Формирование конечно-элементных систем в GPGPU

Новиков А.К., Кузьмин И.М., Недожогин Н.С., Копысов С.П.

Институт механики УрО РАН, г. Ижевск

Международная научная конференция
Параллельные вычислительные технологии ПаВТ'2014
Южный федеральный университет, Ростов-на-Дону,
31 марта – 4 апреля 2014 г.

Содержание

Основные шаги метода конечных элементов:

Шаг 1. Построение и разделение конечно-элементной сетки с заданными физическими параметрами и граничными условиями.

Шаг 2. **Формирование локальных элементных матриц жесткости**[1].

Шаг 3. Формирование глобальной системы уравнений с учетом вкладов от элементов и узлов в соответствии со схемами сборки[2].

Шаг 4. Введение граничных условий.

Шаг 5. Решение системы уравнений[3],[4].

[1] Plaszewski P., Maciol P., Banas K.
[2] Cecka C., Lew A., Darve E.
[3] Dalton S., Bell N.
[4] Губайдуллин Д.А., Никифоров А.И., Садовников Р.В.

$$Ku = f \tag{1}$$

$$K = \int_V B^T DB dV \approx \sum_{e=1}^{m} \int_{V^e} B^{eT} D^e B^e dV^e = \sum_{e=1}^{m} C_e^T K^e C_e, \tag{2}$$

здесь $K^e \in \mathbb{R}^{N_e \times N_e}$ — локальная (элементная) матрица жесткости; $C = [C_e]_{1\times m} \in \mathbb{Z}^{N_e \times N}$ — матрица связности; m — число конечных элементов; N_e — число степеней свободы в одном конечном элементе; N — число неизвестных в системе уравнений.

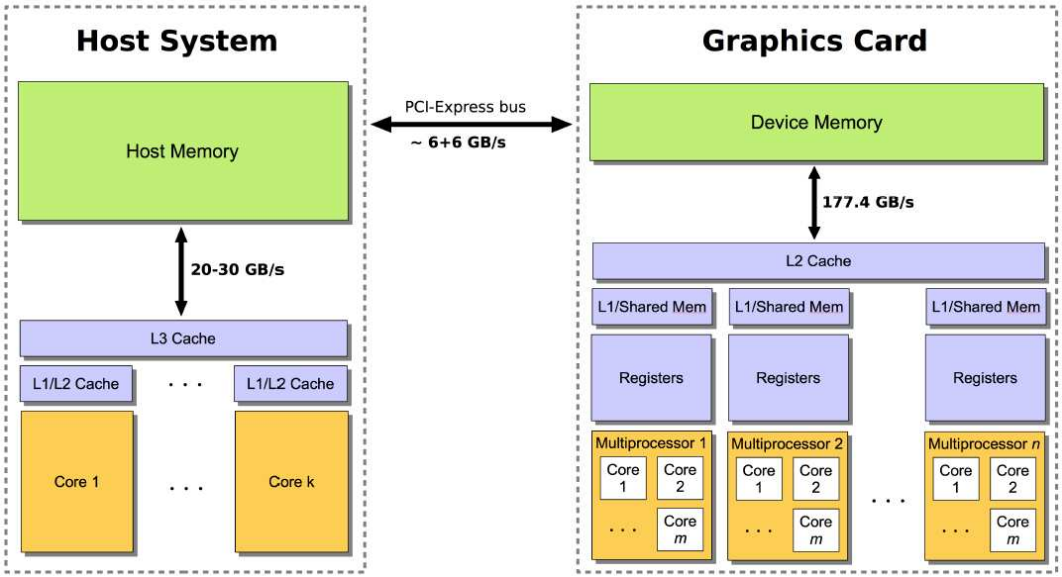
где $B^e = [B_1^e B_2^e \dots B_{N_e}^e]$, $\quad B_i^e = \begin{pmatrix} \frac{\partial \psi_i^e}{\partial x_1} & 0 & 0 \\ 0 & \frac{\partial \psi_i^e}{\partial x_2} & 0 \\ 0 & 0 & \frac{\partial \psi_i^e}{\partial x_3} \\ \frac{\partial \psi_i^e}{\partial x_2} & \frac{\partial \psi_i^e}{\partial x_1} & 0 \\ \frac{\partial \psi_i^e}{\partial x_3} & 0 & \frac{\partial \psi_i^e}{\partial x_1} \\ 0 & \frac{\partial \psi_i^e}{\partial x_3} & \frac{\partial \psi_i^e}{\partial x_2} \end{pmatrix}$.

$$Kp = \sum_{e=1}^{m} C_e^T K^e C_e p = \sum_{e=1}^{m} C_e^T K^e p_e. \tag{3}$$

здесь $p \in \mathbb{R}^N$ — вектор направления, например в методе сопряженных градиентов.

Варианты распределения вычислений между GPU и CPU:

- С решением собранной/крупноблочной системы (1) на GPU[5].
  - Интегрирование матриц K^e на GPU с передачей на CPU для сборки K.
  - Интегрирование матриц K^e и сборка K на GPU.
- Применение элементных схем подпространств Крылова на GPU.
  - Блочно-диагональный вариант[6], с блоками K^e.
  - Вычисление матриц K^e в матрично-векторном произведении.



Коммуникации CPU $\leftrightarrows$ GPU:
– массивы;
– массив конечно-элементных структур;
– одна структура с полями для всех конечных элементов.

Рис. 1: Коммуникации CPU и GPU

[5] Kopysov S.P., Kuzmin I.M., Nedozhogin N.S., Novikov A.K., Sagdeeva Y.A. Hybrid Multi-GPU solver based on Schur complement method // Lecture Notes in Computer Science. 2013. Vol. 7979. P. 65–79.

[6] Кузьмин И.М., Недожогин Н.С., Новиков А.К., Сагдеева Ю.А. Конечно-элементное решение динамических задач деформирования на GPU // Девятая Всероссийская конференция «Сеточные методы для краевых задач и приложения», г. Казань, Изд

Аппроксимация перемещений в шестигранном конечном элементе

$$u_i^p = \sum_{v=1}^{8} \psi_v \cdot u_{v_i} + \sum_{e=1}^{12} \sum_{n=2}^{p} \psi_e^n \cdot a_e^{n_i} + \sum_{f=1}^{6} \sum_{m=2}^{p-2} \sum_{n=2}^{p-m} \psi_f^{mn} \cdot b_f^{mn_i} + \\ + \sum_{l=2}^{p-4} \sum_{m=2}^{p-1} \sum_{n=2}^{p-1-m} \psi^{lmn} \cdot c^{lmn_i}, \tag{4}$$

где u_{v_i}, $a_e^{n_i}$, $b_f^{mn_i}$, c^{lmn_i} — неизвестные; ψ_v, ψ_e^n, ψ_f^{mn}, ψ^{lmn} — функции формы для узлов, ребер, граней и внутренних степеней свободы; p — порядок аппроксимации.

Базисные функции для *p-версии* в случае одного измерения будут иметь вид

$$\psi_i(s) = \frac{1}{2}(1 \pm s), \quad \text{или} \quad \psi_i(s) = \frac{1}{2}(1 + s_i s), \quad i = 0, 1$$

и

$$\psi_j(s) = P_j(s) = \int_{-1}^{s} L_{j-1}(x)dx, \quad j = 2, p$$

где s — локальная координата, $-1 \le s \le 1$; s_i — локальная координата i-го узла, $s_i = -1, 1$; $L_p(x)$ — полиномы Лежандра

$$L_p(x) = \frac{1}{2^p p!} \frac{d^p}{dx^p} \left[(x^2 - 1)\right]^p.$$

Базисные функции для элемента кубической формы с локальными координатами $-1 \le r, s, t \le 1$.

- Узловые базисные функции имеют вид

$$\psi_v = \frac{1}{8}(1 + r_v r)(1 + s_v s)(1 + t_v t), \quad v = 1, 8,$$

где v — номер узла в элементе.

- Базисные функции с обобщенными параметрами на ребрах элемента:

$$\psi_e^p = \frac{1}{4}(1 + r_e r)(1 + s_e s)P_p(t), \quad p \ge 2,$$

где e — ребро в элементе.

- Базисные функции для граней элемента имеют вид

$$\psi_f^{mn} = \frac{1}{2}(1 + r_f r)P_m(s)P_n(t), \qquad p \ge 4,$$

$$m, n \ge 2, \quad m + n \le p.$$

Здесь f — грань элемента, перпендикулярная оси r.

- Базисные функции, связанные с внутренними обобщенными параметрами

$$\psi^{lmn} = P_l(r)P_m(s)P_n(t), \qquad p \ge 6,$$

$$l, m, n \ge 2, \quad l + m + n \le p.$$

Таблица 1: Характеристика иерархических элементов до 10-го порядка

Порядок элемента, p	Число добавляемых функций формы	Число неизвестных в к.э., N_e
1	8 узловых	24
2	12 на ребрах	60
3	12 на ребрах	96
4	12 на ребрах 6 на гранях	150
5	12 на ребрах 12 на гранях	222
6	12 на ребрах 18 на гранях 1 внутренняя	315
7	12 на ребрах 24 на гранях 3 внутренних	432
8	12 на ребрах 30 на гранях 6 внутренних	576
9	12 на ребрах 36 на гранях 10 внутренних	750
10	12 на ребрах 42 на гранях 15 внутренних	957

Таблица 2: Оценка вычислительных затрат[7]

	Квадратуры Гаусса	Векторные квадратуры	Факторизация сумм
2D	$\mathcal{O}(p^6)$	$\mathcal{O}(p^6)$	$\mathcal{O}(p^5)$
3D	$\mathcal{O}(p^9)$	$\mathcal{O}(p^9)$	$\mathcal{O}(p^7)$

[7] Melenk M, Gerdes K and Schwab C. Fully discrete hp-finite elements: fast quadrature // Comput. Methods Appl. Mech. Eng. 2001. Vol. 190. 4339–4369

В общем случае алгоритм формирования локальной матрицы K^e жесткости включает следующие шаги:

- Построение функций формы элемента ψ_e.
- Переход от глобальных системы координат к локальной, вычисление матрицы Якоби J, обратного преобразования J^{-1} и якобиана $detJ$.
- Вычисление $B^e = [B_1^e B_2^e \dots B_{N_e}^e]$.
- Численное интегрирование по объёму для вычисления матрицы жесткости

$$K^e = [K^e_{\alpha\beta}], \quad K^e_{\alpha\beta} = \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} {B^e_\alpha}^T(r,s,t) D B^e_\beta(r,s,t) \det J(r,s,t) dr ds dt. \tag{5}$$

$$K^e_{\alpha\beta} \approx \sum_{i=1}^{n_i}\sum_{j=1}^{n_j}\sum_{k=1}^{n_k} {B^e_\alpha}^T(r_i,s_j,t_k) D^e B^e_\beta(r_i,s_j,t_k) w_i w_j w_k \det J(r,s,t), \tag{6}$$

где $B^e_\alpha(r_i,s_j,t_k)$ — матрица производных ψ_e в (r_i,s_j,t_k); D^e— матрица характеристик материала (среды); w_i, w_j, w_k — весовые коэффициенты квадратур Гаусса.

Число точек интегрирования

$$n_G = \underbrace{\left[\frac{3p-2}{2}\right] \times \dots \times \left[\frac{3p-2}{2}\right]}_{d}. \tag{7}$$

Уровни распараллеливания и алгоритмы

1: В CUDA-нити $th \in [1, m]$ выполнить:
{I уровень}
2: **for** $i = 1$ to n_i **do**
3: **for** $j = 1$ to n_j **do**
4: **for** $k = 1$ to n_k **do**
5:
В CUDA-нити $th \in [1, m \cdot n_G]$ выполнить:
{II уровень}
6: Вычисление производных ψ_e по r, s, t.
7: Вычисление J, $detJ$ и J^{-1}.
8: Вычисление матрицы B^e.
9: **for** $\alpha = 1$ to N_e **do**
10: **for** $\beta = \alpha$ to N_e **do**
11: $\hat{K}^e_{\alpha\beta} = B^{e\,T}_{\alpha} D^e B^e_{\beta} \times$
$\times w_i w_j w_k \det J.$
12: $K^e_{\alpha\beta} \leftarrow K^e_{\alpha\beta} + \hat{K}^e_{\alpha\beta}$.
13: **end for**
14: **end for**
15: **end for**
16: **end for**
17: **end for**

Особенности алгоритмов

AI	AII
$th \leftarrow K^e$	$th \leftarrow K^e_{\alpha\beta}(r_i, s_i, t_i)$
$N_{th} = m$	$N_{th} = m \cdot n_G$
$N_{b_x} = maxGridSize[1]$	$N_{b_x} = \frac{N_{th}}{N_{tib}} + 1$
$N_{b_y} = \frac{N_{th}/N_{tib}}{N_{b_x}-1} + 1$	$N_{b_y} = 1$

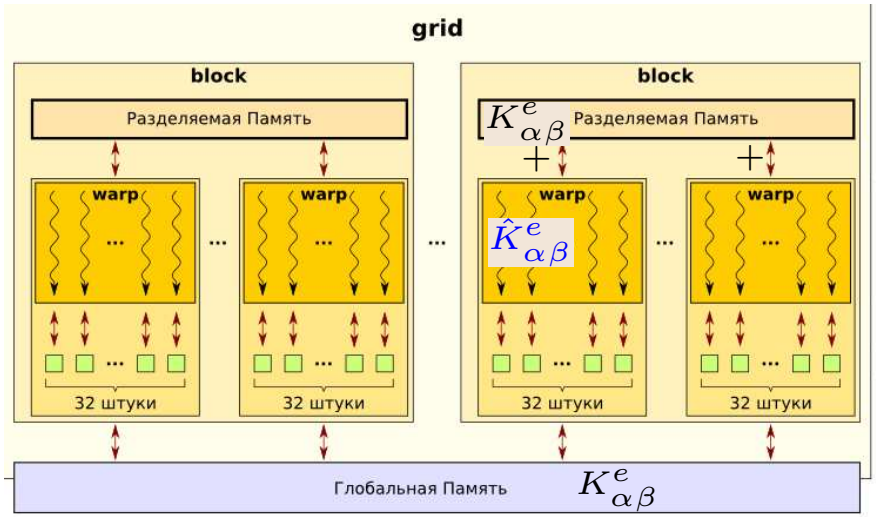


Рис. 2: Суммирование $K^e_{\alpha\beta}$ в AII

Таблица 3: Время вычисления одним ядром CPU (сек.) и ускорение на GPU

3D, $m = 10000$, $(N_e \times N_e)$	t_{CPU} E5430/E5-2609		$\frac{t_{CPU}}{t_{GPU}}$ GTX580/GTX680	
	-O0	-O3	AI	AII
$p = 1, n_G = 8, (24 \times 24)$	1.23 /1.12	0.17/0.14	0.41/0.74	0.39/0.67
$p = 2, n_G = 8, (60 \times 60)$	5.73 /5.20	0.74/0.58	1.09/1.08	0.90/1.05
$p = 3, n_G = 64, (96 \times 96)$	108.3 /97.45	13.30/10.52	2.68/1.48	7.07/6.41
$p = 4, n_G = 125, (150 \times 150)$	491.1/441.04	59.02/47.52	2.69/1.41	8.41/6.75
$p = 5, n_G = 343, (222 \times 222)$	2867.6/2598.90	337.83/293.00	2.81/1.50	5.82/5.01

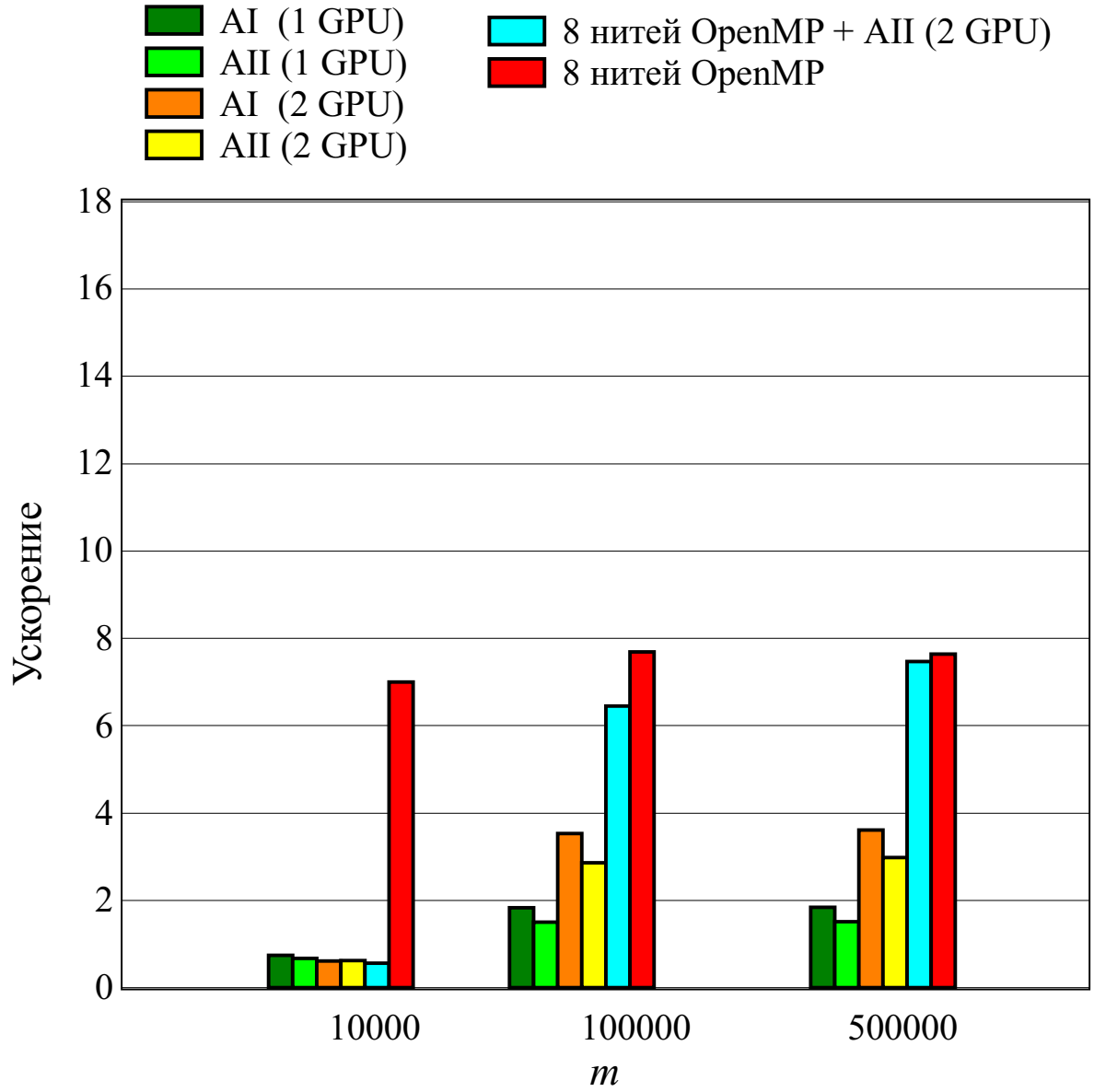


Рис. 3: Интегрирование K^e при $p = 1$.

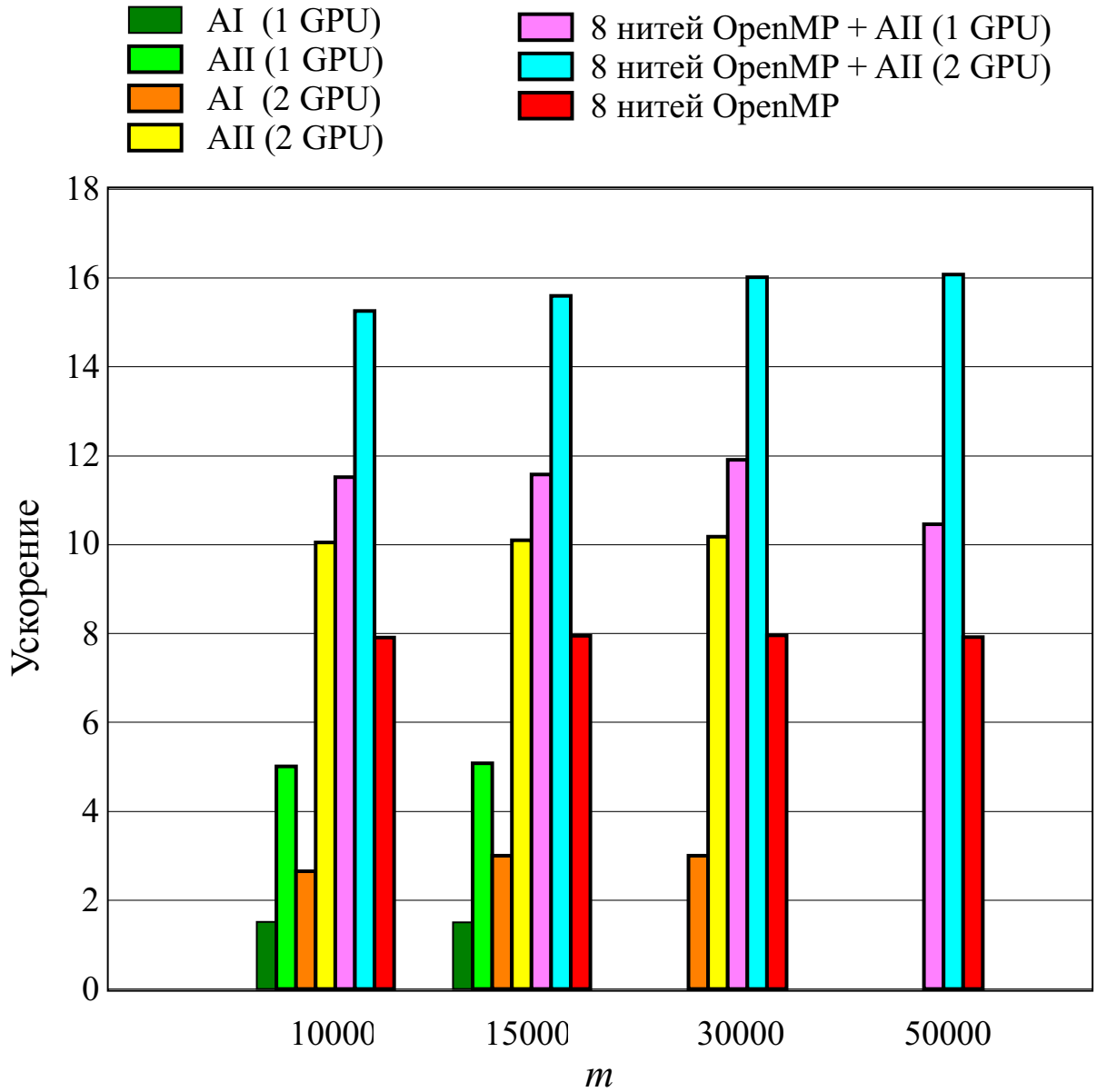


Рис. 4: Интегрирование K^e при $p = 5$.

- **Выводы**
  - В случае конечных элементов высоких порядков, существенное ускорение достигается, когда на одном ускорителе интегрируется в несколько раз больше матриц конечных элементов, чем на одном ядре CPU.
  - При неоднородном распределении порядков (p-/hp-версии МКЭ) конечные элементы меньших степеней предпочтительнее интегрировать на ядрах CPU, а больших степеней — на GPU.
  - Параллелизм GPU наиболее эффективно использовался на уровне точек интегрирования.
- **Продолжение исследований**
  - Распределение вычислений на cluster GPU.
  - Интегрирование локальных матриц в рамках матрично-векторных произведений элементных (element-by-element) схем.
- **Благодарности**

Работа выполнена при финансовой поддержке РФФИ (грант 14-01-00055-а, 14-01-31066-мол_а, 13-01-00101-а) и программы Президиума РАН №18 при поддержке УрО РАН (проект 12-П-1-1005).