
$$s_y^{n+1/2} = s_{Ey}^{n+1/2} - a_1D_xs_{Hz}^{n+1/2} \quad (4b)$$
$$s_z^{n+1/2} = s_{Ez}^{n+1/2} - a_1D_ys_{Hx}^{n+1/2}. \quad (4c)$$

B. Second procedure from $n+1/2$ to $n+1$

(i) Auxiliary updating for $h^{n+1/2}$, $e^{n+1/2}$:

$$h_x^{n+1/2} = h_x^n - a_2D_yE_z^{n+1/2} + \frac{1}{2}s_{Hx}^{n+1/2} \quad (5a)$$
$$h_y^{n+1/2} = h_y^n - a_2D_zE_x^{n+1/2} + \frac{1}{2}s_{Hy}^{n+1/2} \quad (5b)$$
$$h_z^{n+1/2} = h_z^n - a_2D_xE_y^{n+1/2} + \frac{1}{2}s_{Hz}^{n+1/2} \quad (5c)$$

$$e_\xi^{n+1/2} = E_\xi^{n+1/2} - e_\xi^n, \quad \xi = x, y, z. \quad (6)$$

(ii) Implicit updating for E^{n+1}:

$$\frac{1}{2}E_x^{n+1} - \frac{a_1a_2}{2}D_y^2E_x^{n+1} = e_x^{n+1/2} + a_1D_yh_z^{n+1/2} \quad (7a)$$
$$\frac{1}{2}E_y^{n+1} - \frac{a_1a_2}{2}D_z^2E_y^{n+1} = e_y^{n+1/2} + a_1D_zh_x^{n+1/2} \quad (7b)$$
$$\frac{1}{2}E_z^{n+1} - \frac{a_1a_2}{2}D_x^2E_z^{n+1} = e_z^{n+1/2} + a_1D_xh_y^{n+1/2}. \quad (7c)$$

(iii) Auxiliary updating for h^{n+1}:

$$h_x^{n+1} = h_x^{n+1/2} + a_2D_zE_y^{n+1} \quad (8a)$$
$$h_y^{n+1} = h_y^{n+1/2} + a_2D_xE_z^{n+1} \quad (8b)$$
$$h_z^{n+1} = h_z^{n+1/2} + a_2D_yE_x^{n+1}. \quad (8c)$$

The algorithm above preserves the convenience of source incorporation (in one sub-step) like the original CNDS method. Furthermore, it does not return the final magnetic fields H^{n+1} for output directly. Such implementation is more efficient since it exploits the (often) reduced or infrequent output data processing, especially when only the electric (and not the magnetic) field data is needed. If desired, the magnetic fields may still be obtained simply by

$$H_\xi^{n+1} = h_\xi^{n+1} + h_\xi^{n+1/2}, \quad \xi = x, y, z. \quad (9)$$

When there are non-zero initial fields E^0, H^0, the algorithm should take the initialization as

$$e_x^{-1/2} = \frac{1}{2}E_x^0 - \frac{a_1}{2}D_yH_z^0 \quad (10a)$$
$$e_y^{-1/2} = \frac{1}{2}E_y^0 - \frac{a_1}{2}D_zH_x^0 \quad (10b)$$
$$e_z^{-1/2} = \frac{1}{2}E_z^0 - \frac{a_1}{2}D_xH_y^0 \quad (10c)$$

$$h_x^0 = \frac{1}{2}H_x^0 + \frac{a_2}{2}D_zE_y^0 \quad (11a)$$
$$h_y^0 = \frac{1}{2}H_y^0 + \frac{a_2}{2}D_xE_z^0 \quad (11b)$$
$$h_z^0 = \frac{1}{2}H_z^0 + \frac{a_2}{2}D_yE_x^0. \quad (11c)$$

III. Efficient CNCSU Algorithm

In this section, the update equations are presented for the new efficient CNCSU algorithm. We shall follow the previous notations in the implementation below:

A. First procedure from n to $n+1/2$

(i) Auxiliary updating for e^n:

$$e_\xi^n = E_\xi^n - e_\xi^{n-1/2}, \quad \xi = x, y, z. \quad (12)$$

(ii) Implicit updating for $E^{n+1/2}$:

$$\frac{1}{2}E_x^{n+1/2} - \frac{a_1a_2}{2}D_y^2E_x^{n+1/2} = e_x^n + \frac{a_1}{2}D_yH_z^n - \frac{a_1}{2}D_zH_y^n + \frac{1}{4}f_{s1}^{n+1/2} \quad (13a)$$

$$\frac{1}{2}E_y^{n+1/2} - \frac{a_1a_2}{2}D_z^2E_y^{n+1/2} = e_y^n + \frac{a_1}{2}D_zH_x^n - \frac{a_1}{2}D_xH_z^n - \frac{a_1a_2}{2}D_xD_yE_x^{n+1/2} + \frac{1}{4}f_{s2}^{n+1/2} \quad (13b)$$

$$\frac{1}{2}E_z^{n+1/2} - \frac{a_1a_2}{2}D_x^2E_z^{n+1/2} = e_z^n + \frac{a_1}{2}D_xH_y^n - \frac{a_1}{2}D_yH_x^n - \frac{a_1a_2}{2}D_xD_zE_x^{n+1/2} - \frac{a_1a_2}{2}D_yD_zE_y^{n+1/2} + \frac{1}{4}f_{s3}^{n+1/2} \quad (13c)$$

where

$$f_{s1}^{n+1/2} = s_{Ex}^{n+1/2} + a_1D_ys_{Hz}^{n+1/2} - a_1D_zs_{Hy}^{n+1/2} \quad (14a)$$
$$f_{s2}^{n+1/2} = s_{Ey}^{n+1/2} + a_1D_zs_{Hx}^{n+1/2} - a_1D_xs_{Hz}^{n+1/2} \quad (14b)$$
$$f_{s3}^{n+1/2} = s_{Ez}^{n+1/2} + a_1D_xs_{Hy}^{n+1/2} - a_1D_ys_{Hx}^{n+1/2}. \quad (14c)$$

B. Second procedure from $n+1/2$ to $n+1$

(i) Auxiliary updating for $e^{n+1/2}$:

$$e_\xi^{n+1/2} = E_\xi^{n+1/2} - e_\xi^n, \quad \xi = x, y, z. \quad (15)$$

(ii) Implicit updating for E^{n+1}:

$$\frac{1}{2}E_z^{n+1} - \frac{a_1a_2}{2}D_y^2E_z^{n+1} = e_z^{n+1/2} \tag{16a}$$

$$\frac{1}{2}E_y^{n+1} - \frac{a_1a_2}{2}D_x^2E_y^{n+1} = e_y^{n+1/2} - \frac{a_1a_2}{2}D_zD_yE_z^{n+1} \tag{16b}$$

$$\frac{1}{2}E_x^{n+1} - \frac{a_1a_2}{2}D_z^2E_x^{n+1} = e_x^{n+1/2} - \frac{a_1a_2}{2}D_zD_xE_z^{n+1} - \frac{a_1a_2}{2}D_yD_xE_y^{n+1} \tag{16c}$$

(iii) Explicit updating for H^{n+1}:

$$H_x^{n+1} = H_x^n + a_2D_z(E_y^n + E_y^{n+1}) - a_2D_y(E_z^n + E_z^{n+1}) + s_{Hx}^{n+1/2} \tag{17a}$$

$$H_y^{n+1} = H_y^n + a_2D_x(E_z^n + E_z^{n+1}) - a_2D_z(E_x^n + E_x^{n+1}) + s_{Hy}^{n+1/2} \tag{17b}$$

$$H_z^{n+1} = H_z^n + a_2D_y(E_x^n + E_x^{n+1}) - a_2D_x(E_y^n + E_y^{n+1}) + s_{Hz}^{n+1/2}. \tag{17c}$$

Again, the algorithm above preserves the convenience of source incorporation like the original CNCSU method. For non-zero initial fields, the initialization reads

$$e_x^{-1/2} = \frac{1}{2}E_x^0 - \frac{a_1a_2}{2}D_z^2E_x^0 + \frac{a_1a_2}{2}D_yD_xE_y^0 + \frac{a_1a_2}{2}D_zD_xE_z^0 \tag{18a}$$

$$e_y^{-1/2} = \frac{1}{2}E_y^0 - \frac{a_1a_2}{2}D_x^2E_y^0 + \frac{a_1a_2}{2}D_zD_yE_z^0 \tag{18b}$$

$$e_z^{-1/2} = \frac{1}{2}E_z^0 - \frac{a_1a_2}{2}D_y^2E_z^0 \tag{18c}$$

IV. Analytical Proof of Equivalence

The algorithms presented in Sections II and III above correspond respectively to the efficient implementations of CNDS and CNCSU FDTD methods. Despite taking much simpler and more concise form, these methods are equivalent to the original ones described in [6]. Furthermore, the final fields E^{n+1} and H^{n+1} herein coincide with those in that paper, and their intermediate fields (in terms of $E^{n+1/2}$ here and E^* in [6]) are related by

$$E_\xi^* = 2E_\xi^{n+1/2} - E_\xi^n, \quad \xi = x, y, z. \tag{19}$$

While one can easily demonstrate such correspondence by means of numerical tests, we shall provide the analytical proof in the sequel.

Let us examine the CNDS-FDTD method in Section II. For the first procedure, we consider (7a), (8b), (8c) and (9) at one time step backward:

$$e_x^{n-1/2} = \frac{1}{2}E_x^n - \frac{a_1a_2}{2}D_y^2E_x^n - a_1D_yh_z^{n-1/2} \tag{20}$$

$$h_y^{n-1/2} = h_y^n - a_2D_xE_z^n = -h_y^n + H_y^n \tag{21}$$

$$h_z^{n-1/2} = h_z^n - a_2D_yE_x^n = -h_z^n + H_z^n. \tag{22}$$

Manipulating these equations along with (2) gives

$$e_x^n = \frac{1}{2}E_x^n + \frac{a_1}{2}D_yH_z^n \tag{23}$$

$$h_y^n = \frac{1}{2}H_y^n + \frac{a_2}{2}D_xE_z^n \tag{24}$$

$$h_z^n = \frac{1}{2}H_z^n + \frac{a_2}{2}D_yE_x^n. \tag{25}$$

Substituting (23) and (24) into (3a) leads to

$$\frac{1}{2}E_x^{n+1/2} - \frac{a_1a_2}{2}D_z^2E_x^{n+1/2} = \frac{1}{2}E_x^n - \frac{a_1a_2}{2}D_zD_xE_z^n + \frac{a_1}{2}D_yH_z^n - \frac{a_1}{2}D_zH_y^n + \frac{1}{4}s_x^{n+1/2}. \tag{26}$$

Multiplying (26) by 4 and using (19), we obtain the intermediate implicit update equation

$$E_x^* - a_1a_2D_z^2E_x^* = E_x^n + a_1a_2D_z^2E_x^n - 2a_1a_2D_zD_xE_z^n + 2a_1D_yH_z^n - 2a_1D_zH_y^n + s_x^{n+1/2}. \tag{27}$$

For the second procedure, we utilize (23) and (25) in (5c) and (6) to get

$$e_x^{n+1/2} = E_x^{n+1/2} - \frac{1}{2}E_x^n - \frac{a_1}{2}D_yH_z^n \tag{28}$$

$$h_z^{n+1/2} = \frac{1}{2}H_z^n + \frac{a_2}{2}D_yE_x^n - a_2D_xE_y^{n+1/2} + \frac{1}{2}s_{Hz}^{n+1/2}. \tag{29}$$

Substituting (28) and (29) into (7a) gives

$$\frac{1}{2}E_x^{n+1} - \frac{a_1a_2}{2}D_y^2E_x^{n+1} = E_x^{n+1/2} - \frac{1}{2}E_x^n + \frac{a_1a_2}{2}D_y^2E_x^n - a_1a_2D_yD_xE_y^{n+1/2} + \frac{a_1}{2}D_ys_{Hz}^{n+1/2}. \tag{30}$$

Multiplying (30) by 2 and using (19), we arrive at the final implicit update equation

$$E_x^{n+1} - a_1a_2D_y^2E_x^{n+1} = E_x^* + a_1a_2D_y^2E_x^n - a_1a_2D_yD_x(E_y^* + E_y^n) + a_1D_ys_{Hz}^{n+1/2}. \tag{31}$$

For the explicit update equation, utilizing (29) along with (19) in (8c) and (9) yields

$$H_z^{n+1} = H_z^n + a_2D_y(E_x^n + E_x^{n+1}) - a_2D_x(E_y^* + E_y^n) + s_{Hz}^{n+1/2}. \tag{32}$$

Equations (27), (31) and (32) can be found to coincide with equations (11a), (12a) and (13c) of [6] respectively. Note that while the magnetic source terms have been incorporated correctly in the explicit update equations therein, cf. [6, (13c)], they are missing in the implicit update equations, cf. [6, (11a),(12a)]. Although we have proven the update equations for E_x and H_z components only in the above, the correspondence for other update equations may be shown analytically in the same way, or simply by permuting the indices above, e.g. $x \rightarrow y$, $y \rightarrow z$, $z \rightarrow x$, etc. In a similar manner, one can prove the equivalence of CNCSU-FDTD method in Section III and [6]. It should be noted that the analytical proof of equivalence provided here is exact with no additional approximation throughout the analysis. Therefore both methods simply correspond to those of [6], but with their implementations simplified considerably via the present efficient algorithms. Such equivalence is again borne out in their same numerical characteristics (accuracy, stability/instability, etc.) and simulation results as to be illustrated later.

V. Discussion

A. Comparison of Algorithms

Having proven the new algorithms, some comparison with the original implementations are in order. Let us first determine the flops count taking into account the number of multiplications/divisions (M/D) and additions/subtractions (A/S) required for one complete time step. Table I lists the flops count for both original [6] and present algorithms of CNDS and CNCSU FDTD methods. The count is based on the right-hand sides of their respective implicit, explicit and auxiliary update equations using central difference operators. For simplicity, the source terms have been excluded and the number of electric and magnetic field components in all directions have been taken to be the same. It is also assumed that all multiplicative factors have been precomputed and stored.

From the table, it is clear that the flops count has been reduced substantially for both CNDS and CNCSU FDTD methods using the present efficient algorithms. In particular, the CNDS has flops count reduction from 117 to 42, which corresponds to an efficiency gain of 2.79 for the right-hand sides of update equations. For the CNCSU, its flops count has been reduced from 153 to 84, which corresponds to an efficiency gain of 1.82. Besides the right-hand sides of update equations, there is also the cost of solving (implicit) tridiagonal systems, i.e. typically about $5N$ flops for a system of order N using precomputed bidiagonally factorized elements. Taking such cost into account, the present CNDS and CNCSU algorithms still achieve overall efficiency gains of 2 and 1.6 respectively in flops count reductions over the original ones of [6]. More importantly, coding these algorithms is much simpler due to their concise form comprising many fewer terms.

Apart from arithmetic operations, the for-looping overhead incurred in most programming languages should also be considered, cf. Table I. Each for-loop is to perform the whole sweep along x, y and z directions for one field component. To avoid introducing additional for-loops in the present algorithms, some of the auxiliary updatings may be incorporated into the same loops of implicit updatings. In particular, for the efficient CNDS algorithm, the update component equations of (2) and (6) can be tied with those of (3) and (7) respectively. This then constitutes 12 for-loops altogether for all update equations in both procedures. Although it seems to take fewer (9+1) loops for the original CNDS implementation of [6], the gain in loops reduction is not that significant compared to the much more gain attained in flops reduction for the present case. Similarly for the efficient CNCSU algorithm, the update component equations of (12) and (15) can be inside the loops of (13) and (16) respectively. Since there is now no intermediate updating of magnetic field, the number of for-loops needed is only 9. For the original CNCSU implementation of [6], one extra loop is required to store the field components at time step n. This has been saved via proper field array pointer indexing in the present case.

In regard to the memory space requirement, Table I lists the field arrays needed in the original and present algorithms of CNDS and CNCSU FDTD methods. In particular, both original methods of [6] require the storage for their variables including E^n, E^*, E^{n+1} and H^{n+1}/H^n. (The slash symbol of H^{n+1}/H^n means that one field array is used alternately to store H^n and H^{n+1}.) Note that all initial, intermediate and final electric field values are required simultaneously for magnetic field updating, cf. (32), although the space of H^n may be reusable for H^{n+1}. In the present algorithms, many auxiliary field variables may occupy the same spaces of intermediate and final field variables. For instance, in the program for efficient CNDS algorithm, all $E^{n+1}/e^n/E^n$ may share the same field array to be assigned with new values successively. Hence the memory size does not increase with auxiliary variables. Furthermore, when the final magnetic field data is not needed from the present CNDS algorithm, one may omit the spaces for H^{n+1}, thereby demanding even less memory than the original implementation.

Numerical simulations have been carried out using various algorithms discussed above. The programs have been compiled using Microsoft Visual C++ under Microsoft Windows XP OS. To determine the actual computation efficiency gains in CPU time, we have run the programs on ten computers of the same NEC model with Intel Pentium 4 CPU 3.4 GHz and 1 GB RAM. (Typical available physical memory is only $\sim$ 700-800 MB due to some system processes.) Different numbers of time steps from 10 to 100 (sometimes 500) have been adopted with the CPU time found to be varying quite linearly. Two cases of $50 \times 50 \times 50$ and $200 \times 200 \times 200$ cells have been tested. In the first case, the mode CPU time for

TABLE I

COMPARISON OF ALGORITHMS

Method		Crank-Nicolson Direct-Splitting (CNDS)		Crank-Nicolson Cycle-Sweep-Uniform (CNCSU)	
Algorithm		[6]	Section II	[6]	Section III
Implicit	M/D	21	6	30	12
	A/S	66	12	93	36
Explicit +Auxiliary	M/D	6	6	6	6
	A/S	24	18	24	30
Total	M/D	27	12	36	18
	A/S	90	30	117	66
	M/D+A/S	117	42	153	84
For-Loops		9+1	12	9+1	9
Memory (Field Arrays)		E^n, E^*, E^{n+1}, H^{n+1}/H^n	$E^{n+1}/e^n/E^n$, $e^{n+1/2}/E^{n+1/2}/e^{n-1/2}$, $h^{n+1}/h^{n+1/2}/h^n$; for output: H^{n+1}	E^n, E^*, E^{n+1}, H^{n+1}/H^n	E^n, $e^{n+1/2}/E^{n+1/2}/e^{n-1/2}$, E^{n+1}/e^n, H^{n+1}/H^n

M/D – Multiplications/Divisions; A/S – Additions/Subtractions

100 time steps is 22.3 and 26.3 seconds using the CNDS and CNCSU algorithms of [6], while it is 7.50 and 16.6 seconds using the algorithms of Sections II and III. This corresponds to actual efficiency gains of 2.97 and 1.58 respectively. It is interesting to find that the gain of new CNDS algorithm can be higher than expected, which may be attributed to much less memory indexing required for many fewer terms. In the second case, the mode CPU time for 100 time steps is 2160 and 2330 seconds using the CNDS and CNCSU algorithms of [6], while it is 1460 and 1730 seconds using the algorithms of Sections II and III. This corresponds to actual efficiency gains of 1.48 and 1.35 respectively. Such reduced gains may be caused by additional OS overheads due to program executions close to available physical memory. Even though the specific gains may be higher or lower, the numerical experiments above have demonstrated that the new CNDS and CNCSU algorithms are indeed more efficient.

B. Numerical Results

For numerical illustration, we simulate an air-filled cavity meshed with $50\times50\times50$ uniform grid cells of size 2 mm each. The cavity is initialized with TM_{111} mode fields at time $t = 0$. Fig. 1 plots the time-domain electric fields at cell (31,31,26) computed using Yee and CNDS FDTD methods. Yee-FDTD is based on Courant limit time step size $\Delta t = \Delta t_{\text{CFL}}$, whereas CNDS-FDTD uses larger time step size with Courant number $\text{CFLN} = \Delta t/\Delta t_{\text{CFL}}$ being 4, 8 and 16. (CFLN cannot be too large due to limitations as discussed in [6] and [9].) The numbers of time steps to generate Fig. 1 are 4000, 1000, 500 and 250 for Yee and CNDS FDTD methods with $\text{CFLN} = 4$, 8 and 16 respectively. The CNDS-FDTD results are close to that of Yee's when the time step size is not too large and the simulation duration is not too long. Both original and present algorithms of CNDS-FDTD method have been implemented, and their results are the same with their plots not distinguishable. This again verifies the equivalence of these

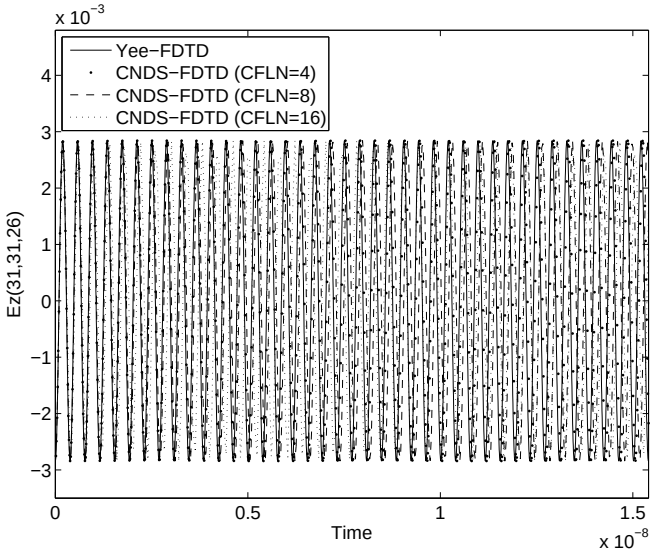

Fig. 1. Time-domain E_z computed using Yee and CNDS FDTD.

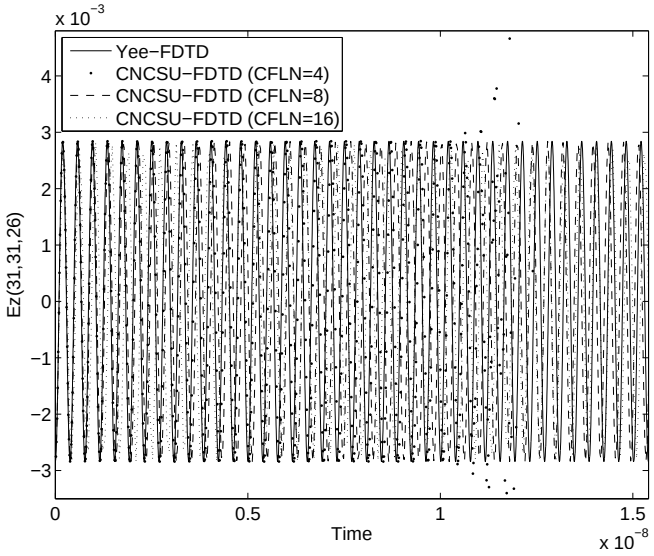

Fig. 2. Time-domain E_z computed using Yee and CNCSU FDTD.

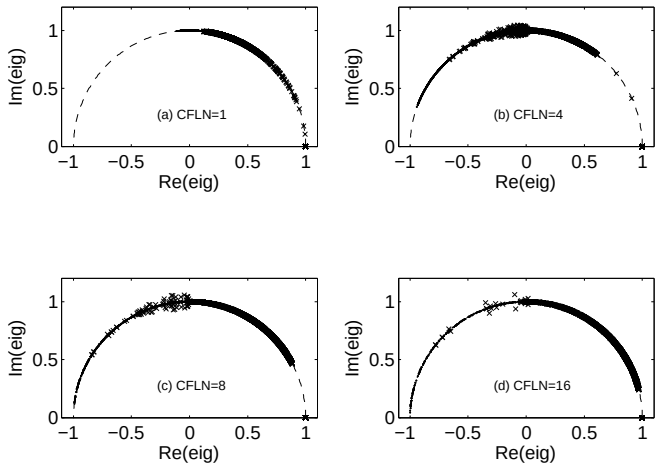

Fig. 3. Sample plots of the eigenvalues of CNCSU (cross) and CNDS (dot) Fourier-updating matrix for CFLN $= 1$, 4, 8 and 16 around unit semi-circle.

algorithms, even though they were subject to quite different codings and executions (simple and efficient in our case). In Fig. 2, the time-domain electric fields are recomputed using CNCSU-FDTD method. It is found that the computation may become unstable using either original or present algorithm. For instance, the electric field amplitude is seen to start growing after ~ 720 time steps for CFLN $= 4$ (or after ~ 550 and ~ 500 time steps – beyond Fig. 2 duration – for CFLN $= 8$ and 16). The instability of CNCSU is contrary to the previous claim [6] that it is also an unconditionally stable FDTD method.

To ascertain that instability does exist, we resort to the independent means of the Fourier method and investigate the eigenvalues of Fourier-updating (amplification) matrix. These eigenvalues may be obtained from the roots of amplification factor polynomial, cf. [6, (24)]. Note that the last term in equation (24b) of [6] should have a factor of 2 instead of 4, and the correct nonstationary amplification factors ξ should be found as solutions of

$$\xi = \frac{\chi \pm \sqrt{\chi^2 - 4}}{2} \tag{33}$$

$$\chi = \frac{-p_1 \pm \sqrt{p_1^2 - 4p_0p_2 + 8p_2^2}}{2p_2} \tag{34}$$

(p_i's are defined in [6]). For the CNCSU in our example, calculations reveal that some of their eigenvalues may be larger than one in magnitude for certain Fourier wavenumbers. To clarify further, Fig. 3 shows the sample plots of the eigenvalues of CNCSU Fourier-updating matrix for CFLN $= 1$, 4, 8 and 16 around unit semi-circle. The eigenvalues correspond to some (1000) random sets of Fourier wavenumbers along x, y and z directions. It is observed that for CFLN > 1, there may exist eigenvalues outside the unit circle, i.e., they have magnitudes greater than one. On the other hand, all the eigenvalues for CNDS are lying on the unit circle as in Fig. 3, i.e. they are always of unity magnitude. Therefore, it is ascertained that only the CNDS but not CNCSU is unconditionally stable.

VI. Conclusion

This paper has presented new efficient algorithms for implementing the three-dimensional CNDS and CNCSU FDTD methods. The algorithms involve update equations whose right-hand sides are much simpler and more concise than those of [6]. Analytical proof has been provided to show the equivalence of original and present methods. Comparison of their implementations signifies substantial reductions of flops count in the new algorithms. Other computational aspects have also been optimized, particularly in regard to the for-looping overhead and the memory space requirement. Through numerical simulation and Fourier stability analysis, it has been found that while the CNDS-FDTD is unconditionally stable, the CNCSU-FDTD may actually become unstable.

Although not all Crank-Nicolson-based FDTD methods are unconditionally stable, there may be certain features attributed to them that can be exploited at advantage, see [4]-[6]. Since the emphasis of this paper has been to devise efficient algorithms, such exploitation is beyond the scope of this paper. Meanwhile, thanks to their simplicity and efficiency, the present algorithms would be useful for further investigations and applications of CNDS and CNCSU FDTD methods (possibly along with some means of stabilization). Moreover, the implementations of many other Crank-Nicolson-based methods including CNAFS, CNDG, CNCS, etc., may be made simpler and more efficient in the similar manner.

References

[1] A. Taflove and S. C. Hagness, *Computational Electrodynamics: The Finite-Difference Time-Domain Method* (Boston, M. A.: Artech House, 2005).

[2] J. Lee and B. Fornberg, "A split step approach for the 3-D Maxwell's equations", *J. Comput. Appl. Math.*, vol. 158, pp. 485-505, 2003.

[3] J. Shibayama, M. Muraki, J. Yamauchi and H. Nakano, "Efficient implicit FDTD algorithm based on locally one-dimensional scheme," *Electron. Lett.*, vol. 41, no. 19, pp. 1046-1047, Sep. 2005.

[4] G. Sun and C. W. Trueman, "Approximate Crank-Nicolson Schemes for the 2-D Finite-Difference Time-Domain Method for TE$_z$ Waves," *IEEE Trans. Antennas Propagat.*, vol. 52, no. 11, pp. 2963-2972, Nov. 2004.

[5] G. Sun and C. W. Trueman, "Unconditionally-stable FDTD method based on Crank-Nicolson scheme for solving three-dimensional Maxwell equations," *Electron. Lett.*, vol. 40, no. 10, pp. 589-590, May 2004.

[6] G. Sun and C. W. Trueman, "Efficient Implementations of the Crank-Nicolson Scheme for the Finite-Difference Time-Domain Method," *IEEE Trans. Microwave Theory Tech.*, vol. 54, no. 5, pp. 2275-2284, May 2006.

[7] E. L. Tan, "Efficient Algorithm for the Unconditionally Stable 3-D ADI-FDTD Method," *IEEE Microw. Wireless Comp. Lett.*, vol. 17, no. 1 pp. 7-9, Jan. 2007.

[8] E. L. Tan, "Unconditionally Stable LOD-FDTD Method for 3-D Maxwell's Equations," *IEEE Microw. Wireless Comp. Lett.*, vol. 17, no. 2, pp. 85-87, Feb. 2007.

[9] G. Sun and C. W. Trueman, "Some Fundamental Characteristics of the One-Dimensional Alternate-Direction-Implicit Finite-Difference Time-Domain Method," *IEEE Trans. Microwave Theory Tech.*, vol. 52, no. 1, pp. 46-52, Jan. 2004.