



Московский Государственный Университет им. М.В. Ломоносова
Факультет Вычислительной Математики и Кибернетики

Кожевников Евгений Владимирович

**Отчёт заданию №3 в рамках курса
«Суперкомпьютерное моделирование
и технологии»**

Вариант 8

Москва, 2023

Содержание

1 Постановка задачи	3
2 Численный метод решения задачи	4
3 Программная реализация алгоритма с использованием технологий OpenMP и MPI	4
4 Результаты запусков на IBM Polus	5
4.1 $L = 1$	6
4.1.1 Графики для $N=128$	6
4.1.2 Графики для $N=256$	8
4.2 $L = \pi$	10
4.2.1 Графики для $N=128$	10
4.2.2 Графики для $N=256$	12
5 Заключение	13

1 Постановка задачи

Рассматривается дифференциальная задача для трёхмерного гиперболического уравнения в области, представляющей из себя прямоугольный параллелепипед.

В трёхмерной замкнутой области

$$\Omega = [0 \leq x \leq L_x] \times [0 \leq y \leq L_y] \times [0 \leq z \leq L_z]$$

для $0 < t \leq T$ требуется найти решение $u(x, y, z, t)$ уравнения в частных производных

$$\frac{\partial^2 u}{\partial t^2} = a^2 \Delta u$$

с начальными условиями

$$\begin{aligned} u|_{t=0} &= \varphi(x, y, z) \\ \left. \frac{\partial u}{\partial t} \right|_{t=0} &= 0 \end{aligned}$$

при условии, что на границе области заданы периодические граничные условия

$$\begin{aligned} u(0, y, z, t) &= u(L_x, y, z, t) & u_x(0, y, z, t) &= u_x(L_x, y, z, t) \\ u(x, 0, z, t) &= u(x, L_y, z, t) & u_y(x, 0, z, t) &= u_y(x, L_y, z, t) \\ u(x, y, 0, t) &= u(x, y, L_z, t) & u_z(x, y, 0, t) &= u_y(x, y, L_z, t) \end{aligned}$$

В соответствии с **вариантом 8**, аналитическое решение задано следующим образом:

$$\begin{aligned} u_{analytical} &= \sin \left(\frac{2\pi}{L_x} x \right) \cdot \sin \left(\frac{4\pi}{L_y} y \right) \cdot \sin \left(\frac{6\pi}{L_z} z \right) \cdot \cos(a_t \cdot t) \\ a_t &= \sqrt{\frac{4}{L_x^2} + \frac{16}{L_y^2} + \frac{36}{L_z^2}} \\ a^2 &= 1 \end{aligned}$$

2 Численный метод решения задачи

Введём на Ω сетку $\omega_{h\tau} = \bar{\omega}_h \times \omega_\tau$. Через ω_h обозначим множество внутренних, а через γ_h — множество граничных узлов сетки $\bar{\omega}_h$.

Через Δ_h обозначим семиточечный разностный аналог оператора Лапласа.

Для начала счёта определим значения u для первых двух итераций:

$$u_{ijk}^0 = \varphi(x_i, y_j, z_k) \in \bar{\omega}_h$$

$$u_{ijk}^1 = \begin{cases} u_{ijk}^0 + a^2 \frac{\tau^2}{2} \Delta_h \varphi(x_i, y_j, z_k), & (x_i, y_j, z_k) \in \omega_h \\ u_{analytical}(x_i, y_j, z_k, \tau), & (x_i, y_j, z_k) \in \gamma_h \end{cases}$$

Из явной разностной схемы

$$\frac{u_{ijk}^{n+1} - 2u_{ijk}^n + u_{ijk}^{n-1}}{\tau^2} = a^2 \Delta_h u^n, \quad (x_i, y_j, z_k) \in \omega_h$$

выразим значения u_{ijk}^{n+1} на $(n+1)$ -м шаге через значения на предыдущих слоях:

$$u_{ijk}^{n+1} = 2u_{ijk}^n - u_{ijk}^{n-1} + \tau^2 a^2 \Delta_h u^n$$

3 Программная реализация алгоритма с использованием технологий OpenMP и MPI

В соответствии с заданием используется блочное разбиение. Для достижения наибольшей производительности выбраны фиксированные варианты разбиения пространства на блоки в зависимости от числа MPI процессов:

Число MPI процессов	Число блоков по X	Число блоков по Y	Число блоков по Z
1	1	1	1
2	2	1	1
4	2	2	1
8	2	2	2
16	2	2	4
32	2	4	4

Для получения блока в разбиении и создания топологии используется MPI_Comm_Cart.

Аналогично OpenMP и однопоточной реализации, для расчётов используются 3 массива, соответствующие текущему, предыдущему и пред-предыдущему моменту времени. Каждый блок разбиения использует свою тройку таких массивов.

В **варианте 8** периодические граничные условия по всем 3 осям. Они реализуются тем же механизмом, с помощью которого блоки передают свои внешние грани следующим блокам в топологии, с точностью до замыкания координат на 4-х мерном торе.

Для распараллеливания внутреннего цикла численного решения задачи каждый блок независимо использует #pragma omp parallel for collapse(N).

4 Результаты запусков на IBM Polus

Для компиляции использовался компилятор IBM mpixlC из модуля SpectrumMPI. В силу перебоев в работе гибридных программ на СК Polus, были произведены замеры с и без использования OpenMP.

```
# mpi+openmp
mpixlC v8.cpp -std=c++14 -DL=1 -DN=128 -DUSE_OMP -qsmp=omp -o v8
# mpi
mpixlC v8.cpp -std=c++14 -DL=1 -DN=128 -o v8
```

Для запуска использовалась утилита mpisubmit.pl:

```
# mpi+openmp
mpisubmit.pl -p N -t M ./v8
# mpi
mpisubmit.pl -p N ./v8
```

где M — число OpenMP нитей, с которым запустится программа, а N — число MPI процессов.

Ускорение S и погрешность δ вычислялись следующим образом:

$$S = \frac{T|_{nthreads,nprocesses}}{T|_{nthreads=min,nprocesses=min}}$$
$$\delta = \sum_{(x_i,y_j,z_k) \in \omega_h} |u_{analytical}(x_i, y_j, z_k, T) - u_{calculated}(x_i, y_j, z_k, T)|$$

Рассматривалось $K = 20$ эпох симуляции от $T = 0$ до $T = 0.0001$. В итоговых таблицах представлено значение погрешности δ на заключительной эпохе симуляции.

4.1 $L = 1$

Число MPI процессов N_p	Число OpenMP нитей	Число точек сетки N^3	Время решения $T (ms)$	Ускорение S	Погрешность δ
1	1	128^3	5296.57	1	0.00125278
2	1	128^3	2740.82	1.93	0.00125278
4	1	128^3	1614.98	3.28	0.00125278
8	1	128^3	1067.11	4.96	0.00125278
2	1	128^3	5401.47	1	0.00125278
2	2	128^3	5880.62	0.92	0.00125278
2	4	128^3	4807.44	1.12	0.00125278
2	8	128^3	7637.96	0.71	0.00125278
1	1	256^3	42004.3	1	0.00933043
2	1	256^3	21627.4	1.94	0.00933043
4	1	256^3	12960.3	3.24	0.00933043
8	1	256^3	6790.63	6.19	0.00933043
4	1	256^3	17919.4	1	0.00933043
4	2	256^3	18464.2	0.97	0.00933043
4	4	256^3	32406.9	0.55	0.00933043
4	8	256^3	30289.9	0.59	0.00933043

4.1.1 Графики для $N=128$

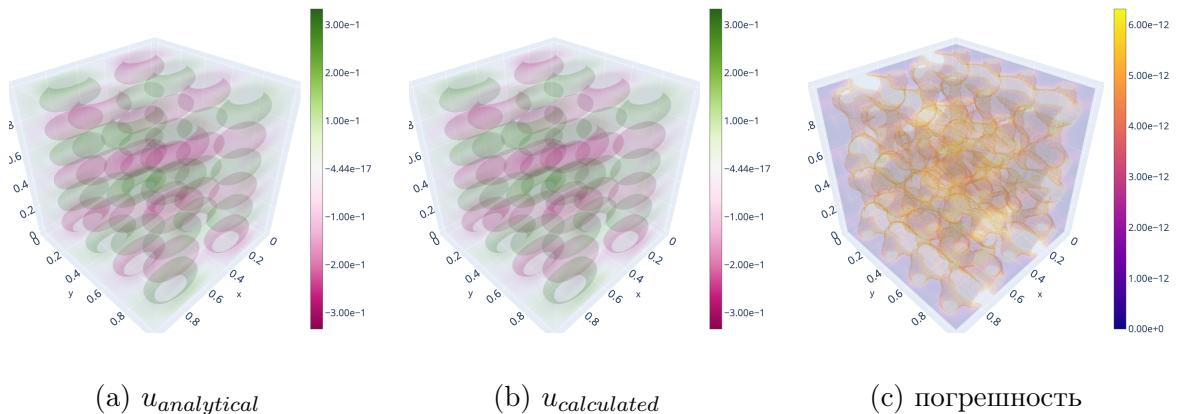
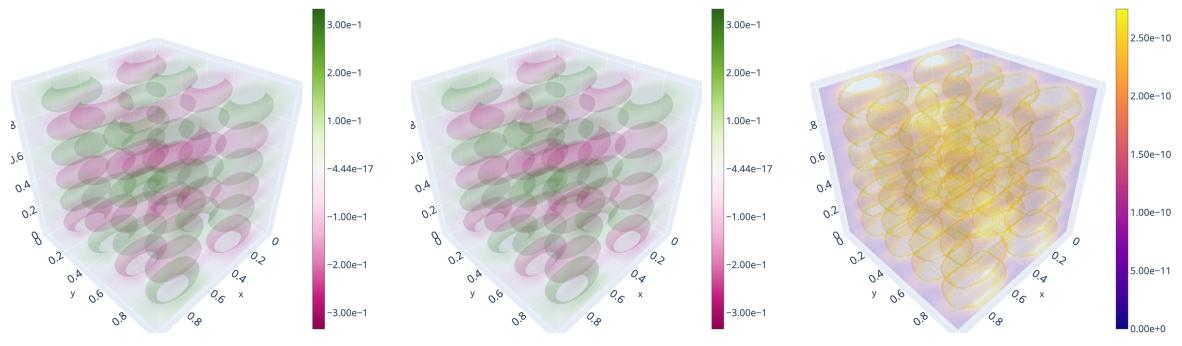
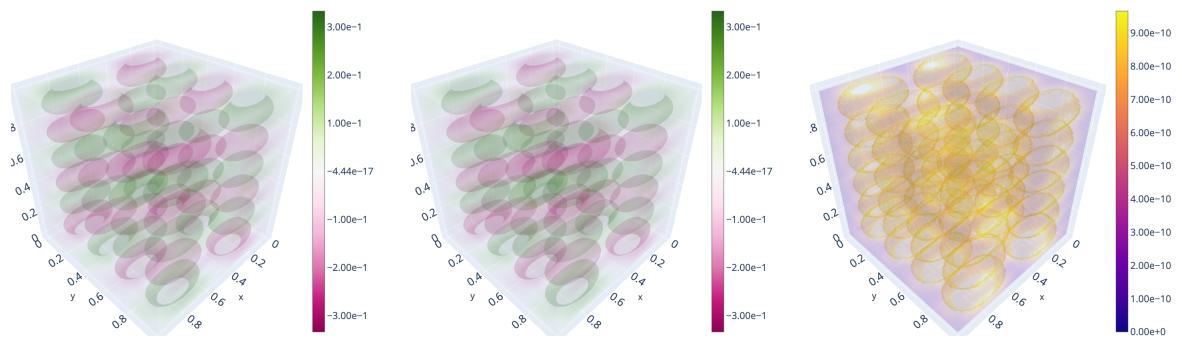


Рис. 1: 1 эпоха

(a) $u_{analytical}$ (b) $u_{calculated}$

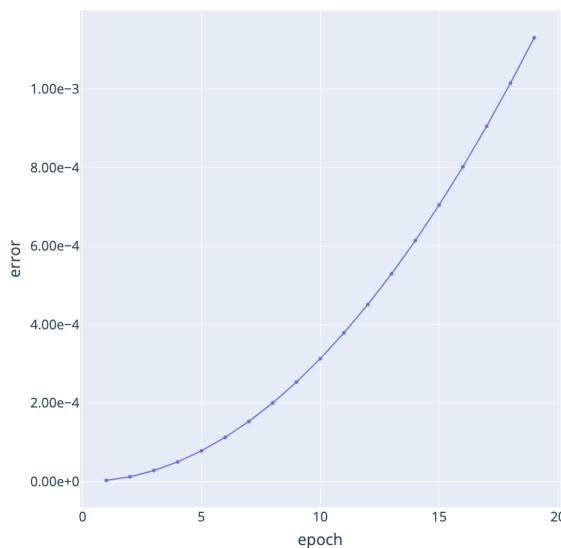
(c) погрешность

Рис. 2: 10 эпоха

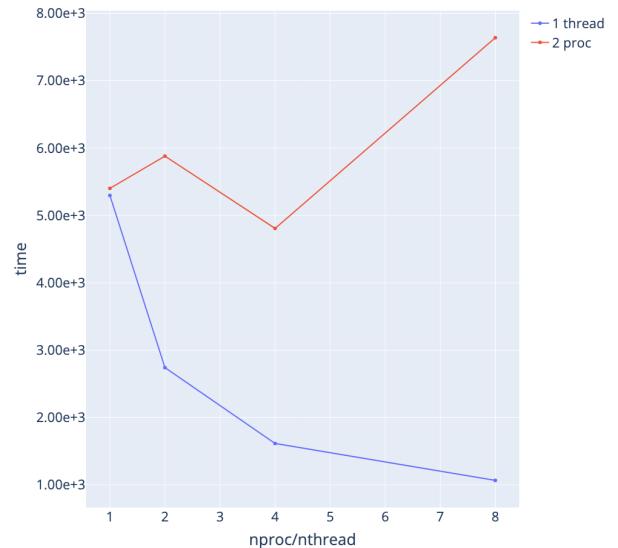
(a) $u_{analytical}$ (b) $u_{calculated}$

(c) погрешность

Рис. 3: 20 эпоха



(a) Погрешность от эпохи



(b) Время работы от конфигурации

4.1.2 Графики для N=256

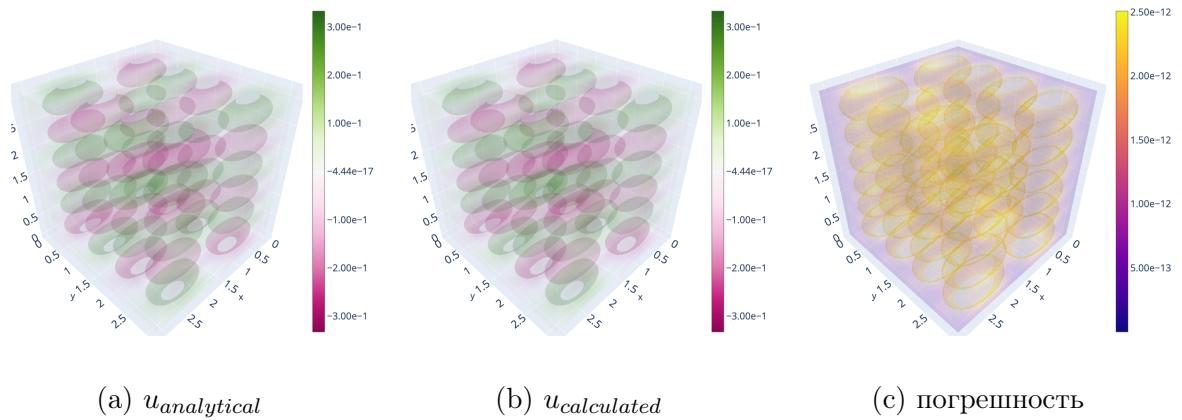


Рис. 5: 1 эпоха

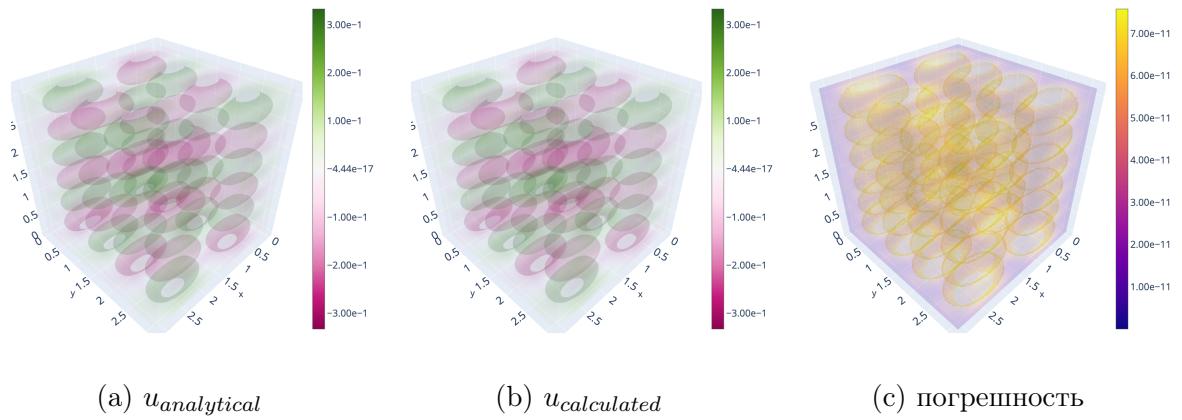


Рис. 6: 10 эпоха

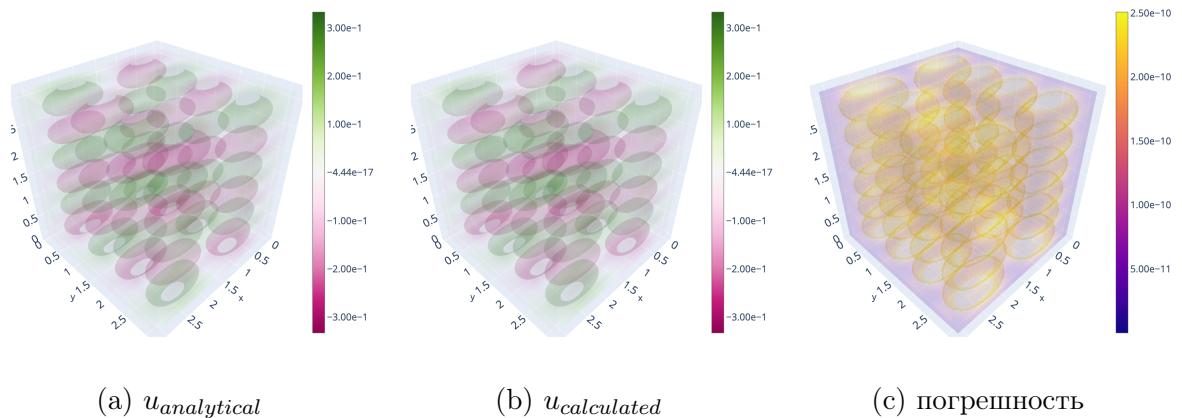
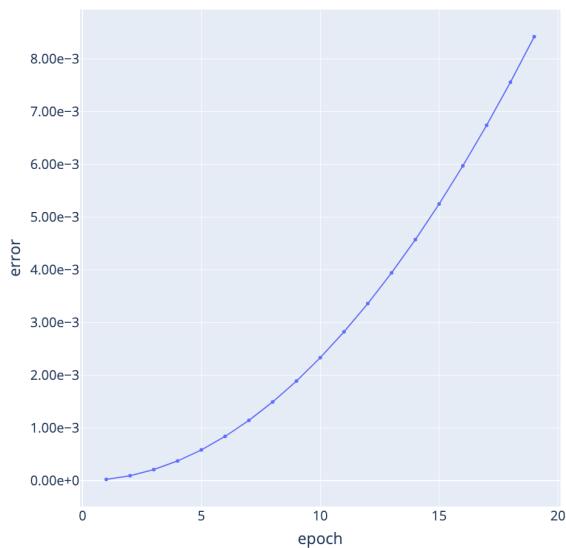
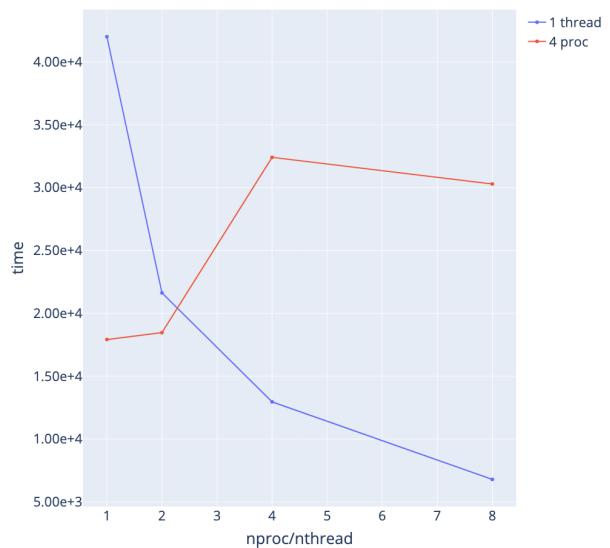


Рис. 7: 20 эпоха



(a) Погрешность от эпохи



(b) Время работы от конфигурации

4.2 $L = \pi$

Число MPI процессов N_p	Число OpenMP нитей	Число точек сетки N^3	Время решения T (ms)	Ускорение S	Погрешность δ
1	1	128^3	5743.76	1	0.000126933
2	1	128^3	3214.88	1.79	0.000126933
4	1	128^3	2137.36	2.69	0.000126933
8	1	128^3	1089.79	5.27	0.000126933
2	1	128^3	7928.95	1	0.000126933
2	2	128^3	9870.61	0.8	0.000126933
2	4	128^3	12763.1	0.62	0.000126933
2	8	128^3	17839	0.44	0.000126933
1	1	256^3	45737.6	1	0.00094537
2	1	256^3	28286.4	1.62	0.00094537
4	1	256^3	13910.3	3.29	0.00094537
8	1	256^3	9115.46	5.02	0.00094537
4	1	256^3	19964.4	1	0.00094537
4	2	256^3	20680.2	0.97	0.00094537
4	4	256^3	24246.2	0.82	0.00094537
4	8	256^3	30478.3	0.66	0.00094537

4.2.1 Графики для $N=128$

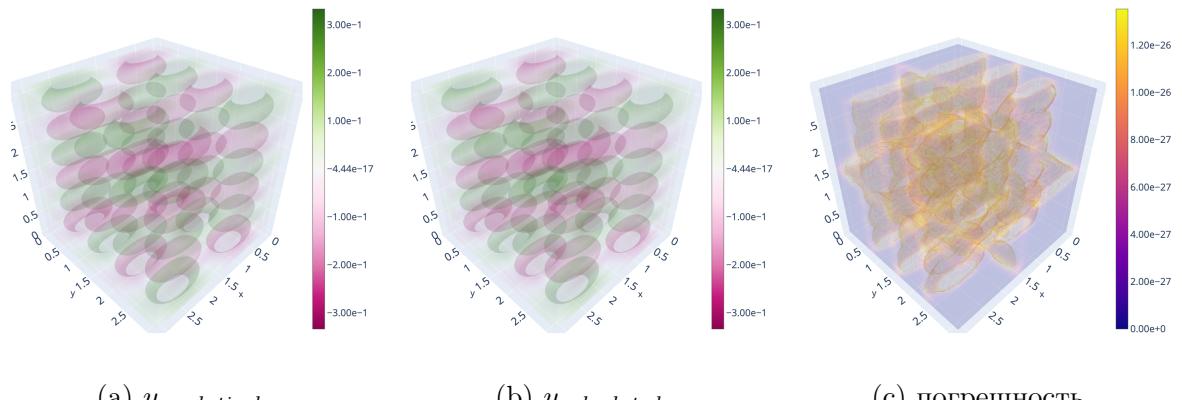
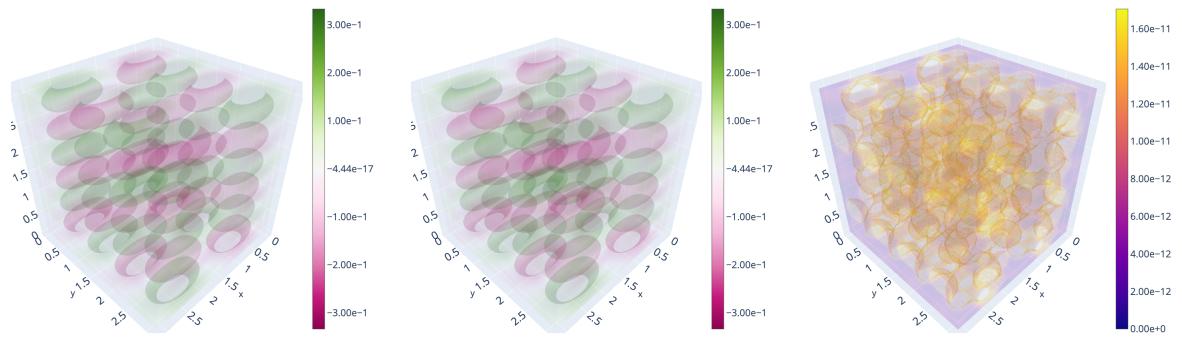
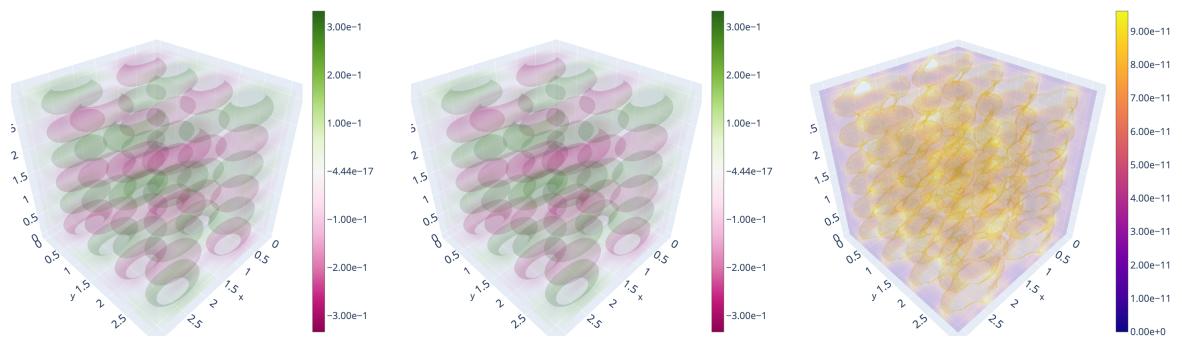


Рис. 9: 1 эпоха

(a) $u_{analytical}$ (b) $u_{calculated}$

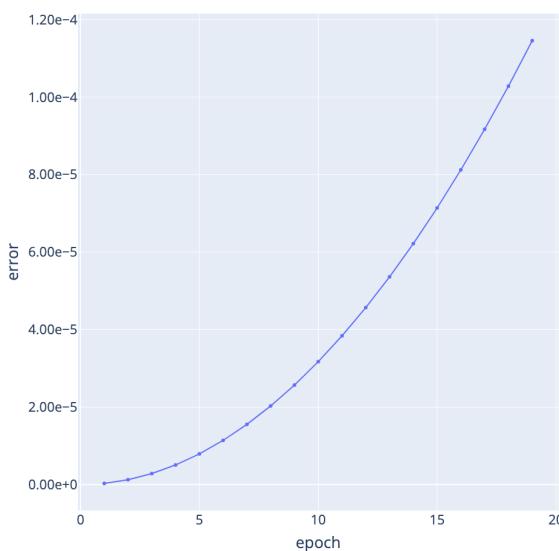
(c) погрешность

Рис. 10: 10 эпоха

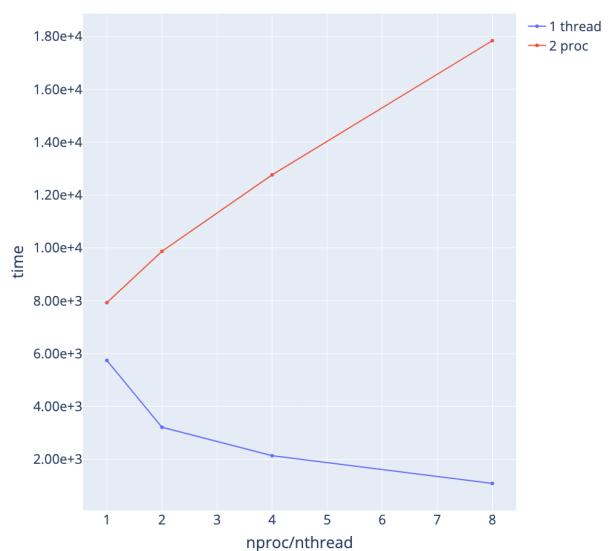
(a) $u_{analytical}$ (b) $u_{calculated}$

(c) погрешность

Рис. 11: 20 эпоха



(a) Погрешность от эпохи



(b) Время работы от конфигурации

4.2.2 Графики для N=256

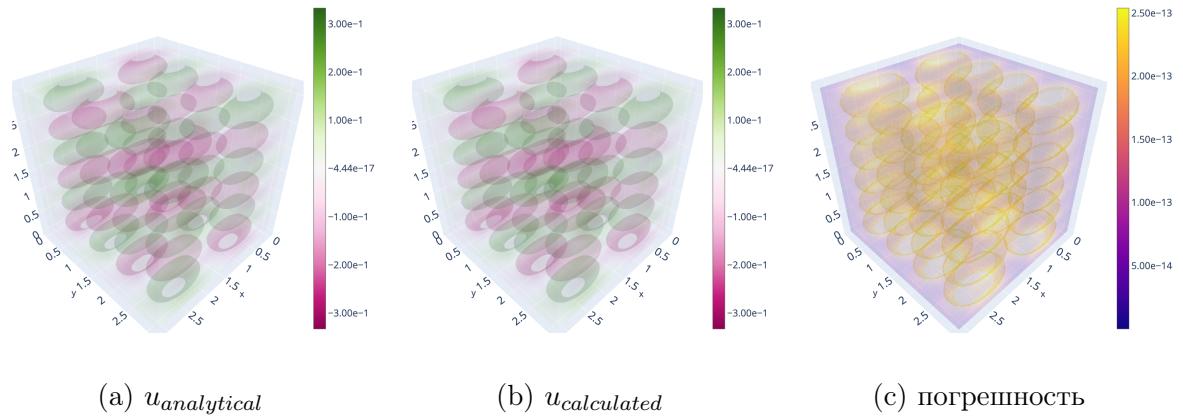


Рис. 13: 1 эпоха

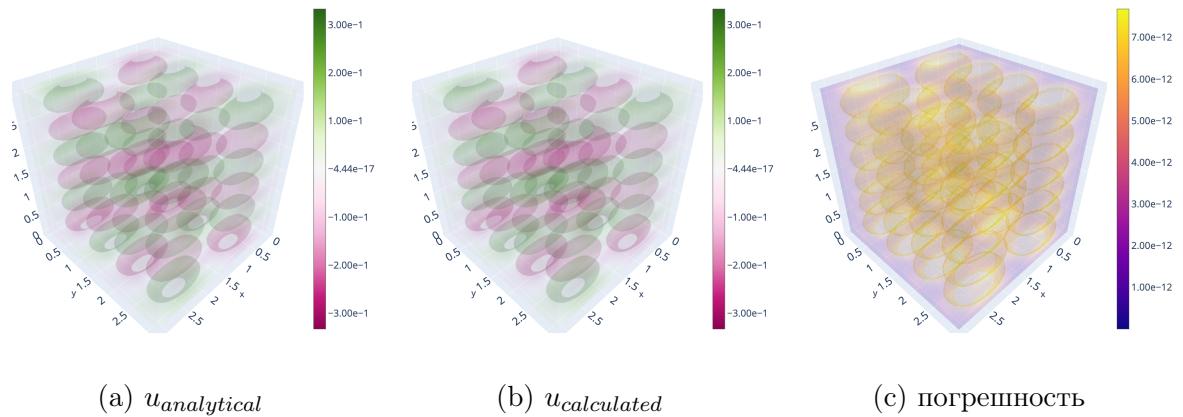


Рис. 14: 10 эпоха

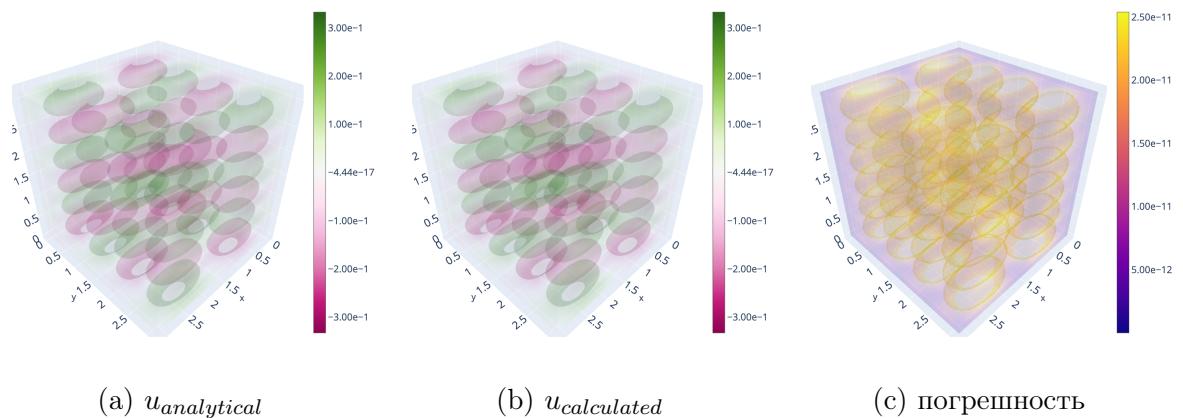
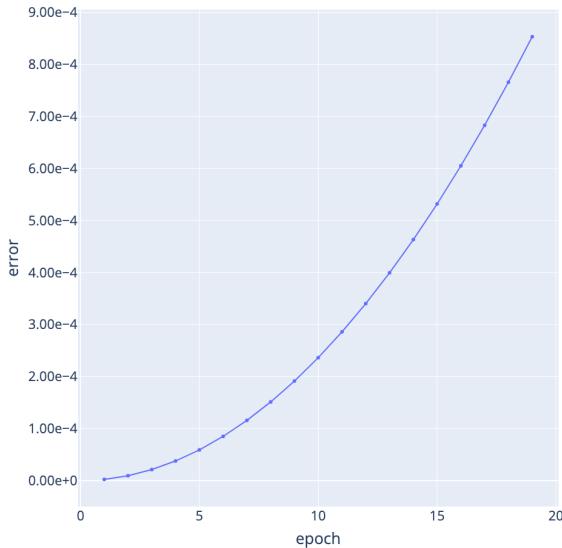
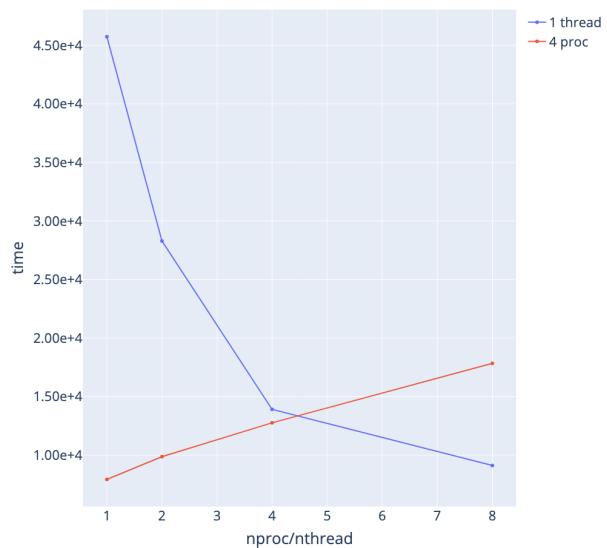


Рис. 15: 20 эпоха



(a) Погрешность от эпохи



(b) Время работы от конфигурации

5 Заключение

На основе анализа данных о запуске программы с различным числом процессов и на различных сетках можно сделать следующие выводы:

- При запуске на суперкомпьютере рост производительности достигается только с ростом числа MPI процессов. Скорее всего, это связано с программным или аппаратным сбоем, мешающим работе гибридных программ.
- По графикам погрешности на обеих сетках (128, 256) хорошо видно, по какой функции расходится разностная схема. Вероятно, это говорит об отсутствии ошибок в программной реализации численного решения.