Высокопроизводительное моделирование распространения электромагнитного поля на графических процессорах с гибким использованием ресурсов*

Д.А. Жердев, В.А. Фурсов

Самарский государственный аэрокосмический университет имени С.П. Королева,
Институт систем обработки изображений РАН

Рассматривается задача моделирования диаграмм рассеяния техногенных объектов на подстилающей поверхности. Расчет электромагнитного поля в непосредственной близости от объекта производится с помощью явного разностного решения уравнений Максвелла в 3-х измерениях. По сравнению с предшествующей версией программы изменен порядок формирования электромагнитного отклика на заданной структуре, что обеспечивает существенную экономию памяти.

1. Постановка задачи

В предыдущей работе авторов [1] был проведен обзор различных методов решения задачи определения рассеяния электромагнитного поля от техногенных объектов на подстилающих поверхностях. Из всех рассмотренных методов был выбран конечно-разностный явный метод решения уравнений Максвелла во временной области, поскольку он позволяет строить достаточно точные радиолокационные портреты объектов, а так же хорошо декомпозируется по данным.

Следуя методу, описанному в работе [2], предлагается находить поле в дальней точке во временной области. При этом достигается экономия ресурсов, т.к. можно хранить меньшее количество данных в оперативной памяти компьютера и в оперативной памяти видео карты. Достоинством данного метода является моделирование распространения электромагнитного поля на больших сетках для расчета облучения моделей разного размера с одной длиной волны.

2. Схема расчета поля в дальней точке

В работе [1] для нахождения поля на значительном удалении от рассеивающего объекта, когда к принимающей антенне приходят плоские волны, применяется метод преобразования ближнего поля в дальнее поле. Ближнее поле определяется с помощью эквивалентной поверхностной теоремы [3]. Величины эквивалентных электрических и магнитных токов определяются на некоторой замкнутой поверхности, которая заключает в себе исследуемый объект:

$$\mathbf{M} = -\left[\mathbf{E}\times\mathbf{n}\right]\delta\left(\mathbf{r}-\mathbf{r}'\right),$$

$$\mathbf{J} = \left[\mathbf{H}\times\mathbf{n}\right]\delta\left(\mathbf{r}-\mathbf{r}'\right),$$

где $\mathbf{r}'$ - точка на замкнутой поверхности. $\delta\left(\mathbf{r}-\mathbf{r}'\right)$ - функция, сингулярная на замкнутой поверхности, $\mathbf{n}$ - вектор внешний нормали замкнутой поверхности.

Поля $\hat{\mathbf{E}}(\mathbf{r},\omega)$, $\hat{\mathbf{H}}(\mathbf{r},\omega)$, $\hat{\mathbf{M}}(\mathbf{r},\omega)$ и $\hat{\mathbf{J}}(\mathbf{r},\omega)$ в свою очередь можно представить как прямое преобразование Фурье временного сигнала $\mathbf{E}(\mathbf{r},t)$, $\mathbf{H}(\mathbf{r},t)$, $\mathbf{M}(\mathbf{r},t)$ и $\mathbf{J}(\mathbf{r},t)$:

$$\mathbf{E}(\mathbf{r},t) = \int \hat{\mathbf{E}}(\mathbf{r},\omega)e^{i\omega t}d\omega,$$

$$\mathbf{H}(\mathbf{r},t) = \int \hat{\mathbf{H}}(\mathbf{r},\omega)e^{i\omega t}d\omega,$$

$$\mathbf{M}(\mathbf{r},t) = \int \hat{\mathbf{M}}(\mathbf{r},\omega)e^{i\omega t}d\omega,$$

* Работа выполнена при поддержке РФФИ (проекты № 12-07-00581, 13-07-13166).

$$\mathbf{J}(\mathbf{r},t)=\int\hat{\mathbf{J}}(\mathbf{r},\omega)e^{i\omega t}d\omega\,.$$

Следуя рассуждениям, представленным в работе [4], поле в дальней точке в сферической системе координат (r,θ,φ) можно представить в виде:

$$\mathbf{H}=\frac{ike^{-ikr}}{4\pi r}\begin{pmatrix}0\\-F_{\varphi}(\hat{\mathbf{M}})-Z_0F_{\theta}(\hat{\mathbf{J}})\\F_{\theta}(\hat{\mathbf{M}})-Z_0F_{\varphi}(\hat{\mathbf{J}})\end{pmatrix}, \tag{1}$$

$$\mathbf{E}=\frac{ike^{-ikr}}{4\pi r}\begin{pmatrix}0\\F_{\varphi}(\hat{\mathbf{J}})-Z_0^{-1}F_{\theta}(\hat{\mathbf{M}})\\-F_{\theta}(\hat{\mathbf{J}})-Z_0^{-1}F_{\varphi}(\hat{\mathbf{M}})\end{pmatrix}, \tag{2}$$

где функции $F_{\varphi}(\mathbf{a})$ и $F_{\theta}(\mathbf{a})$ вычисляются как:

$$F_{\varphi}(\mathbf{a})=\iint(-a_x\sin\varphi+a_y\cos\varphi)e^{ikr'\cos\psi}dS\,,$$

$$F_{\theta}(\mathbf{a})=\iint(a_x\cos\theta\cos\varphi+a_y\cos\theta\sin\varphi-a_z\sin\theta)e^{ikr'\cos\psi}dS,$$

а $r'\cos\psi=x'\cos\varphi\sin\theta+y'\sin\varphi\sin\theta+z'\cos\theta$ - проекция вектора r' на единичный орт компоненты r сферической системы координат. Здесь $Z_0=\sqrt{\mu_0/\varepsilon_0}$ – импеданс свободного пространства. Для известной частоты сигнала путем преобразования ближнего поля в дальнее определяем поля $\mathbf{E}$ и $\mathbf{H}$ в дальней точке $\mathbf{r}$. Тогда мощность рассеянной волны рассчитывается по формуле:

$$P_{отр}=\mathrm{Re}(\hat{E}_{\theta}\hat{H}_{\varphi}^{*}-\hat{E}_{\varphi}\hat{H}_{\theta}^{*}).$$

3. Алгоритм расчета поля в дальней точке с гибким использованием ресурсов

Чем больше фиктивная граница, на которой определяются эквивалентные электрические и магнитные токи, тем больше необходимо временных шагов для нахождения временной формы поля на данной поверхности. Следуя работе [2] хранить поле на фиктивной границе необходимо только для двух соседних временных шагов, а поле в дальней точке вычислять в заданный момент времени.

Запишем для формул (1) и (2) аналоги с временными компонентами, волновое число представим как $k=\frac{\omega}{c}$. Тогда поле в дальней точке представимо в виде

$$\mathbf{H}=\frac{1}{4\pi rc}\begin{pmatrix}0\\-G_{\varphi}(\mathbf{M})-Z_0G_{\theta}(\mathbf{J})\\G_{\theta}(\mathbf{M})-Z_0G_{\varphi}(\mathbf{J})\end{pmatrix}, \tag{3}$$

$$\mathbf{E}=\frac{1}{4\pi rc}\begin{pmatrix}0\\G_{\varphi}(\mathbf{J})-Z_0^{-1}G_{\theta}(\mathbf{M})\\-G_{\theta}(\mathbf{J})-Z_0^{-1}G_{\varphi}(\mathbf{M})\end{pmatrix}, \tag{4}$$

где функции $G_{\varphi}(\vec{a})$ и $G_{\theta}(\vec{a})$ имеют вид:

$$G_{\varphi}(\mathbf{a}) = \frac{\partial}{\partial t}\iint (-a_x(\tau)\sin\varphi + a_y(\tau)\cos\varphi)dS\,, \tag{5}$$

$$G_{\theta}(\mathbf{a}) = \frac{\partial}{\partial t}\iint (a_x\cos\theta\cos\varphi + a_y\cos\theta\sin\varphi - a_z\sin\theta)dS\,. \tag{6}$$

Здесь аргумент $\tau = t - \dfrac{r - r'\cos\psi}{c}$ определяет время задержки отклика поля на фиктивной границе в дальнюю точку.

Для численного моделирования производная и интеграл в формулах (5) и (6) заменяются разностными аналогами. При вычислении разностей, так же как в явной схеме решения разностных уравнений Максвелла, рассмотренной в работе [1], порядок аппроксимации второй, а при замене интеграла интегральной суммой порядок аппроксимации первый.

Запишем разностные формулы для времени задержки $\tau_n = t_n - \dfrac{r - r'\cos\psi}{c\Delta t}$ и соотношений (3), (4):

$$\mathbf{H}^{n+f} = \frac{1}{4\pi rc}\begin{pmatrix} 0 \\ -G_{\varphi}(\mathbf{M}^{n+1/2}) - Z_0 G_{\theta}(\mathbf{J}^n) \\ G_{\theta}(\mathbf{M}^{n+1/2}) - Z_0 G_{\varphi}(\mathbf{J}^n) \end{pmatrix}, \tag{7}$$

$$\mathbf{E}^{n+1/2+f} = \frac{1}{4\pi rc}\begin{pmatrix} 0 \\ G_{\varphi}(\mathbf{J}^n) - Z_0^{-1} G_{\theta}(\mathbf{M}^{n+1/2}) \\ -G_{\theta}(\mathbf{J}^n) - Z_0^{-1} G_{\varphi}(\mathbf{M}^{n+1/2}) \end{pmatrix}. \tag{8}$$

Разностные формулы, построенные по выражениям (5) и (6) для плоскости $x = x_0$ будут иметь вид:

$$G_{\varphi}(\mathbf{a}) = \sum_j\sum_k (-\frac{a_x^{n+1}(x_0, j, k) - a_x^n(x_0, j, k)}{\Delta t}\sin\varphi + \frac{a_y^{n+1}(x_0, j, k) - a_y^n(x_0, j, k)}{\Delta t}\cos\varphi)\Delta y\Delta z\,, \tag{9}$$

$$\begin{aligned} G_{\theta}(\mathbf{a}) = \sum_j\sum_k (&\frac{a_x^{n+1}(x_0, j, k) - a_x^n(x_0, j, k)}{\Delta t}\cos\theta\cos\varphi + \frac{a_y^{n+1}(x_0, j, k) - a_y^n(x_0, j, k)}{\Delta t}\times \\ &\times\cos\theta\sin\varphi - \frac{a_z^{n+1}(x_0, j, k) - a_z^n(x_0, j, k)}{\Delta t}\sin\theta)\Delta y\Delta z. \end{aligned} \tag{10}$$

3. Параллельная реализация алгоритма

Основная трудность при реализации параллельного алгоритма суммирования на CUDA [4] заключается в разделении ячеек памяти, в которые записывается сумма, на блок. При реализации разностных соотношений (7) – (10) в каждой точке фиктивной границы S вычисляется количество временных шагов распространения возмущения до дальней точки $f = \dfrac{r - r'\cos\psi}{c\Delta t}$. В массивах f_X, f_Y, f_Z размерами cubeY*cubeZ, cubeY*cubeZ, cubeX*cubeY соответственно будет храниться величина f, где cubeX, cubeY, cubeZ – линейные размеры фиктивной границы. На рисунке 1 наглядно продемонстрировано расположение фиктивной границы S и распределения величины f на ней.

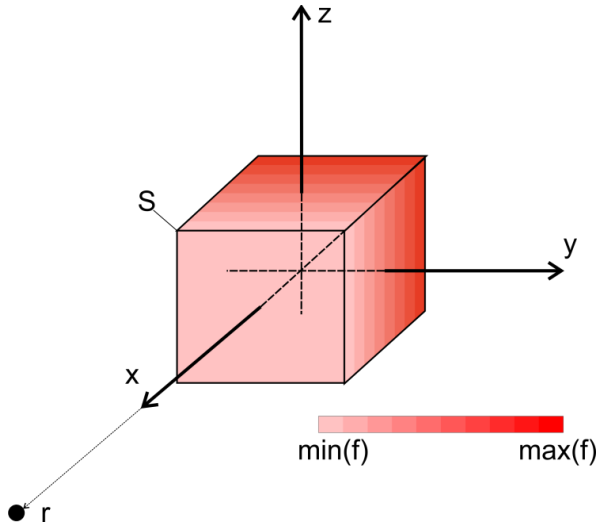

Рис. 1. Диаграмма распределения величины f по фиктивной границе S

Для наглядной иллюстрации работы алгоритма рассмотрим численное интегрирование в плоскости $x = x_0$. На первом шаге алгоритма запускается ядро, выполняющееся на видеокарте с размерностью сетки равной cubeY и размерностью блока равной cubeZ, с помощью которого запишем в массив результаты суммирования в (9) и (10). В момент запуска вычислительного ядра на блок динамически выделяется разделяемая память размером 5*cubeZ*sizeof(float). Листинг алгоритма приведен ниже.

```
      unsigned int bx = blockIdx.x;
      extern __shared__ float Jy[];
      extern __shared__ float Jz[];
      extern __shared__ float My[];
      extern __shared__ float Mz[];
      extern __shared__ int delay[];

      Jy[threadIdx.x] = JyX[k + bx*2 + threadIdx.x*2*cubeY];
      Jz[threadIdx.x + cubeZ] = JzX[k + bx*2 +
threadIdx.x*2*cubeY];
      My[threadIdx.x + 2*cubeZ] = MyX[k + bx*2 +
threadIdx.x*2*cubeY];
      Mz[threadIdx.x + 3*cubeZ] = MzX[k + bx*2 +
threadIdx.x*2*cubeY];
      delay[threadIdx.x + 4*cubeZ] = f_X[bx + threadIdx.x*cubeY +
k*cubeY*cubeZ + n*2*cubeY*cubeZ];
      __syncthreads();

      if(threadIdx.x == 0){
            for(unsigned int j = 0; j < blockDim.x; j++){
                  HtetaTemporary[bx + delay[j +
         4*cubeZ]*blockDim.x] +=
         g(0, Jy[j], 0, My[j + 2*cubeZ], Mz[j + 3*cubeZ], fi, teta,
         S0, S1, consX);
                  HfiTemporary[bx + delay[j + 4*cubeZ]*blockDim.x]
         +=
         g(0, Jy[j], Jz[j + cubeZ], 0, My[j + 2*cubeZ], fi, teta,
         S2, S1, consX);
            }
      }
```

Рис. 2. Первый этап алгоритма вычисления поверхностного интеграла

Следующим этапом алгоритма является суммирование с размерностью блока cubeY и размерностью сетки равной максимальному значению из массива f_X. В момент запуска вычислительного ядра на блок динамически выделяется разделяемая память размером 2*cubeY*sizeof(float). Листинг алгоритма приведен ниже.

```
unsigned int bx = blockIdx.x;
      extern __shared__ float sHteta[];
      extern __shared__ float sHfi[];

      sHteta[threadIdx.x] = HtetaTemporary[threadIdx.x +
bx*cubeY];
      sHfi[threadIdx.x + cubeY] = HfiTemporary[threadIdx.x +
bx*cubeY];
      __syncthreads();

      if(threadIdx.x == 0){
            for(unsigned int j = 0; j < blockDim.x; j++){
                        Hteta[timeCount + bx + n*timeStep] =
         cuCaddf(Hteta[timeCount + bx + n*timeStep],
         make_cuComplex(sHteta[j], 0.0f));
                        Hfi[timeCount + bx + n*timeStep] =
         cuCaddf(Hfi[timeCount + bx + n*timeStep],
         make_cuComplex(sHfi[j + cubeY], 0.0f));
                        HtetaTemporary[j + bx*cubeY] = 0;
                        HfiTemporary[j + bx*cubeY] = 0;
            }
      }
```

Рис. 3. Первый этап алгоритма вычисления поверхностного интеграла

На рисунке 4 приведен график ускорения работы алгоритма на видеокарте Geforce 8800GT, по сравнению со скоростью реализации алгоритма на процессоре Athlon X6400 3Ghz.

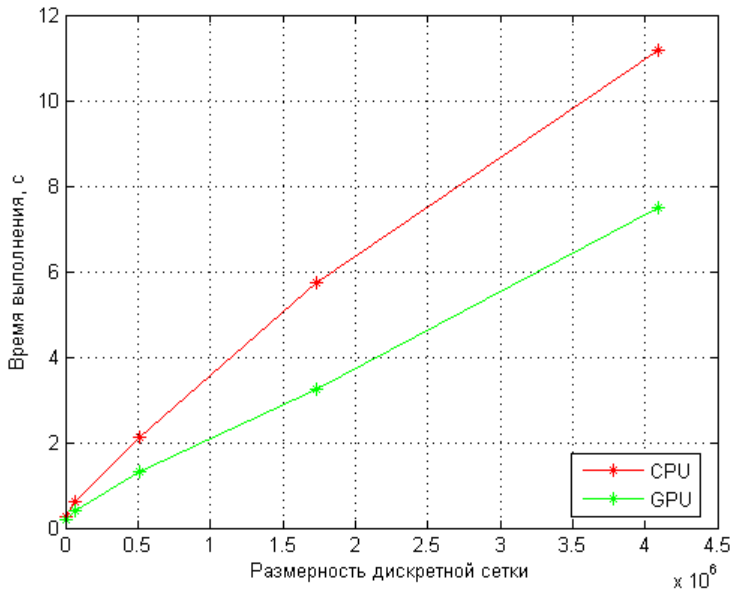

Рис. 4. Сравнение времени выполнения программы

4. Заключение

Полученные результаты показывают возможность существенного увеличения области моделирования. Это позволит расширить классы моделируемых объектов для решения задач распознавания по по радиолокационным портретам.

Литература

1. Жердев Д.А. Фурсов В.А. Высокопроизводительное моделирование распространения электромагнитного поля с использованием технологии CUDA // Сборник трудов международной научной конференции Параллельные вычислительные технологии 2013, г. Челябинск. С. 338-345.
2. Luebbers, R.J. Kunz K.S., Schneider M., Hunsberger F., “A finite-difference time-domain near zone to far zone transformation”, IEEE Trans. Antennas and Propogation, Vol. 39, 1991, P. 429-433.
3. Schelkunoff, S. A., “Some equivalence theorems of electromagnetic and their application to radiation problems”, Bell System Technical Journal, Vol. 15, 1936, P. 92-112.
4. NVIDIA CUDA. Nvidia CUDA C Programming Guide // Version 4.2. *16.4.2012. URL:* http://developer.download.nvidia.com/compute/DevZone/docs/html/C/doc/CUDA_C_Programming_Guide.pdf *(*request data: 29.11.2012*).*