ИСПОЛЬЗОВАНИЕ МНОГОПОТОЧНЫХ ТЕХНОЛОГИЙ В РАСЧЁТАХ ЗАДАЧ РАСПРОСТРАНЕНИЯ ЛАЗЕРНЫХ ПУЧКОВ В УСЛОВИЯХ САМОВОЗДЕЙСТВИЯ

Пластун И.Л., Мисюрин А.Г.
Саратовский государственный технический университет

Аннотация

На основе пространственно-временной численной модели распространения лазерного пучка в условиях самовоздействия было проведено сравнение производительности различных технологий параллельных вычислений: CUDA, OpenCL, GLSL, OpenMP. Показано, что реализация расчётной схемы, основанной на методе расщепления и разложении по модам Гаусса–Лагерра, даёт наивысший прирост производительности на основе использования технологии программирования GLSL, реализованной на видеоускорителе ATI Radeon HD4890 от AMD, что в 3 раза превышает по скорости расчёты этой же задачи на базе технологии CUDA на видеоускорителе NVidia.

Ключевые слова: резонансное самовоздействие, распространение лазерного пучка, частотная модуляция, метод расщепления, разложение по модам Гаусса–Лагерра, параллельные вычисления, CUDA, OpenCL, GLSL, NVidia, ATI.

Введение

Как правило, численное моделирование задач нелинейной оптики, связанных с распространением лазерных пучков на значительные расстояния, требует очень объёмных вычислений, особенно в случае построения полной пространственно-временной модели поведения протяжённого лазерного пучка и его воздействия на нелинейно-оптическую среду [1]. Это объясняется вычислительной сложностью численных методов, применимых к подобным задачам, большим количеством расчётных координат, а также тем, что основные вычисляемые параметры, такие как напряжённость электромагнитного поля и поляризация среды, являются комплексными величинами, что вдвое увеличивает объём массивов данных и требует дополнительных действий по работе с комплексной арифметикой. В таких условиях количество элементов в массивах расчётных величин насчитывает десятки тысяч, а время вычислений по одному параметрическому варианту составляет несколько десятков минут. Соответственно, вычисления, необходимые для построения только одного графика зависимостей, требуют обычно до нескольких часов. Подобная ситуация приводит к необходимости поиска различных технологических возможностей ускорения вычислений. Особенно это актуально при исследовании нелинейной динамики рассматриваемой системы, где обычно анализируется и сравнивается до нескольких сотен параметрических вариантов.

Хорошим выходом из сложившейся ситуации может стать использование современных вычислительных технологий, реализующих параллельные вычисления на основе организации многопоточности, за счёт чего время вычислений сокращается в несколько раз. Технологии параллельных вычислений, основанные на использовании графических процессоров, наиболее известной из которых является CUDA (см., например, [2]), начали широко использоваться для масштабных вычислительных задач примерно с 2007–2008 года, причём спектр их применения в оптических исследованиях чрезвычайно разнообразен: от задач распознавания изображений (см., например, [3]) до использования при моделировании распространения лазерных пучков в волноводах на основе метода распространяющегося пучка (Beam Propagation Method) [4]. В приведённых выше примерах реализация на основе технологии CUDA продемонстрировала ускорение вычислений примерно в 10-20 раз по сравнению с однопоточными реализациями. Это даёт основание предполагать существенное сокращение времени вычислений в результате применения мультипоточности в расчётах распространения частотно-модулированных лазерных пучков в нелинейно-оптических резонансных средах.

Помимо CUDA, существует ряд других более новых технологий, реализующих параллельные вычисления на основе видеоускорителей, например, OpenCL и GLSL, которые в настоящее время пока менее популярны среди исследователей, но не менее эффективны. Поэтому наиболее интересным представляется комплексный анализ различных вычислительных технологий, реализующих параллельные вычисления, на предмет оптимальности их применения в задачах нелинейной оптики на примере численного моделирования пространственно-временного поведения лазерного пучка, модулированного по частоте, в условиях резонансного самовоздействия, что также может быть полезно и при решении других ресурсоёмких вычислительных задач.

1. Описание математической модели и расчётной схемы

Используемая нами математическая модель [5] строится на основе совместного решения трёх уравнений: волнового уравнения (1), описывающего пространственно-временную эволюцию распространяющегося лазерного пучка, и уравнений Блоха (2) и (3), выводимых из уравнений для матрицы плотности (см., например, [6]) и описывающих отклик среды, а именно: изменение заселённостей уровней D (2) и поляризации среды P (3):

$$\left[\nabla\times(\nabla\times)+\frac{1}{c^2}\frac{\partial^2}{\partial t^2}\right]\vec{E}(\vec{r},t)=-\frac{4\pi}{c^2}\frac{\partial^2}{\partial t^2}\vec{P}(\vec{r},t)\,; \qquad (1)$$

$$\frac{\partial D}{\partial t}=-\gamma\left[D-1+i\left(E^{*}\,P-E\,P^{*}\right)\right]; \qquad (2)$$

$$\frac{\partial P}{\partial t}=-(\Gamma+i\Delta)P-\frac{i}{2}\Gamma DE\,. \qquad (3)$$

Для подобных задач традиционным является использование приближения медленно меняющихся амплитуд [6], применимого в случае, когда волна проходит расстояние много больше её длины. Возможность применения этого приближения обусловлена тем, что, как нами уже отмечалось в [7], в исходном состоянии среда является слабо нелинейной и слабо поглощающей и амплитуды волн будут изменяться на малую величину при прохождении волной расстояния порядка её длины, иначе говоря, амплитуды волн будут медленно изменяющимися функциями эволюционной координаты z и времени t, и в уравнении (1) можно принять, что

$$\left|\frac{\partial^2 E(z)}{\partial z^2}\right|<<\left|\frac{k\partial E(z)}{\partial z}\right|,$$

поскольку $z_{\max} >> \lambda$. Это означает, что волновое уравнение (1) можно преобразовать к виду

$$2i\left(\frac{\partial E}{\partial z}+\frac{1}{c}\frac{\partial E}{\partial t}\right)+\left(\frac{\partial^2}{\partial r^2}+\frac{1}{r}\frac{\partial}{\partial r}\right)E-\nu^2 r^2 E=gP \qquad (4)$$

и исследовать поведение огибающей модулированного лазерного сигнала. В уравнениях (2)–(4) g – линейное поглощение на единице длины распространения, γ, Γ – скорости релаксации заселённости и затухания поляризации, соответственно, $D(z,r,t)$ – разность заселённостей, нормированная на её величину в отсутствие насыщения, $E(z,r,t)$, $P(z,r,t)$ – медленно меняющиеся амплитуды электрического поля и поляризации, соответственно, Δ – отстройка несущей частоты от частоты атомного перехода, ν – волноводный параметр.

В уравнениях (2)–(4) все величины нормированы. Единица амплитуды поля соответствует уровню насыщения D=0,5. Продольная координата z измеряется в единицах дифракционной длины пучка, поперечная координата r нормирована на характерный радиус пучка a.

Уравнения (2)–(4) решались при начальных условиях, соответствующих задаче Коши:

$$\begin{aligned}&E(z=0,r,t)=E^0(r,t);\quad E(z,r,t=0)=0;\\&D(z,r,t=0)=1;\quad P(z,r,t=0)=0.\end{aligned} \qquad (5)$$

Были рассмотрены пучки, симметричные относительно оси распространения**,** что соответствует большинству физических экспериментальных задач по распространению лазерных сигналов. Частота пучка на входе в среду гармонически модулировалась по времени, $\omega=\omega_0+\omega_1\sin\Omega t$, где ω_0 – несущая лазерная частота, ω_1 – амплитуда модуляции частоты, Ω – частота модуляции. Профиль пучка на входе в среду брался гауссовым, таким образом, начальная комплексная амплитуда входного поля имела вид:

$$\begin{aligned}&E^0(r,t)=E(0,r,t)=\\&=E_0\exp\left(-\frac{r^2}{2a^2}\right)\exp\left[i\frac{\omega_1}{\Omega}\cos(\Omega\cdot t)\right].\end{aligned} \qquad (6)$$

Начальный радиус пучка a во всех рассматриваемых случаях был взят равным 1. Мы предполагаем, что центральная несущая частота ω_0 равна частоте атомного перехода, таким образом, $\Delta=0$ в (3). В этом случае частота модулированного поля осциллирует симметрично по отношению к точной величине резонанса. Время и частота нормированы на времена релаксации. Для упрощения были взяты равные значения $\gamma=\Gamma=1$.

Амплитуда частотной модуляции бралась порядка $\omega_1=1$. Это означает, что отстройка частоты поля от резонанса составляет одну полуширину линии.

Для решения волнового уравнения (4) была использована неявная разностная схема второго порядка, основанная на итерационном методе расщепления, а точнее – его модификации, известной как метод переменных направлений [8] (называемой также схемой расщепления по направлениям), согласно которому исходный сложный оператор можно представить в виде суммы двух более простых, что в нашем случае предполагает независимое решение уравнения относительно продольной и поперечной координаты.

Вычисление поперечного распределения поля основано на разложении по модам Гаусса–Лагерра. Эти моды являются собственными модами прозрачного световода с параболическим профилем преломления без нелинейности с $\nu\neq 0$.

В общем случае поле распространяющегося пучка представляется в виде разложения по модам Гаусса–Лагерра как

$$E(z,\vec{r},t)=\sum_{m,n}C_n^m(z,t)\phi_{mn}(\vec{r},t)\,, \qquad (7)$$

где $C^m{}_n$ – амплитуды мод, m – азимутальный (угловой) индекс, n – радиальный индекс;

$$C_n^m=N_{mn}^{-1}\int_0^{2\pi}d\varphi\rho E(z,\vec{r},t)\phi_{mn}^{*}(\rho,\varphi)\,, \qquad (8)$$

$$\vec{r}=(\rho,\varphi)\,,$$

где ρ,φ – радиальная и угловая поперечные координаты, $N_{mn}=n!/(n+m)!$

Набор мод Лагерра-Гаусса:

$$\begin{aligned}&\phi_{mn}(\rho,\varphi)=L_n^{|m|}(\eta(z)\rho^2)(\eta\rho^2)^{\frac{|m|}{2}}\times,\\&\times\exp(-R(z)\rho^2/2+im\varphi),\end{aligned} \qquad (9)$$

где $L_n{}^m$ – полиномы Лагерра, $R(z)=\eta(z)+i\xi(z)$ – комплексный параметр пучка, где η определяет обратную ширину пучка, ξ – кривизну волнового фронта.

Координаты дискретной четырёхмерной расчётной сетки $(z_k, \rho_i, \varphi_j, t_n)$ представлены следующим образом: $t_n = nh/c$, $z_k = kh$ – координаты по времени и по продольной координате, соответственно, ρ_i – i-й радиальный узел, принятый равным нулю у моды Гаусса–Лагерра высшего порядка, $i = 0, \ldots, L+;$ $\varphi_j = j\Delta\varphi$ – j-й угловой (азимутальный) узел, где $\Delta\varphi = 2\pi/(2M+1)$, $j = 0, \ldots, 2M;$ $l = 0, \ldots, L$ – номер моды Гаусса–Лагерра, $m = -M, \ldots, M$ – порядок вихревого дефекта.

Дискретные расчётные функции, описывающие распространяющееся поле, поляризацию и разность заселённостей среды, обозначим как

$$E_{n,k,i,j} = E(z_k,\rho_i,\varphi_j,t_n),\ P_{n,k,i,j} = P(z_k,\rho_i,\varphi_j,t_n),$$
$$D_{n,k,i,j} = D(z_k,\rho_i,\varphi_j,t_n),$$

а через $C_{n,k,l,m}$ и $F_{n,k,l,m}$ обозначим зависящие от z и t коэффициенты разложения по модам Гаусса–Лагерра для поля и поляризации, соответственно.

Для расчёта поперечного распределения было использовано 30 мод Гаусса–Лагерра ($L = 30$), что является достаточным для описания поведения распространяющегося лазерного пучка. Исследовалось распространение пучков без дефектов ($M = 0$).

При вычислении временной эволюции пучка для удобства была использована система так называемых «бегущих» координат, когда время выражается через продольную координату: $(z,t) \to (z, t - z/c)$, где c – скорость распространения пучка, выраженная в условных единицах: $c = 1$ (в единицах $ka^2/2\pi\gamma$, где a – начальный радиус пучка, γ – скорость релаксации, k – волновое число). Расчёт эволюции по времени происходит по слоям (шагам), причём на каждом новом слое решается задача пространственной эволюции пучка, где в качестве начальных условий используются значения, полученные на предыдущем слое, а приращение по времени происходит по шагам, связанным с шагом по продольной координате z: $t = t + h_z/c$, где h_z – шаг по z.

Таким образом, в разработанной нами итерационной расчётной схеме, согласно методу расщепления [8], каждый эволюционный шаг разбивается на два и включает следующие основные этапы:

1) Решение волнового уравнения (4) для начальных значений поля и поляризации на n-м слое по времени.
2) На следующем эволюционном шаге (слой $n + 1$) решается система уравнений Блоха (2, 3) для D и P методом прямой подстановки, где используются значения поля E, полученные на предыдущем этапе при решении волнового уравнения.
3) Найденные значения поляризации P подставляются в правую часть волнового уравнения (4), после чего оно решается на слое $n + 1$. В результате этого получаем значения поля E, являющиеся начальными для повторного выполнения этапа 1 на следующем эволюционном шаге.

Более подробно расчётный алгоритм может быть представлен в следующем виде:

Шаг 1: Представление поля и поляризации в виде набора мод Гаусса–Лагерра:

$$C_{n,k,l,m} = \sum_{j=0}^{2M}\sum_{i=1}^{L+1} \varphi_{lm}(\rho_i\phi_j) E_{n,k,i,j}\lambda_i\Delta\phi, \qquad (10)$$

$$F_{n,k,l,m} = \sum_{j=0}^{2M}\sum_{i=1}^{L+1} \varphi_{lm}(\rho_i\phi_j) P_{n,k,i,j}\lambda_i\Delta\phi, \qquad (11)$$

где $\varphi_{lm}(\rho,\phi) = K_{lm}\rho^{|m|}L_l^{|m|}(\nu\rho^2)\exp(-\nu\rho^2/2)\dfrac{\exp(im\phi)}{\sqrt{2\pi}}$ – набор мод Гаусса–Лагерра, $L_l^{|m|}(x)$ – полином Лагерра, $K_{l\ m}$ – нормировочный множитель: $K_{lm} = l!/(l+m)!$.

Узлы решётки ρ_i и веса λ_i есть узлы и веса квадратуры Гаусса–Лагерра, задающиеся уравнением:

$$L_{L+1}^0(\nu\rho_i^2) = 0 \qquad (12)$$

и выражением

$$\lambda_i = \left[\sum_{l=0}^{L} |\varphi_{l0}(\rho_i, 0)|^2\right]^{-1}. \qquad (13)$$

В общем виде шаг 1 представляется как замена дискретных значений поля $E_{n,k,i,j}$ и поляризации среды $P_{n,k,i,j}$ на дискретные значения коэффициентов в разложении по модам Гаусса–Лагерра $C_{n,k,l,m}$ (для поля) и $F_{n,k,l,m}$ (для поляризации) в соответствии с (10) и (11). Данная операция имеет высокую вычислительную сложность, однако вычисление коэффициентов не требует последовательного выполнения, поэтому алгоритм был модифицирован для возможности векторизации. Каждый шаг эволюции поля и поляризации преобразуется независимо от остальных для всех точек по поперечной координате (листинг 1).

Для иллюстрации особенностей векторизации используемого алгоритма ниже привёден листинг части программного кода, реализованного на языке C++ с применением библиотеки Thrust. Выбор был сделан на основе выгодных преимуществ, которые даёт эта библиотека. Thrust позволяет писать код таким образом, как будто мы используем стандартную библиотеку шаблонов, в то же время становится доступной вся мощь вычислений на основе технологии CUDA без необходимости писать много специфичного кода для выделения и освобождения видеопамяти, пересылки данных на видеоускоритель и других подобных операций.

Для полученных значений $C_{n,k,l,m}$ и $F_{n,k,l,m}$ решается волновое уравнение (4) в соответствии с методом расщепления по направлениям.

Шаг 2. Решение волнового уравнения (4) на основе разностной схемы:

$$i\frac{C_{n+1,k+1,l,m} - C_{n,k,l,m}}{h} = \\ = b_{l,m}\frac{C_{n+1,k+1,l,m} + C_{n,k,l,m}}{2} + g\frac{F_{n,k+1,l,m} + F_{n,k,l,m}}{4}, \qquad (14)$$

где

$b_{l,m} = (2l + |m| + 1)\,\nu$ – константа распространения.

Листинг 1

```
struct glatrans_direct
{
        const float expm;
        const float dvarPi;
        const float[30][30] gla;
        const float[30] ww;

      glatrans_direct(float _expm,float _dvarPi
,float[30][30] _gla, float[30] _ww)
      {
        expm = _expm;
        dvarPi= _dvarPi;
        gla = _gla;
        ww = _ww;
      }
        __host__ __device__
        std:: vector<float > operator()(std::vector<float>
        source)
        {
          float sum1= 0;
          float sum2[30];
          std::vector<float> result;
          for (int i = 0; i <= 30; i++)
            result [i].push_back(source [i]);

          for (var i = 0; i <= 30; i++)
          {
            sum1 = expm * result [i] * dvarPi;
            result [i] = sum1;
          }
          for (int j = 0; j <= 30; j++)
            sum2[j]= 0;
          for (int j = 1; j <= 30; j++)
            for (int k = 0; k < 30; k++)
              sum2[k] = sum2[k] + gla[j, k] * result [j] *
ww[j];
          for (int j = 0; j <= 30; j++)
            result [j] = sum2[j];
        return result;
        }
    };

    //Вызов функции
    //e-поле, p-поляризация, cc-разложение поля по модам,
ff – разложение поляризации по модам
thrust::host_vector<std::vector<<float>> e, p, cc, ff;

//перенос данных из системной памяти в память GPU
thrust::device_vector< std::vector<<float>> tempE = e;
thrust::device_vector< std::vector<<float>> tempP = p;

//прямое преобразование Гаусса-Лаггера, выполняемое
параллельно для всех элементов
thrust::transform(tempE.begin(),tempE.end(),cc.begin(),
glatrans_direct(expm, dvarPi , gla, ww))
thrust::transform(tempP.begin(),tempP.end(),ff.begin(),
    glatrans_direct(expm, dvarPi , gla, ww))
```

Разностная схема представляет собой итерационный метод, где массив значений поля, распространяющегося вдоль продольной координаты z, вычисляется на основе значений, полученных на предыдущем шаге распространения.

Шаг 3. Обратное преобразование Гаусса–Лагерра, согласно которому получаем истинные значения поля, которые используем при решении уравнений Блоха (2, 3).

Общее выражение для обратного преобразования Гаусса–Лагерра для поля:

$$E_{n,k,i,j} = \sum_{m=-M}^{M} \sum_{l=0}^{L} C_{n,k,l,m} \varphi_{lm}(\rho_i \phi_j) \qquad (15)$$

и для поляризации:

$$P_{n,k,i,j} = \sum_{m=-M}^{M} \sum_{l=0}^{L} F_{n,k,l,m} \varphi_{lm}(\rho_i \phi_j)\,. \qquad (16)$$

Таким образом, на этом шаге получаем выражение для поля по формуле (15) для следующего момента времени t_{n+1}., которое подставляем в уравнения Блоха (2, 3).

С точки зрения векторизации алгоритма обратное преобразование Гаусса–Лаггера основано на аналогичных программных шагах, где последовательность действий меняется на обратную, что позволяет реализовать параллельные вычисления на основе идентичной схемы векторизации.

Шаг 4: Решение системы уравнений Блоха (2, 3) относительно двух переменных: действительной $D_{n+2,k,i,j}$ и комплексной $P_{n+2,k,i,j}$ согласно схеме (17).

В результате решения этих уравнений получаем выражение для поляризации среды в момент времени t_{n+2}, которое подставляется в правую часть волнового уравнения на следующем шаге решения.

$$\begin{aligned}
&c\frac{D_{n+2,k,i,j} - D_{n,k,i,j}}{2h} = \\
&= -\gamma\left[\frac{D_{n+2,k,i,j} + D_{n,k,i,j}}{2} - 1 - \right. \\
&\left. -2\,\mathrm{Im}\left(E^{*}_{n+1,k,i,j}\frac{P_{n+2,k,i,j} + P_{n,k,i,j}}{4}\right)\right]; \\
&c\frac{P_{n+2,k,i,j} - P_{n,k,i,j}}{2h} = \\
&= -\left(\Gamma + i\Delta\right)\frac{P_{n+2,k,i,j} + P_{n,k,i,j}}{2} - \\
&-i\frac{\Gamma}{2}E_{n+1,k,i,j}\frac{D_{n+2,k,i,j} + D_{n,k,i,j}}{2}.
\end{aligned} \qquad (17)$$

К этой части алгоритма также были применены возможности векторизации.

По известному массиву значений распространяющегося поля E на каждом шаге по продольной координате z мы находим отклик среды в виде массивов значений разности заселённостей D и поляризации среды P (листинг 2).

Шаг 5: Функцию, описывающую поляризацию среды в момент времени t_{n+2}, $P_{n+2,k,i,j}$, преобразуем к виду набора мод Гаусса–Лагерра $F_{n+2,k,i,j}$, для чего используем выражение (11). Данный шаг выполняется по алгоритму, приведённому в листинге 1.

Листинг 2

```
struct evolDP
{
        __host__ __device__
        thrust::tuple<float, float, float> opera-
        tor()(thrust::tuple<float, float, float> v)
        {
        float gamma = 1;
        float bigGamma = 1;
        float c = 1;
        float h=0.001;

        float ein = get<0>(v);
        float pin = get<1>(v);
        float din = get<2>(v);

        float c2H = c / (2 * h);
          float cbgbd2 = bigGamma / 2.0;
          float cig4E = -bigGamma / 4 * ein;
          float cd = c2H + cbgbd2;
          float ca = cig4E / cd;
          float cf = ((c2H - cbgbd2) * pin + cig4E * din) /
cd;
          float gam2 = gamma / 2.0;

        float dout = ((c2H - gam2) * din + gamma + gam-
        ma * (ein* (cf + pin))) / (c2H + gam2 - gamma *
        (ein * ca));

          float pout = ca * din + cf;

        return thrust::make_tuple(ein, pout, dout);
        }
    };

    //Вызов функции
thrust::device_vector<float> e, p, d;
thrust::transform(thrust::make_zip_iterator(thrust::make_tupl
e(e.begin(), p.begin(), d.begin())),
    thrust::make_zip_iterator(thrust::make_tuple(e.end(),
p.end(), d.end())),evolDP());
```

Шаг 6: Подставляем значения поляризации, представленные в виде мод Гаусса–Лагерра $F_{n+2,k,l,m}$, в правую часть волнового уравнения (4) и решаем его согласно разностной схеме:

$$i\frac{C_{n+2,k+1,l,m}-C_{n+1,k,l,m}}{h}= \\ =b_{l,m}\frac{C_{n+2,k+1,l,m}+C_{n+1,k,l,m}}{2}+ \\ +g\frac{F_{n+2,k+1,l,m}+F_{n+2,k,l,m}}{4}. \qquad (18)$$

В результате решения получаем поле в момент времени t_{n+2} в виде набора мод Гаусса–Лагерра.

Шаг 7: Получаем истинные значения поля E в момент времени t_{n+2} согласно выражению (15) (обратное преобразование Гаусса–Лагерра). В данном случае используется та же схема векторизации, что и на шаге 3.

Длина распространения берётся конечной: $z_{max}=Nh$, причём особенностью алгоритма является то, что для каждой поперечной точки считается и хранится весь проход по координате z с теми условиями, которые сложились на данном этапе распространения. Иначе говоря, мы сразу имеем некоторую матрицу значений, которая обновляется по мере прохождения каждого итерационного шага. Таким образом, в памяти компьютера постоянно хранятся четыре массива, каждый из которых имеет размерность $(L+1)(2M+1)N$. Данная расчётная схема имеет второй порядок по h. Устойчивость схемы в режиме распространения непрерывного лазерного излучения проверялась на основе примеров, решаемых аналитически, и путём сравнения с результатами, полученными на основе других численных методов.

Особенностью разработанной нами расчётной схемы является то, что при выборе шагов расчётной сетки вместо традиционной для подобных методов схемы «квадрат» используется схема «параллелограмм», за счёт чего на двух подшагах расщепления (слой $n+1$ (14) и слой $n+2$ (18)) решаются одинаковые уравнения, чем обеспечивается высокая степень устойчивости. Это объясняется тем, что при постоянном входном пучке не возникают высокочастотные колебания (осцилляции) нефизической природы, нарушающие устойчивость метода расщепления при других способах построения схемы второго порядка. Кроме того, благодаря представлению поперечного профиля поля в виде суммы мод Гаусса–Лагерра, имеется возможность учёта дифракционной расходимости лазерного пучка.

Анализ скорости вычислений, выполняемых средствами различных мультипоточных технологий, проводился на основе задачи исследования нелинейно-динамических свойств системы: лазерный пучок, модулированный по частоте, распространяющийся в нелинейно-оптической двухуровневой среде в условиях резонансного самовоздействия. Необходимо отметить, что в условиях резонансного самовоздействия, возникающего при определённой достаточно высокой интенсивности лазерного сигнала, параметры среды (а именно коэффициент поглощения и показатель преломления) начинают существенно изменяться, причём это изменение происходит неравномерно по сечению пучка. Это, в свою очередь, вызывает изменения условий распространения лазерного пучка и его параметров, что можно рассматривать как некоторую систему с обратной связью.

Физические результаты решения этой задачи были представлены нами в статье [7].

Исследовалось пространственно-временное распределение интенсивности пучка на выходе из среды $I(z,r,t,\omega)$, представляемой как квадрат модуля комплексной амплитуды поля E: $I(z,\rho,\varphi,t,\omega)=|E(z,\rho,\varphi,t,\omega)|^2$, а также динамика отклика среды, представляемая в виде проекций фазового пространства системы на плоскость «поляризация среды P – разность заселённостей D». Кроме того, рассчитывался спектр мощности по реализации

(временной зависимости) интенсивности на оси пучка на выходе из среды.

Длина распространения составляла шесть дифракционных длин пучка: $Z = z_{\max} = 6L_d$, что является достаточным для наблюдения изменений в поведении лазерного пучка. Линейное поглощение в рассматриваемых случаях принималось равным $g = 1$, и, таким образом, интенсивность на выходе из среды оказывалась небольшой из-за поглощения и дифракционного расплывания.

Работы по векторизации алгоритма можно считать удачными, так как удалось оптимизировать 1, 3, 4, 5, 7 шаги алгоритма.

Наибольшей вычислительной сложностью в алгоритме является вычисление поперечного распределения поля, основанное на разложении по модам Гаусса–Лагерра. По предварительным оценкам, расчёт прямого и обратного преобразования Гаусса–Лаггера занимает 85 % машинного времени. Количество операций, необходимых для расчёта данной части алгоритма, обратно пропорционально количеству потоковых процессоров. Распараллеливание этой части алгоритма позволило ускорить расчёт в 10–15 раз, в зависимости от выбранной архитектуры и технологии параллельных вычислений.

С некоторым приближением вычислительную сложность общего алгоритма можно оценивать по сложности вычисления мод Гаусса–Лагерра. Объём вычислений, необходимых для расчёта мод при последовательной реализации алгоритма, можно оценить как $O(n^2)$, где n – количество точек по продольной координате. Время же, затраченное на решение системы с тем же набором данных при использовании параллельных вычислений, стремится к $O(n)$, и это зависит от архитектуры и количества сопроцессоров в вычислительной системе.

2. Технологии параллельных вычислений

Конечной целью исследования нелинейной динамики подобной системы является построение карты динамических режимов. Проблема достижения результата заключается в больших объёмах вычислений для получения выходных данных. Требуется около 50000 итераций расчёта системы при разных входных параметрах, которые займут очень много времени. Используя идею векторизации алгоритма и реализации его на видеоускорителе на платформе CUDA [4], был проведён анализ и подбор наиболее оптимальной программно-аппаратной мультипоточной технологии для решения вычислительной задачи на основе описанной выше расчётной схемы. Для отображения результатов расчёта была разработана специальная сервисная программа на языке C# (рис. 1).

Необходимо отметить, что организация мультипоточных вычислений строится на основе реализации алгоритма в некоторой программной среде, ориентированной на распараллеливание вычислений, и последующем выполнении расчётов на некоторой мультипроцессорной системе.

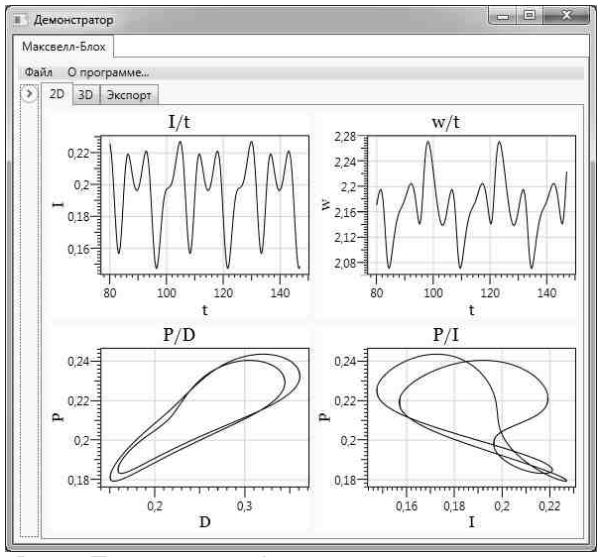

Рис. 1. Программа отображения результатов расчёта

Наиболее часто в качестве подобной аппаратной платформы используются графические процессоры как относительно дешёвые и доступные, а главное – высокопроизводительные устройства. Таким образом, для того чтобы выбрать наиболее оптимальную технологию мультипоточных вычислений, необходимо проанализировать различные сочетания программных технологий и аппаратных платформ видеоускорителей.

Для программной реализации разработанного алгоритма (10)–(18) использовались средства современных систем программирования на базе языков C# и C++ (OpenMP, OpenCL, CUDA, GLSL), рассчитанные на работу в четырёх различных технологиях параллельных вычислений. Опишем более подробно каждый из четырёх комплексов, позволяющих наиболее эффективно использовать параллельные методики на графических ускорителях.

Наиболее популярным средством организации параллельных вычислений является CUDA (Compute Unified Device Architecture) – программно-аппаратная архитектура, позволяющая производить вычисления с использованием графических процессоров NVidia, поддерживающих технологию GPGPU (произвольных вычислений на видеокартах). Впервые эта архитектура появилась на рынке с выходом чипа NVidia восьмого поколения – G80 и присутствует во всех последующих сериях графических чипов, которые используются в семействах ускорителей GeForce, Quadro и NVidia Tesla.

Для понимания работы с платформой обычно рассматривается некоторая обобщённая логическая архитектура организации мультипоточных вычислений (рис. 2).

Основным исполнительным элементом в GPU являются так называемые потоковые процессоры (Streaming Processor – SP) или CUDA-ядра (CUDA core). Это скалярные процессоры, устроенные гораздо проще по сравнению с центральным процессором (CPU). Каждые 8 потоковых процессоров

объединяются в потоковые мультипроцессорные блоки, или мультипроцессоры (Streaming Multiprocessor). Также в состав мультипроцессора входят два дополнительных вычислительных элемента (Special Function Unit), предназначенных для интерполяции и некоторых других операций, специфичных для обработки графики. Все потоковые процессоры, входящие в состав мультипроцессора, имеют общий кэш, предназначенный для констант и текстур, разделяемую (или общую – shared) память и регистры.

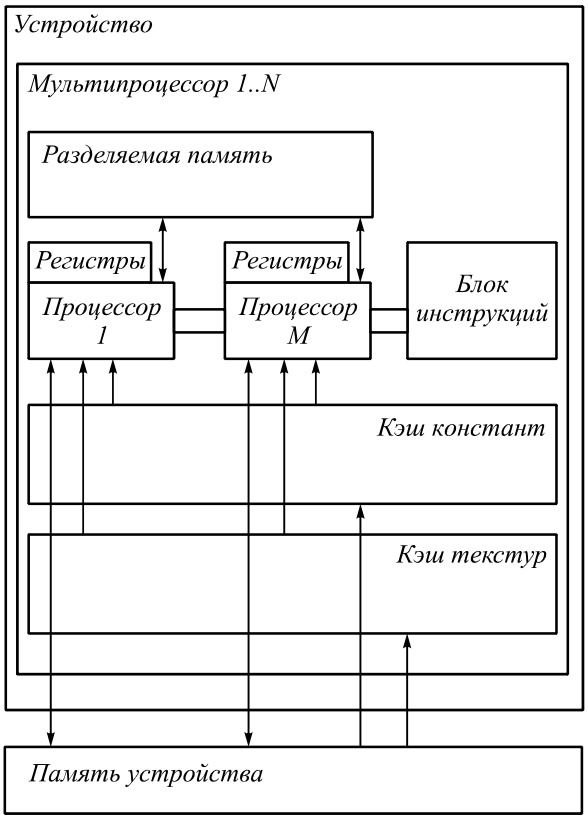

Рис. 2. Логическая архитектура организации мультипоточных вычислений (на примере платформы CUDA)

Вторая технология, используемая нами, – OpenCL (Open Computing Language – открытый язык вычислений). Она представляет собой фреймворк для написания компьютерных программ, связанных с параллельными вычислениями на различных графических (GPU) и центральных процессорах (CPU). В архитектуру OpenCL входят язык программирования, который базируется на стандарте C99, и прикладной программный интерфейс API (Advanced Program Interface). OpenCL обеспечивает параллелизм как на уровне инструкций, так и на уровне данных и является реализацией техники GPGPU.

Технология OpenCL очень похожа на CUDA, так как NVidia принимала активное участие в разработке стандарта OpenCL, и, также как и CUDA, соответствует логической архитектуре, представленной на рис. 2.

Третья технология – GLSL (OpenGL Shading Language) представляет собой язык высокого уровня, разработанный в 2009 году для программирования шейдеров. Ше́йдер (англ. Shader) – это программа для одной из ступеней графического конвейера, используемая в трёхмерной графике для определения окончательных параметров объекта или изображения. Она может включать в себя произвольной сложности описание поглощения и рассеяния света, наложения текстуры, отражение и преломление, затенение, смещение поверхности и эффекты пост-обработки.

Четвёртая технология – OpenMP (Open Multi-Processing) – это открытый стандарт для распараллеливания программ на языках Си, Си++ и Фортран. Описывает совокупность директив компилятора, библиотечных процедур и переменных окружения, которые предназначены для программирования многопоточных приложений на многопроцессорных системах с общей памятью.

Предполагается, что «главный» (master) поток создаёт набор подчинённых (slave) потоков, выполняющихся параллельно на машине с несколькими процессорами (количество процессоров не обязательно должно быть больше или равно количеству потоков).

Для сравнения различных технологий был рассчитан следующий тестовый пример: число радиальных узлов в разложении по модам Гаусса–Лагерра $L = 30$, число азимутальных узлов $M = 0$ (пучок полностью осесимметричен), число точек по продольной координате z (и, соответственно, отсчётов по времени) бралось различным: $N = 200$, 400, 800 и 1600. При этом длина распространения во всех случаях оставалась неизменной – шесть дифракционных длин пучка. Таким образом, значение N, по сути, является параметром аппроксимации, или параметром степени точности расчёта. Пример вычислений с различной степенью точности продемонстрирован на рис. 3. Значения физических параметров исследуемой системы взяты такими же, как в статье [7].

Как видно из рисунка, использование большой степени точности обязательно для решаемой задачи. Выходными данными являются два массива по 24 тысячи элементов в каждом и один массив из 12 тысяч элементов, представляющих собой эволюцию во времени и пространстве интенсивности лазерного пучка I, поляризации среды P и разности заселённостей энергетических уровней D, соответственно.

3. Результаты сравнительного анализа

Тестирование скорости мультипоточной реализации нашей расчётной схемы проводилось на трёх различных устройствах – NVidia Quadro FX5600, ATI Radeon HD4890, Intel Core2Duo E8200, поскольку представленные технологии параллельных вычислений оптимизированы под различные аппаратные вычислительные системы.

На первом этапе вычисления производились на видеоускорителе NVidia Quadro FX5600, в тесте принимали участие реализации на OpenCL, CUDA, GLSL, версия драйвера видеокарты – 197.45. Результаты представлены в табл. 1.

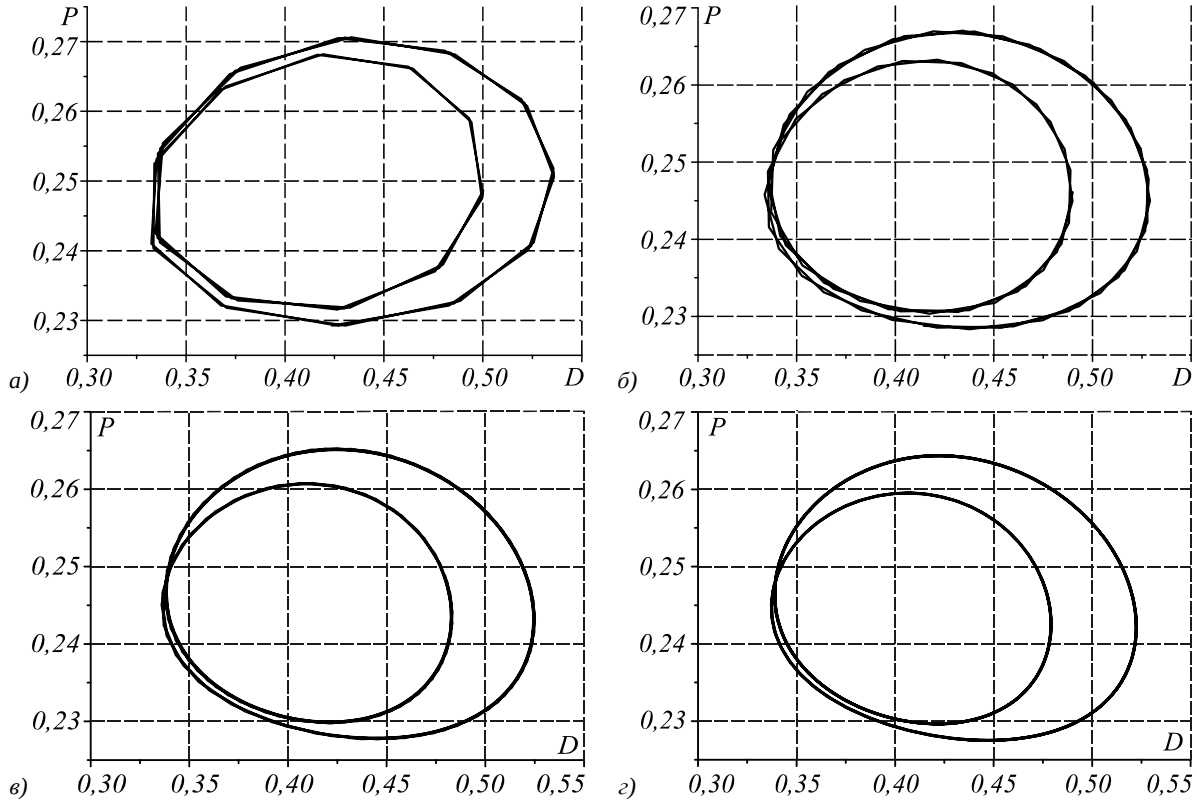

Рис. 3. Проекции фазового пространства системы на плоскость «поляризация среды P – разность заселённостей D», рассчитанные при разных значениях точности N: N = 200 (а), N = 400 (б), N = 800 (в), N = 1600

Таблица. 1. Сравнение производительности различных технологий на видеоускорителе NVidia Quadro FX5600

Точность	OpenCL, с	CUDA, с	GLSL, с
200	57	50	61
400	119	105	126
800	244	216	259
1600	512	453	544

Единицей измерения производительности является время, необходимое для завершения вычислений по одному параметрическому варианту.

Из таблицы видно, что CUDA опережает OpenCL по производительности приблизительно на 13%. При этом, если оценивать теоретически возможную производительность реализации этой задачи для данной архитектуры, реализация на CUDA достигает её. (В [9] говорится о том, что производительность ядра OpenCL проигрывает CUDA от 13% до 63%.)

Несмотря на то, что тесты проводились на видеоускорителе серии Quadro, понятно, что GeForce 8800 GTS или GeForce 250 GTS дадут схожие результаты, поскольку все три видеокарты основаны на чипе G92.

Во втором тесте использовался видеоускоритель от ATI – Radeon HD4890 и технологии OpenCL и GLSL, поскольку технология CUDA может использоваться только на видеокартах от NVidia (результаты тестирования представлены в табл. 2).

OpenCL проигрывает шейдерам на видеоадаптере от AMD, так как вычислительные блоки на них имеют архитектуру VLIW (англ. very long instruction word – «очень длинная машинная команда»), на которую (после оптимизации) могут хорошо лечь многие шейдерные программы, но компилятор для кода OpenCL (который является частью драйвера) плохо справляется с оптимизацией.

Таблица 2. Сравнение производительности различных технологий на видеоускорителе ATI Radeon HD4890

Точность	OpenCL, с	GLSL, с
200	201	17
400	431	36
800	885	76
1600	1858	159

Также этот весьма скромный результат может быть вызван тем, что видеоускорители от AMD не поддерживают локальную память на физическом уровне, а отображают область локальной памяти на глобальную.

На третьем этапе тестирования использовался центральный процессор (CPU) от компании Intel – Intel Core2Duo E8200 и технологии OpenCL и OpenMP. Код с использованием OpenMP был скомпилирован при помощи компиляторов от Intel и Microsoft (табл. 3).

Компания Intel не выпустила своих драйверов для запуска кода OpenCL на центральном процессоре, поэтому был использован ATI Stream SDK версии 2.01.

Код на OpenMP, скомпилированный при помощи MS VC++, имеет практически идентичную производительность с OpenCL.

Таблица 3. Сравнение производительности различных технологий на центральном процессоре Intel Core2Duo E8200

Точность	.Net, с	OpenCL, с	OpenMP	
			MS VC++, с	Intel PC, с
200	816	715	698	523
400	1940	1724	1641	1222
800	7015	6723	6235	4645
1600	26903	24877	22448	16258

Компилятор от Intel поступил не совсем «честно»: он полностью развернул основной цикл программы, повторив его где-то 8000 раз (число точек было задано константой в коде) и получил прирост производительности также благодаря использованию SSE инструкций.

Кроме того, было проведено сравнение данных технологий с технологией .Net, реализующей параллельные вычисления на самом простейшем уровне и дающей минимальный выигрыш в скорости по сравнению с однопоточными реализациями.

На рис. 4 представлены результаты численных экспериментов.

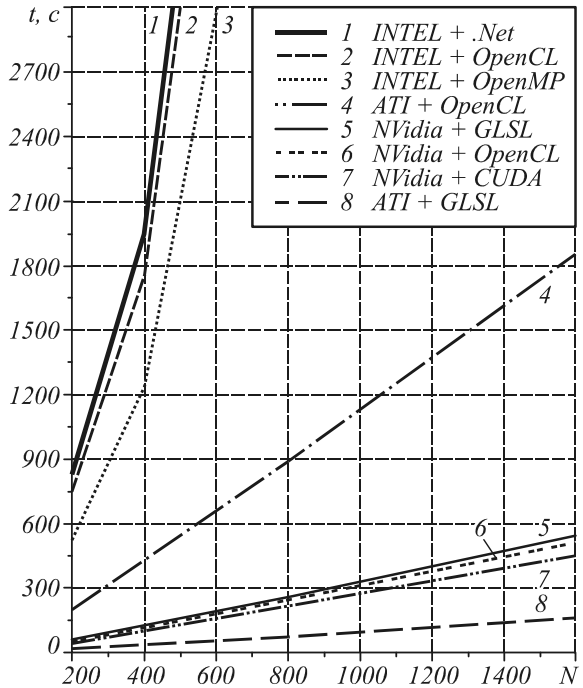

Рис. 4. Сравнительный анализ времени выполнения расчёта на основе различных мультипоточных технологий в зависимости от степени точности N

Видно, что при реализации на центральном процессоре Intel Core2Duo E8200 технология OpenCL показывает результат, практически идентичный с технологией .Net. Технология OpenMP даёт несколько лучший результат, но всё равно производительность существенно ниже, чем на видеоускорителях.

Видеоускорители NVidia дают результаты, примерно идентичные для всех рассмотренных мультипоточных технологий, при этом CUDA, как наиболее оптимизированная под аппаратуру NVidia технология, показывает наилучший результат по производительности.

Наиболее интересным является результат тестирования, проведённого на видеоадаптере от AMD ATI Radeon HD4890.

Технология OpenCL в реализации на ATI существенно проигрывает реализации на NVidia, и это легко объяснимо, поскольку OpenCL представляет собой технологию, очень похожую на CUDA, и даже в какой-то степени надстройку над CUDA, которая, также как и CUDA, ориентирована в большей степени на видеоускорители NVidia и хуже стыкуется с другими типами видеоускорителей.

Существенное отличие реализации алгоритма по технологии GLSL на видеоускорителе ATI от AMD, дающее значительное превышение в скорости (в 3 раза) даже перед самым эффективным и наиболее популярным сочетанием CUDA и NVidia, объясняется тем, что видеокарта ATI имеет более высокую тактовую частоту графических процессоров и большее их количество [10], а также более высокую частоту работы видеопамяти по сравнению с NVidia [11]. Весьма скромные вычислительные результаты реализации технологии OpenCL на этой же видеокарте связаны с тем, что OpenCL, как уже упоминалось, менее адаптирован под ATI, чем под NVidia, а сравнительно новая технология GLSL является кроссплатформенной и одинаково хорошо работает с различными типами видеоускорителей. Кроме того, GLSL – более низкоуровневое средство программирования по сравнению с OpenCL, что даёт более глубокий уровень взаимодействия с аппаратурой.

Необходимо отметить, что на данный момент не до конца доработаны драйверы и случаются такие ситуации, когда что-то есть в стандарте, а использовать в реальном коде нельзя (например, поддержка текстур в программах на OpenCL появилась у видеоускорителей ATI только с серии HD5xxx). Кроме того, очень часто драйверы генерируют не оптимальный код для данной конкретной платформы, и в этом плане, конечно, очевидны пути дальнейшего совершенствования мультипоточных технологий.

Заключение

В заключение сравнительного анализа мультипоточных технологий необходимо подчеркнуть, что при выборе оптимальной технологии реализации параллельных вычислений необходимо учитывать степень совместимости технологии программирования и аппаратуры мультипроцессорной системы.

Стоит отметить, что программы на платформе OpenMP лишены возможности выполняться в параллельных потоках на видеоускорителях из-за ограничений платформы, что объясняет большую разницу с платформой CUDA и OpenCL.

Уже сейчас программы на OpenCL показывают достойную производительность по сравнению с конкурентами и могут успешно использоваться для

параллельных вычислений общего назначения, что подтверждают результаты численных экспериментов по реализации этой технологии на видеоускорителе NVidia.

Наибольшее преимущество связки «видеоускоритель ATI и технология GLSL» обеспечено не столько гибкостью и оптимизацией языка GLSL, сколько большей мощностью видеокарты от ATI по сравнению с NVidia, обеспеченной за счёт большей тактовой частоты GPU, большего количество сопроцессоров, а также высокой частоты видеопамяти [10, 11].

Результаты тестирования разных технологий и инструментов для параллельных вычислений показывают, что для решения уравнений нелинейной оптики оптимальным является сочетание технологии GLSL и видеокарты ATI от AMD. Также хорошо подходит традиционная связка CUDA – графический ускоритель NVidia.

Кроме того, достаточно перспективной является возможность объединения видеоускорителей в вычислительный кластер (NVidia SLI, ATI CrossFire), что, скорее всего, позволит получить ещё более значительный прирост производительности.

Литература

1. **Карамзин, Ю.Н.** Математическое моделирование в нелинейной оптике / Ю.Н. Карамзин, А.П. Сухоруков, В.А. Трофимов. – М.: Изд-во МГУ, 1989. – 156 с.
2. **Sanders, J.** CUDA by Example. An Introduction to General-Purpose GPU Programming / J. Sanders, E. Kandrot. – Addison-Wesley Professional, 2010. – 312 p.
3. **Изотов, П.Ю.** Технология реализации нейросетевого алгоритма в среде CUDA на примере распознавания рукописных цифр / П.Ю. Изотов, С.В. Суханов, Д.Л. Головашкин // Компьютерная оптика. – 2010. – Т. 34, № 2. – С. 243-251.
4. **Алексеев, В.А.** Векторизация метода распространяющегося пучка и его реализация по технологии CUDA / В.А. Алексеев, Д.Л. Головашкин // Компьютерная оптика. – 2010. – Т. 34, № 2. – С. 225-230.
5. **Пластун, И.Л.** Исследование влияния нестационарных когерентных эффектов и резонансного самовоздействия на характеристики лазерного пучка, модулированного по частоте / И.Л. Пластун, В.Л. Дербов // Компьютерная оптика. – 2009. – Т. 33, № 3. – С. 233-239.
6. **Шен, И.Р.** Принципы нелинейной оптики / И.Р. Шен; пер. с англ. – М.: Наука, 1989. – 560 с.
7. **Пластун, И.Л.** Оптическое деление частоты при распространении лазерного излучения в среде с насыщением поглощения и дисперсии / И.Л. Пластун, А.Г. Мисюрин // Компьютерная оптика. – 2010. – Т. 34, № 3. – С. 292-296.
8. **Марчук, Г.И.** Методы вычислительной математики / Г.И. Марчук. – М.: Наука, 1989. – 608 с.
9. **Karimi, K.** A Performance Comparison of CUDA and OpenCL / K. Karimi, N.G. Dickson, F. Hamze. – D-Wave Systems Inc. 100-4401 Still Creek Drive Burnaby, British Columbia Canada, 2011. – 10 p.
10. **Воробьёв, А.** 3D-ускоритель от AMD:ATI RADEON HD 4890 1024MB [Электронный ресурс] / А. Воробьёв, А. Берилло // IXBT[сайт].[2009].URL: www.ixbt.com/video3/rv790.shtml (дата обращения: 10.06.2012).
11. Технические характеристики NVidia Quadro SDI [Электронный ресурс] // NVidia [сайт]. [2010]. URL: http://www.nvidia.ru/page/qfx_4000sdi_techspec.html (дата обращения: 10.06.2012).

References

1. **Karamzin, Y.N.** Mathematical modeling in nonlinear optics / Y.N. Karamzin, A.P. Sukhorukov, V.A. Trofimov. – Moscow, Moscow State University Publisher, 1989. – 156 p. – (In Russian)
2. **Sanders, J.** CUDA by Example. An Introduction to General-Purpose GPU Programming / J. Sanders, E. Kandrot - Addison-Wesley Professional, 2010. – 312 p.
3. **Izotov, P.Y.** Technology of implementation of neural network algorithm in CUDA environment at the example of handwritten digits recognition / P.Y. Izotov, S.V. Sukhanov, D.L. Golovashkin // Computer Optics. – 2010. – V. 34, N 2. – P. 243-251. – (In Russian).
4. .**Alekseev, V.A.** Vectorization of the beam propagation method using CUDA technology / V.A. Alekseev, D.L. Golovashkin // Computer Optics. – 2010. – V. 34, N 2. – P. 225-230. – (In Russian).
5. .**Plastun, I.L.** Investigation of the nonstationary coherent effects and resonant self-action influence on the characteristics of a frequency-modulated laser beam / I.L. Plastun, V.L. Derbov // Computer Optics. – 2009. – V.33, N 3. – P. 233-239. – (In Russian).
6. **Shen**, **Y.R.** The principles of nonlinear optics / Y.R. Shen. – N.Y.: A Wiley-Interscience Publication, John Wiley&Sons, Inc., 1984.
7. **Plastun, I.L.** Observation of optical frequency division in medium with saturable absorption and dispersion/ I.L. Plastun, A.G. Misurin // Computer Optics. – 2010. – V. 34, N 3. – P. 292-296. – (In Russian)
8. **Marchuk, G.I.** Methods of Computational Mathematics / G.I. Marchuk. – Moscow: Nauka Publisher, 1989. – 608 p. – (In Russian).
9. **Karimi, K.** A Performance Comparison of CUDA and OpenCL / K. Karimi, N.G. Dickson, F. Hamze. – D-Wave Systems Inc. 100-4401 Still Creek Drive Burnaby, British Columbia Canada, 2011. – 10 p.
10. **Vorobiev A., Berillo A.** 3D-accelerator from AMD:ATI RADEON HD 4890 1024MB / A. Vorobiev, A. Berillo // IXBT [site]. [2009]. URL: www.ixbt.com/video3/rv790.shtml (verified at: 10.06.2012).
11. NVidia Quadro SDI // NVidia [2010]. URL: http://www.nvidia.ru/page/qfx_4000sdi_techspec.html (verified at: 10.06.2012).

MULTI-THREADED TECHNOLOGY IN NUMERICAL SIMULATION OF LASER BEAMS PROPAGATION IN A SELF-ACTION CONDITIONS

I.L. Plastun, A.G. Misurin
Saratov State Technical University

Abstract

On the basis of spatio-temporal numerical model of the frequency - modulated cw laser beam propagating in resonance self-action conditions the performance of various parallel computing technologies: CUDA, OpenCL, GLSL, OpenMP was compared. It is shown that the numerical scheme based on the method of splitting and decomposition of Gauss-Laguerre modes, gives the highest performance on the basis of the use of programming technique GLSL, realized on the video card ATI Radeon HD4890 from AMD, which is 3 times the speed of calculations of the same problem based on the CUDA technology to NVidia video card.

Key words: resonance self-action, laser beam propagation, frequency modulation, method of splitting, decomposition of Gauss-Laguerre modes, parallel computing, CUDA, OpenCL, GLSL, NVidia, ATI.

Сведения об авторах

Пластун Инна Львовна, 1969 года рождения. В 1991 году с отличием окончила Саратовский государственный университет им. Н.Г.Чернышевского по специальности «Физика». Доктор физико-математических наук по специальностям «Лазерная физика» и «Математическое моделирование» (2011 год), доцент, является доцентом Саратовского государственного технического университета (СГТУ). Область научных интересов: нелинейная оптика, лазерная физика, математическое моделирование физических процессов, параллельные вычисления. Имеет около 80 научных работ, из них – 60 статей и 1 монография.

E-mail: *inna_pls@mail.ru* .

Inna Lvovna Plastun (b. 1969) graduated with honours (1991) from the N.G. Chernyshevsky Saratov State University, majoring in Physics. Doctor of Physical and Mathematical Sciences (2011). Associated professor of Saratov State Technical University. Her research interests include nonlinear optics, laser physics, mathematical modeling and parallel computing. She is co-author of 80 scientific works including 60 papers and 1 monography.

Мисюрин Артём Геннадиевич, 1986 года рождения, в 2009 году окончил Саратовский государственный технический университет (СГТУ) по специальности «Программное обеспечение вычислительной техники и автоматизированных систем», является аспирантом СГТУ. Область научных интересов: параллельные вычисления, численное моделирование нелинейно-оптических процессов, разработка интерфейсных и сетевых приложений.

E-mail: *drdoomenator@yandex.ru* .

Artem Gennadievich Misurin (b. 1986) graduated from Saratov State Technical University in 2009, majoring in Computers and Automatic Systems Software. He is post-graduate student in Saratov State Technical University. His research interests are: parallel computing, numerical simulation of nonlinear optics processes, developing of interface and network applications.

В редакцию поступила 14 июня 2012 г.