ЧИСЛЕННЫЕ МЕТОДЫ И АНАЛИЗ ДАННЫХ

Особенности построения блочных алгоритмов FDTD-метода при организации вычислений на графическом процессоре с использованием языка MATLAB

Н.Д. Морунов[1], Д.Л. Головашкин [1,2]
[1] Самарский национальный исследовательский университет имени академика С.П. Королёва, 443086, Россия, Самарская область, Самара, Московское шоссе, д. 34;
[2] ИСОИ РАН – филиал ФНИЦ «Кристаллография и фотоника» РАН, 443001, Россия, Самарская область, Самара, ул. Молодогвардейская, д. 151

Аннотация

Работа посвящена исследованию особенностей реализации блочных алгоритмов FDTD-метода на графических процессорах. Обсуждается понятие блочного алгоритма в целом и блочного алгоритма FDTD-метода в частности. Основное внимание уделяется специфике подбора объёма и формы блоков, связанной с использованием языка MATLAB и его дополнения Parallel Computing Toolbox. Демонстрируется практическая эффективность предложенных авторских приемов, раскрываются перспективы их применения.

Ключевые слова: FDTD-метод, блочные алгоритмы, ускорение вычислений.

Цитирование: **Морунов, Н.Д.** Особенности построения блочных алгоритмов FDTD-метода при организации вычислений на графическом процессоре с использованием языка MATLAB / Н.Д. Морунов, Д.Л. Головашкин // Компьютерная оптика. – 2019. – Т. 43, № 4. – С. 671-676. – DOI: 10.18287/2412-6179-2019-43-4-671-676.

Введение

Растущая, несмотря на почтенный возраст [1], популярность применения FDTD-метода для решения широкого круга задач, связанных с распространением электромагнитного излучения (например [2]), обуславливает живой интерес к реализации данного метода на современных процессорных архитектурах, в том числе на графических ускорителях (GPU).

Потребность в такой реализации оказалась настолько велика, что к ней приступили [3] ещё до появления средств программирования высокого уровня на GPU; среди прочих и такой признанный авторитет в вычислительной электродинамике, как Allen Taflove, посвятивший главу фундаментальной монографии [4] реализации FDTD-метода на GPU с использованием интерфейса компьютерной графики OpenGL. С появлением программно-аппаратной архитектуры CUDA реализацией FDTD-метода на GPU и созданием соответствующих программных продуктов занялись как коммерческие компании (например, пакет XFdtd с модулем XStream GPU Acceleration компании REMCOM [5]), так и независимые группы исследователей (пакет B-CALM Калифорнийского технологического института [6]).

Вместе с тем использование готовых программных продуктов в инициативных исследованиях небольших научных групп не всегда возможно. В силу немалой стоимости (для коммерческих пакетов), прекращения поддержки (для свободно распространяемых) либо отсутствия необходимых возможностей (зачастую исследования в новых областях связаны с новыми требованиями). При этом специалисту в области дифракционной оптики, к примеру, зачастую не с руки осваивать тонкости программирования на языке CUDA Си, а хотелось бы использовать после необходимой модификации готовые реализации FDTD-метода с открытым кодом на MATLAB'е, известном широкому кругу специалистов-физиков.

Таким реализациям отводится место как в монографиях, освещающих FDTD-метод в целом (к [4] прилагается диск с программами Susan C. Hagness), так и в специализированных работах, посвящённых исключительно представлению FDTD на языке MATLAB (например, [7]). Отличительной особенностью последнего упомянутого труда является присутствие главы, в которой описана организация вычислений по FDTD-методу на GPU, но не с помощью MATLAB'а, в котором возможность работы с GPU к тому времени была уже реализована, а посредством языка Brook, разработанного в Стэндфордском университете ещё до появления CUDA Си. Видимо, стремясь исправить данное упущение, нивелирующее преимущества использования MATLAB'а, один из авторов [7], A.Z. Elsherbeni, недавно опубликовал работу [9], в которой FDTD-метод реализовывался на GPU с помощью Parallel Computing Toolbox – дополнения MATLAB'а.

Значимым общим недостатком реализации из [9] и упомянутых ранее пакетов [5, 6] является проблема, связанная с небольшой ёмкостью видеопамяти по сравнению с памятью оперативной, приобретающая особую остроту при использовании MATLAB'а в силу известной «жадности» последнего к любой памяти, а к видеопамяти особенно. К примеру, желая ускорить вычисления на сеточной области (при том, что все с ней связанные массивы вполне помещаются в ОЗУ компьютера) переходом к расчётам на GPU, исследователь зачастую сталкивается с переполнением видеопамяти и невозможностью производства расчётов.

Предложенное в работе [10] решение данной проблемы посредством перехода к блочным алгоритмам (там они называются DiamondTorre), по мнению авторов настоящей статьи, открывает возможности для разработки новых подходов к реализации FDTD-метода на GPU. При этом указанные возможности в тексте [10] далеко не исчерпываются. Так, не разработан блочный алгоритм FDTD-метода для языка MATLAB (в [10] используется CUDA Си), очевидно имеющий свои особенности при выборе оптимального объёма блока и его формы. Восполнению данных пробелов и посвящена предлагаемая работа.

1. Идея блочной реализации FDTD-метода

Не претендуя на полноту освещения (настоящая статья не носит обзорного характера), авторы считают полезным познакомить читателя с ключевыми особенностями блочной реализации FDTD-метода, необходимыми для погружения в обсуждаемую предметную область.

С наиболее общей точки зрения смысл блочности состоит в управлении (не всегда прямом) коммуникациями между различными областями иерархически структурированной памяти ЭВМ при реализации численного метода. Количество арифметических операций в блочном алгоритме метода не изменяется, а коммуникационные издержки минимизируются. Для такой минимизации необходима интенсификация использования загруженных на верхние иерархические уровни памяти данных через совершение над ними наибольшего из возможных, в рамках рассматриваемого численного метода, количества арифметических операций до выгрузки [11]. Поэтому некоторые авторы [12] связывают блочность с представлением численного метода через арифметические операции над массивами высокой размерности при возможности варьировать параметры этих массивов. По сути оба взгляда являются разными точками зрения на один предмет.

Синтез блочных алгоритмов рассматриваемого здесь FDTD-метода представляется весьма перспективным издавна в силу подавляющего преобладания коммуникационных издержек над длительностью арифметических операций при классической [4] реализации данного метода на ЭВМ. Действительно, основная арифметическая операция метода конечных разностей, вычитание двумерных или трёхмерных массивов (в зависимости от размерности задачи), производится на современных процессорах с векторизацией вычислений значительно быстрее, чем коммуникации между различными областями иерархической памяти фрагментов самих массивов: трёх массивов в двумерном и шести в трёхмерном случаях. Однако работы в этом направлении до констатации «кремниевого тупика» не велись, очевидно, в силу упования на дальнейший рост скоростных характеристик вычислительной техники. Избегая детального исторического обзора, отметим две важные современные публикации: [13] с блочными алгоритмами FDTD-метода для CPU и уже упоминавшуюся [10] с алгоритмами для GPU. При определённой схожести, обусловленной общностью цели – ускорение вычислений, алгоритмы отличаются как по решаемым задачам (минимизация коммуникаций между разными парами объектов: оперативная память и видеопамять, оперативная память и кэш-процессора), так и по средствам реализации (в первом случае коммуникациями можно управлять напрямую даже на языках высокого уровня).

Тем не менее, ключевым в обоих случаях является понятие блока – фрагмента сеточной области, для которого вычисление значений определённых внутри него функций не сопровождается производством упомянутых коммуникаций. Форма блока зачастую определяет название алгоритма, а его геометрические параметры варьируются при экспериментальном исследовании ускорения. Объём блока связывается с ёмкостью выбранного верхнего иерархического уровня памяти ЭВМ и не должен превышать его либо быть значительно меньше. В первом случае необходимость в коммуникациях возникнет при вычислениях внутри одного блока (тогда это уже и не блок вовсе), во втором уменьшение объёма блока вызовет рост их количества и, как следствие, также увеличение коммуникационных издержек. Исследование особенностей построения блочных алгоритмов FDTD-метода при организации вычислений на графическом процессоре с использованием языка MATLAB предлагается читателю далее.

2. Определение оптимального объёма блока

При организации блочных вычислений исследователю необходимо задать форуму и объём блока. Если выбор формы сопровождает разработку собственно блочного алгоритма, являясь важным её этапом, то для демонстрации особенностей определения объёма авторы настоящей работы считают возможным обратиться к исследованию известного не блочного. В самом деле, если в этом простом случае очевидная методика, основанная на сложении объёма всех обрабатываемых данных и сопоставлении его с ёмкостью видеопамяти, окажется несостоятельной, то и в более сложном случае исследования блочного алгоритма ей также не следует доверять.

Выбор не блочного алгоритма FDTD-метода, реализующего его на видеопроцессоре с использованием языка MATLAB, ограничен единственной [9], но достаточно подробно описанной реализацией, базирующейся, в свою очередь, на алгоритмах для центральных процессоров, представленных в [7] с исчерпывающей детализацией. В силу чего здесь остановимся лишь на указании размерности задачи (двумерный случай распространения TE-волны, вычислительная область – квадрат), типа и ширины поглощающих PML слоев (CPML в 8 узлов), дискретизации сеточной области по времени и пространству ($N_t = 100$, $316 \le N \le 3\,873$). Размерность обусловлена ориентацией на персональные ЭВМ, для которых мо-

делирование дифракции на обширных трёхмерных объектах с использованием MATLAB заведомо избыточно; дискретизация обеспечивает решение поставленной задачи за приемлемое время (ускорение вычислений теоретически не зависит от N_t), а поглощающие слои типа CPML считаются [10] наиболее современными из хорошо зарекомендовавших себя.

В качестве аппаратного обеспечения использовался центральный процессор Intel Core i5-6300HQ 2,3 ГГц; материнская плата ASUS GL552VW; ОЗУ DIMM DDR4-2133 МГц объёмом 8 Гб; видеокарта NVIDIA GeForce GTX 960 M, подключенная через шину PCI Express ×16, с внутренней оперативной памятью GDDR5 объёмом 2 Гб и графическим процессором GM107 с тактовой частотой 1,1 ГГц (640 потоковых CUDA процессоров, 5 блоков SMM). Аппаратная база работала под управлением операционной системы Windows 10 64bit; использовался MATLAB R2017b с Parallel Computing Toolbox 6.11 (CUDA v9.2).

На рис. 1 представлены результаты вычислительных экспериментов по оценке оптимального объёма сеточной области по пространству, под ускорением понималось отношение длительности вычислений на центральном процессоре ко времени расчётов на графическом.

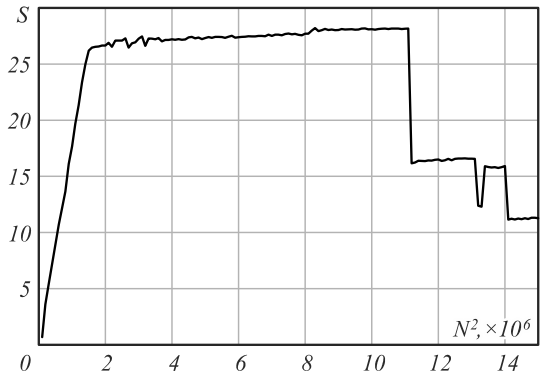

Рис. 1. Зависимость ускорения вычислений (S) от дискретизации сеточной области

Выделим на обсуждаемом графике три фрагмента, соответствующие росту ($N^2<2e6$), стабилизации ($2e6\le N^2\le 11{,}1e6$) и падению ($N^2>11{,}1e6$) ускорения. Первый согласуется с постепенной загрузкой вычислительных ресурсов графического процессора при расчётах на увеличивающихся сеточных областях; второй – полной загрузкой такого процессора; третий – снижением загрузки ещё до исчерпания свободного объёма видеопамяти, за которым ($N^2=15e6$) вычисления по не блочному алгоритму невозможны.

Поведение ускорения на заключительном фрагменте обусловлено особенностями организации вычислений на языке MATLAB; в частности, увеличением объёма занимаемой памяти через дублирование части данных с целью снижения длительности вычислений. Не вдаваясь здесь в тонкости данного приёма (по-видимому, связанные с организацией параллельных вычислений и дублированием данных для соблюдения информационных зависимостей [15]), механизм которого скрыт разработчиками MATLAB'а, свяжем наблюдаемый эффект падения ускорения на рис. 1 с затруднительностью упомянутого дублирования, начиная с $N^2=11{,}1e6$. Очевидно, вычислительный процесс при $11{,}1e6\le N^2\le 15e6$ организуется иначе в силу нарастающей нехватки видеопамяти, менее эффективным (в смысле сокращения длительности вычислений), чем при $2e6\le N^2\le 11{,}1e6$ образом. Следовательно, значение $N^2=11{,}1e6$ (а не 15e6, как теоретически ожидалось) и будет задавать наилучший в смысле ускорения вычислений объём сеточной области по пространству.

Аналогичный вывод о неудачности использования всего объёма видеопамяти, как оценки оптимального объёма блока, следует сделать и для блочных алгоритмов. Вместе с тем необходимость в отыскании оценки очевидна: при объёме меньшем оптимального потребуется больше блоков (следовательно, количество коммуникаций блочного алгоритма возрастёт), при большем (пусть даже и помещающемся в видеопамять целиком) загрузка вычислительных ресурсов графического процессора драматически снизится.

Авторы предлагают воздержаться от построения теоретических оценок, аналогичных из [10, 16] (по крайней мере до появления удовлетворительной модели вычислительного процесса для расчётов такого типа) и обращаться в каждом конкретном случае к постановке предваряющей серии экспериментов для нахождения обсуждаемой величины, как, например, это предлагается в [14] при организации блочных вычислений на центральном процессоре.

3. Выбор формы блока

Следуя [10, 16], где приведены соответствующие обоснования, первоначально выберем в качестве формы блока наклоненную башню (*DiamondTorre*). На рис. 2 представлено применение этой стратегии в рассматриваемом случае решения двумерной по пространству задачи с одномерной (пока для простоты) декомпозицией сеточной области по блокам.

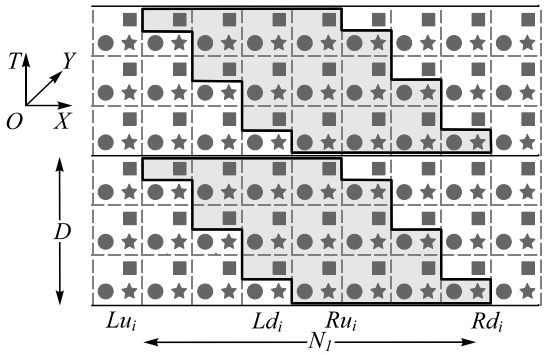

Рис. 2. Одномерная декомпозиция сеточной области на блоки без окаймления. Звездочки соответствуют области определения сеточного аналога Hz (проекции магнитного поля на ось Z), круги – области определения Ey, квадраты – Ex

Далее будем различать ширину (число узлов сеточной области по направлению OX, N_1 на рис. 2), высоту (по направлению OT, D на рис. 2) и глубину (по OY, N в силу одномерности декомпозиции) блока.

Ступенчатые границы выделенного i-го блока определены на рисунке значениями Lu_i, Ld_i – координатами узлов левой границы блока по оси абсцисс, Ru_i, Rd_i – координатами правой границы по той же оси и высотой блока D. Тогда $D = Rd_i - Ru_i + 1 = Ld_i - Lu_i + 1$, $N_1 = Rd_i - Lu_i + 1$. Переходя к следующему $i+1$-му блоку, примем $Lu_{i+1} = Ru_i$, $Ld_{i+1} = Rd_i$. В общем случае ширина отдельных блоков в одном блочном слое по времени может различаться, высота же внутри одного слоя всегда одинакова и может изменяться лишь при переходе на следующий слой. В настоящей работе авторы полагали обе величины постоянными в рамках одного эксперимента.

Теперь, после детального описания блока, поясним, что под его объёмом понимается не геометрический объём соответствующей ему фигуры на рис. 2, а объём видеопамяти (в узлах сеточной области), который блок занимает, будучи там размещенным. Геометрически такой объём соответствует площади (в узлах сеточной области) проекции блока на плоскость $OXY - N_1 N$. Предостережем также читателя от соблазна перехода к расчёту в мегабайтах, напомнив о сокрытии разработчиками MATLAB'а механизмов дублирования данных с целью ускорения вычислений.

Основной фрагмент алгоритма – вычисления внутри выделенного на рис. 2 блока, состоящие в переборе временных слоёв сеточной области Yee вдоль оси ординат. При переходе к новому слою последовательно, с замещением прежних значений новыми, вычисляются сеточные напряжённости компонент электромагнитного поля. Перебор блоков производится слева направо внутри одного блочного слоя высотой D, по его исчерпанию начинается работа с новым слоем, расположенным над данным. Переход к новому блоку сопровождается коммуникациями между оперативной и видеопамятью, связанными с выгрузкой из второй в первую значений сеточных функций, определённых на верхнем слое блока, вычисления в котором завершились, и загрузкой в обратном направлении нижнего слоя следующего блока. Наклонные границы блоков обеспечивают соблюдение информационных зависимостей при производстве расчётов. Отметим также, что при работе с крайними блоками одного блочного слоя площадь их горизонтального сечения варьируется, в отличие от случая центральных блоков, в силу прямоугольности сеточной области и наклонения границ блоков, что обуславливает определённые особенности организации вычислений в этих блоках.

Все вычисления производятся над матрицами (случай двумерный по пространству) – фрагментами массивов, хранящих значения сеточных функций, что теоретически должно обусловить высокое ускорение в силу использования операций второго уровня [11], наивысшего для данной задачи. Ожидается, что производство коммуникаций несколько снизит ускорение. Впрочем, совокупная длительность арифметических операций растёт с увеличением дискретизации сеточной области значительно быстрее, чем общее время коммуникаций, следовательно, такое снижение будет незначительным.

Однако на практике (рис. 3, непрерывная линия) переход к организации вычислений по представленному блочному алгоритму приводит к пятикратному падению ускорения по сравнению с не блочным случаем (рис. 2).

При этом применение блочности (разбиение на два блока) все же позволило избавиться от ограничения на размер видеопамяти – расчёты для дискретизации $N^2 > 15e6$ по не блочному алгоритму на графическом процессоре невозможны.

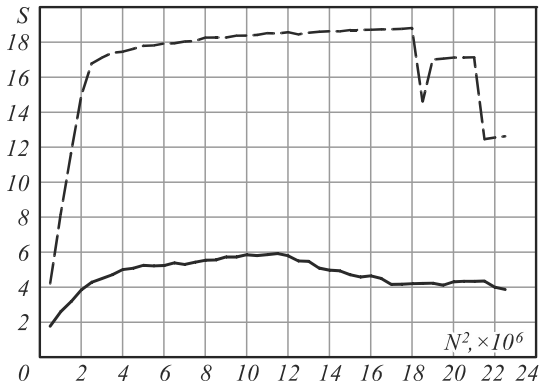

Рис. 3. Сравнение ускорений по блочным алгоритмам без (непрерывная линия) и с окаймлением блоков (пунктирная)

По мнению авторов, причиной резкого падения ускорения является увеличение использования в тексте программы обращений к фрагментам массивов с помощью выражений, включающих индексы. Действительно, при расчётах внутри одного блока индексы массивов меняются при переходе к каждому следующему временному слою. В не блочной версии FDTD-метода такого не происходит.

Решение данной проблемы авторы видят в отказе от такой работы с индексами и переходе к новой форме блока (рис. 4).

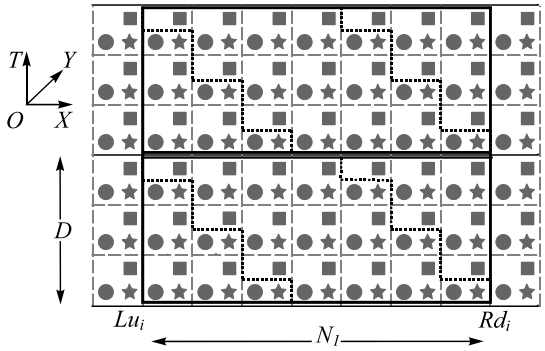

Рис. 4. Одномерная декомпозиция сеточной области на блоки с окаймлением. Новые вертикальные границы i-го блока отмечены линией увеличенной ширины

Арифметические операции по схеме Yee, как и ранее, производятся внутри подобласти, ограниченной ступенчатыми линиями, но в выражениях указываются горизонтальные сечения блока, выделенного на рис. 4, границы которых в пределах одного блока совпадают. На каждом таком сечении нового блока дополнительно предписывается сравнение координат узлов с заранее подготовленной маской, выделяющей

ступенчатую подобласть из прямоугольной. Это позволяет оперировать постоянными индексами при работе с массивами, отказавшись от переменных. От прежней формы блока данная отличается ступенчатыми окаймлениями, дополняющими его до параллелепипеда.

В итоге ускорение нового блочного алгоритма (рис. 3, пунктирная линия) троекратно увеличивается по сравнению с прежним, отличаясь от не блочного падением в полтора раза, что уже объяснимо дополнительными коммуникационным издержками, неизбежными при организации блочных вычислений на видеопроцессоре. Абсолютное значение достигнутого ускорения, 18,5 раз по сравнению с центральным процессором, весьма значительно. Кроме того, сравнение графиков на рис. 3 позволяет говорить о разном оптимальном объёме блоков в случае отличающихся блочных алгоритмов, что очевидно связано с хранением и использованием дополнительной маски.

Заключение

Использование языка MATLAB для составления блочных алгоритмов FDTD-метода и организации вычислений по ним на графических процессорах уместно, что позволяет говорить о преодолении ограничения на объём видеопамяти, сдерживающего широкое применение реализаций FDTD-метода на языке MATLAB из [7, 9]. В свою очередь, использование распространенного языка MATLAB, вместо программирования на достаточно сложном CUDA Си, открывает доступ к богатому инструментарию FDTD-метода специалистам в вычислительных оптике и электродинамике.

Благодарности

Работа выполнена при поддержке гранта РФФИ 19-07-00423 А и Министерства науки и высшего образования РФ в рамках выполнения работ по Государственному заданию ФНИЦ «Кристаллография и фотоника» РАН (соглашение 007-ГЗ/Ч3363/26).

Литература

1. **Yee, K.S.** Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media / K.S. Yee // IEEE Transactions on Antennas and Propagation. – 1966. – Vol. AP-14. – P. 302-307. – DOI: 10.1109/TAP.1966.1138693.
2. **Котляр, В.В.** Острая фокусировка лазерного света с помощью микрооптики / В.В. Котляр, С.С. Стафеев, А.Г. Налимов. – Самара: Новая техника, 2018. – 344 с. – ISBN: 978-5-88940-148-3.
3. **Krakiwsky, S.E.** Graphics processor unit (GPU) acceleration of finite-difference time-domain (FDTD) algorithm / S.E. Krakiwsky, L.E. Turner, M.M Okoniewski // Proceedings of the 2004 International Symposium on Circuits and Systems (ISCAS '04). – 2004. – P. 1033-1036. – DOI: 10.1109/ISCAS.2004.1329513.
4. **Taflove, A.** Computational electrodynamics: The finite-difference time-domain method / A. Taflove, S. Hagness. – 3th ed. – Boston: Arthech House Publishers, 2005. – 1006 p. – ISBN: 978-1-58053-832-9.
5. XStream® GPU Acceleration [Electronical Resource].– URL: https://www.remcom.com/xf-xstream (request date 08.04.2019).
6. **Wahl, P.** B-CALM: An open-source GPU-based 3D-FDTD with multi-pole dispersion for plasmonics / P. Wahl, D.-S. LyGagnon, Ch. Debaes, D.A.B. Miller, H. Thienpont // Optical and Quantum Electronics. – 2012. – Vol. 44, Issue 3. – P. 285-290. – DOI: 10.1007/s11082-012-9558-z.
7. **Elsherbeni, A.Z.** The finite-difference time-domain method for electromagnetics with MATLAB simulations / A.Z. Elsherbeni, V. Demir. – Ralrigh, NC: SciTech Publishing, Inc., 2009. – 426 p. – ISBN: 978-1-891121-71-5.
8. BrookGPU [Electronical Resource].– URL: http://graphics.stanford.edu/projects/brookgpu (request date 08.04.2019).
9. **Diner, J.E.** FDTD Acceleration using MATLAB parallel computing toolbox and GPU/ J.E. Dinier, A.Z. Elsherbeni // Applied Computational Electromagnetics Society Journal. – 2017. – Vol. 32, Issue 4. – P. 283-288.
10. **Закиров, А.В.** Алгоритм DiamondTorre и высокопроизводительная реализация FDTD метода для суперкомпьютеров с графическими ускорителями / А.В. Закиров, В.Д. Левченко, А.Ю. Перепёлкина, Я. Земпо // Труды международной конференции Суперкомпьютерные дни в России. – 2016. – С. 80-94.
11. **Golub, G.H.** Matrix computations / G.H. Golub, Ch.F. Van Loan. – 3rd ed. – Baltimore, London: Johns Hopkins University Press, 1996. – 694 p. – ISBN: 978-0-8018-5414-9.
12. **Деммель, Дж.** Вычислительная линейная алгебра. Теория и приложения / Дж. Деммель. – М.: Мир, 2001. – 435 с. – ISBN: 5-03-003402-1.
13. **Orozco, D.** Mapping the FDTD application to many-core chip architectures / D. Orozco, G. Guang // International Conference on Parallel Processing (ICPP '09). – 2009. – P. 309-316. – DOI: 10.1109/ICPP.2009.44.
14. **Minami, T.** Automatic parameter tuning of three-dimensional tiled FDTD kernel / T. Minami, M. Hibino, T. Hiraishi, T. Iwashita, H. Nakashima. – In: High performance computing for computational science. VECPAR 2014 / ed. by M. Daydé, O. Marques, K. Nakajima. – 2014. – P. 284-297. – DOI: 10.1007/978-3-319-17353-5_24.
15. **Вальковский, В.А.** Параллельное выполнение циклов. Метод пирамид / В.А. Вальковский// Кибернетика. – 1983. – № 5. – С. 51-55.

Сведения об авторах

Морунов Никита Дмитриевич, аспирант кафедры прикладных математики и физики Самарского национального исследовательского университета имени академика С. П. Королева. Область научных интересов: разностные схемы, FDTD-метод, векторные и матричные вычисления. E-mail: *niknixad@mail.ru* .

Головашкин Димитрий Львович, доктор физико-математических наук, ведущий научный сотрудник лаборатории дифракционной оптики Института систем обработки изображений РАН – филиала Федерального государственного учреждения Федеральный научно-исследовательский центр «Кристаллография и фотоника» Российской академии наук, профессор кафедры прикладных математики и физики Самарского национального исследовательского университета имени академика С. П. Королева. Область научных интересов: FDTD-метод, векторные и матричные вычисления, математическое моделирование оптических элементов и устройств нанофотоники. E-mail: *dimitriy@smr.ru* .

ГРНТИ: 27.41.19.
Поступила в редакцию 11 апреля 2019 г. Окончательный вариант – 3 июня 2019 г.

Design features of block algorithms of FDTD-method implemented on a GPU using MATLAB

N.D. Morunov [1], D.L. Golovashkin [1, 2]
[1] Samara National Research University, Moskovskoye Shosse 34, 443086, Samara, Russia;
[2] IPSI RAS – Branch of the FSRC "Crystallography and Photonics" RAS, Molodogvardeyskaya 151, 443001, Samara, Russia

Abstract

The paper is devoted to the investigation of the implementation features of a block algorithm for the FDTD-method on GPU. The block algorithm in general and in the context of the FDTD-method in particular is discussed. The main attention is paid to specifics of determining the shape and volume of blocks, which stem from the use of MATLAB and its Parallel Computing Toolbox. The practical efficiency of the proposed techniques is shown. Possible applications and prospects are discussed.

Keywords: FDTD-method, block algorithms, computational speed-up.

Citation: Morunov ND, Golovashkin DL. Design features of block algorithms of FDTD-method implemented on a GPU using MATLAB. Computer Optics 2019; 43(4): 671-676. DOI: 10.18287/2412-6179-2019-43-4-671-676.

Acknowledgements: The work was funded by the Russian Foundation for Basic Research under grant 19-07-00423 A the RF Ministry of Science and Higher Education under the government project of the FSRC "Crystallography and Photonics", RAS No. 007-GZ/C3363/26.

References

[1] Yee KS. Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media. IEEE Transactions on Antennas and Propagation 1966; AP-14: 302-307. DOI: 10.1109/TAP.1966.1138693.

[2] Kotlyar VV, Stafeev SS, Nalimov AG. Sharp focusing of laser light with micro-optics [In Russian]. Samara: "Novaya Tehnika" Publisher; 2018. ISBN: 978-5-88940-148-3.

[3] Krakiwsky SE, Turner LE, Okoniewski MM. Graphics processor unit (GPU) acceleration of finite-difference time-domain (FDTD) algorithm. Proc ISCAS'04 2004: 1033-1036. DOI: 10.1109/ISCAS.2004.1329513.

[4] Taflove A, Hagness S. Computational electrodynamics: The finite-difference time-domain method. 3th ed. Boston: Arthech House Publishers, 2005. ISBN: 978-1-58053-832-9.

[5] XStream® GPU Acceleration. Source: ⟨https://www.remcom.com/xf-xstream⟩.

[6] Wahl P, LyGagnon DS, Debaes Ch, Miller DAB, Thienpont H. B-CALM: An open-source GPU-based 3D-FDTD with multi-pole dispersion for plasmonics. Opt Quant Electron 2012; 44(3): 285-290. DOI: 10.1007/s11082-012-9558-z.

[7] Elsherbeni AZ, Demir V. The finite-difference time-domain method for electromagnetics with MATLAB simulations. Ralrigh, NC: SciTech Publishing Inc; 2009. ISBN: 978-1-891121-71-5.

[8] BrookGPU. Source: ⟨http://graphics.stanford.edu/projects/brookgpu⟩.

[9] Dinier JE, Elsherbeni AZ. FDTD Acceleration using MATLAB parallel computing toolbox and GPU. Applied Computational Electromagnetics Society Journal 2017; 32(4): 283-288.

[10] Zakirov AV, Levchenko VD, Perepelkina AYu, Zempo Y. DiamondTorre algorithm and high-performance implementation of the FDTD-method for supercomputers with graphics accelerators [In Russian]. Proc "Supercomputer days in Russia '16" 2016: 80-94.

[11] Golub GH, Van Loan ChF. Matrix Computations. 3rd ed. Baltomore, London: Johns Hopkins University Press; 1996. ISBN: 978-0-8018-5414-9.

[12] Demmel J. Applied numerical linear algebra. Philadelphia: SIAM; 1997. ISBN: 978-0-89871-389-3.

[13] Orozco D, Guang G. Mapping the FDTD application to many-core chip architectures. Proc ICPP '09 2009: 309-316. DOI: 10.1109/ICPP.2009.44.

[14] Minami T, Hibino M, Hiraishi T, Iwashita T, Nakashima H. Automatic parameter tuning of three-dimensional tiled FDTD kernel. In Book: Daydé M, Marques O, Nakajima K, eds. High performance computing for computational science (VECPAR 2014) 2014: 284-297. DOI: 10.1007/978-3-319-17353-5_24.

[15] Valkovskii V. Parallel execution cycles. Method of the pyramids [In Russian]. Cybernetics 1983; 5: 51-55.

Authors' information

Nikita Dmitrievich Morunov study as graduate student of the Applied Mathematics and Physics sub-department at Samara University. Research interests: finite-difference approximations, FDTD-method, vector and matrix computations. E-mail*:* *niknixad@mail.ru*.

Dimitriy Lvovich Golovashkin Doctor of Physics and Mathematics, leader researcher Diffractive Optics laboratory at the Image Processing Systems Institute of RAS – Branch of the FSRC "Crystallography and Photonics" RAS. Works as professor of the Applied Mathematics and Physics sub-department at Samara University. Research interests are FDTD-method, vectors and matrix computation, mathematical modeling of optical components and nanophotonic devices. E-mail: *dimitriy@smr.ru*.

Received April 11, 2019. The final version – June 03, 2019.