УДК 517:519.6

А.В. Гладкий, В.А. Богаенко

Моделирование переноса загрязнений в атмосфере с использованием параллельных вычислений

Рассмотрена задача моделирования переноса загрязнений в атмосфере на основе нестационарного уравнения конвективной диффузии в двух- и трехмерной постановке. Для вычислительной схемы, базирующейся на методах расщепления с использованием явных разностных схем бегущего счета, разработаны параллельные алгоритмы для графических процессоров и получены теоретические оценки времени их работы.

Modelling of pollution transfer in atmosphere has been considered on the base of unsteady convection-diffusion equation in two- and three-dimensional formulation. Parallel algorithms for graphical processors have been developed for computational scheme based on explicit finite-difference splitting methods. Theoretical estimations of algorithms execution time have been obtained.

Розглянуто задачу моделювання переносу забруднень в атмосфері на основі нестаціонарного рівняння конвективної дифузії у дво- та тривимірній постановці. Для обчислювальної схеми, що базується на методах розщеплення з використанням явних різницевих схем біжучої хвилі, розроблено паралельні алгоритми для графічних процесорів та отримано теоретичні оцінки часу їх роботи.

Введение. Математическое моделирование и вычислительный эксперимент становятся основными способами изучения процессов переноса загрязняющих веществ в экосистемах [1–6]. Компьютерная реализация математических моделей, включающих многомерные уравнения конвективной диффузии, основана на применении нетривиальных вычислительных алгоритмов и требует использования высокопроизводительных вычислительных средств, в частности, многопроцессорных кластерных систем и систем с графическими процессорами.

Постановка задачи

Целью статьи – построение параллельных алгоритмов расщепления для решения задач переноса загрязнений в атмосфере с использованием метеорологических данных или модели потенциального течения перемещения воздушных масс. Вычислительная схема решения многомерных задач основана на методах расщепления [7, 8] с использованием явных разностных схем бегущего счета, что позволяет существенно снизить сложность дискретных алгоритмов и применять технологию параллельных вычислений для их реализации на ЭВМ.

Кроме того, разработаны параллельные алгоритмы для графических процессоров (*GPU*), получены теоретические оценки времени их работы, позволяющие выбирать оптимальную с учетом быстродействия схему организации вычислений. Отметим, что большинство параллельных (в том числе для *GPU*) алгоритмов разработаны для неявных схем расщепления [9–14], а существующие параллельные алгоритмы для явных схем разработаны преимущественно для систем с распределенной памятью [15].

Математическая модель переноса и трансформации примесей

Для моделирования нестационарного процесса конвективно-диффузионного распространения вредных веществ в атмосфере, производственных помещениях, на промплощадках и других объектах будем использовать уравнение миграции примеси [1, 2, 6, 8]

$$\frac{\partial C}{\partial t}+u\frac{\partial C}{\partial x}+v\frac{\partial C}{\partial y}+w\frac{\partial C}{\partial z}+\sigma C=$$
$$=\frac{\partial}{\partial x}(\mu_1\frac{\partial C}{\partial x})+\frac{\partial}{\partial y}(\mu_2\frac{\partial C}{\partial y})+\frac{\partial}{\partial z}(\mu_3\frac{\partial C}{\partial z})+ \qquad (1)$$
$$+f(x,y,z,t),\ (x,y,z)\in G,\ t\in(0,T],$$

где $\mathbf{x}=(x,y,z)$ – декартовы координаты, $C(x,y,z,t)$ – концентрация примеси, $(u(\mathbf{x}), v(\mathbf{x}), w(\mathbf{x}))$ – компоненты вектора скорости воз-

душных масс $\mathbf{V}=(u,v,w)$, $\mu=(\mu_1,\mu_2,\mu_3)$ – коэффициенты турбулентной диффузии, σ – коэффициент трансформации примеси, $f(x,y,z,t)$ – функция, характеризующая распределение источников загрязнения, G – область с цилиндрической поверхностью ∂G, состоящей из боковой поверхности Γ, нижнего основания Γ_0 (при $z=0$) и верхнего основания Γ_H (при $z=H$).

Компоненты вектора скорости воздушного потока $\mathbf{V}=(u,v,w)$ удовлетворяют уравнению неразрывности (условию несжимаемости среды)

$$\operatorname{div}\mathbf{V}=\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}=0. \qquad (2)$$

В случае постоянно действующих точечных источников загрязнения или залповых аэрозольных выбросов в атмосферу в моменты времени $t_i, i=\overline{1,N}$ правую часть уравнений (1), (3) целесообразно представить в виде

$$f(x,y,z)=\sum_{i=1}^{N}Q_i\,\delta(x-x_i)\delta(y-y_i)\delta(z-z_i),$$

$$f(x,y,z,t)=$$

$$=\sum_{i=1}^{N}Q_i\,\delta(x-x_i)\delta(y-y_i)\delta(z-z_i)\delta(t-t_i)$$

соответственно, где Q_i – интенсивность выброса вредного вещества от *i*-го источника; $\delta(\cdot)$ – дельта-функция Дирака; $(x_i,y_i,z_i), i=\overline{1,N}$ – координаты точечных источников примеси.

Уравнение (1) следует дополнить начальным и граничными условиями [1, 8].

При исследовании переноса и рассеивания загрязнений в промышленных зонах с учетом влияния рельефа местности возникает необходимость расчета компонентов вектора скорости перемещения воздушных масс. Простейшая модель для описания поля скоростей есть модель потенциального течения, тогда $\mathbf{V}=\operatorname{grad}p$, где $p=p(x,y,z)$ – потенциал скорости. В результате, с учетом уравнения (2), для p получаем уравнение Лапласа

$$\Delta p=\frac{\partial^2 p}{\partial x^2}+\frac{\partial^2 p}{\partial y^2}+\frac{\partial^2 p}{\partial z^2}=0. \qquad (3)$$

Для однозначного решения уравнения (3) задаются следующие граничные условия. На твердых стенках производная потенциала по направлению единичной внешней нормали n равняется нулю, поэтому $\partial p/\partial n=0$. На границе попадания воздушного потока значение нормальной компоненты его скорости V_n известно, поэтому $\partial p/\partial n=V_n$. На участке вытекания воздуха задают значение потенциала скорости $p=p_э$.

Двухшаговая схема расщепления для расчета потенциального течения

Следуя идее установления [16], для решения (3) в области $\overline{G}=G\cup\Gamma=\{0\le x\le l_1, 0\le y\le l_2, 0\le z\le l_3\}$ рассмотрим схему расщепления для неоднородного нестационарного уравнения

$$\frac{\partial p}{\partial t}+Ap=f(x,y,z),\ Ap=-\Delta p \qquad (4)$$

с граничным условием первого рода и произвольным начальным условием. Представим оператор A в виде $A=A_1+A_2, A_i=-0{,}5\Delta,\ i=1,2$. Тогда, в результате последовательного решения двух вспомогательных задач

$$\frac{\partial p_1}{\partial t}+A_1p_1=0{,}5f,\ p_1(x,y,z,t)=p(x,y,z,t), \qquad (5)$$

$$\frac{\partial p_2}{\partial t}+A_2p_2=0{,}5f,\ p_2(x,y,z,t)=p_1(x,y,z,\hat{t}), \qquad (6)$$

получаем решение уравнения (4) в момент времени $\hat{t}=t+\tau$ с погрешностью $O(\tau^2)$ [17].

Для численного исследования нестационарных уравнений (5), (6) в области $\overline{G}$ введем пространственную равномерную сетку ($x_i=ih_1, y_j=jh_2, z_k=kh_3$). Во внутренних узлах (x_i,y_j,z_k) в момент времени $t=t_n=\tau n$ уравнениям (5), (6) поставим в соответствие неявную разностную схему

$$\varphi_t-0{,}5\left(\varphi^{n+1}_{x\bar{x}}+\varphi^{n+1}_{y\bar{y}}+\varphi^{n+1}_{z\bar{z}}\right)=0{,}5f, \qquad (7)$$

где φ – сеточная функция, и используются общепринятые обозначения теории разностных схем [8, 16]. На первом шаге явную схему бегущего счета

$$\varphi_t - \frac{1}{2}\left(\frac{1}{h_1}\left(\varphi_x^n - \varphi_{\bar{x}}^{n+1}\right) + \right.$$
$$\left. + \frac{1}{h_2}\left(\varphi_y^n - \varphi_{\bar{y}}^{n+1}\right) + \frac{1}{h_3}\left(\varphi_z^n - \varphi_{\bar{z}}^{n+1}\right)\right) = 0{,}5f \quad (8)$$

можно получить, заменяя в уравнении (7) операторы $\varphi_x^{n+1}, \varphi_y^{n+1}, \varphi_z^{n+1}$ их соответствующими в предыдущий момент времени $t = t_n$. На втором шаге разностную схему получим, заменяя в (7) операторы $\varphi_{\bar{x}}^{n+1}, \varphi_{\bar{y}}^{n+1}, \varphi_{\bar{z}}^{n+1}$ выражениями $\varphi_{\bar{x}}^{n}, \varphi_{\bar{y}}^{n}, \varphi_{\bar{z}}^{n}$:

$$\varphi_t - \frac{1}{2}\left(\frac{1}{h_1}\left(\varphi_x^{n+1} - \varphi_{\bar{x}}^{n}\right) + \right.$$
$$\left. + \frac{1}{h_2}\left(\varphi_y^{n+1} - \varphi_{\bar{y}}^{n}\right) + \frac{1}{h_3}\left(\varphi_z^{n+1} - \varphi_{\bar{z}}^{n}\right)\right) = 0{,}5f. \quad (9)$$

При этом решение уравнения (8) при $t = t_{n+1}$ – стартовое для разностного уравнения (9).

Расщепляя интервал τ между точками t_n и t_{n+1} на две равные части с промежуточной точкой $t_{n+1/2}$, двухшаговую схему можно записать в виде

$$\frac{1}{\tau}\left(\varphi^{n+1/2} - \varphi^n\right) = \frac{1}{2}\left(\frac{1}{h_1}\left(\varphi_x^n - \varphi_{\bar{x}}^{n+1/2}\right) + \right.$$
$$\left. + \frac{1}{h_2}\left(\varphi_y^n - \varphi_{\bar{y}}^{n+1/2}\right) + \frac{1}{h_3}\left(\varphi_z^n - \varphi_{\bar{z}}^{n+1/2}\right)\right) + 0{,}5f, \quad (10)$$

$$\frac{1}{\tau}\left(\varphi^{n+1} - \varphi^{n+1/2}\right) = \frac{1}{2}\left(\frac{1}{h_1}\left(\varphi_x^{n+1} - \varphi_{\bar{x}}^{n+1/2}\right) + \right.$$
$$\left. + \frac{1}{h_2}\left(\varphi_y^{n+1} - \varphi_{\bar{y}}^{n+1/2}\right) + \frac{1}{h_3}\left(\varphi_z^{n+1} - \varphi_{\bar{z}}^{n+1/2}\right)\right) + 0{,}5f. \quad (11)$$

В совокупности они составляют разностную схему бегущего счета, аппроксимирующую исходную дифференциальную задачу с погрешностью $O(|h|^2 + \tau^2 + \tau^2/|h|^2)$. Поэтому точность получаемых результатов зависит от соотношения шагов сетки.

Двухшаговая схема расщепления для задачи конвекции–диффузии

Эту схему рассмотрим в области $\overline{G} = G \cup \Gamma = \{0 \le x \le l_1,\ 0 \le y \le l_2\}$ на примере двумерного уравнения конвективной диффузии

$$\frac{\partial C}{\partial t} + u\frac{\partial C}{\partial x} + v\frac{\partial C}{\partial y} + \sigma C = \mu\left(\frac{\partial^2 C}{\partial x^2} + \frac{\partial^2 C}{\partial y^2}\right) + f(x, y) \quad (12)$$

с начальным условием и однородным граничным условием первого рода. Представим компоненты вектора скорости воздушного потока $u = u(x, y)$ и $v = v(x, y)$ в виде $u = u^+ + u^-$, $v = v^+ + v^-$, где $u^+ = 0{,}5(u + |u|) \ge 0$, $u^- = 0{,}5(u - |u|) \le 0$, $v^+ = 0{,}5(v + |v|) \ge 0$, $v^- = 0{,}5(v - |v|) \le 0$.

Пусть τ – временной интервал между точками t_n и t_{n+1}. Аналогично предыдущему, схема расщепления на дифференциальном уровне принимает вид

$$\frac{\partial C_1}{\partial t} + A_1 C_1 = 0{,}5f,\ C_1^n = C^n = C(x, y, t_n),$$
$$\frac{\partial C_2}{\partial t} + A_2 C_2 = 0{,}5f,\ C_2^n = C_1^{n+1},\ C^{n+1} = C_2^{n+1},$$

где $A_1 C = \left(u^+\frac{\partial C}{\partial x} + v^+\frac{\partial C}{\partial y}\right) + \frac{\sigma}{2}C - \frac{\mu}{2}\Delta C$, $A_2 C = \left(u^-\frac{\partial C}{\partial x} + v^-\frac{\partial C}{\partial y}\right) + \frac{\sigma}{2}C - \frac{\mu}{2}\Delta C$.

Дифференциальные операторы $A_i, i = \overline{1-2}$ во внутренних узлах $(x_i, y_j) \in \omega_h$ двумерной сетки $\overline{\omega}_h = \omega_h \cup \gamma_h = \{(x, y): x = x_i = ih_1,\ i = \overline{0, N_1};\ y = y_j = jh_2, j = \overline{0, N_2};\ h_\alpha = l_\alpha / N_\alpha, \alpha = 1,2\}$ аппроксимируем, используя для этого конвективные слагаемые схемы с направленными разностями [8]. В результате явную разностную схему расщепления бегущего счета для решения уравнения (12) можно получить в следующем виде

$$\varphi_t + u^+\varphi_{\bar{x}}^{n+1} + v^+\varphi_{\bar{y}}^{n+1} + \frac{\sigma}{2}\varphi^{n+1} -$$
$$- \frac{\mu}{2}\left((1 - R_1^+)\frac{1}{h_1}\left(\varphi_x^n - \varphi_{\bar{x}}^{n+1}\right) + \right.$$
$$\left. + (1 - R_2^+)\frac{1}{h_2}\left(\varphi_y^n - \varphi_{\bar{y}}^{n+1}\right)\right) = \frac{1}{2}f^n. \quad (13)$$

$$\varphi_t + u^-\varphi_x^{n+1} + v^-\varphi_y^{n+1} + \frac{\sigma}{2}\varphi^{n+1} -$$

$$-\frac{\mu}{2}\left((1+R_1^-)\frac{1}{h_1}\left(\varphi_x^{n+1}-\varphi_{\bar{x}}^{n}\right)+\right. \qquad (14)$$

$$\left.+(1+R_2^-)\frac{1}{h_2}\left(\varphi_y^{n+1}-\varphi_{\bar{y}}^{n}\right)\right)=\frac{1}{2}f^n,$$

где $R_1^+ = h_1\frac{u^+}{\mu}, R_2^+ = h_2\frac{v^+}{\mu}$, $R_1^- = h_1\frac{u^-}{\mu}$, $R_2^- = h_2\frac{v^-}{\mu}$. При этом решение уравнения (13) в момент времени $t = t_{n+1}$ – стартовое для уравнения (14).

Введя обозначения

$$L_1\varphi^{n+1} = u^+\varphi_{\bar{x}}^{n+1} + v^+\varphi_{\bar{y}}^{n+1} + \frac{\sigma}{2}\varphi^{n+1} -$$

$$-\frac{\mu}{2}\left((1-R_1^+)\frac{1}{h_1}\left(\varphi_x^{n}-\varphi_{\bar{x}}^{n+1}\right)+\right.$$

$$\left.+(1-R_2^+)\frac{1}{h_2}\left(\varphi_y^{n}-\varphi_{\bar{y}}^{n+1}\right)\right),$$

$$L_2\varphi^{n+1} = u^-\varphi_x^{n+1} + v^-\varphi_y^{n+1} + \frac{\sigma}{2}\varphi^{n+1} -$$

$$-\frac{\mu}{2}\left((1+R_1^-)\frac{1}{h_1}\left(\varphi_x^{n+1}-\varphi_{\bar{x}}^{n}\right)+\right.$$

$$\left.+(1+R_2^-)\frac{1}{h_2}\left(\varphi_y^{n+1}-\varphi_{\bar{y}}^{n}\right)\right),$$

для реализации явной двухшаговой схемы φрасщепления (13), (14) можно воспользоваться следующим алгоритмом [17]:

$$\overset{1}{\varphi}{}_t^n + L_1\overset{1}{\varphi}{}^{n+1} - \frac{1}{2}f^n = 0,\ \overset{1}{\varphi}{}^n = \varphi^n = \varphi(x,y,t_n),\ (15)$$

$$\overset{2}{\varphi}{}_t^n + L_2\overset{2}{\varphi}{}^{n+1} - \frac{1}{2}f^n = 0,\ \overset{2}{\varphi}{}^n = \overset{1}{\varphi}{}^{n+1}, \varphi^{n+1} = \overset{2}{\varphi}{}^{n+1}.\ (16)$$

Параллельные алгоритмы

Разностные схемы содержат естественный параллелизм, который может быть использован при их реализации на *GPU*. Суть подхода состоит в том, что к схемам бегущего счета (10), (11) и (15), (16), алгоритмически представимым в виде двух вложенных циклов в которых для каждого узла сетки последовательно проводятся вычисления, возможно применение процедуры скашивания (*loop skewing*) [18]. После этого вычисления, проводимые во внутреннем цикле, становятся независимыми.

Разностные схемы бегущего счета позволяют рассчитывать решение в узлах сетки рекуррентно через известные значения сеточной функции в соседних узлах. При варьировании размеров элементов геометрической декомпозиции алгоритмические свойства не изменяются. Отмеченные свойства положены в основу параллельных алгоритмов для систем с общей памятью, в частности *GPU*. К недостаткам таких параллельных алгоритмов можно отнести неполное задействование вычислительных ресурсов.

Алгоритм решения двумерных задач для графических процессоров. Графические процессоры (*GPU*) – высокопродуктивные параллельные вычислительные системы, имеющие ряд архитектурных особенностей, которые необходимо учитывать при разработке параллельных алгоритмов.

В рамках *GPU* блоки программного кода, именуемые *GPU*-ядрами (*kernel*), исполняются в массово-многопоточной среде, где каждый поток – код, исполняемый последовательно и обрабатывающий элементы данных согласно своему номеру. Количество потоков, исполняющих *GPU-kernel*, обычно превышает количество потоков, которое *GPU* может исполнить параллельно. Это приводит к невозможности проводить глобальную синхронизацию их исполнения и к необходимости декомпозиции алгоритма на фрагменты, которые возможно выполнить параллельно без этой операции. При этом, однако, потоки могут организовываться в группы, в рамках которых реализована операция барьерной синхронизации.

При распараллеливании рассматриваемых вычислительных схем на *GPU* также возникает необходимость учета специфики организации памяти современных графических процессоров – разделения ее на сравнительно медленную глобальную и существенно более быструю локальную, для эффективного использования ко-

торой необходима совместная работа групп потоков.

Без ограничения общности рассмотрим алгоритм для разностной схемы (15), распараллеливающий один ее шаг и применяемый как к двух-, так и к четырехшаговым схемам.

1. Множество внутренних узлов $(x_i, y_j) \in \omega_h$ двумерной сетки $\overline{\omega}_h$ разбивается на квадратные, кроме, возможно, граничных, блоки фиксированного размера $L \times L$, где $L \geq 1$ – целое.

2. Пусть $M = \max(N_1 - 1, N_2 - 1)$, $N = \min(N_1 - 1, N_2 - 1)$, тогда на каждом из $\lfloor N/L \rfloor + \lfloor M/L \rfloor - 1$ шагов ($\lfloor x \rfloor$ – операция округления вещественного числа x до меньшего целого) выполняются независимые вычисления по схеме (15) в от единицы до $\lfloor N/L \rfloor$ блоках узлов сетки. При $N_2 \geq N_1$ координаты узлов блоков лежат в диапазоне от $(iL+1,\ (j-i)L+1)$ до $((i+1)L,\ (j-i+1)L)$ включительно, где $j = 0,...,\lfloor N/L \rfloor + \lfloor M/L \rfloor - 2$ – номер шага, $i = 0,...,\lfloor N/L \rfloor - 1$ – номер блока. В случае когда $N_1 > N_2$, координаты узлов блоков будут лежать в диапазоне от $((j-i)L+1, iL+1)$ до $((j-i+1)L, (i+1)L)$. Вычисления проводятся, если блок находится в пределах сетки, чем объясняется разное количество обрабатываемых блоков в зависимости от номера шага. На каждом шаге управляющая программа, выполняемая на центральном процессоре, задает входные параметры и запускает на *GPU* соответствующее ядро (*kernel*).

3. *GPU-kernel*, отвечающий за исполнение одного шага вычислений в п. 2, обрабатывает данные построчно: каждый поток отвечает за одну строку узлов в пределах блока. Потоки, обрабатывающие строки в одном блоке, объединяются в группу для ускорения вычислений путем использования локальной памяти. Перед началом вычислений все необходимые данные, используя возможности по объединению запросов к памяти, параллельно загружаются в локальную память, а затем – в глобальную.

4. При вычислениях в пределах каждой группы потоков используется схема, аналогичная п. 2. Проводится $j = 0,...,2L-2$ шагов, на каждом из которых поток $i = 0,...,L-1$ обрабатывает узел $(i, j-i)$ блока размера $L \times L$, после чего проводится операция барьерной синхронизации.

Оценим время работы предложенного алгоритма (здесь и в дальнейшем без учета процедур инициализации и сбора результатов).

Время, затраченное L-потоками на обработку блока размера $L \times L$, можно оценить как

$$T_1(L) = (5L+16)t_g + (2L-1)(13t_l + t_c + t_b), \quad (17)$$

где t_g и t_l – время исполнения всеми потоками операций чтения или записи в глобальную и локальную память соответственно, t_c – время обработки данных в одном узле сетки, t_b – время исполнения операции барьерной синхронизации.

В случае когда локальная память не используется, оценка (17) принимает вид

$$T_{1g}(L) = (2L-1)(13t_g + t_c + t_b).$$

Пусть *GPU* позволяет параллельно исполнять bL потоков, где b – известное положительное целое. Тогда общее время работы можно оценить как

$$T(N,M,L) = \left(2\sum_{i=1}^{\lfloor N/L \rfloor}\left(\left\lfloor \frac{i}{b} \right\rfloor + 1\right) + \left(\left\lfloor \frac{M}{L} \right\rfloor - \left\lfloor \frac{N}{L} \right\rfloor + 1\right)\left(\left\lfloor \frac{N/L}{b} \right\rfloor + 1\right)\right) T_1(L). \quad (18)$$

Оценка (18) упрощается

$$T(N,L) = \left(2\lfloor N/L \rfloor - 1\right)\left((5L+16)t_g + (2L-1)(t_c + t_b)\right) \quad (19)$$

при следующих допущениях: время доступа к локальной памяти существенно меньше времени доступа к глобальной, и тогда значением t_l можно пренебречь; *GPU* исполняет все созданные потоки одновременно, т.е. $b > \lfloor N/L \rfloor$; $M = N$.

Функция $T(N,L)$ при $L = [1,...,N]$ имеет один максимум, а ее минимум достигается при

$L=1$ или $L=N$. Локальный минимум при $N>13$ будет достигаться в точке $L=N$ и равняться $(5N+4)t_g+(2N-1)(t_c+t_b)$. Так как значение L ограничено количеством вычислительных ресурсов видеокарты, а условие $N>13$ всегда выполняется для задач, требующих распараллеливания, наилучшей стратегией выбора L будет его максимально возможное увеличение.

Алгоритмы решения трехмерных задач на GPU. В случае таких задач в сеточной области

$$\overline{\omega}_h=\omega_h\cup\gamma_h=\{(:x_i=ih_1,\ i=\overline{0,N_1};y_j=jh_2,\ j=\overline{0,N_2};z_j=kh_3,k=\overline{0,N_3};h_\alpha=l_\alpha/N_\alpha\}$$

предлагается два алгоритма организации вычислений:

- разностная схема рассматривается как три вложенных цикла, внутренние два из которых могут быть распараллелены, используя алгоритмы для двумерных задач (алгоритм 1);
- разностная схема рассматривается как три вложенных цикла, к двум из которых применяется операция скашивания (алгоритм 2).

Пусть $K=\max(N_1-1,N_2-1,N_3-1)$, $N=\min(N_1-1,N_2-1,N_3-1)$,

$$M=\begin{cases}N_1-1,(N_2\ge N_1\ge N_3)\vee(N_3\ge N_1\ge N_2),\\ N_2-1,(N_1\ge N_2\ge N_3)\vee(N_3\ge N_2\ge N_1),\\ N_3-1,(N_1\ge N_3\ge N_2)\vee(N_2\ge N_3\ge N_1).\end{cases}$$

Тогда, в *первом* случае, алгоритм для двумерных задач модифицируется следующим образом:

- для трехмерной сетки размера $N\times M\times K$, единицей вычислений в п. 2 алгоритма решения двумерных задач будут блоки размера $L\times L\times K$;
- вычисления в п. 4 проводятся последовательно для слоев размера $L\times L$ трехмерного блока узлов сетки размера $L\times L\times K$ с локальными координатами узлов от $(0,0,k)$ до $(L-1,L-1,k)$, $k=0,...,K-1$.

Время, затраченное L-потоками на обработку блока размера $L\times L$ в рамках блока размера $L\times L\times K$, в этом случае можно оценить как

$$T_{21}(L)=(15L+48)t_g+(2L-1)(14t_l+t_c+t_b),$$

а общее время работы (если *GPU* позволяет параллельно исполнять bL потоков) как

$$T_{22}(N,M,K,L)=K\left(2\sum_{i=1}^{\lfloor N/L\rfloor}\left(\left\lfloor\frac{i}{b}\right\rfloor+1\right)+\left(\left\lfloor\frac{M}{L}\right\rfloor-\left\lfloor\frac{N}{L}\right\rfloor+1\right)\left(\left\lfloor\frac{N/L}{b}\right\rfloor+1\right)\right)T_{21}(L).\quad(20)$$

Если *GPU* способен исполнить все созданные потоки одновременно, а $M=N=K$, оценка (20) приобретает вид $T_{22}(N,L)=N(2\lfloor N/L\rfloor-1)T_{21}(L)$.

Во *втором* случае, алгоритм, по аналогии с двумерным, принимает следующий вид:

1. Множество внутренних узлов $(x_i,y_jz_k)\in\omega_h$ разбивается на кубические (кроме, возможно, граничных) блоки фиксированного размера $L\times L\times L$.

2. На каждом из $\lfloor N/L\rfloor+\lfloor M/L\rfloor+\lfloor K/L\rfloor-2$ шагов выполняются независимые вычисления от единицы до $\lfloor N/L\rfloor\lfloor M/L\rfloor$ блоках. При $N_1\ge N_2\ge N_3$ координаты узлов блоков будут находиться в диапазоне от $(iL+1,jL+1,(k-i-j)L+1)$ до $((i+1)L,(j+1)L,(k-i-j+1)L)$ включительно, где $k=0,...,\lfloor N/L\rfloor+\lfloor M/L\rfloor+\lfloor K/L\rfloor-3$ – номер шага п. 2, (i,j), $i=0,...,\lfloor N/L\rfloor-1$, $j=0,...,\lfloor M/L\rfloor-1$ – номер блока. Вычисления проводятся только если блок находится в пределах сетки. Для других случаев п. 2 алгоритма может быть легко модифицирован.

3. Для обработки каждого блока запускается группа из L^2 потоков.

4. При вычислениях в пределах каждой группы потоков используется схема, аналогичная п. 2. На каждом шаге $k=0,...,3L-3$, поток (i,j), $i=0,...,L-1$, $j=0,...,L-1$ обрабатывает узел $(i,j,(k-i-j))$ блока размера $L\times L\times L$, после чего проводится операция барьерной синхронизации.

Время, затраченное L^2-потоками на обработку блока размера $L\times L\times L$, можно оценить как

$$T_{31}(L) = (5L+16)t_g + (3L-1)(13t_l + t_c + t_b),$$

а общее время работы, если *GPU* позволяет параллельно исполнять bL^2 потоков, как:

$$\begin{aligned} T_{32}(N,M,K,L) = & \left(2\sum_{i=1}^{\lfloor N/L \rfloor}\left(\left\lfloor \frac{i(i+1)/2}{b} \right\rfloor + 1\right) + \right. \\ & + 2\sum_{i=1}^{\lfloor M/L \rfloor - \lfloor N/L \rfloor}\left(\left\lfloor \frac{i\lfloor N/L \rfloor + \lfloor N/L \rfloor(\lfloor N/L \rfloor + 1)/2}{b} \right\rfloor + 1\right) + \\ & + \sum_{i=1}^{\lfloor N/L \rfloor}\left(\left\lfloor \frac{(\lfloor M/L \rfloor - \lfloor N/L \rfloor)\lfloor N/L \rfloor + \lfloor N/L \rfloor(\lfloor N/L \rfloor + 1)/2) + (1-i)/2}{b} \right\rfloor + 1\right) + \\ & \left. + \left(\left\lfloor \frac{K}{L} \right\rfloor - \left\lfloor \frac{M}{L} \right\rfloor - 1\right)\left(\left\lfloor \frac{\lfloor M/L \rfloor \lfloor N/L \rfloor}{b} \right\rfloor + 1\right)\right) T_{31}(L). \end{aligned} \quad (21)$$

Если *GPU* способен исполнить все созданные потоки одновременно, а $N = M = K$, оценка (21) принимает вид $T_{32}(N,L) = (3\lfloor N/L \rfloor - 1)T_{31}(L)$.

Тестирование быстродействия параллельных алгоритмов для двумерных задач

Алгоритмы тестировались на задачах разного размера на кластере СКІТ-4 Института кибернетики им. В.М. Глушкова НАНУ (12 узлов с 4 *Intel Xeon E*5-2600 и 3 *NVidia Tesla M*2075 на каждом; *CentOS* 5.9, *Cuda toolkit* 4.2, *Open-MPI* 1.6.5).

При решении двумерного уравнения конвекции–диффузии с использованием одного *GPU* для задачи размера (3600×1800) максимальное ускорение составило 9,6. Рассчитанная средняя погрешность оценивания времени обработки одного блока по формуле (17) составила ~12 процентов. Время обработки оценивалось используя информацию о количестве вычислительных ресурсов видеокарты *NVidia Tesla M*2075, а максимальный размер блока ограничивался объемом локальной памяти.

В соответствии с теоретическими оценками, полученное время работы незначительно зависит от размера обрабатываемой сетки и линейно зависит от размера блока. Считая, что на вспомогательные операции при запуске каждого ядра (*kernel*) тратится фиксированное время $t_{kc} = 0{,}5$ мс, построены линейные зависимости для алгоритма с и без использования локальной памяти в виде $t_1(L) = 0{,}1855L + 0{,}1032$ мс и $t_1(L) = 0{,}1664L + 0{,}5408$ мс соответственно. На основе этих зависимостей получены значения коэффициентов, входящих в формулу (17), и оценки времени обработки одного блока согласно этой формуле.

Полученные результаты показывают, что использование локальной памяти приводит к увеличению ускорения при уменьшении размера блока. Это может быть объяснено меньшим количеством объединенных операций доступа к памяти при больших размерах блока, что не учитывается в оценках (17), (18) и приводит к увеличению погрешности оценивания.

Минимальное время работы получено для алгоритма с использованием локальной памяти при $L = 16$, а для алгоритма без ее использования – при $L = 32$. В оптимальной ситуации второй алгоритм на шесть процентов быстрее первого.

Тестирование быстродействия параллельных алгоритмов для трехмерных задач

Алгоритмы с распараллеливанием одного цикла (алгоритм 1) и двух циклов (алгоритм 2) были реализованы и тестировались для уравнения Пуассона и уравнения конвекции–диффузии. Размер блока во всех случаях был равен восьми и все алгоритмы реализованы по схемам, использующим локальную память *GPU*.

Максимальное ускорение, в сравнении с вычислениями на центральном процессоре *CPU*, полученное при решении алгоритмом 1 трехмерных задач, ниже, чем в случае двумерных: 4,72 для уравнения конвекции–диффузии и 4,39 для уравнения Пуассона, тогда как при использовании алгоритма 2 – 28,53 для уравнения Пуассона и 30,66 – для уравнения конвекции–диффузии. Отметим, что увеличение ускорения при решении более сложной задачи.

В случае алгоритма 2 существенный вклад в общее время вычислений вносит операция учета краевых условий. Эта операция была реализована как *GPU-kernel*, каждый поток которого обрабатывает один узел сетки путем получения значения маркера из массива–маски краевых условий и внесения изменений в массив значений искомой функции. Такая операция есть плохо распараллеливаемой на *GPU* в силу того, что количество операций с памятью существенно превышает количество арифметических операций.

Были рассчитаны оценки времени работы (согласно схемам расщепления) по формулам (20), (21) и относительные погрешности оценивания, которые, в целом, находятся в приемлемом диапазоне ~15 процентов. Факторы, влияющие на точность оценок, могут быть выявлены при анализе времени работы *GPU-kernel* на отдельных шагах алгоритмов.

Время обработки одного слоя размера $L \times L$ в рамках блока размера $L \times L \times K$ на шагах п. 2 алгоритма 1 для различных задач приведено на рис. 1.

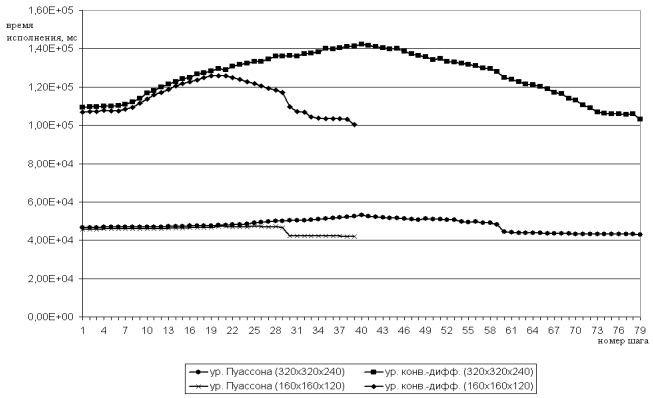

Рис. 1. Время исполнения шагов п. 2 алгоритма 1

В рамках описанных процедур оценки времени работы алгоритмов время обработки одного слоя размера $L \times L$ принято постоянным, исходя из допущения о постоянности времени доступа к глобальной памяти. Однако, как следует из рис. 1, это время не есть постоянным. Временные вариации увеличиваются при решении уравнения конвенции–диффузии, требующего большего количества операций обращения к глобальной памяти в сравнении с решением уравнения Пуассона.

Время исполнения шагов п. 2 алгоритма 2 для задачи размера 320×320×240 при решении уравнения Пуассона приведено на рис. 2.

На рис. 2 наблюдаются участки, соответствующие слагаемым формулы (21): с квадратичным (шаги 1–30 и 81–110), линейным ростом количества обрабатываемых блоков (шаги 31–40 и 71–80), а также неизменным их количеством (шаги 41–70). Однако различные факторы, в частности кэширование, приводят к неточному оцениванию согласно формуле. Кроме того, результаты экспериментов показали, что время обработки одного блока, в соответствии с исходными допущениями, не зависит от размера сетки.

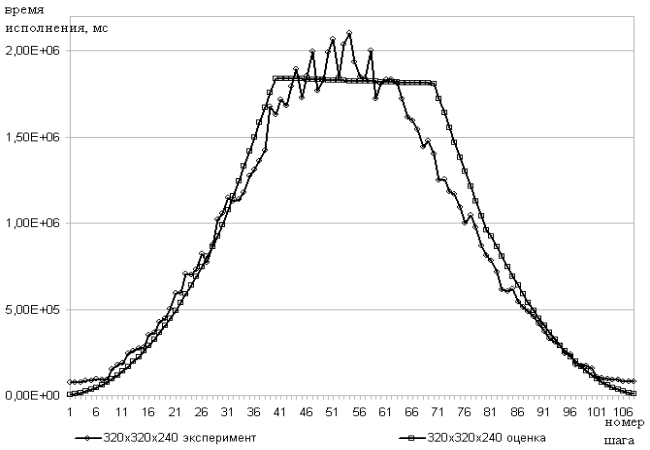

Рис. 2. Время исполнения шагов п. 2 алгоритма 2

Заключение. В работе для разностных схем расщепления с явной организацией вычислений применительно к дву- и трехмерным задачам моделирования распространения загрязнений в атмосфере разработаны параллельные алгоритмы для графических процессоров. Получены теоретические оценки времени работы алгоритмов, позволяющие выбирать оптимальную с помощью быстродействия схему организации вычислений. Результаты тестирования показали высокую эффективность алгоритмов для *GPU*: ускорение для двумерных задач составило ~10, а для трехмерных – ~30.

1. *Марчук Г.И.* Математическое моделирование в проблеме окружающей среды. – М.: Наука, 1982. – 320 с.
2. *Численное* моделирование распространения загрязнения в окружающей среде / М.З. Згуровский, В.В. Скопецкий, В.К. Хрущ и др. – К.: Наук. думка, 1997. – 368 с.
3. *Аргучинцев В.К.*, *Аргучинцева А.В.* Модели и методы для решения задач охраны атмосферы, гидросферы и подстилающей поверхности. – Иркутск: ИГУ, 2001. – 114 с.
4. *Алоян А.Е.*, *Пененко В.В.*, *Козодеров В.В.* Математическое моделирование в проблеме окружающей среды // Современные проблемы вычислительной математики и математического моделирования. – 2005. – Т. 2. – С. 279–351.
5. *Алоян А.Е.* Моделирование динамики и кинетики газовых примесей и аэрозолей в атмосфере. – М.: Наука, 2008. – 415 с.

6. *Основи* математичного моделювання в екології / А.В. Гладкий, І.В. Сергієнко, В.В. Скопецький та ін. – К.: НТУУ «КПІ», 2009. – 240 с.
7. *Марчук Г.И.* Методы расщепления. – М.: Наука, 1988. – 264 с.
8. *Самарский А.А.*, *Вабищевич П.Н.* Численные методы решения задач конвекции–диффузии. – М.: Эдиториал УРСС, 2004. – 248 с.
9. *Zhang Y.*, *Cohen J.*, *Owens J.* Fast tridiagonal solvers on the GPU // PPoPP '10 Proc. of the 15th ACM SIGPLAN Symp. on Principles and Practice of Parallel Program., Bangalor, Ind., Jan. 9–14 2010. – P. 127–136.
10. *Davidson A.*, *Zhang Y.*, *Owens J.* An auto-tuned method for solving large tridiagonal systems on the GPU // Proc. of the 25th IEEE Intern. Parallel and Distributed Processing Symp., May 2011. – P. 956–965,
11. *Goddeke D.*, *Strzodka R.* Cyclic reduction tridiagonal solvers on GPUs applied to mixed precision multigrid // IEEE Trans. Parallel Dist. Syst. – 2010. – **21**. – P. 22–32.
12. *GPGPU*-based ADE-FDTD method for fast electromagnetic field simulation and its estimation / Y. Inoue, M. Unno, S. Aono et al. // Microwave Conf. Proc. (APMC), 2011 Asia-Pacific. – P. 733–736.
13. *Автоматично* налагоджуваний паралельний алгоритм чисельного розв'язання багатовимірної задачі моделювання навколишнього середовища / П.А. Іваненко, А.Ю. Дорошенко, Л.М. Суслова та ін. // Пробл. програм. – 2010. – № 2–3. – С. 202–208.
14. *Черниш Р.І.*, *Тирчак Ю.М.*, *Іваненко П.А.* Побудова паралельного алгоритму чисельного розв'язання багатовимірної задачі моделювання навколишнього середовища // Там же. – 2009. – № 1. – С. 85–91
15. *Скопецкий В.В.*, *Богаенко В.А.* Моделирование прямых и обратных задач распространения загрязнений в воздушной среде с помощью кластерной системы СКИТ // УСиМ. – 2007. – № 5. – С. 86–92.
16. *Самарский А.А.*, *Николаев Е.С.* Методы решения сеточных уравнений. – М.: Наука, 1987. – 588 с.
17. *Гладкий А.В.* Об исследовании алгоритмов расщепления в задачах конвекции–диффузии // Кибернетика и системный анализ. – 2014. – № 4. – С. 76–88.
18. *David F. Bacon*, *Susan L. Graham*, *Oliver J. Sharp.* Compiler transformations for high-performance computing // ACM Comp. Surveys (CSUR), Dec. 1994. – **26**, Issue 4, – P. 345–420.

Поступила 29.09.2014
Тел. для справок: +38 044 526-4167, 067 134-7585, 425-6472 (Киев)
E-mail: *gladky@ukr.net*, *sevab@ukr.net*