РЕАЛИЗАЦИЯ РАЗНОСТНОГО РЕШЕНИЯ УРАВНЕНИЙ МАКСВЕЛЛА НА ГРАФИЧЕСКИХ ПРОЦЕССОРАХ МЕТОДОМ ПИРАМИД

С.А. Малышева [1,2], Д.Л. Головашкин [1,2]

[1] Институт систем обработки изображений РАН, Самара, Россия,

[2] Самарский государственный аэрокосмический университет имени академика С.П. Королёва (национальный исследовательский университет) (СГАУ), Самара, Россия

Аннотация

Работа посвящена развитию метода пирамид в приложении к решению системы уравнений Максвелла во временной области посредством конечных разностей (FDTD) и его реализации на графическом процессоре. Применение этого метода позволяет снизить влияние ограниченного объема памяти графического вычислительного устройства на длительность расчётов, существенное для FDTD.

Ключевые слова: FDTD, уравнения Максвелла, метод пирамид, GPU, CUDA.

Цитирование: **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. – DOI: 10.18287/2412-6179-2016-40-2-179-187.

Введение

Метод разностного решения уравнений Максвелла во временной области (FDTD) [1] широко используется в современных исследованиях при моделировании: распространения излучения [2], материалов с новыми свойствами [3], взаимодействия излучения с биологическими объектами [4] и во многих других приложениях.

Популярность метода обусловлена методической наглядностью (основан на замене производных разностными отношениями), обширностью сферы применения (строгая теория дифракции) и обилием готовых программных реализаций (множество коммерческих и свободно распространяемых пакетов для различных архитектур и операционных систем). Платой за универсальность являются высокие системные требования к быстродействию процессора и объему оперативной памяти, в силу которых разработанный полвека назад метод начал активно применяться в вычислительной практике только сейчас.

Сталкиваясь с известными трудностями («кремниевый тупик»), развитие аппаратной базы вычислительной техники в последнее десятилетие связывается не с наращиванием тактовой частоты процессоров, а со специализированными архитектурными решениями, позволяющими значительно увеличить быстродействие операций определенного типа. Наиболее заметные успехи в этом направлении достигнуты при развитии архитектуры графических процессоров (GPU) и соответствующих программных инструментов (CUDA [5], OpenCL [6]), позволяющих многократно ускорить действия с векторами и матрицами.

Современные реализации FDTD-метода на GPU [7, 8], в основе которых лежит операция вычитания матриц, характеризуются повышением производительности на порядок по сравнению с вычислениями на центральном процессоре. Однако ограниченный объем видеопамяти (2–6 Гб. по сравнению с 4–32 Гб. оперативной памяти) не позволяет в полной мере использовать преимущество GPU в быстродействии.

Снижение влияния упомянутого фактора представляется авторам весьма актуальной задачей. Для ее решения в настоящей работе предлагается развить метод пирамид [9, 10, 11], основанный на уменьшении интенсивности коммуникаций между оперативной и видеопамятью за счёт дублирования арифметических операций (в частности, FDTD-метода).

1. Разностная схема Yee

Иллюстрируя применение метода пирамид к разностному решению уравнений Максвелла, авторы остановились на классической схеме Yee для трехмерного случая [12], являющейся в настоящее время наиболее популярным инструментом вычислительной электродинамики [1] и наноплазмоники [13]. Основная особенность этой схемы состоит в раздельном расположении узлов сеточной области для каждой проекции напряженностей электрического и магнитного полей, обеспечивающем повышенный порядок аппроксимации исходной дифференциальной задачи по времени и пространству.

Табл. 1. Коэффициенты сеточных проекций электрического и магнитного полей

A	d	M_t	a	I_x	b	J_y	c	K_z
E_x			0,5	1				
E_y					0,5	1		
E_z							0,5	1
H_x	0,5	1			0,5	1	0,5	1
H_y	0,5	1	0,5	1			0,5	1
H_z	0,5	1	0,5	1	0,5	1		

Рассмотрим трехмерную по пространству область вычислительного эксперимента D^3 ($0<t\le T$, $0\le x\le L_x$, $0\le y\le L_y$, $0\le z\le L_z$) с наложенной на нее сеточной структурой D_h^3, в узлах которой $\{(t_{m+d}, x_{i+a}, y_{j+b}, x_{k+c})$: $t_{m+d}=(m+d)h_t$, $m=0, 1,\ldots,. M–M_t$, $(M=T/h_t)$, $x_{i+a}=(i+a)h_x$, $i=0, 1,\ldots, I–I_x$, $(I=L_x/h_x)$, $y_{j+b}=(j+b)h_y$, $j=0, 1,\ldots,J–Jy$, $(J=Ly/hy)$, $z_{k+c}=(k+c)h_z$, $k=0, 1,\ldots,K–K_z$, $K=L_z/h_z\}$ определены сеточные проекции поля $A^{m+d}_{i+a,\,j+b,k+c}$.

Значения коэффициентов a, b, c, d для различных сеточных проекций электрического и магнитного полей

представлены в табл. 1, пустые ячейки соответствуют нулевому значению коэффициента.

Разностная схема для этой области представлена уравнениями (1)–(6):

$$H^{m+0.5}_{x_{i,j+0.5,k+0.5}} = D_{a_{i,j+0.5,k+0.5}} H^{m-0.5}_{x_{i,j+0.5,k+0.5}} - D_{b_{i,j+0.5,k+0.5}} \left(\frac{E^m_{z_{i,j+1,k+0.5}} - E^m_{z_{i,j,k+0.5}}}{h_y} - \frac{E^m_{y_{i,j+0.5,k+1}} - E^m_{y_{i,j+0.5,k}}}{h_z} \right), \quad (1)$$

$$H^{m+0.5}_{y_{i+0.5,j,k+0.5}} = D_{a_{i+0.5,j,k+0.5}} H^{m-0.5}_{y_{i+0.5,j,k+0.5}} - D_{b_{i+0.5,j,k+0.5}} \left(\frac{E^m_{x_{i+0.5,j,k+1}} - E^m_{x_{i+0.5,j,k}}}{h_z} - \frac{E^m_{z_{i+1,j,k+0.5}} - E^m_{z_{i,j,k+0.5}}}{h_x} \right), \quad (2)$$

$$H^{m+0.5}_{z_{i+0.5,j+0.5,k}} = D_{a_{i+0.5,j+0.5,k}} H^{m-0.5}_{z_{i+0.5,j+0.5,k}} - D_{b_{i+0.5,j+0.5,k}} \left(\frac{E^m_{y_{i+1,j+0.5,k}} - E^m_{y_{i,j+0.5,k}}}{h_x} - \frac{E^m_{x_{i+0.5,j+1,k}} - E^m_{x_{i+0.5,j,k}}}{h_y} \right), \quad (3)$$

$$E^{m+1}_{x_{i+0.5,j,k}} = C_{a_{i+0.5,j,k}} E^m_{x_{i+0.5,j,k}} + C_{b_{i+0.5,j,k}} \left(\frac{H^{m+0.5}_{z_{i+0.5,j+0.5,k}} - H^{m+0.5}_{z_{i+0.5,j-0.5,k}}}{h_y} - \frac{H^{m+0.5}_{y_{i+0.5,j,k+0.5}} - H^{m+0.5}_{y_{i+0.5,j,k-0.5}}}{h_z} \right), \quad (4)$$

$$E^{m+1}_{y_{i,j+0.5,k}} = C_{a_{i,j+0.5,k}} E^m_{y_{i,j+0.5,k}} + C_{b_{i,j+0.5,k}} \left(\frac{H^{m+0.5}_{x_{i,j+0.5,k+0.5}} - H^{m+0.5}_{x_{i,j+0.5,k-0.5}}}{h_z} - \frac{H^{m+0.5}_{z_{i+0.5,j+0.5,k}} - H^{m+0.5}_{z_{i-0.5,j+0.5,k}}}{h_x} \right), \quad (5)$$

$$E^{m+1}_{z_{i,j,k+0.5}} = C_{a_{i,j,k+0.5}} E^m_{z_{i,j,k+0.5}} + C_{b_{i,j,k+0.5}} \left(\frac{H^{m+0.5}_{y_{i+0.5,j,k+0.5}} - H^{m+0.5}_{y_{i-0.5,j,k+0.5}}}{h_x} - \frac{H^{m+0.5}_{x_{i,j+0.5,k+0.5}} - H^{m+0.5}_{x_{i,j-0.5,k+0.5}}}{h_y} \right), \quad (6)$$

где

$$C_{a_{i,j,k}} = \left(1 - \frac{\sigma_{i,j,k} h_t}{2\varepsilon_{i,j,k}\varepsilon_0}\right) \Big/ \left(1 + \frac{\sigma_{i,j,k} h_t}{2\varepsilon_{i,j,k}\varepsilon_0}\right),$$

$$C_{b_{i,j,k}} = \left(\frac{h_t}{\varepsilon_{i,j,k}\varepsilon_0}\right) \Big/ \left(1 + \frac{\sigma_{i,j,k} h_t}{2\varepsilon_{i,j,k}\varepsilon_0}\right),$$

$$D_{a_{i,j,k}} = \left(1 - \frac{\sigma^*_{i,j,k} h_t}{2\mu_{i,j,k}\mu_0}\right) \Big/ \left(1 + \frac{\sigma^*_{i,j,k} h_t}{2\mu_{i,j,k}\mu_0}\right),$$

$$D_{b_{i,j,k}} = \left(\frac{h_t}{\mu_{i,j,k}\mu_0}\right) \Big/ \left(1 + \frac{\sigma^*_{i,j,k} h_t}{2\mu_{i,j,k}\mu_0}\right),$$

ε_0, μ_0 – электрическая и магнитная постоянные, $\varepsilon_{j,j,k}$, $\mu_{j,j,k}$ – сеточные относительные электрическая и магнитная проницаемости изотропной линейной бездисперсной среды, $\sigma_{i,j,k}$, $\sigma^*_{i,j,k}$ – сеточные удельные электрическая и магнитная проводимости.

При моделировании бесконечного свободного пространства вокруг D^3 наложим на сеточную область поглощающие слои CPML [1] шириной *nxPML*_1 отсчётов слева и *nxPML*_2 справа вдоль оси *x*, *nyPML*_1 и *nyPML*_2, *nzPML*_1 и *nzPML*_2 по осям *y* и *z* соответственно. В табл. 2 представлены диапазоны индексов (в нотации Голуба [13]) для каждой сеточной функции, соответствующие подобластям с наложенными слоями. Выражения (7)–(12) описывают сеточные уравнения в поглощающих слоях.

Табл. 2. Диапазоны индексов для проекций электрического и магнитного полей

	i		*j*		*k*	
E_x	–	–	1:*nyPML*_1	*J*–*nyPML*_2+1:*J*	1:*nzPML*_1	*K*–*nzPML*_2:*K*
E_y	1:*nxPML*_1	*I*–*nxPML*_2+1:*I*	–	–	1:*nzPML*_1	*K*–*nzPML*_2+1:*K*
E_z	1:*nxPML*_1	*I*–*nxPML*_2+1:*I*	1:*nyPML*_1	*J*–*nyPML*_2+1:*J*	–	–
H_x	–	–	1:*nyPML*_1–1	*J*–*nyPML*_2+1:*J*–*1*	1:*nzPML*_1–1	*K*–*nzPML*_2+1:*K*–1
H_y	1:*nxPML*_1–1	*I*–*nxPML*_2–1:*I*–1	–	–	1:*nzPML*_1–1	*K*–*nzPML*_2+1:*K*–1
H_z	1:*nxPML*_1–1	*I*–*nxPML*_2+1:*I*–1	1:*nyPML*_1–1	*J*–*nyPML*_2+1:*J*–1	–	–

$$H^{m+0.5}_{x_{i,j+0.5,k+0.5}} = D_{a_{i,j+0.5,k+0.5}} H^{m-0.5}_{x_{i,j+0.5,k+0.5}} - D_{b_{i,j+0.5,k+0.5}} \left(\frac{E^m_{z_{i,j+1,k+0.5}} - E^m_{z_{i,j,k+0.5}}}{\kappa_{y_{j+0.5}} h_y} - \frac{E^m_{y_{i,j+0.5,k+1}} - E^m_{y_{i,j+0.5,k}}}{\kappa_{z_{k+0.5}} h_z} + \Psi^m_{H_{xy_{i,j+0.5,k+0.5}}} - \Psi^m_{H_{xz_{i,j+0.5,k+0.5}}} \right), \quad (7)$$

$$H^{m+0.5}_{y_{i+0.5,j,k+0.5}} = D_{a_{i+0.5,j,k+0.5}} H^{m-0.5}_{y_{i+0.5,j,k+0.5}} - D_{b_{i+0.5,j,k+0.5}} \left(\frac{E^m_{x_{i+0.5,j,k+1}} - E^m_{x_{i+0.5,j,k}}}{\kappa_{z_{k+0.5}} h_z} - \frac{E^m_{z_{i+1,j,k+0.5}} - E^m_{z_{i,j,k+0.5}}}{\kappa_{x_{i+0.5}} h_x} + \Psi^m_{H_{yz_{i+0.5,j,k+0.5}}} - \Psi^m_{H_{xyx_{i+0.5,j,k+0.5}}} \right), \quad (8)$$

$$H^{m+0.5}_{z_{i+0.5,j+0.5,k}} = D_{a_{i+0.5,j+0.5,k}} H^{m-0.5}_{z_{i+0.5,j+0.5,k}} - D_{b_{i+0.5,j+0.5,k}} \left(\frac{E^m_{y_{i+1,j+0.5,k}} - E^m_{y_{i,j+0.5,k}}}{\kappa_{x_{i+0.5}} h_x} - \frac{E^m_{x_{i+0.5,j+1,k}} - E^m_{x_{i+0.5,j,k}}}{\kappa_{y_{j+0.5}} h_y} + \Psi^m_{H_{zx_{i+0.5,j+0.5,k}}} - \Psi^m_{H_{zy_{i+0.5,j+0.5,k}}} \right), \quad (9)$$

$$E^{m+1}_{x_{i+0.5,j,k}} = C_{a_{i+0.5,j,k}} E^m_{x_{i+0.5,j,k}} + C_{b_{i+0.5,j,k}} \left(\frac{H^{m+0.5}_{z_{i+0.5,j+0.5,k}} - H^{m+0.5}_{z_{i+0.5,j-0.5,k}}}{\kappa_{y_j} h_y} - \frac{H^{m+0.5}_{y_{i+0.5,j,k+0.5}} - H^{m+0.5}_{y_{i+0.5,j,k-0.5}}}{\kappa_{z_k} h_z} + \Psi^{m+0.5}_{E_{x,y_{i+0.5,j,k}}} - \Psi^{m+0.5}_{E_{x,z_{i+0.5,j,k}}} \right), \quad (10)$$

$$E^{m+1}_{y_{i,j+0.5,k}} = C_{a_{i,j+0.5,k}} E^{m}_{y_{i,j+0.5,k}} + C_{b_{i,j+0.5,k}} \left(\frac{H^{m+0.5}_{x_{i,j+0.5,k+0.5}} - H^{m+0.5}_{x_{i,j+0.5,k-0.5}}}{\kappa_{z_k} h_z} - \frac{H^{m+0.5}_{z_{i+0.5,j+0.5,k}} - H^{m+0.5}_{z_{i-0.5,j+0.5,k}}}{\kappa_{z_i} h_x} + \Psi^{m+0.5}_{E_{yz_{i,j+0.5,k}}} - \Psi^{m+0.5}_{E_{yz_{i,j+0.5,k}}} \right), \quad (11)$$

$$E^{m+1}_{z_{i,j,k+0.5}} = C_{a_{i,j,k+0.5}} E^{m}_{z_{i,j,k+0.5}} + C_{b_{i,j,k+0.5}} \left(\frac{H^{m+0.5}_{y_{i+0.5,j,k0.5}} - H^{m+0.5}_{y_{i-0.5,j,k+0.5}}}{\kappa_{x_i} h_x} - \frac{H^{m+0.5}_{x_{i,j+0.5,k+0.5}} - H^{m+0.5}_{x_{i,j-0.5,k+0.5}}}{\kappa_{y_j} h_y} + \Psi^{m+0.5}_{E_{zx_{i,j,k+0.5}}} - \Psi^{m+0.5}_{E_{zy_{i,j,k+0.5}}} \right), \quad (12)$$

где

$$\Psi^{m}_{H_{xy_{i,j+0.5,k+0.5}}} = b_{y_{j+0.5}} \Psi^{m-1}_{H_{xy_{i,j+0.5,k+0.5}}} + c_{y_{i+0.5}} \left(\frac{E^{m}_{z_{i,j+1,k+0.5}} - E^{m}_{z_{i,j,k+0.5}}}{h_y} \right),$$

$$\Psi^{m+0.5}_{E_{xy_{i+0.5,j,k}}} = b_{y_j} \Psi^{m-0.5}_{E_{xy_{i+0.5,j,k}}} + c_{y_j} \left(\frac{H^{m+0.5}_{z_{i+0.5,j+0.5,k}} - H^{m+0.5}_{z_{i+0.5,j-0.5,k}}}{h_y} \right),$$

$$b_w = e^{-\left(\frac{\sigma_w}{\varepsilon_0 \kappa_w} + \frac{a_w}{\varepsilon_0}\right) h_t}, \; c_w = \frac{\sigma_w}{\sigma_w \kappa_w + \kappa_w^2 a_w} \left(e^{-\left(\frac{\sigma_w}{\varepsilon_0 \kappa_w} + \frac{a_w}{\varepsilon_0}\right) h_t} - 1 \right),$$

σ_w, κ_w, a_w – коэффициенты CFS (*Complex-Frequency-Shifted*)-тензора. $\Psi^{m}_{H_{\alpha\beta_{i,j,k}}}$, $\Psi^{m+0.5}_{E_{\alpha\beta_{i,j,k}}}$ для остальных компонент определяются аналогично с точностью до замены x, y, z и i, j, k. Коэффициенты b_w и c_w отличны от 0 только в поглощающих слоях.

По краям сеточной области зададим граничное условие Дирихле, соответствующее идеальному проводнику, когда приведенные в табл. 3 компоненты вектора электрического поля $A^{m+d}_{i+a,j+b,k+c}$ равны 0 на соответствующих границах при $0 \le m \le M$:

Табл. 3. Граничное условие Дирихле для проекций электрического и магнитного полей

	$0 \le i \le I–I_x$	$0 \le j \le J–J_y$	$0 \le k \le K–K_z$
$0 \le i \le I–I_x$	-	$E^{m}_{x_{i+0.5,j,0}} = 0,\ E^{m}_{x_{i+0.5,j,K}} = 0$	$E^{m}_{x_{i+0.5,0,k}} = 0,\ E^{m}_{x_{i+0.5,J,k}} = 0$
$0 \le j \le J–J_y$	$E^{m}_{y_{i,j+0.5,0}} = 0,\ E^{m}_{y_{i,j+0.5,K}} = 0$	-	$E^{m}_{y_{0,j+0.5,k}} = 0,\ E^{m}_{y_{I,j+0.5,k}} = 0$
$0 \le k \le K–K_z$	$E^{m}_{z_{i,0,k+0.5}} = 0,\ E^{m}_{z_{i,J,k+0.5}} = 0$	$E^{m}_{z_{0,j,k+0.5}} = 0,\ E^{m}_{z_{I,j,k+0.5}} = 0$	-

Начальное условие зададим как $E^{0}_{x_{i+0.5,j,k}} = E^{0}_{y_{i,j+0.5,k}} = E^{0}_{z_{i,j,k+0.5}} = 0$, что соответствует отсутствию излучения в начальный момент времени. Поле вводится в область вычислительного эксперимента с помощью метода результирующего поля [14].

2. Метод пирамид

Приступая к синтезу алгоритма для организации вычислений по (1–12) на графическом процессоре, остановимся на методе пирамид, позволяющем снять остроту известного ограничения на объем видеопамяти.

Суть классического метода пирамид [10] заключается в следующих предписаниях:

- построении пространства итераций циклической конструкции, обеспечивающей вычисления по (1–12) на всей сеточной области D^3_h;
- выделении на этом пространстве результирующих итераций, от которых не зависит никакая другая;
- отнесении к ведению каждой задачи искомого алгоритма (в терминах модели канал/задача) всех итераций, от которых зависит данная результирующая.

Проблема ограничения памяти графического вычислительного устройства решается в [11] последовательной реализацией таких задач на одной видеокарте, а именно:

- пространство итераций разбивается по переменной, отвечающей за шаги по времени, на равные интервалы, к каждому из которых применяется метод пирамид;
- необходимо, чтобы все итерации, соответствующие меньшему значению переменной, не зависели от итераций, соответствующих большему значению переменной;
- высота пирамид выбирается из соображений минимизации длительности вычислений;
- разбиение на задачи может быть одинаковым для всех проходов;
- задачи выполняются квазипараллельно, по очереди занимая вычислительное устройство на время, необходимое для выполнения одного прохода;
- каждая из задач решается в векторизованном виде.

Применение модифицированного подхода к сеточным уравнениям позволяет сократить число коммуникаций, т.к. в нем пересылка данных осуществляется только после расчёта нескольких временных слоёв (их число равно высоте пирамиды), а не на каждом шаге, как в классическом алгоритме. Квазипараллельное исполнение задач позволяет одной задаче монопольно занимать GPU, наиболее полно используя доступную видеопамять.

Особенностью FDTD-метода является выражение каждой сеточной функции через значение сеточных функций на предыдущих временных слоях в явном виде, поэтому метод пирамид применяется ко всем уравнениям (1)–(6) одновременно: на каждом временном шаге рассчитываются все 6 компонент $A^{m+d}_{i+a,j+b,k+c}$ электромагнитного поля; перед началом расчёта и после завершения обработки пирамиды осуществляется пересылка необходимых значений для всех компонент полей. При этом расчёт полей в поглощающих слоях по формулам (7)–(12) будет производиться только для пирамид, расположенных по краям исходной сеточной области. Количество таких пирамид определяется соотношением ширины поглощающего слоя, ширины основания пирамиды и высоты пирамиды, определяющей ширину области перекрытия.

2.1. Случай одномерной сеточной области

Проиллюстрируем применение метода пирамид к реализации схемы Yee на GPU на примере одномерной задачи. Для одномерного случая (плоская электромагнитная волна распространяется по направлению Z) от нуля будут отличны только компоненты E_x, H_y электромагнитного поля и схема Yee сократится до двух уравнений (2), (4), в которых сеточные функции других компонент обнуляются. Расчёт по этим уравнениям требует наличия в оперативной памяти всей сеточной области.

Для обхода этого ограничения в [11] предлагается разбивать сеточную область на подобласти, каждая из которых может быть помещена в память GPU и производить в ней итерации по времени стандартным образом, после чего перемещать обратно на CPU. В случае тривиального алгоритма производится одна итерация по времени, при этом для расчёта r значений $E_{x_k}^m$ и $H_{y_{k+0.5}}^{m+0.5}$ необходимо передать по $R = r+2$ значений каждой из компонент, так как дифференциальный шаблон затрагивает соседние узлы. Однако этот алгоритм предполагает пересылку данных на каждом временном шаге.

Чтобы этого избежать, в [11] предлагается следующая модификация: для вычисления M слоёв сеточных уравнений (2), (4) необходимо последовательно выполнить s проходов методом пирамид, вычисляя каждый раз n слоёв ($M = sn$, n – высота пирамиды). Взаимодействие между задачами происходит через CPU и осуществляется только по окончании прохода, т.е. проходы выполняются независимо каждой из задач. При этом для расчёта в результате прохода r значений E_{x_k} и H_{y_k} необходимо передать по $R = r + 2n$ значений каждой из компонент. Число задач μ выбирается исходя из соотношения $2r\mu = 2K - 1$, где K – количество узлов сеточной области по направлению Z.

Множество итераций, выполняемых каждой задачей, представляет собой усеченную пирамиду высотой n, шириной верхнего основания r и нижнего основания R. Это иллюстрирует рис. 1 для второй задачи на примере разбиения области на 3 подзадачи, $n = 3$, $r = 3$, $R = 9$, где квадраты обозначают значения сеточной функции, столбцы – пространственные узлы сетки, строки – временные слои, цифры – номер задачи, вычисляющей значение в данном узле. На каждом временном слое вычисляются значения E_{x_k} и $H_{y_{k+0.5}}$.

Темно-серым обозначены значения, вычисляемые второй задачей, бледно-серым – передаваемые для ее расчёта.

Оценка сверху числа значений сеточной функции, обрабатываемых за один проход одной задачей, для метода пирамид и тривиального алгоритма приведена в табл. 4, при этом для расчёта n временных слоёв тривиальным алгоритмом необходимо выполнить n проходов.

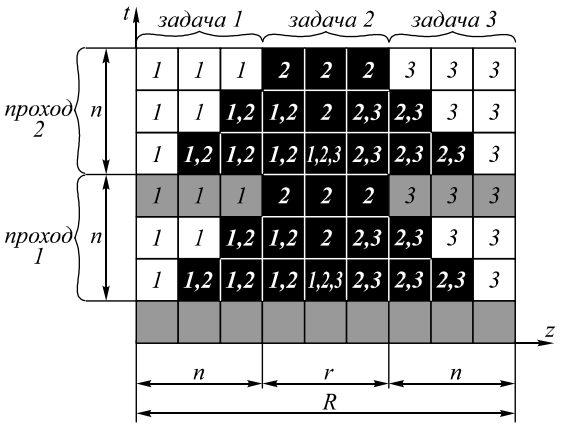

Рис. 1. Декомпозиция сеточной области в одномерном случае для второй задачи

Табл. 4. Оценка числа значений сеточной функции для одной задачи за один проход в одномерном случае

	Тривиальный алгоритм	Метод пирамид
Пересылаемых всего:	$4(R–1)$	$4(R–n)$
- в видеопамять	$2R$	$2R$
- из видеопамяти	$2(R–2)$	$2(R–2n)$
Вычисляемых по шаблону	$2(R–2)$	$2n(R–n–1)$

Таким образом, для одной задачи число пересылок сокращается в $n(R-1)/(R-n)$ раз, а вычислительная сложность увеличивается в $(R-n-1)/(R-2)$ раза.

2.2. Случай двумерной сеточной области

Иллюстрируя применение метода пирамид к двумерной сеточной области, рассмотрим пример распространения ТМ-волны. В этом случае от нуля будут отличны только компоненты E_x, H_y, H_z электромагнитного поля и схема Yee сократится до трёх уравнений (2), (3), (4), в которых сеточные функции других компонент обнуляются.

Тривиальный алгоритм для двумерной по пространству задачи с одномерной декомпозицией сеточной области аналогичен изложенному в п. 2.1. Сеточная область разбивается на равные прямоугольные подобласти размером $J \times (R-2)$, которые могут быть целиком помещены в видеопамять; т.к. для вычисления одного значения требуются значения сеточной функции в соседних узлах. Подобласти должны перекрываться на два сеточных узла по каждой стороне, не прилегающей к краям сеточной области. На каждом шаге по времени производится последовательная обработка подобластей: значения сеточных функций копируются в память GPU, вычисляются значения для следующего временного слоя, вычисленные значения копируются на CPU.

Аналогично для организации вычислений по методу пирамид с одномерной декомпозицией требуется область видеопамяти для размещения по $R = r + 2n$ значений каждой из компонент электромагнитного поля E_x, H_y, H_z для вычисления по r значений для каждой компоненты. Число задач μ выбирается исходя из соотношения $3r\mu = 3K - 1$.

На рис. 2 показана одномерная декомпозиция двумерной области на примере $n=3$, $r=3$, $R=9$, где кубы обозначают значения сеточной функции, столбцы – пространственные узлы сетки, строки – временные слои, цифры – номер задачи, вычисляющей значение в данном узле. На каждом временном слое вычисляются значения $E_{x_{j,k}}$, $H_{y_{j,k+0,5}}$, $H_{z_{j+0,5,k}}$.

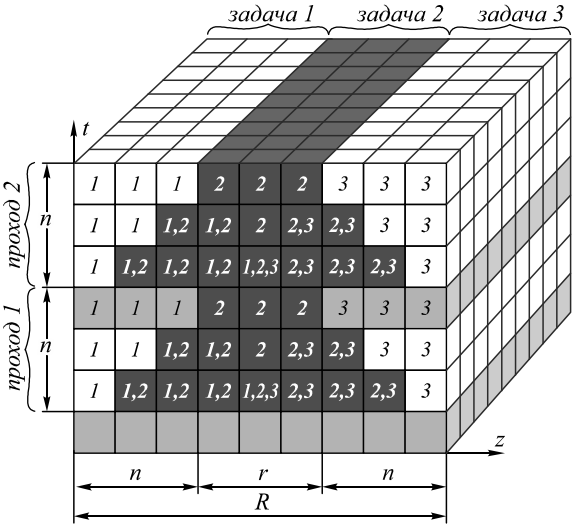

Рис. 2. Декомпозиция сеточной области в одномерном случае для второй задачи

Темно-серым обозначены значения, вычисляемые второй задачей, бледно-серым – получаемые ею от других задач.

Оценка сверху числа значений сеточной функции, обрабатываемых за один проход одной задачей, для метода пирамид и тривиального алгоритма приведена в табл. 5, при этом для расчёта n временных слоёв тривиальным алгоритмом необходимо выполнить n проходов.

Как видно из табл. 5, количество операций отличается от одномерного случая только множителем $(3J-1)$, соответствующим количеству узлов по направлению y для компонент E_x, H_y, H_z. Таким образом, для одной задачи число пересылок, так же как и в случае одномерной задачи, сокращается в $n(R-1)/(R-n)$ раз, а вычислительная сложность увеличивается в $(R-n-1)/(R-2)$ раза.

Табл. 5. Оценка числа значений сеточной функции для одной задачи за один проход в двумерном случае

	Тривиальный алгоритм	Метод пирамид
Пересылаемых всего:	$2(3J–1)(R–1)$	$2(3J–1)(R–n)$
- в видеопамять	$(3J–1)R$	$(3J–1)R$
- из видеопамяти	$(3J–1)(R–2)$	$(3J–1)(R–2n)$
Вычисляемых по шаблону	$(3J–1)(R–2)$	$(3J–1)n(R–n–1)$

2.3. Случай трёхмерной сеточной области с одномерной декомпозицией

В случае трёхмерной сеточной области от нуля отличны все компоненты электромагнитного поля и схема Yee представлена уравнениями (1)–(6).

Методики построения тривиального алгоритма и алгоритма метода пирамид, применяемые к трёхмерной сеточной области при одномерной декомпозиции последней, аналогичны рассмотренным выше с той лишь разницей, что сеточные подобласти, которые выделяются задачам, имеют форму параллелепипедов размером $I\times J\times(R-2)$, которые могут быть целиком помещены в видеопамять.

Оценка сверху числа значений сеточной функции, обрабатываемых за один проход одной задачей, для метода пирамид и тривиального алгоритма приведена в табл. 3, при этом для расчёта n временных слоёв тривиальным алгоритмом необходимо выполнить n проходов.

Как видно из табл. 6, количество операций отличается от одномерного случая только множителем $(6IJ-3I-3J+1)$, соответствующим количеству узлов в плоскости xy для компонент электромагнитного поля.

Табл. 6. Оценка числа значений сеточной функции для одной задачи за один проход в трехмерном случае

	Тривиальный алгоритм	Метод пирамид
Пересылаемых всего ω_c	$2(6IJ–3I–3J+1)$ $(R–1)$	$2(6IJ–3I–3J+1)$ $(R–n)$
- в видеопамять	$(6IJ–3I–3J+1)$ R	$(6IJ–3I–3J+1)R$
- из видеопамяти	$(6IJ–3I–3J+1)$ $(R–2)$	$(6IJ–3I–3J+1)$ $(R–2n)$
Вычисляемых по шаблону ω_n	$(6IJ–3I–3J+1)$ $(R–2)$	$(6IJ–3I–3J+1)$ $n(R–n–1)$

На рис. 3 представлена блок-схема алгоритма расчёта по методу пирамид, вычисления по которой описаны в следующем параграфе.

3. Экспериментальное исследование метода пирамид

Ниже описаны эксперименты, проведенные с целью выявления ускорения вычислений при применении метода пирамид к FDTD.

Вычислительные эксперименты по определению ускорения проводились на видеокарте NVIDIA GeForce GTX 660 Ti (табл. 7) и процессоре Intel Core 2 Duo E8500 (табл. 8). Исследование велось в операционной системе Ubuntu.

3.1. Программная реализация FDTD-метода

Рассмотрим распространение плоской волны в трехмерной области свободного пространства, ограниченной поглощающими слоями CPML, для чего будем рассматривать уравнения (1)–(6), решение которых соответствует моделированию распространения поля в подобласти без PML; (7)–(12) – в подобласти с PML и (13) – как начальное условие.

Зададим сеточную область размером 256×256×256 отсчётов по пространству, ширину поглощающего слоя выберем $nxPML = nyPML = nzPML = 11$ отсчётов. Для размещения такой задачи в памяти видеокарты необходимо 467 Мб.

Рис. 3. Схема алгоритма метода пирамид

Табл. 7. Основные характеристики GPU NVIDIA GeForce GTX 660 Ti

Характеристика	Значение
Количество мультипроцессоров, шт.	7
Размер видеопамяти, Гб	2
Максимальное число потоков в блоке, шт.	1024
Максимальная размерность блока потоков (*x*, *y*, *z*), шт.	1024×1024×128
Максимальная размерность сетки блоков, шт.	2147483×65535×65535
Тактовая частота ядра, МГц	1032
Тактовая частота памяти, МГц	6008

Табл. 8. Основные характеристики CPU Intel Core 2 Duo E8500

Характеристика	Значение
Тактовая частота ядра, ГГц	3,16
Тактовая частота шины CPU, МГц	1333
Кеш L1, Кб	64×2
Кеш L2, Кб	6144

Применим метод пирамид с одномерной декомпозицией и проведем разбиение исходной сеточной области по направлению *K*. Ниже приведен исходный код, описывающий алгоритм, приведённый на рис. 3:

```
__host__ void raschetGPU_pyramid_1d()
{
  ...
  //Число запускаемых блоков
  int SIZEx = ((Imax-1)%BLOCK_DIMx==0) ? (Imax-
1)/BLOCK_DIMx : (Imax-1)/BLOCK_DIMx+1;
int SIZEy = ((Jmax-1)%BLOCK_DIMy==0) ? (Jmax-
1)/BLOCK_DIMy : (Jmax-1)/BLOCK_DIMy+1;
dim3 gridSize = dim3(SIZEx,SIZEy, 1);
dim3 blockSize = dim3(BLOCK_DIMx,BLOCK_DIMy,
BLOCK_DIMz);
create_temp_fields();//Копирование пересекаю-
//щихся областей во временный буфер
 int countPyramidsK = ceil((Kmax-1) /
(pyramidBaseLengthK + 0.0f));
 // Число пирамид (п. 2 на рисунке 3)
 int currentTime = 1;//Текущий временной шаг
 int timeOfPass =((nmax)%PASS==0)? nmax/PASS :
nmax/PASS+1;
// Число проходов (п. 3 на рисунке 3)
//Цикл по проходам (п. 4 на рисунке 3)
for (int pass = 1; pass <= PASS; pass++)
{
  int currentBaseLengthK; //ширина по К верх-
  //него основания текущей пирамиды
  int leftOffsetK;  //ширина левого перекрытия
  //по К нижнего основания пирамиды
  int rightOffsetK;  //ширина правого перекры-
  //тия по К нижнего основания пирамиды
  int fullBaseLengthK;  //ширина по К нижнего
  //основания текущей пирамиды
  int leftPyramidBorderK; //индекс левой грани-
  //цы основания пирамиды
  int rightPyramidBorderK;//индекс правой гра-
  //ницы основания пирамиды
  int durationOfTimePass = timeOfPass * pass;
  //временной шаг, соответствующий окончанию
  //текущего прохода
  //Копируем во временный буфер начальные зна-
  //чения полей для прохода
  copyFields(...);
  //Цикл по пирамидам (п. 5 на рисунке 3)
  for(int pyramidIdK = 0; pyramidIdK <
  countPyramidsK; pyramidIdK++)
  {
    int startPyramidBasePositionK = pyramidIdK
    * pyramidBaseLengthK; //начальный индекс
    //верхнего основания пирамиды в сетке
    getOffsets(pyramidIdK, pyramidBaseLengthK,
    &currentBaseLengthK, &leftOffsetK,
    &rightOffsetK);
    getBorders(currentBaseLengthK, leftOffsetK,
    rightOffsetK, &fullBaseLengthK,
    &leftPyramidBorderK, &rightPyramidBorderK);
    //Копирование полей на видеокарту (п. 6 на
    //рисунке 3)
    create_arrays_on_GPU(Imax,Jmax,fullBaseLengthK);
    copy_temp_arrays_to_GPU(0,0,leftPyramidBord
    erK,Imax,Jmax,fullBaseLengthK);
    copy_constant_to_device();
    //Цикл по времени внутри прохода
    //(п. 7 на рисунке 3)
    for (int n = currentTime; n <= durationOfT-
    imePass && n <=nmax; n++)
    {
      //(п. 8 на рисунке 3)
      //Ядро для пересчёта компонент магнитного
      //поля на GPU
      kernelH_pyramid1<<<gridSize,blockSize>>>(...);
      cudaEventSynchronize(syncEvent);
      //Ядро для пересчёта компонент электриче-
      //ского поля на GPU
      kenelE_pyramid1<<<gridSize,blockSize>>>(...);
      cudaEventSynchronize(syncEvent);
    }
    //Копирование данных из видеопамяти (п. 9
    //на рисунке 3)
    copy_arrays_from_GPU_with_offset(currentBas
    eLengthK, startPyramidBasePositionK, left-
    OffsetK);
    delete_arrays_on_GPU();
  }
  currentTime = durationOfTimePass +1;
}
...
}
```

Сеточные подобласти, соответствующие нижним основаниям пирамид и содержащие начальные данные

для каждого прохода, пересекаются (аналогично показанному на рис. 1, 2 для одномерного и двумерного случаев соответственно), поэтому перед началом очередного прохода необходимо скопировать эти области во временный буфер (реализуется процедурой *create_temp_fields*()), чтобы значения не были изменены при копировании из видеопамяти значений, рассчитанных при обработке соседней пирамиды.

3.2. Постановка вычислительных экспериментов

Исследуя условия эффективности применения метода пирамид, проведем две серии вычислительных экспериментов, ограничив объем используемой видеопамяти 128 и 256 Мб, что позволит построить до 16 (в первом случае) и 4 (во втором) пирамид с разной шириной основания и высотой. Целью экспериментов будет определение зависимости ускорения вычислений от данных параметров.

На рис. 4 и 5, в табл. 9 и 10 приведены результаты измерений длительности вычислений.

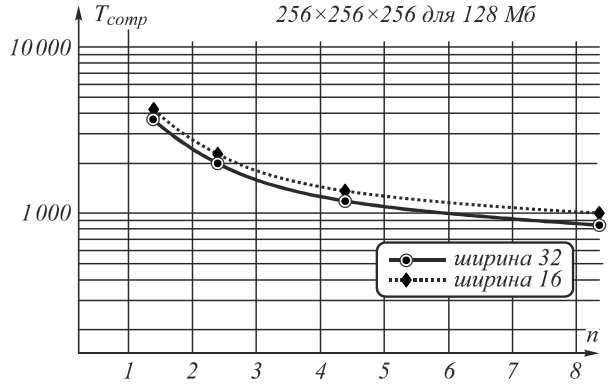

Рис. 4. Зависимость времени расчёта Tcomp. (с) от высоты (число отсчётов) пирамиды для различной ширины основания (128Мб)

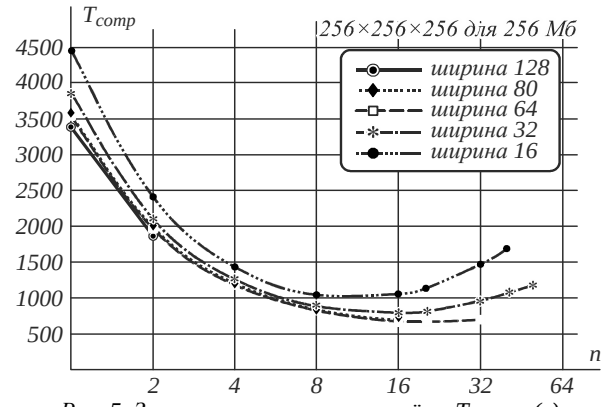

Рис. 5. Зависимость времени расчёта Tcomp. (с) от высоты (число отсчётов) пирамиды для различной ширины основания (256Мб)

Как показали результаты экспериментов, минимальное время вычислений достигается при соблюдении баланса использования памяти устройства и числа пересылок. В эксперименте это достигнуто при использовании пирамид с шириной основания 32 и высотой 8 отсчётов при использовании 128 Мб памяти; пирамид с шириной основания 64 и высотой 20 отсчётов при использовании 256 Мб памяти.

Табл. 9. Зависимость времени расчёта (с) от высоты (число отсчётов) пирамиды для различной ширины основания

μ	*r*	*n*	Объём памяти для расчёта одной пирамиды, Мб	Время расчёта, с	Ускорение
8	32	1	81	3854	1
8	32	2	84	2091	1,84
8	32	4	91	1253	3,07
8	32	8	105	879	4,38
16	16	1	53	4456	0,86
16	16	2	56	2400	1,61
16	16	4	63	1436	2,68
16	16	8	77	1035	3,72

Табл. 10. Зависимость времени расчёта (с) от высоты (число отсчётов) пирамиды для различной ширины основания

μ	*r*	*n*	Объем памяти для расчёта одной пирамиды, Мб	Время расчёта, с	Ускорение
2	128	1	246	3403	1
2	128	2	248	1896	1,79
4	80	1	164	3560	0,96
4	80	2	168	1969	1,73
4	80	4	175	1190	2,86
4	80	8	189	820	4,15
4	80	16	216	683	4,98
4	64	1	136	3561	0,96
4	64	2	140	1960	1,74
4	64	4	147	1184	2,87
4	64	8	161	814	4,18
4	64	16	189	673	5,05
4	64	20	202	660	5,16
4	64	32	244	694	4,90
8	32	1	81	3855	0,88
8	32	2	84	2092	1,63
8	32	4	91	1254	2,71
8	32	8	105	880	3,86
8	32	16	133	795	4,28
8	32	20	147	809	4,21
8	32	32	189	956	3,55
8	32	40	216	1062	3,20
8	32	50	251	1177	2,89
16	16	1	53	4457	0,76
16	16	2	56	2400	1,41
16	16	4	63	1436	2,37
16	16	8	77	1035	3,29
16	16	16	105	1054	3,23
16	16	20	119	1121	3,03
16	16	32	161	1464	2,32
16	16	40	189	1682	2,02

При дальнейшем увеличении высоты пирамиды время расчёта начинает увеличиваться за счёт увеличения объема дублирующихся данных, что иллюстрируется U-образной зависимостью времени вычислений от высоты пирамиды. На рис. 4 зависимость имеет линейный характер, однако ограничение памяти в 128 Мб не позволяет более увеличивать высоту пирамиды при указанной ширине основания, в результате чего невозможно достигнуть превышения объема дублирующихся

вычислений над временем пересылки, что хорошо видно при ограничении памяти в 256 Мб.

Заключение

Представленная реализация FDTD с использованием метода пирамид позволяет снизить влияние ограниченного объема памяти графического процессора и использовать преимущество GPU в быстродействии за счёт снижения интенсивности коммуникаций между оперативной и видеопамятью за счёт дублирования арифметических операций.

При реализации вычислений по методу пирамид важно соблюдать баланс между объемом пересылок и задействованным объемом памяти: определяя ширину основания пирамиды при фиксированной высоте, необходимо выбирать максимально допустимую ширину основания, так как это сокращает число пересылок. Расчёты, произведенные с использованием пирамид одинаковой высоты, имеют близкие значения ускорения, что также подчеркивает значительность затрат на пересылку данных по сравнению с затратами на непосредственный расчёт.

Благодарности

Работа выполнена при поддержке гранта РФФИ 14-07-00291-а.

Литература

1. **Taflove A.** Computational Electrodynamics: The Finite-Difference Time-Domain Method / A. Taflove, S. Hagness. – 3th ed. – Boston: Arthech House Publishers, 2005. – 1006 p.
2. **Котляр, В.В.** Фотонные струи, сформированные квадратными микроступеньками / В.В. Котляр, С.С. Стафеев, А.Ю. Фельдман // Компьютерная оптика. – 2014. – Т. 38, № 1. – С. 72-80.
3. **Тиранов, А.Д.** Коллективное спонтанное излучение в волноводе с близким к нулю показателем преломления / А.Д. Тиранов, А.А. Калачёв // Известия РАН, серия физическая. – 2014. – Т. 78, № 3. – С. 271-275.
4. **Перов, С.Ю.** Теоретическая и экспериментальная дозиметрия в оценке биологического действия электромагнитных полей носимых радиостанций. Сообщение 1. Плоские фантомы / С.Ю. Перов, Е.В. Богачёва // Радиационная биология: Радиоэкология. – 2014. – Т. 54, № 1. – С. 57-61.
5. Основы работы с технологией CUDA / А.В. Боресков, А.А. Харламов. – М.: ДМК Пресс, 2010. – 232 с.
6. OpenCL – The open standard for parallel programming of heterogeneous systems / URL: http://www.khronos.org/opencl/.
7. B-CALM – Belgium California Light Machine [Электронный ресурс]. – URL: http://b-calm.sourceforge.net/.
8. FDTD solver [Электронный ресурс]. – URL: http://www.acceleware.com/fdtd-solvers.
9. **Lamport, L.** The parallel execution of DO loops / L. Lamport // Communications of the ACM. – 1974. – Vol. 17(2). – P. 83-93.
10. **Вальковский, В.А.** Параллельное выполнение циклов. Метод пирамид / В.А. Вальковский // Кибернетика. – 1983. – № 5. – С. 51-55.
11. **Головашкин, Д.Л.** Решение сеточных уравнений на графических вычислительных устройствах. Метод пирамид [Электронный ресурс] / Д.Л. Головашкин, А.В. Кочуров // Современные проблемы прикладной математики и механики: теория, эксперимент и практика (Новосибирск, Россия, 30 мая – 4 июня 2011 г.). – Новосибирск: ИВТ СО РАН, 2011. – URL: http://conf.nsc.ru/files/conferences/niknik-90/fulltext/37858/46076/kochurov_final.pdf.
12. **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.
13. **Климов, В.В.** Наноплазмоника / В.В. Климов. – М.: Физматлит, 2009. – 480 с.
14. **Taflove, A.** Numerical solution of steady-state electromagnetic scattering problems using the time-dependent Maxwells's equation's / A. Taflove, M. Brodwin // IEEE Transactions of Microwave Theory and Techniques. – 1975. – Vol. mtt-23, Issue 8. – P. 623-630.

Сведения об авторах

Малышева Светлана Александровна, Самарский государственный аэрокосмический университет имени академика С.П. Королева (национальный исследовательский университет), аспирант. Область научных интересов: FDTD-метод, векторные и матричные вычисления, CUDA. E-mail: *s-a-mal@yandex.ru* .

Головашкин Димитрий Львович, доктор физико-математических наук, профессор кафедры прикладной математики Самарского государственного аэрокосмического университета имени академика С.П. Королева (национального исследовательского университета), ведущий научный сотрудник Института систем обработки изображений (ИСОИ) РАН. Область научных интересов: FDTD-метод, векторные и матричные вычисления, математическое моделирование оптических элементов и устройств нанофотоники. E-mail: *dimitriy@smr.ru* .

Поступила в редакцию 15 февраля 2016 г. Окончательный вариант – 5 апреля 2016 г.

IMPLEMENTATION OF THE FDTD ALGORITHM ON GPU USING A PYRAMID METHOD

S.A. Malysheva [1,2], D.L. Golovashkin [1,2]
[1] Image Processing Systems Institute, Russian Academy of Sciences, Samara, Russia,
[2] Samara State Aerospace University, Samara, Russia

Abstract

In this paper we develop a pyramid method in the context of solving time-dependent Maxwell's equations based on the finite difference time domain (FDTD) approach, which is implemented on a

graphics processing unit (GPU). Application of this method allows the impact of the GPU's limited memory capacity on the computation time to be reduced, which is significant for the FDTD method.

Keywords: FDTD, Maxwell's equations, method of pyramids, GPU, CUDA.

Citation: Malysheva SA, Golovashkin DL. Implementation of the FDTD algorithm on GPU using a pyramid method. Computer Optics 2016; 40(2): 179-87. DOI: 10.18287/2412-6179-2016-40-2-179-187.

Acknowledgements: The work was partially funded by the Russian Foundation of Basic Research Grants No. 14-07-00291-a.

References

[1] Taflove A, Hagness S. Computational Electrodynamics: The Finite-Difference Time-Domain Method. 3th ed. Boston: Arthech House Publishers; 2005.

[2] Kotlyar VV, Stafeev SS, Feldman AYu. Photonic Nanojets Formed By Square Microsteps [In Russian]. Computer Optics 2014; 38(1): 72-80.

[3] Tiranov AD, Kalachev AA. Collective spontaneous emission in a waveguide with close to zero refractive index [In Russian]. Bulletin of the Russian Academy of Sciences 2014; 78(3): 271-275.

[4] Petrov SY, Bogacheva EV. Theoretical and experimental dosimetry in the assessment of biological action of electromagnetic fields portable radio. Message 1. Flat phantoms [In Russian]. Radiation Biology: Radioecology 2014; 54(1): 57-61.

[5] Boreskov AV, Harlamov AA. The basics of working with CUDA technology [In Russian]. Moscow: DMK Press, 2010.

[6] OpenCL – The open standard for parallel programming of heterogeneous systems. Source: ⟨http://www.khronos.org/opencl⟩.

[7] B-CALM – Belgium California Light Machine. Source: ⟨http://b-calm.sourceforge.net/⟩.

[8] FDTD solver. Source: ⟨http://www.acceleware.com/fdtd-solvers⟩.

[9] Lamport L. The parallel execution of DO loops. Communications of the ACM 1974; 17(2): 83-93.

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

[11] Golovashkin DL, Kochurov AV. The decision of the grid equations graphical computing devices. Method pyramids. Source: ⟨http://conf.nsc.ru/files/conferences/niknik-90/fulltext/37858/46076/kochurov_final.pdf⟩.

[12] Yee KS. Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media. IEEE Trans Antennas Propag 1966; AP-14: 302-307.

[13] Klimov VV, Nanoplasmonics [In Russian]. Moscow: Fizmatlit; 2009.

[14] Taflove A, Brodwin M. Numerical solution of steady-state electromagnetic scattering problems using the time-dependent Maxwells's equation's. IEEE Transactions of Microwave Theory and Techniques 1975; 23(8): 623-630.

Authors' information

Svetlana Alexandrovna Malysheva, Samara State Aerospace University (National Research University), post-graduate student. Scientific interests: FDTD-method, vector and matrix computations, CUDA. E-mail: *s-a-mal@yandex.ru* .

Dimitriy Lvovich Golovashkin, Doctor of Physical and Mathematical Sciences, Professor of Samara State Aerospace University (National Research University), leading scientist of Image Processing Systems Institute of RAS. Scientific interests: FDTD-method, vector and matrix computations, mathematical modeling optical elements and devices of nanophotonics. E-mail: *dimitriy@smr.ru* .

Received February 15, 2016. The final version – April 5, 2016.