Векторный алгоритм FDTD метода

М.А. Дорофеева[1]

[1]Самарский национальный исследовательский университет им. академика С.П. Королева, Московское шоссе 34А, Самара, Россия, 443086

Аннотация. Метод конечных разностей во временной области (FDTD) – метод исследования и моделирования электромагнитного поля, является прямым методом решения уравнений Максвелла. Данный метод также широко используется в нанофотонике, для моделирования интегральных схем, в повседневных инженерных расчётах и для разработки принципиально новых приборов и их частей.
На данный момент известен следующий прием векторизации вычислений при разностном решении уравнений Максвелла – алгоритм с длинными векторами на основе операции gaxpy. В настоящей работе разработан алгоритм, основанный на повторном использовании попарных сумм. Он позволит уменьшить вычислительную сложность и перейти к другому типу векторных операций.
Специфическое устройство сетки Yee препятствует повторному использованию попарных сумм, необходимо найти другую разностную схему для решения уравнений Максвелла. Такой схемой стала схема типа «крест». Ее принципиальное отличие от схемы Yee в том, что в каждом узле ее сеточной области определяются все проекции сеточных функций. Недостатком данной схемы является то, что при той же плотности сетки, что и у Yee, погрешность вычислений у новой схемы выше. Использование векторизации обеспечит сокращение длительности вычислений и, как следствие, возможность сгущения сетки и снижения погрешности вычислений.

1. Введение

Известно множество методов решения уравнений Максвелла. Выделим в качестве наиболее универсального конечно-разностный метод [1-4]. Таким является FDTD метод. Он строится путем дискретизации по времени и пространству вычислительной области.

При численном решении сеточных уравнений явных разностных схем перейдем от матричных алгоритмов, при которых сеточная область отображается на прямоугольный массив вычислительных потоков, к векторным, позволяющим получить большую степень свободы при конструировании алгоритма.

Начнем с часто применяемого метода векторизации – коротковекторного алгоритма [2]. Несмотря на все достоинства, серьезным его минусом является неполная загрузка микропроцессоров графической карты. Даже при условии достаточно густой сеточной области. Разумеется, со сгущением сеточной области всегда найдется такая дискретизация, при которой почти всем ядрам придется одновременно производить вычисления по разностной схеме. Однако с увеличением длины вектора квадратически возрастают требования к объему видеопамяти, как правило весьма ограниченному.

В силу изложенного, необходимо увеличить длину вектора без изменения параметров сеточной области. Таким образом мы перейдем к алгоритму с длинными векторами.

2. Определение нового разностного аналога системы

Прежде чем приступить к исследованию новой сеточной области, запишем двумерную систему уравнений Максвелла [5]:

$$\begin{aligned}
&\mu_0 \frac{\partial H_y}{\partial t} = -\frac{\partial E_x}{\partial z},\\
&\mu_0 \frac{\partial H_z}{\partial t} = \frac{\partial E_x}{\partial y},\\
&\varepsilon_0 \frac{\partial E_x}{\partial t} = \frac{\partial H_z}{\partial y} - \frac{\partial H_y}{\partial z},
\end{aligned} \tag{1}$$

где E_x, H_y и H_z – проекции векторов напряженностей электрического и магнитного полей, μ_0, ε_0 – магнитная и электрическая константы.

Определим область вычислений D ($0 \le t \le T$, $0 \le y \le L_y$, $0 \le z \le L_z$) и граничные условия Дирихле $E_x(t,y,0)=0$, $E_x(t,y,L_z)=0$, $E_x(t,0,z)=0$, и $E_x(t,L_y,0)=0$.

Далее необходимо задать сеточную область. Наложим на область вычислений D сеточную область D_h. Сеточные проекции электрического поля $E^m_{x_{n,k}}$ и магнитных полей $H^m_{y_{n,k}}$, $H^m_{z_{n,k}}$ определены в узлах $\{(t_m, y_n, z_k)$: $t_m = mh_t$, $m=0,1,\ldots M=T/h_t$, $y_n = nh_y$, $n=0,\ldots N=L_y/h_y$, $z_k = kh_z$, $k=0,\ldots K=L_z/h_z\}$, причем n,k – обозначают узлы по пространству, m – по времени [3]. С помощью пространственных шагов сетки h_z и h_y и временных h_t задается расстояние между узлами сетки.

Тогда для системы (1) запишем разностный аналог:

$$\begin{aligned}
&\mu_0 \frac{H^{m+1}_{y_{n,k}} - H^{m-1}_{y_{n,k}}}{2h_t} = -\frac{E^m_{x_{n,k+1}} - E^m_{x_{n,k-1}}}{2h_z},\\
&\mu_0 \frac{H^{m+1}_{z_{n,k}} - H^{m-1}_{z_{n,k}}}{2h_t} = \frac{E^m_{x_{n+1,k}} - E^m_{x_{n-1,k}}}{2h_y},\\
&\varepsilon_0 \frac{E^{m+1}_{x_{n,k}} - E^{m-1}_{x_{n,k}}}{2h_t} = \frac{H^m_{z_{n+1,k}} - H^m_{z_{n-1,k}}}{2h_y} - \frac{H^m_{y_{n,k+1}} - H^m_{y_{n,k-1}}}{2h_z}.
\end{aligned} \tag{2}$$

Расстояния между узлами задаются пространственными (h_y и h_z) и временным (h_t) шагами сетки. Данная разностная схема имеет порядок аппроксимации $O(h_t^2, h_y^2, h_z^2)$ [5]. Для удобства последующей векторизации перепишем систему уравнений (2) в следующем виде:

$$\begin{aligned}
&H^{m+1}_{y_{n,k}} = \alpha\left(E^m_{x_{n,k+1}} - E^m_{x_{n,k-1}}\right) + H^{m-1}_{y_{n,k}},\\
&H^{m+1}_{z_{n,k}} = \beta\left(E^m_{x_{n+1,k}} - E^m_{x_{n-1,k}}\right) + H^{m-1}_{z_{n,k}},\\
&E^{m+1}_{x_{n,k}} = \theta_1\left(H^m_{z_{n+1,k}} - H^m_{z_{n-1,k}}\right) - \theta_2\left(H^m_{y_{n,k+1}} - H^m_{y_{n,k-1}}\right) + E^{m-1}_{x_{n,k}},
\end{aligned} \tag{3}$$

где $\alpha = -\frac{h_t}{\mu_0 h_z}$, $\beta = \frac{h_t}{\mu_0 h_y}$, $\theta_1 = \frac{h_t}{\varepsilon_0 h_y}$, $\theta_2 = \frac{h_t}{\varepsilon_0 h_z}$.

3. Алгоритмы с короткими векторами

Начнем с наиболее простого приема векторизации, алгоритма с короткими векторами, представляющего из себя работу с векторами, длина которых совпадает с дискретизаций сеточной области по пространству (N – в случае квадратной области).

Данный алгоритм основан на операциях сложения векторов и saxpy в их строчных вариантах (алгоритм 1). Прежде чем переходить к его описанию, для удобства возьмем равные шаги по пространству $h_y = h_z$. Тогда уравнения (3) будут выглядеть следующим образом:

$$\begin{aligned}
&H^{m+1}_{y_{n,k}} = -\alpha\left(E^m_{x_{n,k+1}} - E^m_{x_{n,k-1}}\right) + H^{m-1}_{y_{n,k}},\\
&H^{m+1}_{z_{n,k}} = \alpha\left(E^m_{x_{n+1,k}} - E^m_{x_{n-1,k}}\right) + H^{m-1}_{z_{n,k}},
\end{aligned} \tag{4}$$

$$E_{x_{n,k}}^{m+1}=\theta\left(H_{z_{n+1,k}}^{m}-H_{z_{n-1,k}}^{m}-H_{y_{n,k+1}}^{m}+H_{y_{n,k-1}}^{m}\right)+E_{x_{n,k}}^{m-1},$$

где $\alpha=\frac{h_t}{\mu_0 h_z}=\frac{h_t}{\mu_0 h_y}$, $\theta=\frac{h_t}{\varepsilon_0 h_y}=\frac{h_t}{\varepsilon_0 h_z}$.

Алгоритм 1. Реализация алгоритма с короткими векторами, строчный вариант

```
for m=1:M % проход по слоям по времени
% проход по строкам
for n=1:N
Hy=Hy – c1*(Ex(n,2:N+1) – Ex(n,0:N-1));
Hz=Hz + c1*(Ex(n+1,1:N) – Ex(n-1,1:N));
Ex(n,1:N)=Ex(n,1:N) + c2*(Hz(n+1,1:N) – Hz(n-1,1:N) – Hy(n,2:N+1) + Hy(n,0:N-1));
end
end
```

В алгоритме 1 в массиве Hy хранится значение $H_{y_{n,k}}^{m}$ сеточной функции, в Hz и Ex – $H_{z_{n,k}}^{m}$ и $E_{x_{n,k}}^{m}$ соответственно. Таким образом, получаем 5N операций сложения векторов и 3N операций saxpy с векторами длины N.

Перейдя к замене поэлементного выполнения на векторный метод, можем получить ускорение вычислений алгоритма до N раз при реализации на вычислительных устройствах с одинаковой производительностью.

Следующим шагом является увеличение длины вектора без изменения параметров сеточной области.

4. Алгоритм с длинными векторами, основанный на повторном использовании попарных сумм

Для снижения длительности вычислений при решении сеточных уравнений явных разностных схем, воспользуемся идеей использования данных, полученных в результате нахождения значений сеточной функции, повторно.

Для этого рассмотрим новый способ организации массива (рисунок 1). Пусть U – это одномерный массив длины $(N+2)^2$, в который входят граничные значения.

U(1)	*U(2)*	*U(3)*	*U(4)*	...	*U(N+2)*
U(N+3)	*U(N+4)*	*U(N+5)*	*U(N+6)*	...	*U(2(N+2))*
U(2N+5)	*U(2N+6)*	*U(2N+7)*	*U(2N+8)*	...	*U(3(N+2))*
U(3N+7)	*U(3N+8)*	*U(3N+9)*	*U(3N+10)*	...	*U(4(N+2))*
...	...	...	...	...	...
U((N-1)(N+2)+1)	*U((N-1)(N+2)+2)*	*U((N-1)(N+2)+3)*	...	*U(N(N+2)-1)*	*U(N(N+2))*
U(N(N+2)+1)	*U(N(N+2)+2)*	...	*U((N+1)(N+2)-2)*	*U((N+1)(N+2)-1)*	*U((N+1)(N+2))*
U((N+2)(N+1)+1)	*U((N+2)(N+1)+2)*	*U((N+2)(N+1)+3)*	...	*U((N+2)(N+2)-1)*	*U((N+2)(N+2))*

Рисунок 1. Схема хранения массива U.

На рисунке 1 оттенками серого выделены элементы вектора U, которые участвуют в получении новых значений сеточной функции *U(N+4)*, *U(2N+7)* и *U((N+1)(N+2)-1)* (соответствующих $E_{x_{1,1}}^{m}$, $E_{x_{2,2}}^{m}$ и $E_{x_{N,N}}^{m}$). В получении *U(N+4)* задействованы элементы *U(2)*, *U(N+3)*, *U(N+5)*, *U(2N+6)* (соответствующие $E_{x_{0,1}}^{m}$, $E_{x_{1,0}}^{m}$, $E_{x_{1,2}}^{m}$ и $E_{x_{2,1}}^{m}$). При расчете элемента *U(2N+7)* используются значения *U(N+5)*, *U(2N+6)*, *U(3N+9)*, *U(N+5)* ($E_{x_{1,2}}^{m}$, $E_{x_{2,1}}^{m}$, $E_{x_{3,2}}^{m}$ и $E_{x_{2,3}}^{m}$ соответственно). Таким образом, попарная сумма элементов *U(N+5)*, *U(2N+6)* используется дважды.

Для хранения результатов попарного сложения необходимо ввести вспомогательный вектор T размерности *(N+2)(N+1)+1*, что позволит уменьшить общее количество операций. Запишем данную интерпретацию длинновекторного алгоритма подсчёта сеточной функции на каждом n+1-ом шаге.

Алгоритм 2. Длинновекторный алгоритм, основанный на повторном использовании попарных сумм

% Заполнение вектора T

T(2:(N+1)(N+2)) = H(N+3:(N+2)(N+2)−1) – H(2:(N+1)(N+2))

% Сохранение значения E^m

Z=U

% Подсчёт значений для следующего временного слоя

H(N+5:(N+1)(N+2))= H(N+5:(N+1)(N+2)) – α(U(2N+7:(N+2)(N+2)) – U(3:N(N+2)))

H(2N+6:(N+2)(N+2)-1)= H(2N+6:(N+2)(N+2)-1) + α(U(2N+7:(N+2)(N+2)) – U(2N+5:(N+2)(N+2)−2))

U(N+4:(N+1)(N+2)−1) = θ(T(N+5:(N+1)(N+2)) – T(2:N(N+2)−1)) + U_1(N+4:(N+1)(N+2)−1)

% Переинициализация вектора E^{m-1}

U_1=Z

В качестве векторных операций в данном алгоритме также используются saxpy и векторное сложение. В результате мы получаем 4 операции сложения векторов длины N^2 и 3 saxpy для векторов той же длины. Переход к длинновекторному алгоритму позволил сэкономить одну операцию сложения при вычислении каждого значения сеточной функции. Однако приходится использовать дополнительную память для хранения вектора *T*.

U(1)	*U(2)*	*U(3)*	*U(4)*	…	*U(N+2)*
U(N+3)	*U(N+4)*	*U(N+5)*	*U(N+6)*	…	*U(2(N+2))*
U(2N+5)	*U(2N+6)*	*U(2N+7)*	*U(2N+8)*	…	*U(3(N+2))*
U(3N+7)	*U(3N+8)*	*U(3N+9)*	*U(3N+10)*	…	*U(4(N+2))*
…	…	…	…	…	…
U((N-1)(N+2)+1)	*U((N-1)(N+2)+2)*	*U((N-1)(N+2)+3)*	…	*U(N(N+2)-1)*	*U(N(N+2))*
U(N(N+2)+1)	*U(N(N+2)+2)*	…	*U((N+1)(N+2)-2)*	*U((N+1)(N+2)-1)*	*U((N+1)(N+2))*
U((N+2)(N+1)+1)	*U((N+2)(N+1)+2)*	*U((N+2)(N+1)+3)*	…	*U((N+2)(N+2)-1)*	*U((N+2)(N+2))*

Рисунок 2. Обнуление граничных значений.

Еще один незначительным недостатком алгоритма 2 является замена граничных значений, выделенных серым, при переходе на новый временной слой фиктивными результатами (рисунок 2). Но такая операция обнуления заранее известных позиций вектора не должна заметного увеличивать вычислительную сложность.

5. Заключение

Автором подробно рассмотрено применение методики построения векторных алгоритмов с повторным использованием попарных сумм дифференциального шаблона на примере выбранной разностной схемы [5] для уравнений Максвелла по сравнению со широко известной схемой Yee [6]. Для использованной схемы возможно применение указанного приема векторизации и, следовательно, построение алгоритма с длинными векторами.

6. Литература

[1] Боресков, А.В. Основы работы с технологией CUDA / А.В. Боресков, А.А. Харламов. – М.: ДМК Пресс, 2010. – 232 с.

[2] Воротникова, Д.Г. Алгоритмы с «длинными» векторами решения сеточных уравнений явных разностных схем / Д.Г. Воротникова, Д.Л. Головашкин // Компьютерная оптика. – 2015. – Т. 39, № 1. – С. 87-93.

[3] Головашкин, Д.Л. Разностный метод решения уравнений Максвелла: учеб. пособие / Д.Л. Головашкин, Н.Л. Казанский. – Самара: Изд-во Самар. гос. аэрокосм. ун-та, 2007. – 160 с.

[4] Самарский, А.А. Методы решения сеточных уравнений / А.А. Самарский, Е.С. Николаев. – Москва: Наука, 1978. – 592 с.

[5] Taflove, A. Computational Electrodynamics: The Finite-Difference Time-Domain Method / A. Taflove, S. Hagness. – Boston: Arthech House Publishers, 2005. – 852 p.

[6] Liu, Y. Fourier Analysis of Numerical Algorithms for Maxwell Equations / Y. Liu // Journal of Computational Physics, 1993. DOI:10.1006/jcph.1996.0068.

Vector algorithm of FDTD method

M.A. Dorofeeva[1]

[1]Samara National Research University, Moskovskoe Shosse 34A, Samara, Russia, 443086

Abstract. Finite-difference time-domain method is a direct method used for solving Maxwell's equations. This method is also widely used in the nanophotonic, in daily engineering calculations and in the pipeline of crucially new devices.
Today we know only the algorithm with long vectors based on the gaxpy operation which is used for solving Maxwell's equations. In this paper we present new vector algorithm based on the reusing of pairwise sums. Implementation of this algorithm reduces the computational complexity.
The qualities of the Yee's grid prevent reusing of pairwise sums. For this reason, we need to use new "cross" type scheme. Its fundamental difference from the Yee's scheme is that all projections of the grid functions are defined in every node of grid domain. At the same grid density, the disadvantage of this scheme is the low computational accuracy which is worse than Yee's scheme [1, 6] computational accuracy. Vectorization abbreviate the evaluation length resulting in the step of grid will decline, and so computational error will decrease.

Keywords: FDTD method, Maxwell's equations, difference scheme, vector algorithm.