



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

Кожевников Евгений Владимирович

**Отчёт заданию №2 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	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 $N=256$	7
4.2 $L = \pi$	9
4.2.1 Графики для $N=128$	9
4.2.2 Графики для $N=256$	10
5 Заключение	12

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

Для распараллеливания алгоритма численного решения задачи была использована директива `#pragma omp parallel for collapse(N)` для каждого гнезда циклов.

Для вывода отладочной информации с значениями аналитического и численного решения в узлах была использована однопоточная реализация. Это было сделано из-за незначительности времени вычислений по сравнению с временем записи значений на диск.

4 Результаты запусков на IBM Polus

Для компиляции использовался стандартный компилятор GNU. Оптимизации были отключены, так как они рассчитаны в основном на однопоточные программы и ухудшают производительность программ, распараллеленных с помощью OpenMP.

```
g++ -std=c++11 -O0 -o v8 -fopenmp v8.cpp
```

Для запуска использовался следующий командный файл для утилиты `bsub`:

```
1 #BSUB -n N
2 #BSUB -W 00:15
3 #BSUB -o "thread_M.out"
4 #BSUB -e "thread_M.err"
5 #BSUB -R "span[hosts=1]"
6 OMP_NUM_THREADS=M ./v8
```

где M — число OpenMP нитей, с которым запустится программа, а N — число выделенных ядер процессора. M и N для всех запусков выбирались с соотношением $\frac{M}{N} = 2$.

```
bsub < script.lsf
```

Ускорение S и погрешность δ вычислялись следующим образом:

$$S = \frac{T|_{nthreads}}{T|_{nthreads=1}}$$
$$\delta = \sum_{(x_i, y_j, z_k) \in \omega_h} |u_{analytical}(x_i, y_j, z_k, T) - u_{calculated}(x_i, y_j, z_k, T)|$$

Рассматривалось $K = 20$ эпох симуляции от $T = 0$ до $T = 0.001$. В итоговых таблицах представлено значение погрешности δ на заключительной эпохе симуляции.

4.1 L = 1

<i>Число OpenMP нитей</i>	<i>Число точек сетки N^3</i>	<i>Время реше- ния T (ms)</i>	<i>Ускорение S</i>	<i>Погрешность δ</i>
1	128^3	14827.8	1	0.00209787
2	128^3	6911.27	2.15	0.00209787
4	128^3	5171.96	2.87	0.00209787
8	128^3	3473.81	4.27	0.00209787
16	128^3	3097.21	4.79	0.00209787
1	256^3	94378.8	1	0.00420157
4	256^3	36181.3	2.61	0.00420157
8	256^3	25097.1	3.76	0.00420157
16	256^3	19657.7	4.80	0.00420157
32	256^3	14909	6.33	0.00420157

4.1.1 Графики для N=128

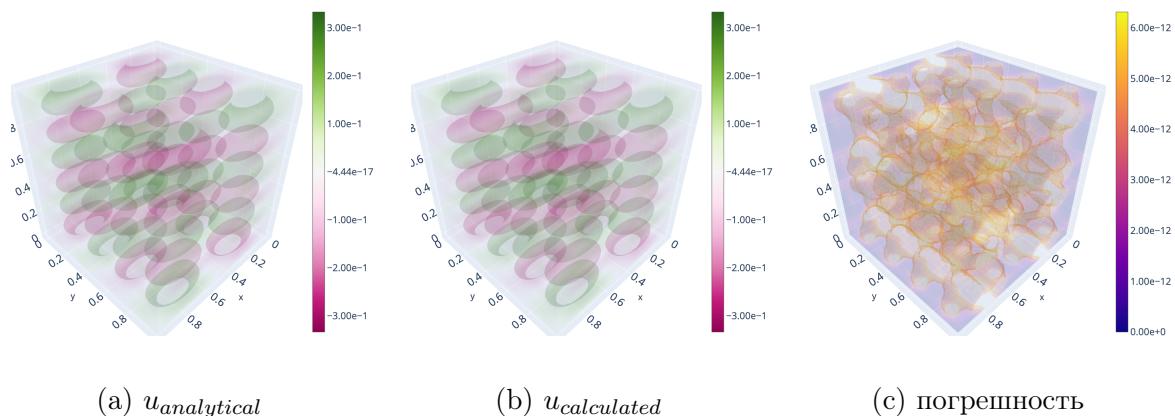
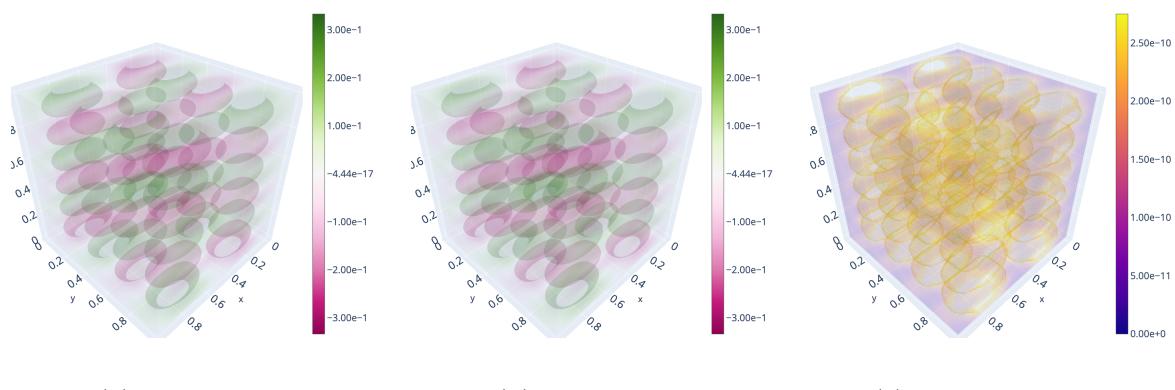
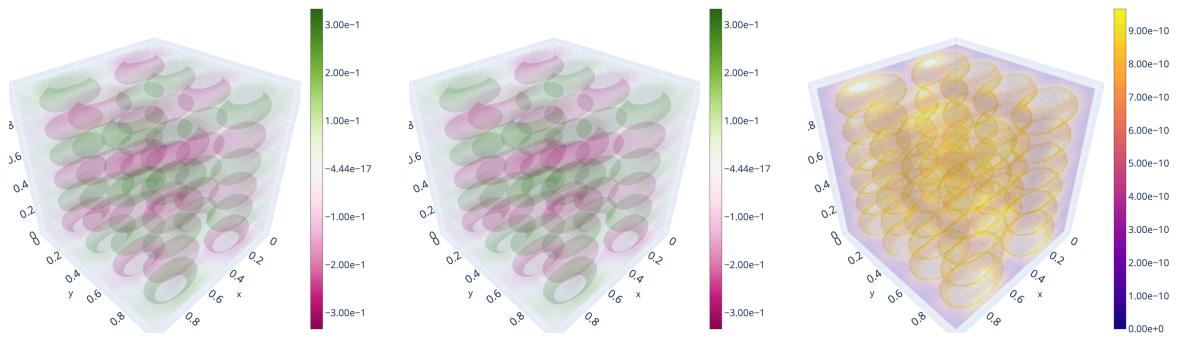


Рис. 1: 1 эпоха



B = 2.10

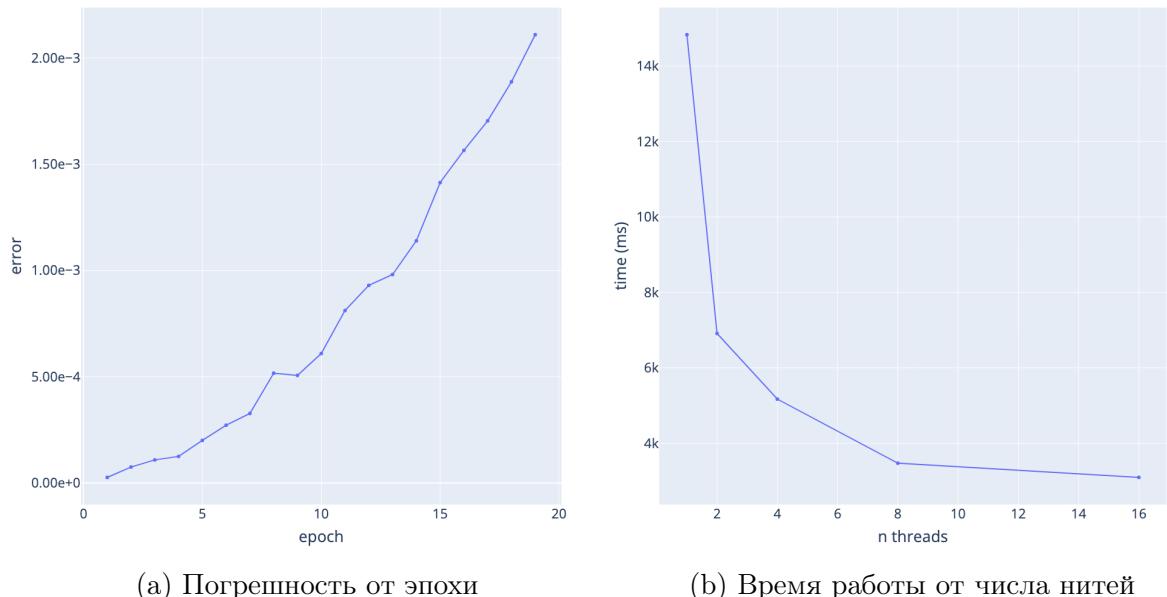


(a) $u_{analytical}$

(b) $u_{calculated}$

(c) погрешность

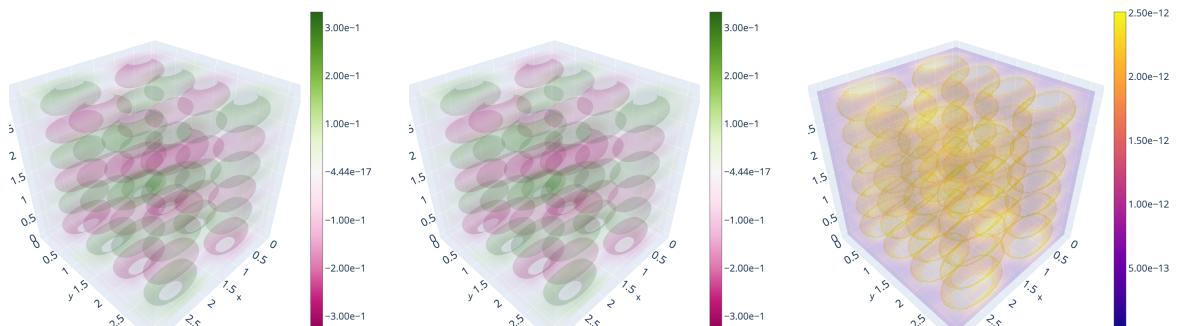
Рис. 3: 20 эпоха



(a) Погрешность от эпохи

(b) Время работы от числа нитей

4.1.2 Графики для N=256

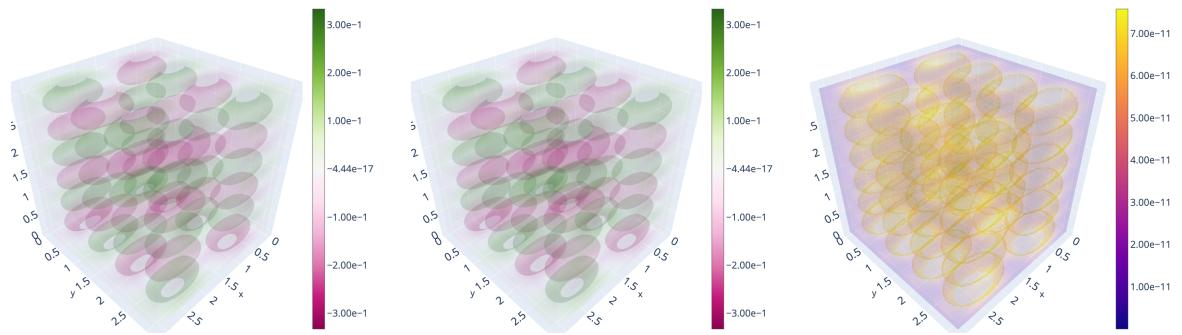


(a) $u_{analytical}$

(b) $u_{calculated}$

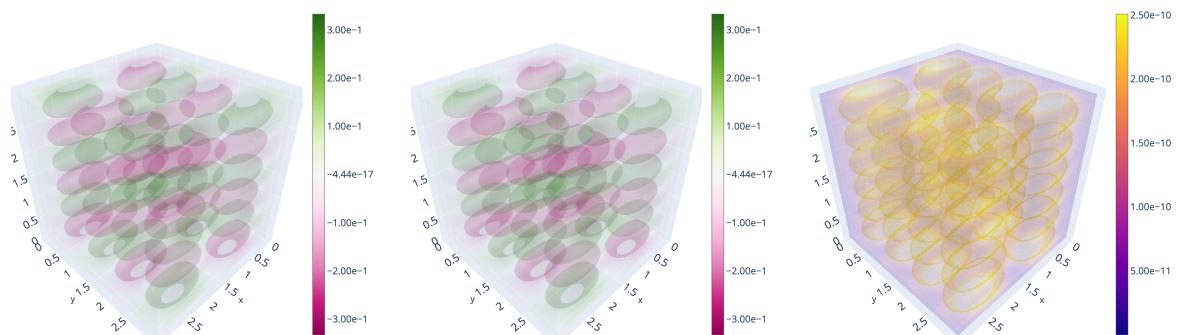
(c) погрешность

Рис. 5: 1 эпоха

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

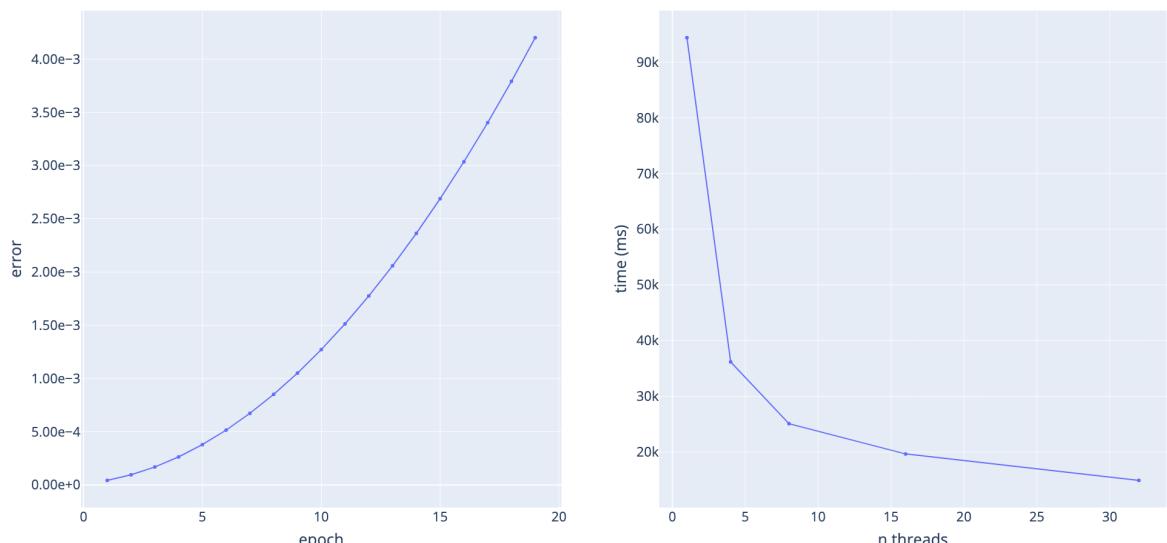
(c) погрешность

Рис. 6: 10 эпоха

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

(c) погрешность

Рис. 7: 20 эпоха



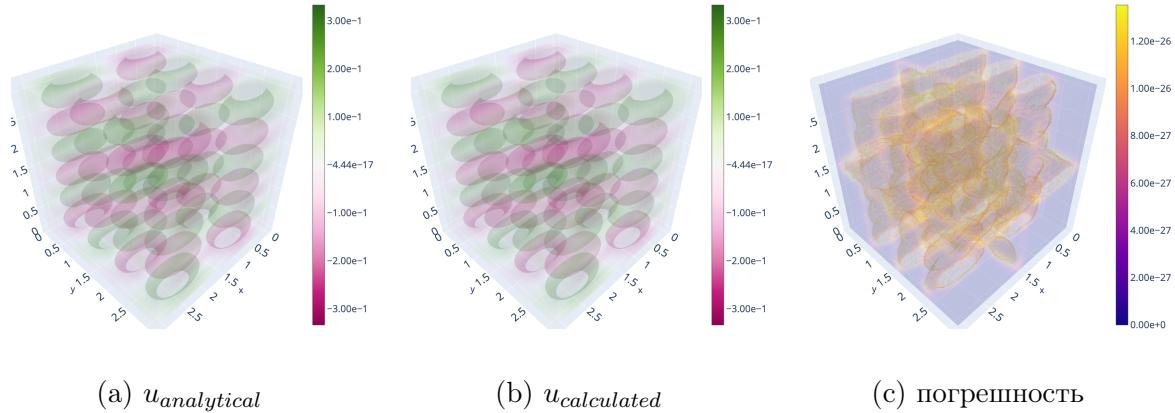
(a) Погрешность от эпохи

(b) Время работы от числа нитей

4.2 $L = \pi$

Число OpenMP нитей	Число точек сетки N^3	Время реше- ния T (ms)	Ускорение S	Погрешность δ
1	128^3	13009.1	1	0.000212559
2	128^3	9545.68	1.36	0.000212559
4	128^3	5383.07	2.42	0.000212559
8	128^3	4221.77	3.08	0.000212559
16	128^3	3620.54	3.59	0.000212559
1	256^3	98665.7	1	0.00042571
4	256^3	37394.8	2.64	0.00042571
8	256^3	24504.9	4.03	0.00042571
16	256^3	19678.3	5.01	0.00042571
32	256^3	16161.1	6.11	0.00042571

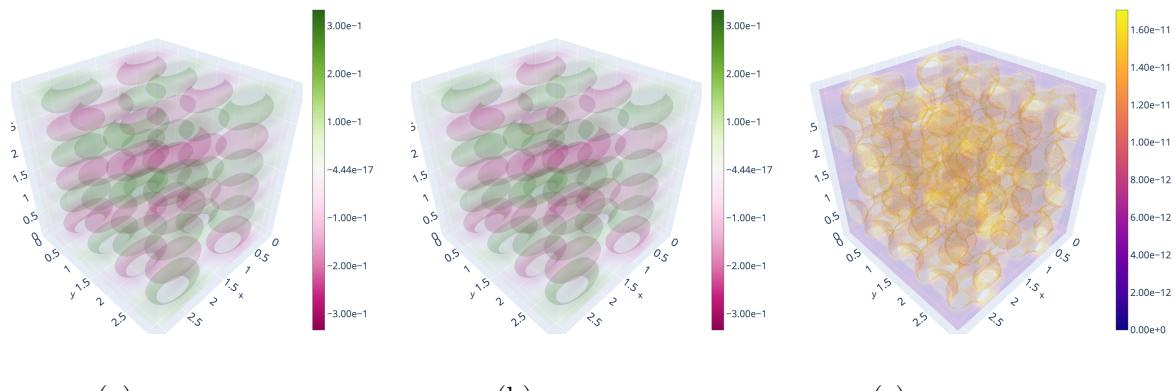
4.2.1 Графики для $N=128$



(b) $u_{calculated}$

(c) погрешность

Рис. 9: 1 эпоха



(b) $u_{calculated}$

(c) погрешность

Рис. 10: 10 эпоха

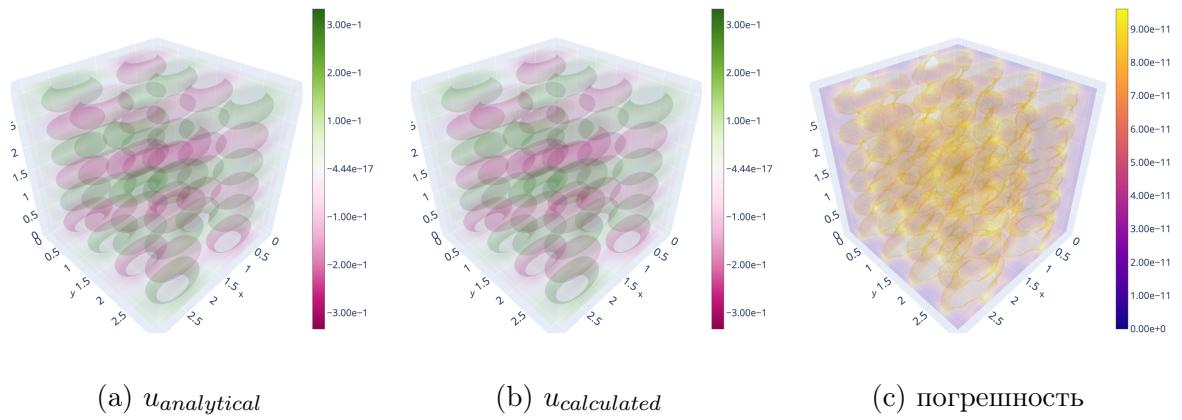
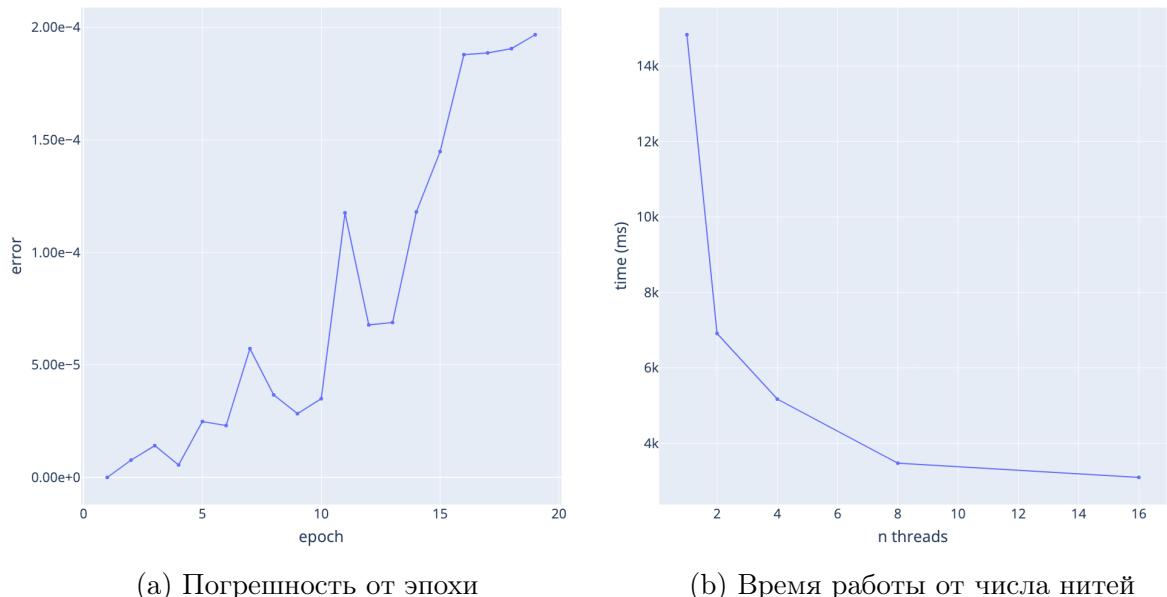


Рис. 11: 20 эпоха



4.2.2 Графики для N=256

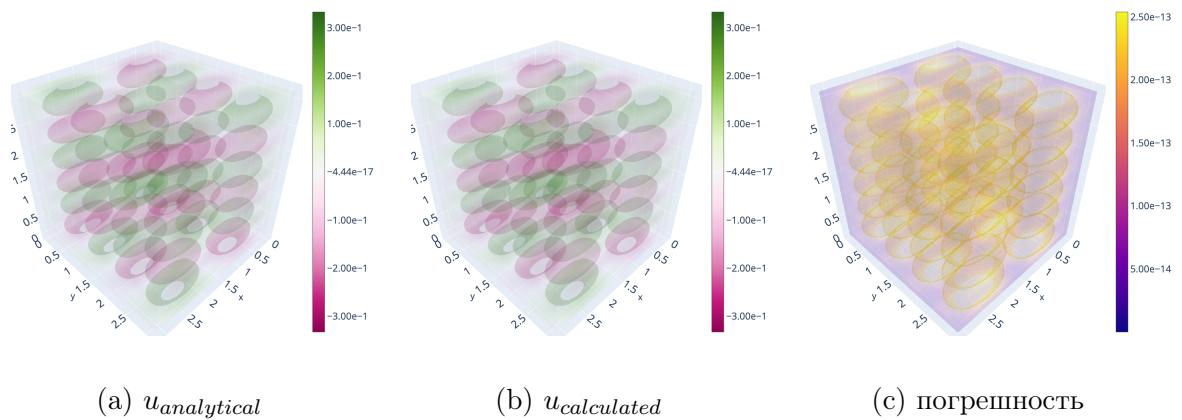


Рис. 13: 1 эпоха

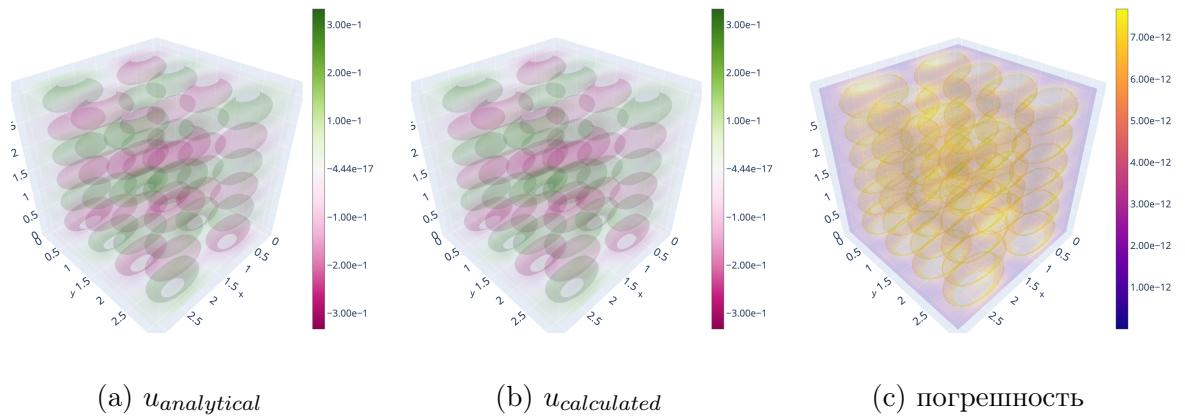


Рис. 14: 10 эпоха

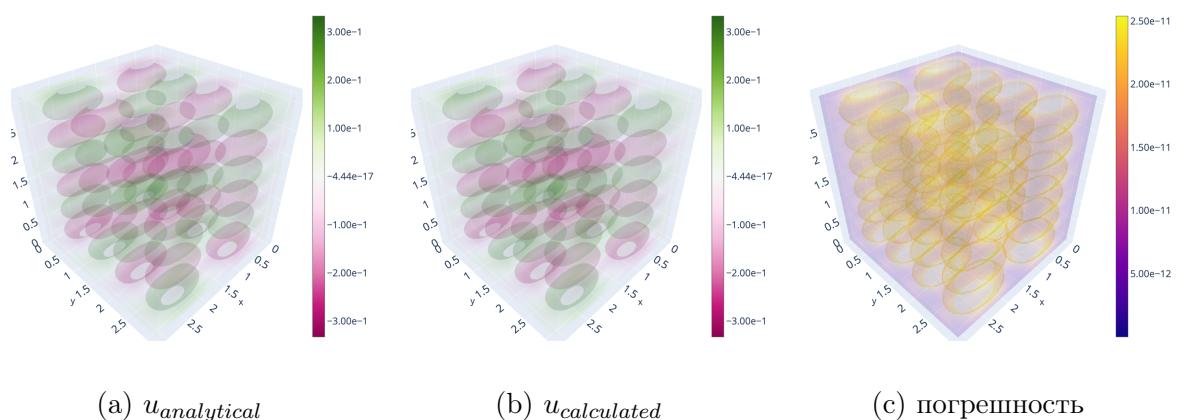
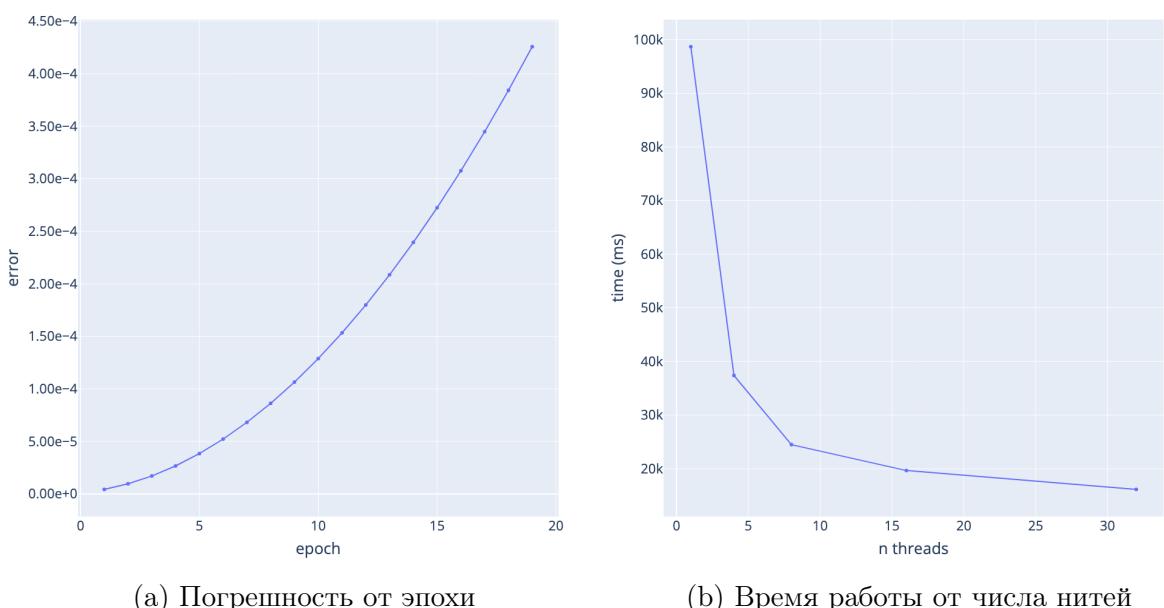


Рис. 15: 20 эпоха



5 Заключение

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

- На сетке 256 более заметно ускорение от числа нитей. Скорее всего, это связано с большей долей во времени работы программы непосредственно вычислений по сравнению с побочными задачами вроде копирования тензоров;
- На сетке 256 лучше видно, по какой функции расходится явная разностная схема;
- На сетке 256 вычисленное решение гораздо ближе прилегает к аналитическому (это видно по форме поверхностей на графиках погрешности).