6. Kirsh D.V., Kupriyanov A.V. Modeling and Identification of Centered Crystal Lattices in Three-Dimensional Space. CEUR Workshop Proceedings, 2015; 1490. P. 162-170.

7. Shirokanev A.S., Kirsh D.V., Kupriyanov A.V., Application of gradient steepest descent method to the problem of crystal lattice parametric identification. CEUR Workshop Proceedings, 2016; 1638. – P. 393-400.

8. Shirokanev, A.S. Development of the crystal lattice parameter identification method based on the gradient steepest descent method / A. S. Shirokanev, D. V. Kirsh, A. V. Kupriyanov // Computer Science Research Notes. - 2016. - Vol. 2603. – P. 65-68.

9. Шаскольская М.П. Кристаллография. - Учеб. пособие. - 2-е изд., перераб. и доп. - М.: Высш. шк., 1984. - 376 с.

10. Боресков А.В., Харламов А.А. Основы работы с технологией CUDA. – М.: ДМК Пресс, 2010. – 232 с.

Л.В. Яблокова, Д.Л. Головашкин, О.В. Калюжная

ПРИМЕНЕНИЕ МЕТОДА ПИРАМИД ПРИ РАЗНОСТНОМ РЕШЕНИИ УРАВНЕНИЯ ДАЛАМБЕРА НА ГРАФИЧЕСКОМ ПРОЦЕССОРЕ С ИСПОЛЬЗОВАНИЕМ ЯЗЫКА MATLAB

(Самарский университет, Институт систем обработки изображений РАН)

Введение

Глубокая взаимосвязь оптики и вычислительной техники обусловлена их взаимным влиянием, в ходе которого на рубеже 70-х и 80-х годов прошлого века возникли две самостоятельные отрасли науки: компьютерная оптика, связанная разработкой численных методов расчета и моделирования оптических устройств на ЭВМ и оптическая системотехника, в рамках которой создаются оптические элементы вычислительных устройств. Рост актуальности упомянутых отраслей в настоящее время обусловлен совершенствованием архитектуры ЭВМ (многопоточность, многоядерность, векторизация вычислений) и технологий формирования оптических элементов (переход от микро- к нано размерам). Первая особенность позволила задействовать для расчетов нано размерных элементов оптических процессоров методы строгой теории дифракции [1], характеризующиеся высокой вычислительной сложностью.

Среди численных методов строгой теории дифракции широкой популярностью заслуженно пользуется метод FDTD [1], характеризующийся высокой универсальностью (уравнения Максвелла описывают все волновые электромагнитные процессы) и простотой понимания (основан на замене производных разностными отношениями). Последнее обстоятельство позволяет записывать вычислительные процедуры метода в ясном виде на популярном языке матричных вычислений MATLAB [2].

К сожалению, программная реализация FDTD-метода на современных графических вычислительных устройствах, обеспечивающих ускорение вычислений относительно CPU на порядок, сталкивается, при использовании этого языка, с высокими требованиями к объему видеопамяти: в ходе производства вычислений необходимо использовать в несколько раз больший объем, чем при работе на центральном процессоре. Это обстоятельство отягчается традиционно небольшими размерами видеопамяти (до 2Гб у современных бюджетных видеокарт) по сравнению с оперативной памятью (не ниже 4Гб даже у офисных ЭВМ).

Выходом из создавшегося положения авторы настоящей публикации видят применение метода пирамид, продемонстрировавшего свою эффективность на примере организации расчетов по разностной схеме Yee [1] на GPU с использованием CUDA C [3].

1. Разностное решение одномерного уравнения Даламбера на графическом процессоре

Традиционно под FDTD-методом понимают исключительно разностное решение уравнений Максвелла, что не вполне правильно. В начале 80-ходов прошлого века [1] к нему также стали относить разностное решение уравнения Даламбера, что делают и поныне [1,4]. Отметим, что при решении волнового уравнения проблема нехватки видеопамяти стоит более остро, чем для уравнений Максвелла в силу необходимости конечно-разностной аппроксимации производных второго, а не первого порядка. Однако решение именно волнового уравнения на GPU представляется более перспективным в силу высокой эффективности векторизации вычислительных процедур [5].

Излагая замысел работы, авторы решили остановиться на одномерном уравнении Даламбера, стремясь к демонстрации возможностей метода пирамид на простом примере. Так известная [1] разностная схема для этого уравнения

$$\frac{E_i^{k+1} - 2E_i^k + E_i^{k-1}}{h_t^2} = c^2 \frac{E_{i-1}^k - 2E_i^k + E_{i+1}^k}{h_z^2} \qquad (1)$$

записана относительно сеточной функции, определенной на области $D^h = \{(t_k, z_i) : t_k = kh_t, k = 0,1,..., N_t = T/h_t, z_i = ih_t, i = 1,..., N_z = L_z/h_z + 1\}$, где E – значение напряженности электрического поля, c – скорость света в свободном пространстве, T и L_z – размеры области по времени и пространству.

Ниже приведен фрагмент вычислительной процедуры по решению (1) на языке MATLAB, где $c_1 = c^2 h_t^2 / h_z^2$, $c_5 = 2\pi c h_t / \lambda$.

```
% размещение сеточных функций на двух временных слоях  в видеопамяти
 E1=zeros(1,Nz,'gpuArray'); E2=zeros(1,Nz,'gpuArray');
   for k=1:2:Nt % проход по временным слоям сеточной области через один
     E1(2:Nz-1)=2*E2(2:Nz-1)-E1(2:Nz-1)+c1*diff(E2,2); % вычисления на слое k
     E1(2)=sin(c5*k);       % жесткое излучающее условие на слое к
     E2(2:Nz-1)=2*E1(2:Nz-1)-E2(2:Nz-1)+c1*diff(E1,2); % вычисления на слое k+1
     E2(2)=sin(c5*(k+1));  % жесткое излучающее условие на слое k+1
   End
```

E=gather(E2); % пересылка результата в оперативную память

Для $N_z = 5\times10^7$ и $N_t = 100$ длительность вычислений на CPU Intel Core i7 составила 57,08 с., на графическом процессоре GeForce GTX TITAN X – 5,55 с. (ускорение в 10,29 раз) при использовании языка MATLAB 2015b и операционной системы CentOS 7.2. Оба используемых массива занимали в памяти 762 Мб, однако в ходе вычислений на CPU требования к памяти возрастали в полтора раза, на GPU – трехкратно. По всей видимости, при реализаций вычислений по конструкции E1(2:Nz-1)=2*E2(2:Nz-1)-E1(2:Nz-1)+c1*diff(E2,2) на CPU исполнение операции численного дифференцирования diff(E2,2) приводило к выделению дополнительной памяти под две копии массива E2, а исполнение на GPU конструкции в целом требовало отдельных областей памяти под все участвующее в ней массивы и также двукратного копирования E2. MATLAB занимает около 0,4 гигабайта в оперативной памяти, но не использует под свое размещение видеопамять. Таким образом, исполнение всего алгоритма на CPU сопровождалось выделением 1,52 Гб, на GPU – 2,24 Гб. И если в распоряжении исследователя имеется видеокарта с 2 Гб памяти (как большинство распространенных в настоящее время видеопроцессоров), то организация вычислений на GPU становится невозможной.

2. Применение метода пирамид

В своей предыдущей публикации [7], на примере разностной схемы для уравнений Максвелла, программного инструмента CUDA C, авторы предложили решать указанную проблему с помощью метода пирамид. Удастся ли это в данном случае, учитывая, что язык MATLAB не специализирован для работы с графическими процессорами и его инструментарий в этой области весьма скуден?

Суть упомянутого метода в авторской модификации состоит в разбиении сеточной области на перекрывающиеся подобласти, умещающиеся в видеопамяти целиком, с последующей организацией коммуникаций при производстве векторных вычислений на каждой подобласти отдельно. При этом, пересылки из оперативной памяти в видео и наоборот производятся не на каждом временном слое, а через определенное их количество h (высота пирамиды), что с одной стороны приводит к сокращению в h раз количества коммуникаций и к дублированию арифметических операций в перекрывающихся фрагментах сеточных подобластей (имеющих в двумерном случае форму пирамид), с другой стороны.

Фрагмент вычислительной процедуры, реализующей данную стратегию в рассматриваемом случае представлен далее, где $N = N_z/2$.

```
% создание временных слоев на CPU и GPU
  E1=zeros(1,Nz); E2=zeros(1,Nz); E1m=zeros(1,h); E2m=zeros(1,h);
  E1g=zeros(1,N+h,'gpuArray'); E2=zeros(1,N+h,'gpuArray');
    for t=1:h:Nt % проход по пирамидам
        % работа с левой подобластью
      E1g=gpuArray(E1(1:N+h)); E2g=gpuArray(E2(1:N+h)); % пересылки CPU ==> GPU
        for k=1:2:h    % вычисления внутри первой пирамид
```

```
    E1g(2:N+h-k)=2*E2g(2:N+h-k)-E1g(2:N+h-k)+c1*diff(E2g(1:N+h-k+1),2);
    E1g(2)=sin(c5*(t+k-1));
    E2g(2:N+h-k-1)=2*E1g(2:N+h-k-1)-E2g(2:N+h-k-1)+c1*diff(E1g(1:N+h-k),2);
    E2g(2)=sin(c5*(t+k));
  end
 E1(2:N-h)=gather(E1g(2:N-h)); E2(2:N-h)=gather(E2g(2:N-h)); % пересылки GPU ==> CPU
 E1m(1:h)=gather(E1g(N-h+1:N)); E2m(1:h)=gather(E2g(N-h+1:N));
    % работа с правой подобластью
 E1g(1:N+h-1)=gpuArray(E1(N-h+1:Nz)); E2g(1:N+h-1)=gpuArray(E2(N-h+1:Nz));
  for t=1:2:h % проход по пирмидам
    E1g(t+1:N+h-2)=2*E2g(t+1:N+h-2)-E1g(t+1:N+h-2)+c1*diff(E2g(t:N+h-1),2);
    E2g(t+2:N+h-2)=2*E1g(t+2:N+h-2)-E2g(t+2:N+h-2)+c1*diff(E1g(t+1:N+h-1),2);
  end
 E1(N+1:Nz-1)=gather(E1g(h+1:N+h-2)); % пересылки GPU ==> CPU
 E2(N+1:Nz-1)=gather(E2g(h+1:N+h-2)); %
 E1(N-h+1:N)=E1m(1:h); E2(N-h+1:N)=E2m(1:h); % восполнение результата
end
```

В ходе экспериментов с новым алгоритмом была исследована зависимость длительности вычислений от высоты пирамиды. Результаты представлены в таблице 1.

Таблица 1. Зависимость длительности вычислений от высоты пирамиды

высота пирамиды, *h*	длительность вычислений с.	ускорение
2	53,02	1,08
4	29,58	1,93
10	15,49	3,7
20	10,75	5,31
50	7,9	7,23

Заключение

Таким образом, метод пирамид может эффективно применяться при организации вычислений по решению разностных уравнений с помощью языка MATLAB на графических процессорах в случае, когда массивы, хранящие значения сеточных функций, не умещаются в видеопамять целиком. Развитие предложенного алгоритма на случаи больших размерностей станет следующим этапом исследований авторов в данном направлении.

Благодарности

Работа выполнена при поддержке гранта РФФИ № 16-47-630560-р_а и частичной поддержке Министерства образования и науки РФ.

Литература

1. Taflove, A. Computational Electrodynamics: The Finite-Difference Time-Domain Method: 2nd. ed. / A. Taflove, S. Hagness // Boston: Arthech House Publishers, 2000. – 852 p.

2. Elsherbeni, A The Finite-Difference Time-Domain Method for Electromagnetics with MATLAB Simulations / A. Elsherbeni and V. Demir // Scitech Publishing Inc, 2009. – 426 p.

3. Малышева, С.А. Реализация разностного решения уравнений Максвелла на графических процессорах методом пирамид/ С.А. Малышева, Д.Л. Головашкин // Компьютерная оптика. – 2016. – Т. 40, № 2. – С. 179-187.

4. Козлова, Е.С. Моделирование распространения короткого двумерного импульса света / Е.С. Козлова, В.В. Котляр // Компьютерная оптика. – 2012. – Т. 36, № 2. – С. 158-164.

5. Воротникова, Д.Г. Разностное решение волнового уравнения на графических процессорах с повторным использованием попарных сумм дифференциального шаблона / Д.Г. Воротникова, Д.Л. Головашкин // Компьютерная оптика. – 2017. – Т. 41, № 1. (в печати)

6. 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.