



**UNIVERSIDAD
DE GRANADA**

Facultad de Ciencias

GRADO EN FISICA

TRABAJO FIN DE GRADO

**IMPLEMENTACIÓN DE
MÉTODOS
EN DIFERENCIAS FINITAS
EN GPU_s**

Presentado por:

D. JUAN JOSÉ SALAZAR LÓPEZ

Curso Académico 2021/2022

Resumen

El método numérico de las diferencias finitas dependientes del tiempo (FDTD) se usa como herramienta para la resolución de ecuaciones diferenciales de primer orden. Este cobra especial importancia en la electrodinámica computacional dado que nos permite resolver las ecuaciones de Maxwell realizando algoritmos con los que calcular punto a punto los valores del campo electromagnético. Generalmente se implementan estos algoritmos usando la potencia de cálculo que nos proporcionan las CPUs, en este TFG se ha generado un algoritmo 'open source' en cuda para exponer las ventajas que se obtienen al implementar este método usando el poder de paralelización que nos proporcionan las GPUs.

Abstract

FDTD method implementation for the resolution of one order differential equation in GPUs to achieve an improvement in the calculus time.

Índice

1	Introducción	4
2	Metodología	5
3	Algoritmo	8
4	CPU	9
5	Resultados	10
6	Una sección	11
7	Conclusiones	11
	Referencias	12

1 Introducción

El método numérico de las diferencias finitas es usado en muchos campos para la resolución de ecuaciones diferenciales de primer orden. En particular, el método de la diferencia central finita cobra especial importancia en la electrodinámica computacional donde se emplea para resolver las ecuaciones de Maxwell en el dominio del tiempo, discretizando tiempo y el espacio para poder implementar la solución en un algoritmo. Actualmente su uso está muy extendido ya que posibilita el estudio de la propagación de señales electromagnéticas por diversos medios. En particular puede usarse para determinar la radiación que actúa sobre una cabeza humana producida por un teléfono móvil o la propagación de una señal en el uso de radares marítimos entre otras aplicaciones.

Desde que se desarrolló el método y se generaron los primeros algoritmos se ha hecho uso del poder de cómputo que nos proporcionan las CPUs, el cual ha ido creciendo con el paso de los años. No obstante, al aumentar la dimensión espacial en el modelo también aumenta el tiempo necesario para obtener la solución dado que se debe calcular punto a punto los valores de los campos al aplicar el método. Este problema puede solventarse haciendo uso del poder de paralelización que nos ofrecen las tarjetas gráficas (GPUs), pudiendo hacer simultáneamente, hasta cierto límite, todas las operaciones que se realizan punto a punto en la dimensión espacial. Proceso que se consigue implementando los algoritmos FDTD en el lenguaje CUDA, con el cual podemos especificar punto a punto el cálculo a realizar por cada hilo dentro de la GPU.

En este TFG el autor ha desarrollado varios códigos en CUDA de libre uso para la resolución de las ecuaciones de Maxwell aplicando el método FDTD en dos dimensiones. Los códigos generados pueden encontrarse en el siguiente repositorio de github:

<https://github.com/juanjo1213/TFG-FDTD>

En el mismo pueden encontrarse los códigos para la solución en 2D implementada en la cpu y en la gpu, siendo todos los algoritmos del repositorio de libre uso.

2 Metodología

El método numérico de las diferencias finitas se emplea en la resolución de ecuaciones diferenciales de primer orden, en particular, el método de la diferencia central finita se basa en aproximar de forma sencilla la derivada de una función en un punto como la diferencia de dicha función evaluada en dos puntos equidistantes al mismo. Haciendo uso del desarrollo en serie de Taylor y aproximando a segundo orden obtenemos:

$$f(x+h) = y(x) + hy'(x) + \frac{h^2}{2}y''(x) + \frac{h^3}{6}y'''(x) \quad (2.1)$$

$$f(x-h) = y(x) - hy'(x) + \frac{h^2}{2}y''(x) - \frac{h^3}{6}y'''(x) \quad (2.2)$$

$$y'(x) = \frac{y(x+h) - y(x-h)}{2h} + O(h^2) \quad (2.3)$$

Donde hemos despejado el valor de la derivada de la función en un punto mediante su desarrollo en serie en $(x+h)$ y $(x-h)$.

Recordando la expresión de las ecuaciones de Maxwell para el campo electromagnético, haciendo uso del vector campo D.

$$\frac{\partial D}{\partial t} = \nabla \times H \quad (2.4)$$

$$D(\omega) = \epsilon_0 \epsilon_r^*(\omega) E(\omega) \quad (2.5)$$

$$\frac{\partial H}{\partial t} = -\frac{1}{\mu_0} \nabla \times E \quad (2.6)$$

Podemos observar que están expresadas en el dominio de la frecuencia para poder aplicarlas a medios cuya constante dieléctrica dependa de la misma. Resaltar que ϵ_0 y ϵ_r corresponden a las constantes dieléctricas del vacío y del medio respectivamente, y μ_0 es la permeabilidad magnética del vacío.

Para simplificar el formalismo es conveniente normalizar las ecuaciones haciendo uso de las constantes dieléctrica y permeabilidad magnética del vacío.

$$\tilde{E} = \sqrt{\frac{\epsilon_0}{\mu_0}} E \quad (2.7)$$

$$\tilde{D} = \frac{1}{\sqrt{\epsilon_0 \mu_0}} D \quad (2.8)$$

De este modo podemos expresar las ecuaciones de Maxwell para el campo electromagnético como:

$$\frac{\partial \tilde{D}}{\partial t} = -\frac{1}{\sqrt{\epsilon_0 \mu_0}} \nabla \times H \quad (2.9)$$

$$\tilde{D}(\omega) = \varepsilon_r^*(\omega) \tilde{E}(\omega) \quad (2.10)$$

$$\frac{\partial H}{\partial t} = -\frac{1}{\sqrt{\varepsilon_0 \mu_0}} \nabla \times E \quad (2.11)$$

En \mathbb{R}^3 tenemos 3 componentes para el campo eléctrico y 3 para el magnético $\tilde{E}_x, \tilde{E}_y, \tilde{E}_z, H_x, H_y, H_z$, por lo que, si queremos aplicar un modelo en dos dimensiones debemos elegir uno de los dos grupos de tres vectores, (i) el transversal magnético, formado por \tilde{E}_z, H_x, H_y , o (ii) el transversal eléctrico, compuesto por los vectores \tilde{E}_x, \tilde{E}_y y H_z .

Usaremos el modo transversal magnético y utilizaremos directamente las letras E y D sin tildar para nombrar a las componentes normalizadas de los campos:

$$\frac{\partial D_z}{\partial t} = \frac{1}{\sqrt{\varepsilon_0 \mu_0}} \left(\frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y} \right) \quad (2.12)$$

$$D(\omega) = \varepsilon_r^*(\omega) \tilde{E}(\omega) \quad (2.13)$$

$$\frac{\partial H_x}{\partial t} = -\frac{1}{\sqrt{\varepsilon_0 \mu_0}} \frac{\partial E_z}{\partial y} \quad (2.14)$$

$$\frac{\partial H_y}{\partial t} = \frac{1}{\sqrt{\varepsilon_0 \mu_0}} \frac{\partial E_z}{\partial x} \quad (2.15)$$

Estas son las ecuaciones de una onda plana moviéndose en la dirección x e y con el campo eléctrico orientado en la dirección z y el magnético en la dirección x e y. Las componentes a calcular son ecuaciones diferenciales de primer orden, por lo que, aplicando el método de la diferencia central término a término obtenemos.

$$\begin{aligned} \frac{D_z^{n+1/2}(i, j) - D_z^{n-1/2}(i, j)}{\Delta t} &= \frac{1}{\sqrt{\varepsilon_0 \mu_0}} \left[\frac{H_y^n(i + \frac{1}{2}, j) - H_y^n(i - \frac{1}{2}, j)}{\Delta x} \right] \\ &\quad - \frac{1}{\sqrt{\varepsilon_0 \mu_0}} \left[\frac{H_x^n(i, j + \frac{1}{2}) - H_x^n(i, j - \frac{1}{2})}{\Delta x} \right] \end{aligned} \quad (2.16)$$

$$\frac{H_x^{n+1}(i, j + \frac{1}{2}) - H_x^n(i, j + \frac{1}{2})}{\Delta t} = -\frac{1}{\sqrt{\varepsilon_0 \mu_0}} \left[\frac{E_z^{n+1/2}(i, j + 1) - E_z^{n+1/2}(i, j)}{\Delta x} \right] \quad (2.17)$$

$$\frac{H_y^{n+1}(i + \frac{1}{2}, j) - H_y^n(i + \frac{1}{2}, j)}{\Delta t} = \frac{1}{\sqrt{\varepsilon_0 \mu_0}} \left[\frac{E_z^{n+1/2}(i + 1, j) - E_z^{n+1/2}(i, j)}{\Delta x} \right] \quad (2.18)$$

En estas ecuaciones el tiempo se especifica con el superíndice, donde n es el paso temporal, de modo que $t = \Delta t \cdot n$, en los paréntesis especificamos los valores de i, j para determinar la posición, sin olvidar que el campo eléctrico y el magnético están intercalados en tiempo y espacio, lo cual se especifica con la suma y resta de valores $\frac{1}{2}$ en los superíndices y paréntesis. Asumiremos $\Delta x = \Delta y$.

Una vez se ha elegido el tamaño de celda ($\Delta x \cdot \Delta y$) podemos calcular el salto temporal como:

$$\Delta t = \frac{\Delta x}{2 \cdot c_0} \quad (2.19)$$

Donde c_0 es la velocidad de la luz en el vacío. Recordando que $\epsilon_0 \mu_0 = 1/(c_0)^2$ y teniendo en cuenta la normalización utilizada:

$$\frac{\Delta t}{\sqrt{\epsilon_0 \mu_0} \Delta x} = \frac{\Delta x}{2 \cdot c_0} \cdot \frac{1}{\sqrt{\epsilon_0 \mu_0} \Delta x} = \frac{1}{2} \quad (2.20)$$

Lo que nos permite, junto a las expresiones discretizadas en tiempo y espacio de las ecuaciones de Maxwell, generar un algoritmo de forma sencilla para calcular los campos:

$$dz[i, j] = dz[i, j] + 0.5 * (hy[i, j] - hy[i - 1, j] - hx[i, j] + hx[i, j - 1]) \quad (2.21)$$

$$ez[i, j] = gaz[i, j] * dz[i, j] \quad (2.22)$$

$$hx[i, j] = hx[i, j] + 0.5 * (ez[i, j] - ez[i, j + 1]) \quad (2.23)$$

$$hy[i, j] = hy[i, j] + 0.5 * (ez[i + 1, j] - ez[i, j]) \quad (2.24)$$

3 Algoritmo

```

73 //Calculate Dz
74 if (0 < fil && fil < ydim && 0 < col && col < xdim) {
75     field[fil * xdim + col].dz += 0.5 * (field[fil * xdim + col].hy - field[fil * xdim + col - 1].hy -
76     field[fil * xdim + col].hx + field[(fil - 1) * xdim + col].hx);
77 }
78 __syncthreads();
79
80 //Gaussian pulse in the middle
81 double pulse = exp(-0.5 * ((t0 - ts) / spread) * ((t0 - ts) / spread));
82 field[jc * xdim + ic].dz = pulse;
83 __syncthreads();
84
85 //Calculate the Ez field from Dz
86 if (0 < fil && fil < ydim && 0 < col && col < xdim) {
87     field[fil * xdim + col].ez = field[fil * xdim + col].gaz * field[fil * xdim + col].dz;
88 }
89 __syncthreads();
90
91 //Calculate the Hx field
92 if (fil < ydim - 1 && col < xdim - 1) {
93     field[fil * xdim + col].hx += 0.5 * (field[fil * xdim + col].ez - field[(fil + 1) * xdim + col].ez);
94 }
95 __syncthreads();
96
97 //Calculate the Hy field
98 if (fil < ydim - 1 && col < xdim - 1) {
99     field[fil * xdim + col].hy += 0.5 * (field[fil * xdim + col + 1].ez - field[fil * xdim + col].ez);
100 }

```

Figura 1: Implementación de las ecuaciones (2.21-2.24) para el cálculo de los campos en la GPU


```

66 //Main loop
67 for (int time_step = 1; time_step < nsteps + 1; time_step++)
68 {
69     //Calculate Dz
70     for (int j = 1; j < ydim; j++) {
71         for (int i = 1; i < xdim; i++) {
72             field[j*xdim+i].dz = field[j * xdim + i].dz + 0.5 * (field[j*xdim+i].hy - field[j * xdim + i-1].hy -
73                 field[j*xdim+i].hx + field[(j-1) * xdim + i].hx);
74         }
75     }
76
77     //Gaussian pulse in the middle
78     double pulse = exp(-0.5 * ((t0 - time_step) / spread) * ((t0 - time_step) / spread));
79     field[jc * xdim + ic].dz = pulse;
80
81     //Calculate the Ez field from Dz
82     for (int j = 1; j < ydim; j++) {
83         for (int i = 1; i < xdim; i++) {
84             field[j * xdim + i].ez = gaz[j * xdim + i] * field[j * xdim + i].dz;
85         }
86     }
87
88     //Calculate the Hx field
89     for (int j = 0; j < ydim-1; j++) {
90         for (int i = 0; i < xdim-1; i++) {
91             field[j * xdim + i].hx = field[j * xdim + i].hx + 0.5 * (field[j * xdim + i].ez - field[(j + 1) * xdim + i].ez);
92         }
93     }
94
95     //Calculate the Hy field
96     //Calculate the Hx field
97     for (int j = 0; j < ydim - 1; j++) {
98         for (int i = 0; i < xdim - 1; i++) {
99             field[j * xdim + i].hy = field[j * xdim + i].hy + 0.5 * (field[j * xdim + i + 1].ez - field[j * xdim + i].ez);
100         }
101     }
102 }
103

```

Figura 2: Implementación de las ecuaciones (2.21-2.24) para el cálculo de los campos en la CPU

4 CPU

5 Resultados

6 Una sección

7 Conclusiones

Referencias