

Отчет о научно-исследовательской практике

ФИО и должность практиканта: Макоев Артур Аланович, студент

Научный руководитель: Ганкевич Иван Геннадьевич, кандидат физ.-мат. наук, доцент

Руководитель практики от СПбГУ: Погожев Сергей Владимирович, кандидат физ.-мат. наук, доцент

Тема задания: Сравнение производительности методов моделирования распространения звука в замкнутом помещении с использованием вычислений на процессоре и на видеокарте

Место и сроки прохождения: СПбГУ, 28.04.2021 - 14.05.2021

Цели и задачи практики

- Сравнительный анализ производительности методов моделирования распространения звука в замкнутом помещении с использованием вычислений на процессоре и на видеокарте, работа с вычислительной платформой СПбГУ
- Виды выполненных работ:
 1. Изучение разработки приложений с использованием библиотеки OpenGL
 2. Написание алгоритмов решения волнового уравнения на C++ для платформы CUDA
 3. Написание программы, которая использует написанные алгоритмы и визуализирует результаты работы метода.
 4. Проведение сравнительного анализа производительности алгоритмов, запуск бенчмарка на нескольких конфигурациях железа, в том числе на базе вычислительной платформы СПбГУ.

Основные результаты практики

- Изучены способы работы с библиотекой OpenGL и инструменты для работы с программным кодом CUDA.
- Написаны 8 методов, вычисляющие приближенное решение волнового уравнения в двумерной области, основанные на методе конечных разностей.
- Создано кроссплатформенное приложение, применяющее методы решения волнового уравнения и визуализирующее результаты вычислений.
- Проведен сравнительный анализ производительности методов и рассмотрено влияние приемов оптимизации кода на производительность.

Алгоритм №1

Волновое уравнение задается в виде:

$$\frac{\partial^2 u}{\partial t^2} + b \frac{\partial u}{\partial t} - v^2 \Delta u = f(x, t), \quad \Delta u = \sum_{i=1}^n \frac{\partial^2 u}{\partial x_i^2}$$

с нулевым граничным условием $u(\Gamma) = 0$

В алгоритме используется трёхслойная разностная схема с приближением второго порядка:

$$u_{ij}^{t+1} = \frac{2u_{ij}^t - u_{ij}^{t-1} \left(1 - \frac{\Delta t}{2} b\right) + v^2 \frac{\Delta t^2}{\Delta x^2} (u_{i-1j}^t - 2u_{ij}^t + u_{i+1j}^t + u_{ij-1}^t - 2u_{ij}^t + u_{ij+1}^t)}{\left(1 + \frac{\Delta t}{2} b\right)}$$

Алгоритм №2

Схема вычисления (p, v_x, v_y) :

$$p^+ = B(p - C \nabla \cdot \vec{v})$$

$$v_x^+ = BB_{<}(v_x - C \nabla_x p^+) + (B_{<} - B)(BY_{<} + B_{<}Y)(Bp^+ + B_{<}p_{<}^+)$$

$$v_y^+ = BB_V(v_y - C \nabla_y p^+) + (B_V - B)(BY_V + B_VY)(Bp^+ + B_Vp_V^+)$$

$*^+$ – знак плюс обозначает значения переменных на следующем шаге,

$*_{<}$ и $*_V$ обозначают значения в положениях $\{(x - \Delta x, y), (x, y - \Delta y)\}$ соответственно.

C – константа, равная $C = c \frac{\Delta t}{\Delta x}$, где c – скорость распространения волны, Δt и Δx – шаги сетки.

B – индикатор твердой поверхности, равен нулю если в точке твердая граница, единице для свободного пространства.

$Y = (1 - R)/(1 + R)$, где R – коэффициент отражательной способности поверхности. Для среды считается равным нулю.

$$\nabla_x p^+ = p^+ - p_{<}^+$$

$$\nabla_y p^+ = p^+ - p_V^+$$

$$\nabla \cdot \vec{v} = (v_x(x + \Delta x, y) - v_x(x, y)) + (v_y(x, y + \Delta y) - v_y(x, y))$$

Список методов

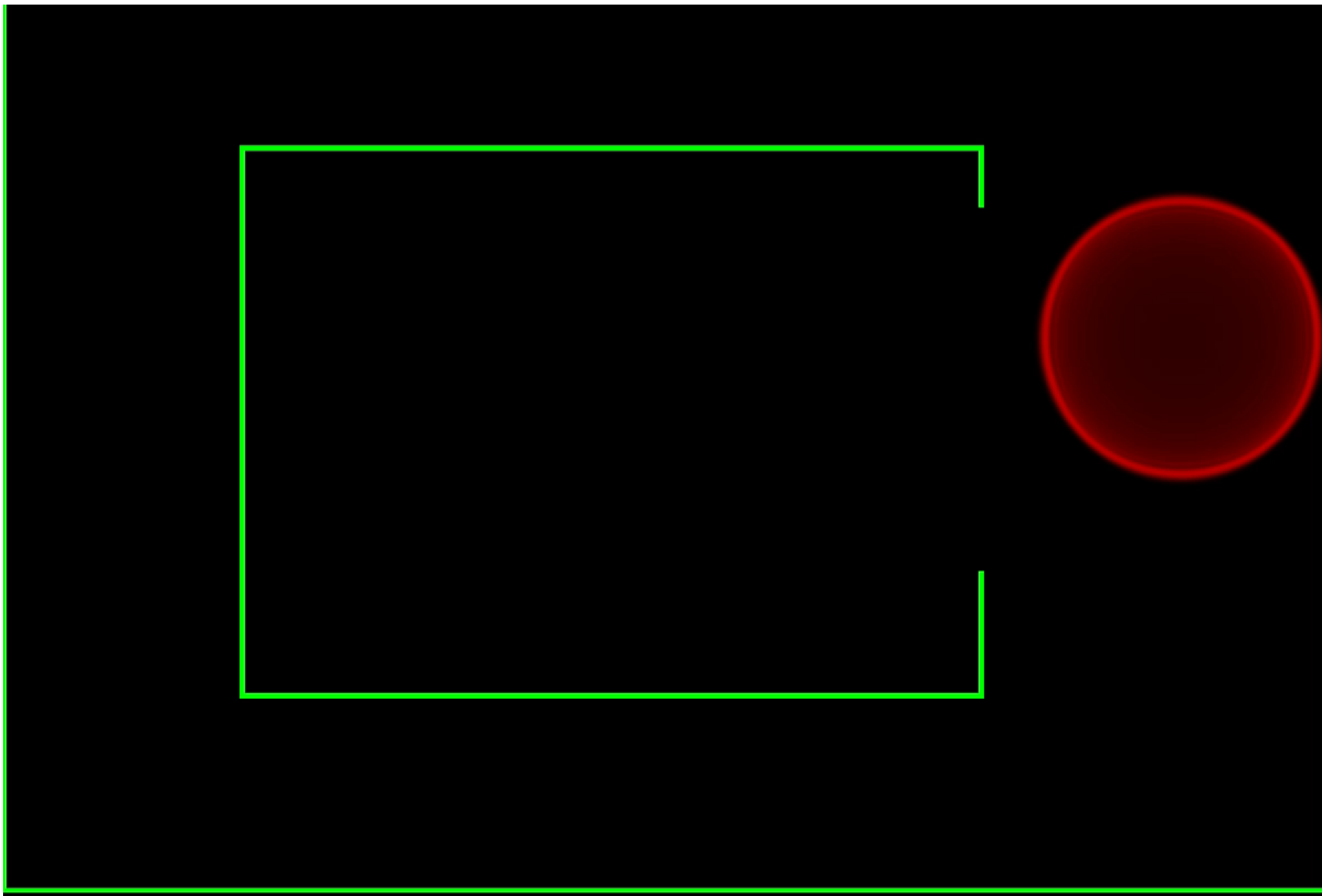
1. Wavelteration – однопоточный метод на CPU, выполняющий одну итерацию симуляции.
2. WavelterationOMP – многопоточный метод на CPU, выполняющий одну итерацию симуляции.
3. WavelterationOMPMultipleFrames – многопоточный метод на CPU, выполняющий несколько итераций симуляции за раз.
4. WavelterationKernel – метод на GPU, выполняющий одну итерацию симуляции.
5. MultipleIterationsKernel – метод на GPU, выполняющий несколько итераций симуляции за раз с помощью динамического параллелизма. В этом случае запускается малое основное ядро на сетке с только одним потоком, которое уже запускает основную большую сетку для расчета итерации. В этом случае синхронизация выполнения работы не требует обращения к CPU, что потенциально увеличивает производительность.
6. WavelterationKernelSM – метод на GPU, выполняющий одну итерацию симуляции, использующий общую память. В этом случае необходимые данные сначала копируются в память, доступную одному блоку. Производительность повышается за счет того, что обращение к элементу в общей памяти блока происходит быстрее, чем обращение в глобальную область памяти, и за счет того, что один элемент используется в нескольких вычислительных нитях ядра.
7. WavelterationAltOMP – многопоточный метод на CPU, выполняющий одну итерацию симуляции для альтернативного алгоритма №2.
8. WavelterationKernelAlt – метод на GPU, выполняющий одну итерацию симуляции для альтернативного алгоритма №2.

Режимы бенчмарков

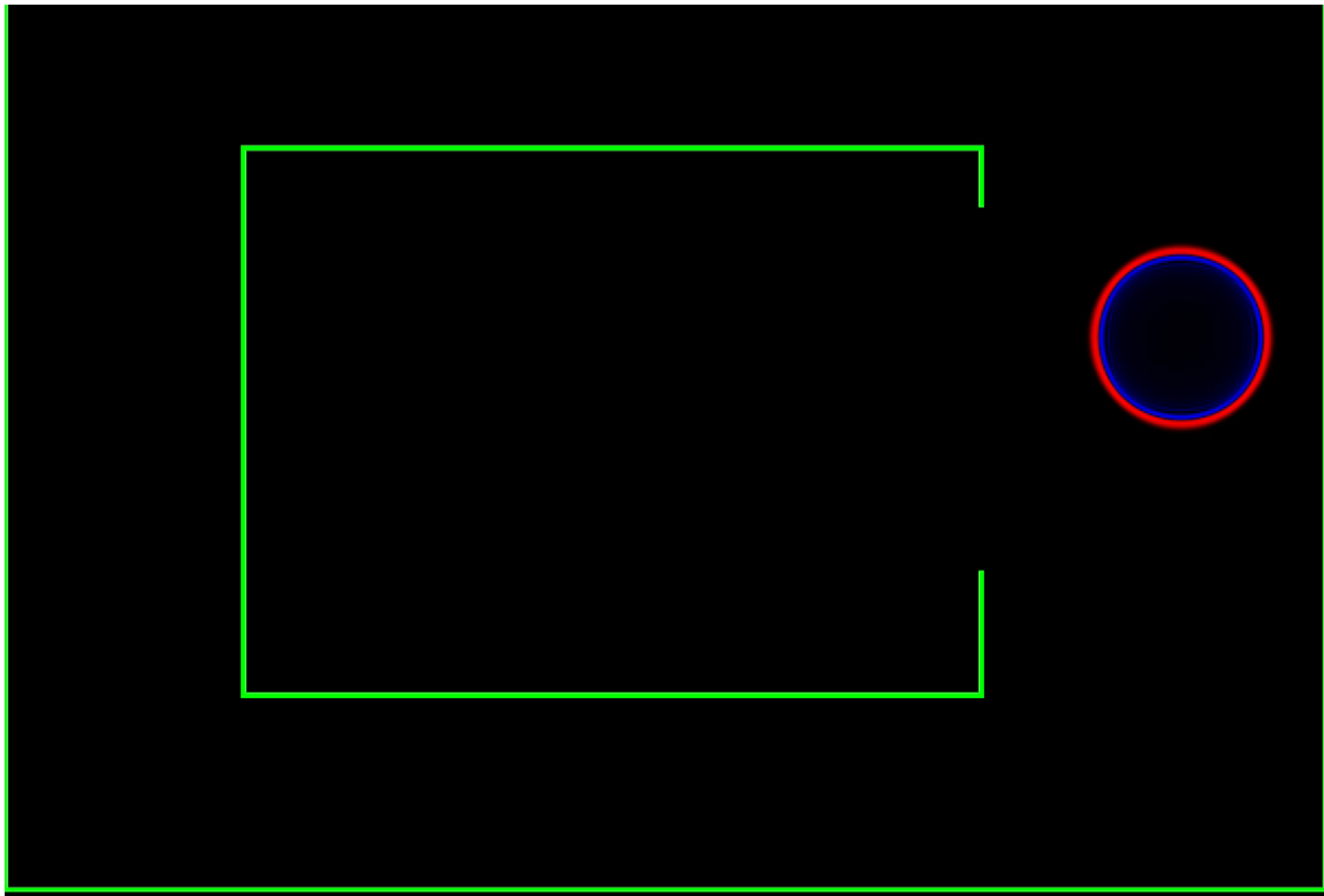
- 1) Сравнение производительности, времени одной итерации симуляции, для всех методов на разных размерах поля $N \times N$ для N от 500 до 5000 с шагом в 500.
- 2) Сравнение производительности метода WavelterationOMP на различном числе потоков.
- 3) Сравнение производительности метода WavelterationOMPAlt на различном числе потоков.
- 4) Сравнение производительности метода WavelterationKernel на сетке с блоками размером в $N \times N$ нитей, для N от 1 до 32.
- 5) Сравнение производительности метода WavelterationKernelAlt на сетке с блоками размером в $N \times N$ нитей, для N от 1 до 32.

Для замера времени одной итерации берется среднее арифметическое от 100 итераций. Кроме того, для этих 100 итераций высчитывается среднеквадратичное отклонение, максимальное и минимальное значения.

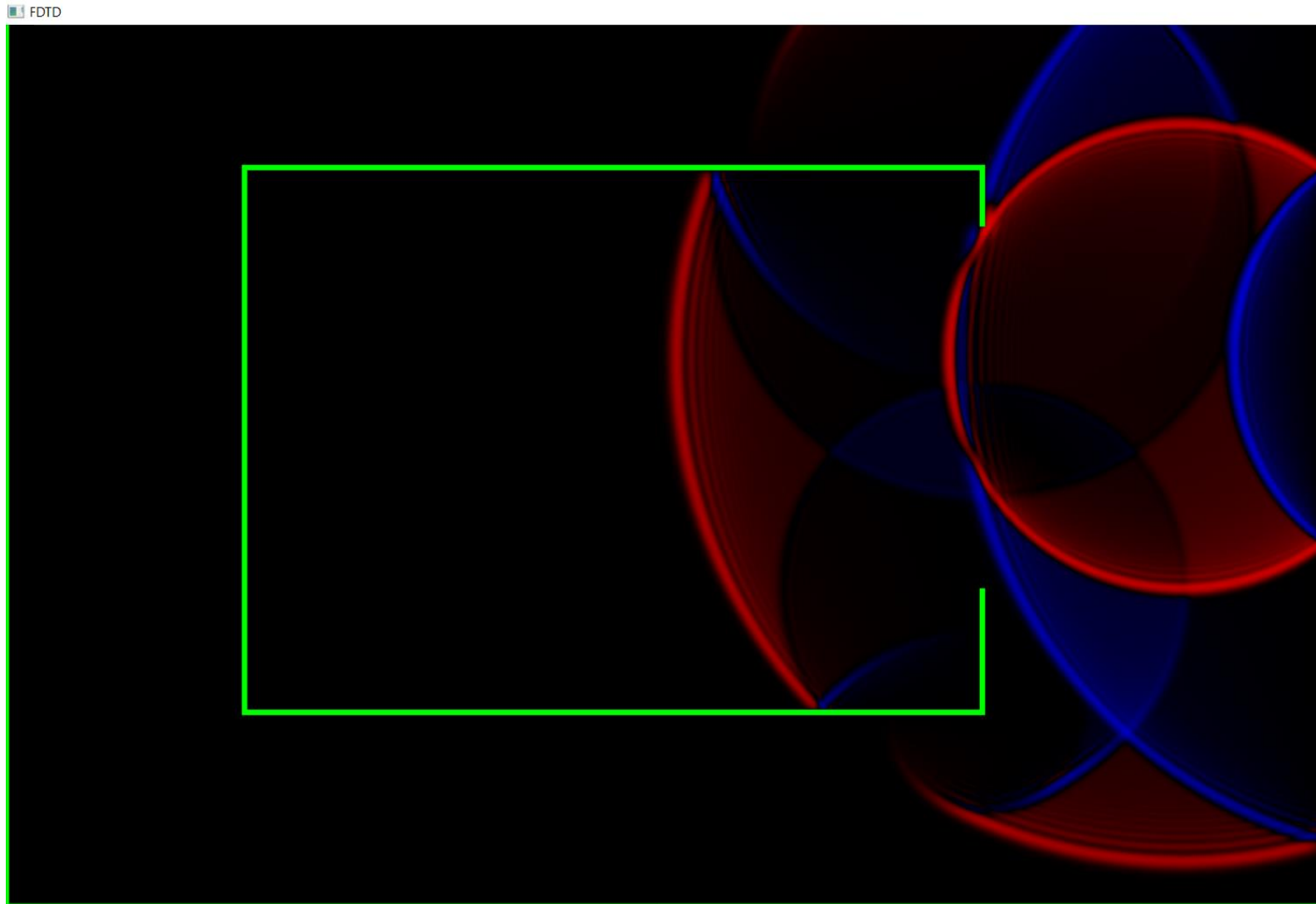
Примеры работы программы

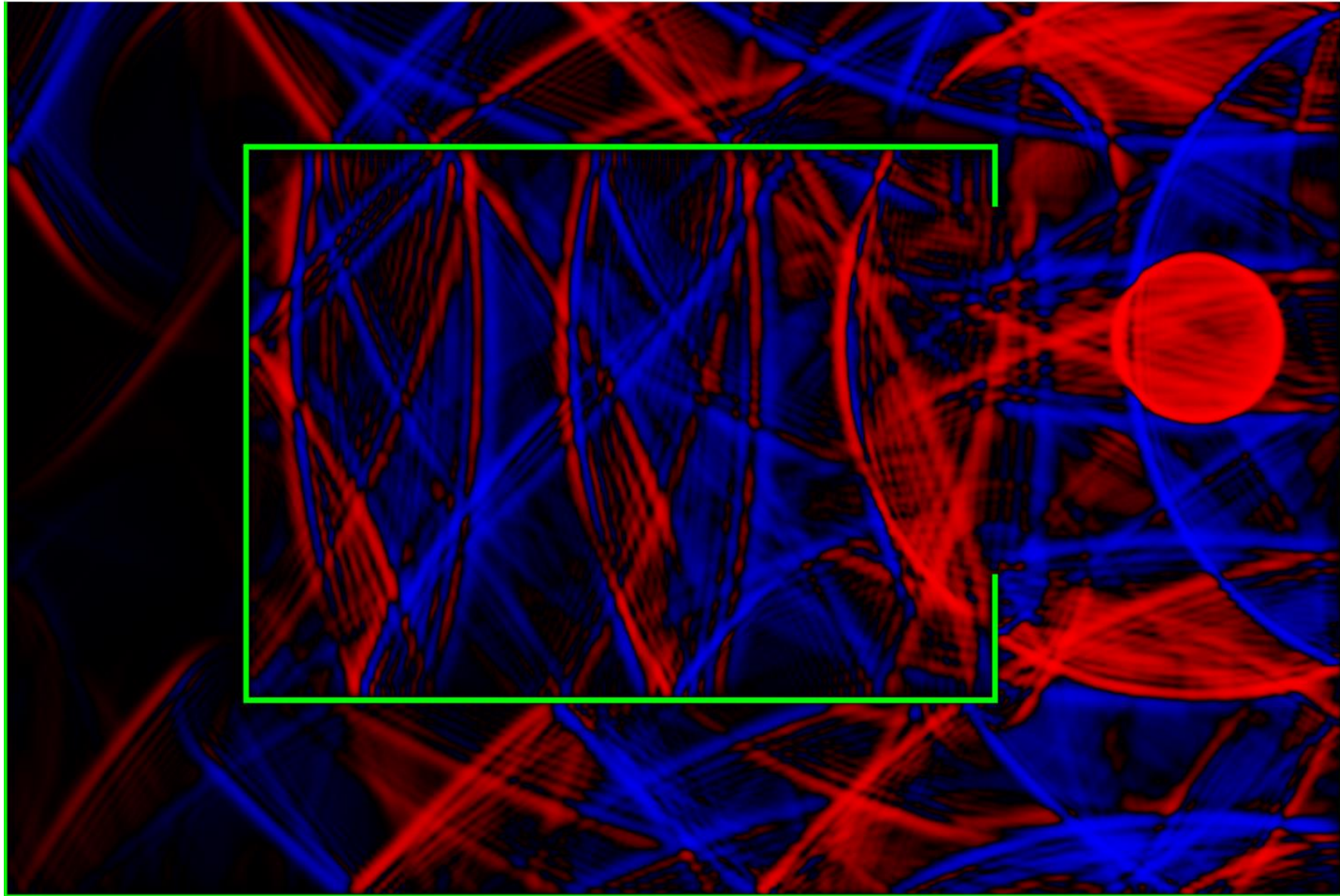


Примеры работы программы

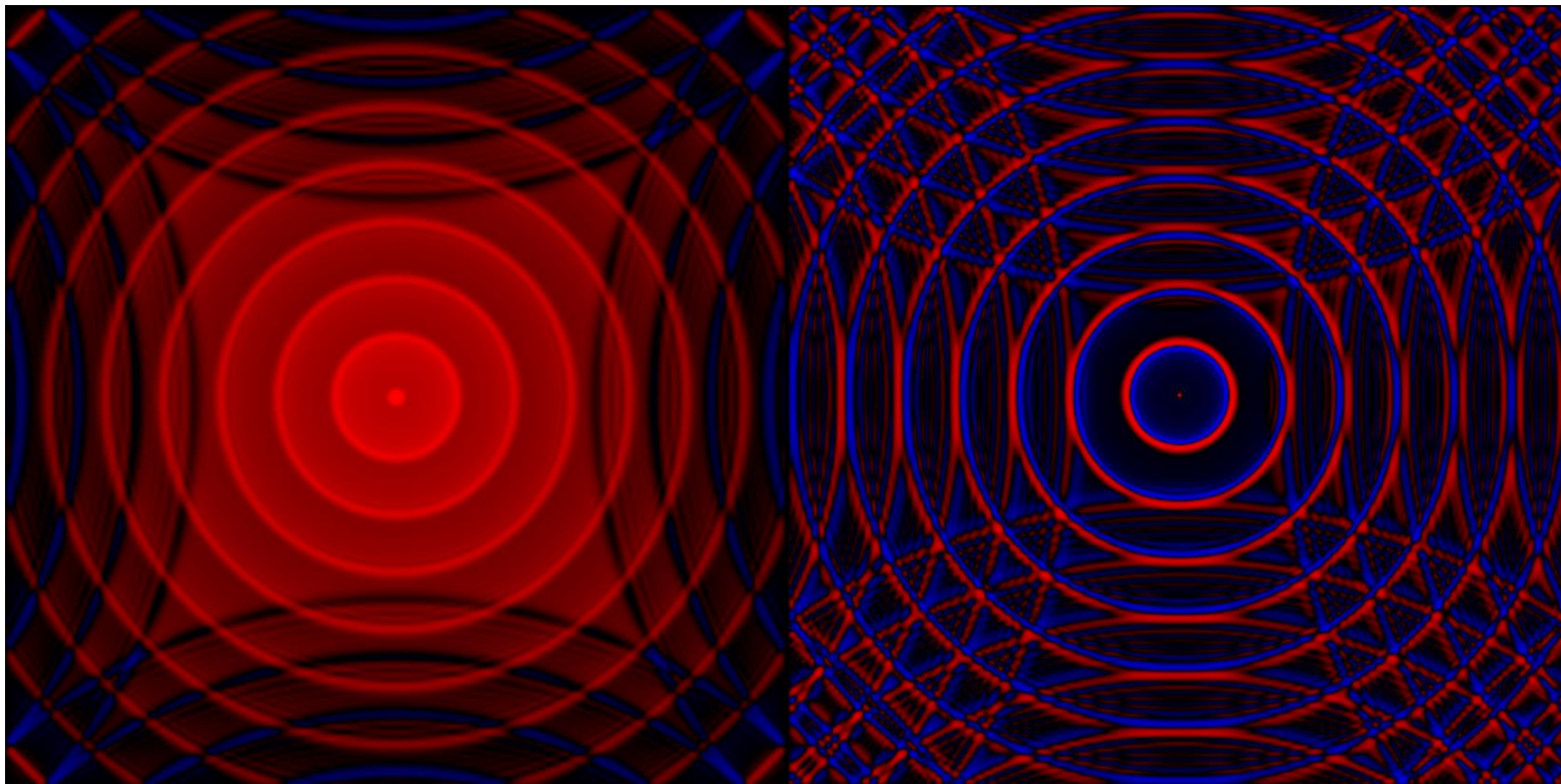


Примеры работы программы






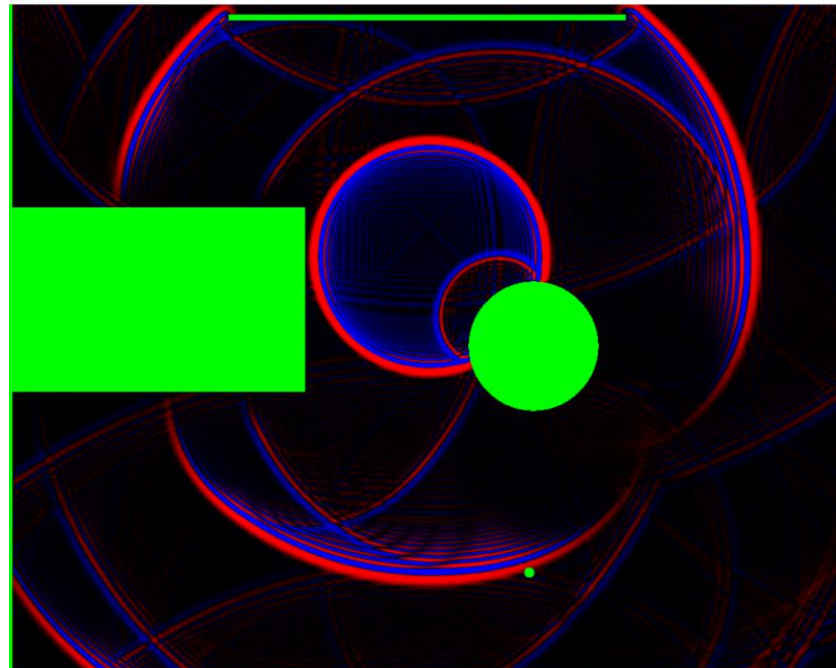
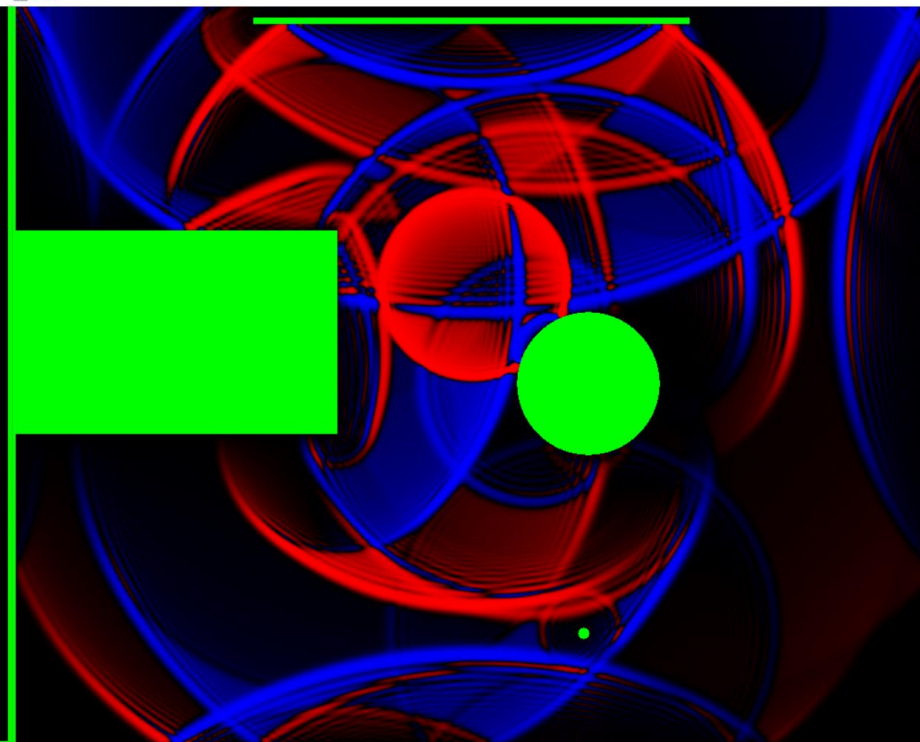
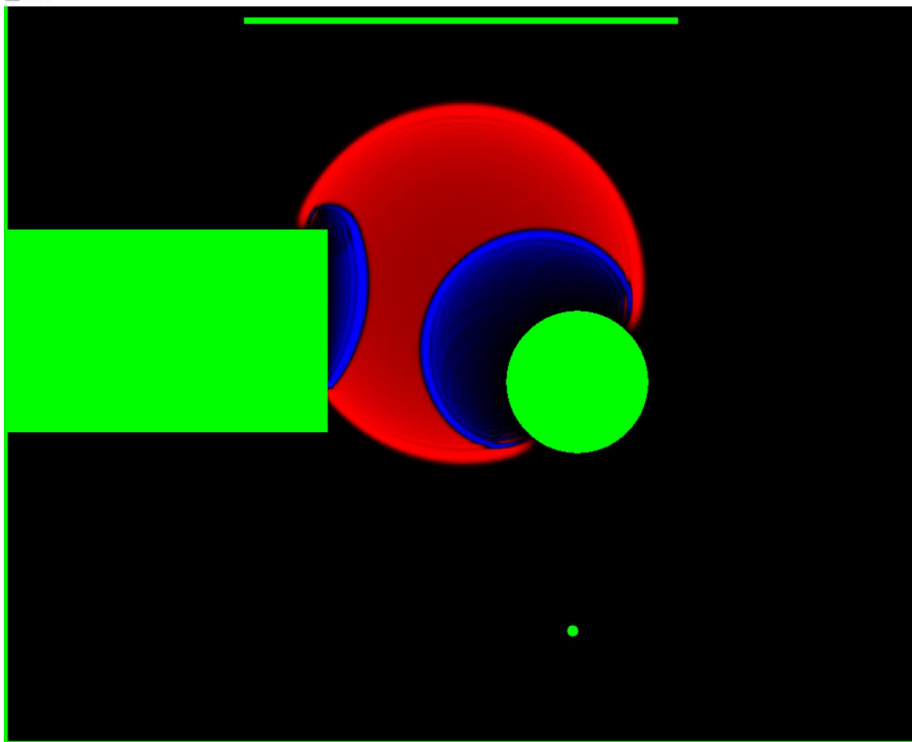
Слева - алгоритм №1, справа - алгоритм №2



Пример вывода в консоль

 Консоль отладки Microsoft Visual Studio

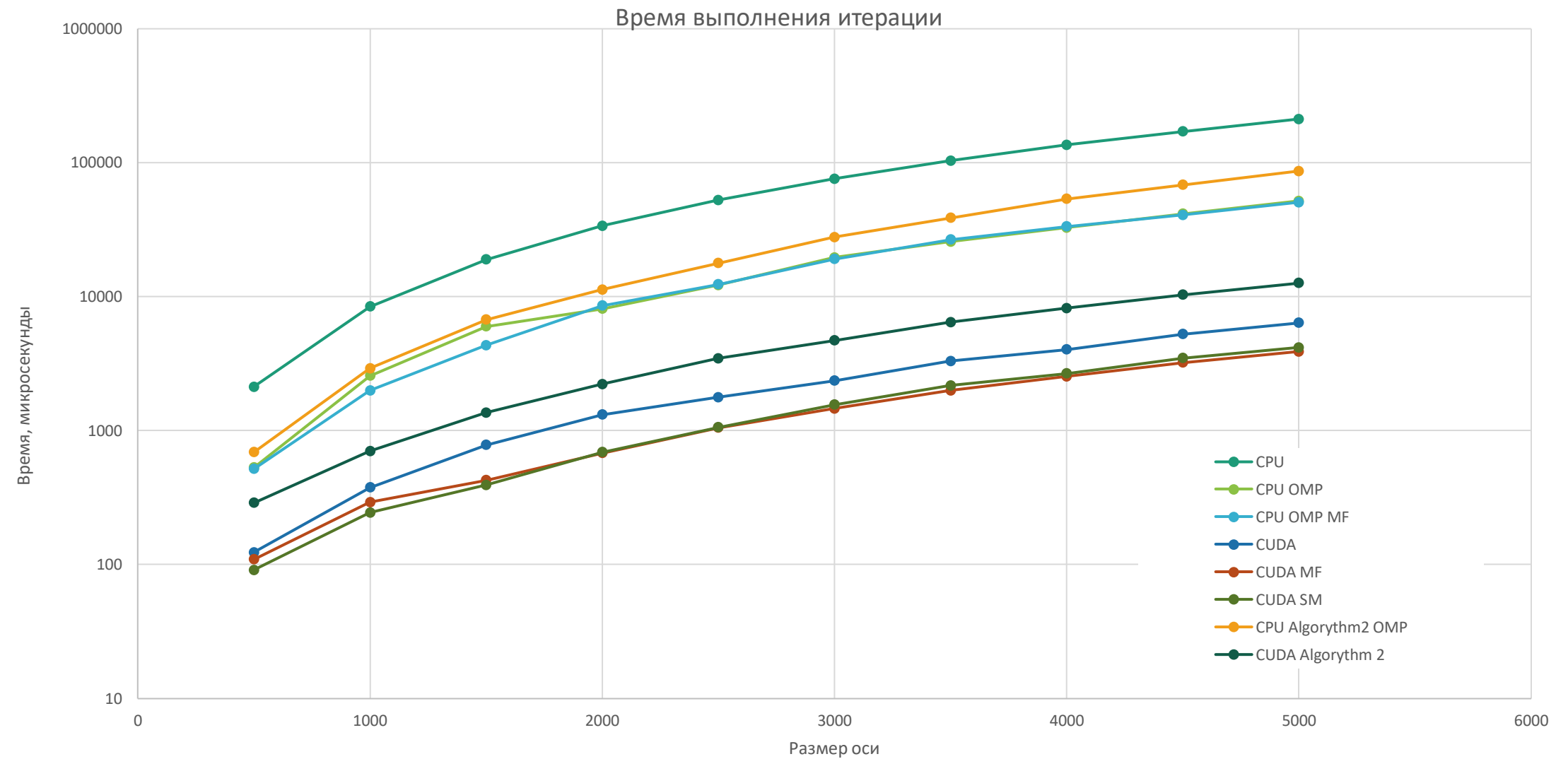
```
Method: WaveIterationKernelSM
mean: 265 microseconds, Sigma = 45.8476 min,max= 218 541
mean: 287 microseconds, Sigma = 29.8161 min,max= 228 379
mean: 284 microseconds, Sigma = 35.609 min,max= 231 364
mean: 291 microseconds, Sigma = 41.6653 min,max= 234 595
mean: 283 microseconds, Sigma = 33.2114 min,max= 234 358
mean: 286 microseconds, Sigma = 33.8083 min,max= 228 352
mean: 278 microseconds, Sigma = 36.2767 min,max= 224 357
mean: 277 microseconds, Sigma = 31.0966 min,max= 222 359
mean: 284 microseconds, Sigma = 36.7287 min,max= 227 395
mean: 278 microseconds, Sigma = 33.3766 min,max= 237 334
mean: 280 microseconds, Sigma = 33.8674 min,max= 229 343
mean: 276 microseconds, Sigma = 33.7194 min,max= 211 333
mean: 284 microseconds, Sigma = 43.909 min,max= 220 601
mean: 271 microseconds, Sigma = 34.6121 min,max= 220 347
mean: 279 microseconds, Sigma = 28.9482 min,max= 234 338
mean: 279 microseconds, Sigma = 29.7993 min,max= 229 350
```



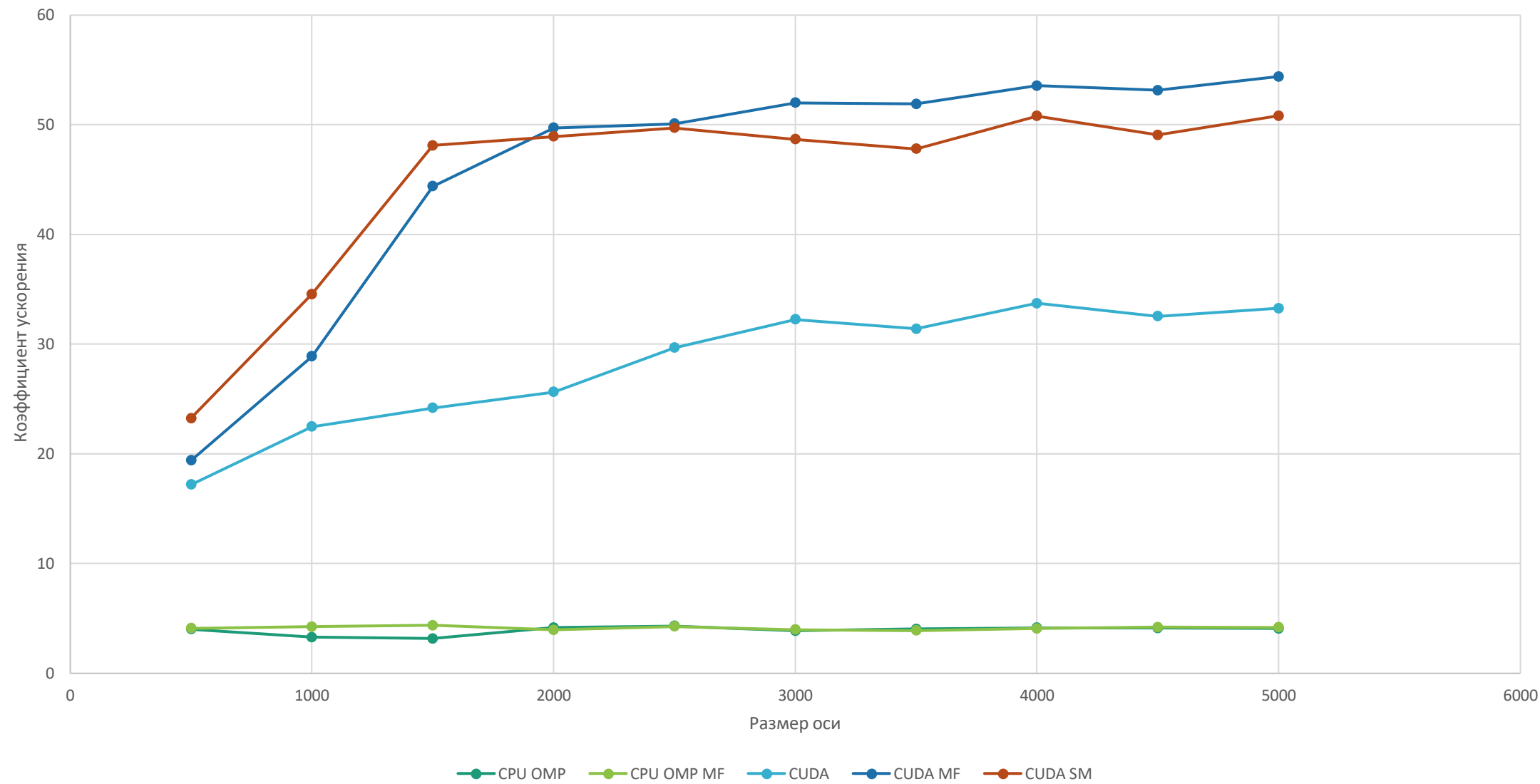
Вывод режима бенчмарка №1

size = 500	Method: WaveIteration	mean = 2114 microseconds, Sigma = 63.1744 min,max= 2071 2186
size = 500	Method: WaveIterationOMP	mean = 528 microseconds, Sigma = 33.6006 min,max= 516 560
size = 500	Method: WaveIterationOMPMultipleFrames	mean = 516 microseconds, Sigma = 15.0997 min,max= 508 786
size = 500	Method: WaveIterationKernel	mean = 123 microseconds, Sigma = 10.3441 min,max= 121 133
size = 500	Method: MultipleIterationsKernel	mean = 109 microseconds, Sigma = 12.3693 min,max= 108 133
size = 500	Method: WaveIterationKernelSM	mean = 91 microseconds, Sigma = 9.16515 min,max= 89 94
size = 500	Method: WaveIterationAltOMP	mean = 689 microseconds, Sigma = 48.9694 min,max= 664 977
size = 500	Method: WaveIterationKernelAlt	mean = 288 microseconds, Sigma = 17.9444 min,max= 284 303
size = 1000	Method: WaveIteration	mean = 8430 microseconds, Sigma = 116.563 min,max= 8350 9068
size = 1000	Method: WaveIterationOMP	mean = 2561 microseconds, Sigma = 760.293 min,max= 1866 3664
size = 1000	Method: WaveIterationOMPMultipleFrames	mean = 1990 microseconds, Sigma = 124.98 min,max= 1910 2253
size = 1000	Method: WaveIterationKernel	mean = 375 microseconds, Sigma = 7.74597 min,max= 372 389
size = 1000	Method: MultipleIterationsKernel	mean = 292 microseconds, Sigma = 22.9347 min,max= 292 372
size = 1000	Method: WaveIterationKernelSM	mean = 244 microseconds, Sigma = 19.799 min,max= 232 403
size = 1000	Method: WaveIterationAltOMP	mean = 2917 microseconds, Sigma = 158.386 min,max= 2761 3784
size = 1000	Method: WaveIterationKernelAlt	mean = 703 microseconds, Sigma = 31.5278 min,max= 688 754
size = 1500	Method: WaveIteration	mean = 18858 microseconds, Sigma = 84.1724 min,max= 18782 18945
size = 1500	Method: WaveIterationOMP	mean = 5984 microseconds, Sigma = 1677.98 min,max= 4234 7974
size = 1500	Method: WaveIterationOMPMultipleFrames	mean = 4327 microseconds, Sigma = 72.298 min,max= 4258 4393
size = 1500	Method: WaveIterationKernel	mean = 780 microseconds, Sigma = 64.3972 min,max= 602 828
size = 1500	Method: MultipleIterationsKernel	mean = 425 microseconds, Sigma = 29.4279 min,max= 409 444
size = 1500	Method: WaveIterationKernelSM	mean = 392 microseconds, Sigma = 24.0832 min,max= 376 418
size = 1500	Method: WaveIterationAltOMP	mean = 6706 microseconds, Sigma = 580.681 min,max= 6085 8444
size = 1500	Method: WaveIterationKernelAlt	mean = 1360 microseconds, Sigma = 117.162 min,max= 1285 1714
size = 2000	Method: WaveIteration	mean = 33647 microseconds, Sigma = 421.305 min,max= 33505 36784
size = 2000	Method: WaveIterationOMP	mean = 8097 microseconds, Sigma = 826.986 min,max= 7552 14103
size = 2000	Method: WaveIterationOMPMultipleFrames	mean = 8533 microseconds, Sigma = 1374.35 min,max= 7644 12275
size = 2000	Method: WaveIterationKernel	mean = 1313 microseconds, Sigma = 86.3308 min,max= 1258 1694

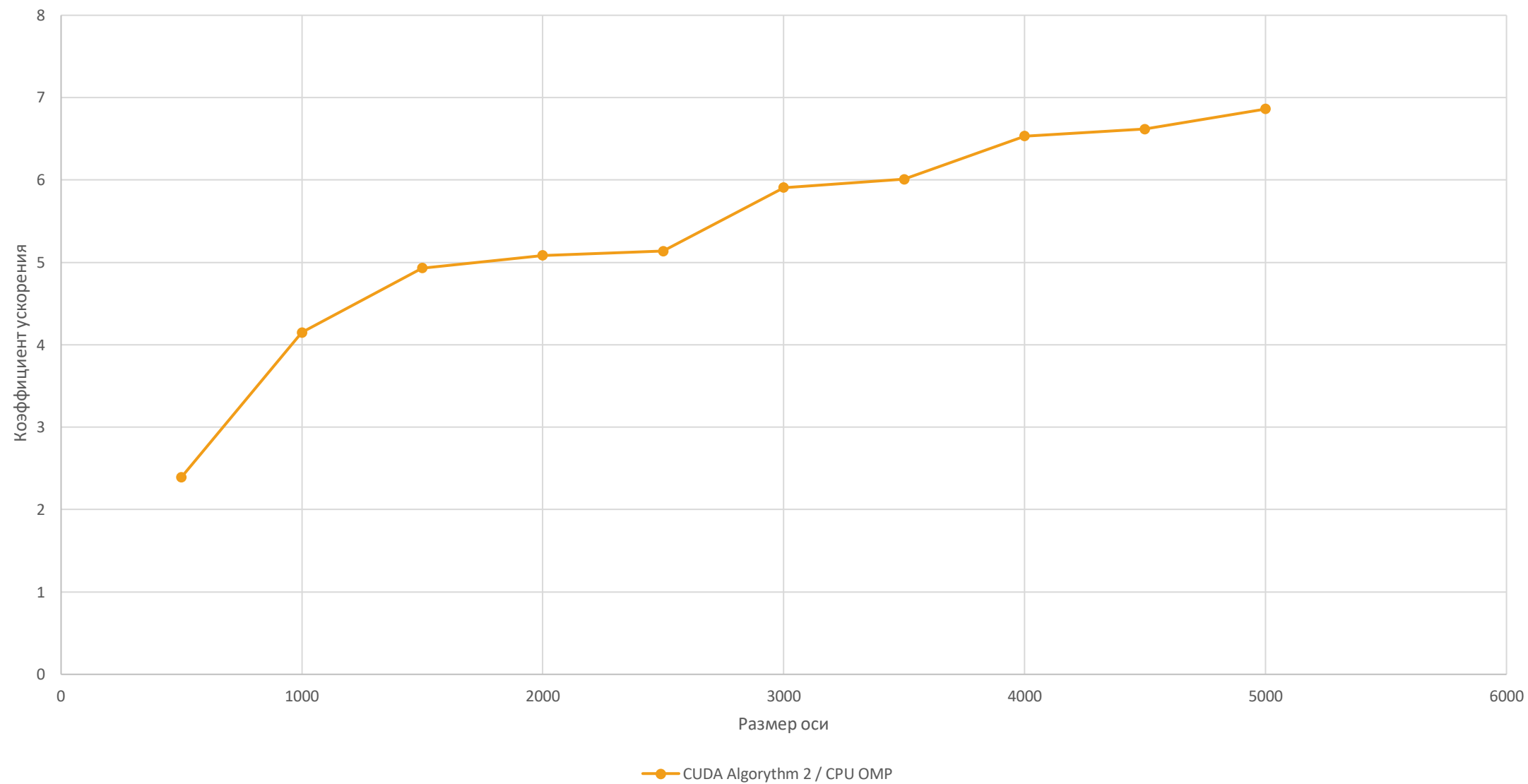
Результаты бенчмарка для типа данных float, запущенном на ПК



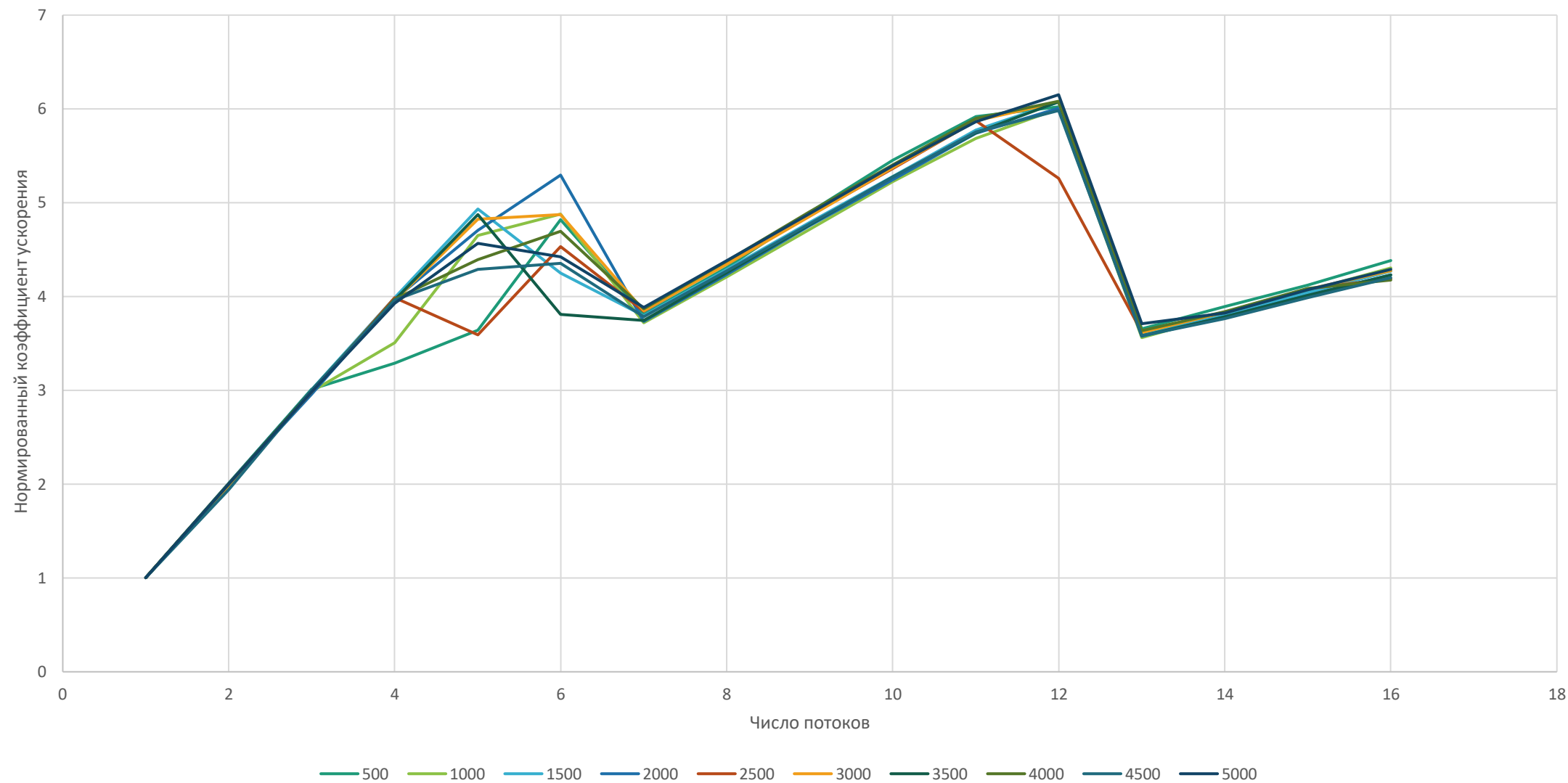
Ускорение по сравнению с однопоточным алгоритмом



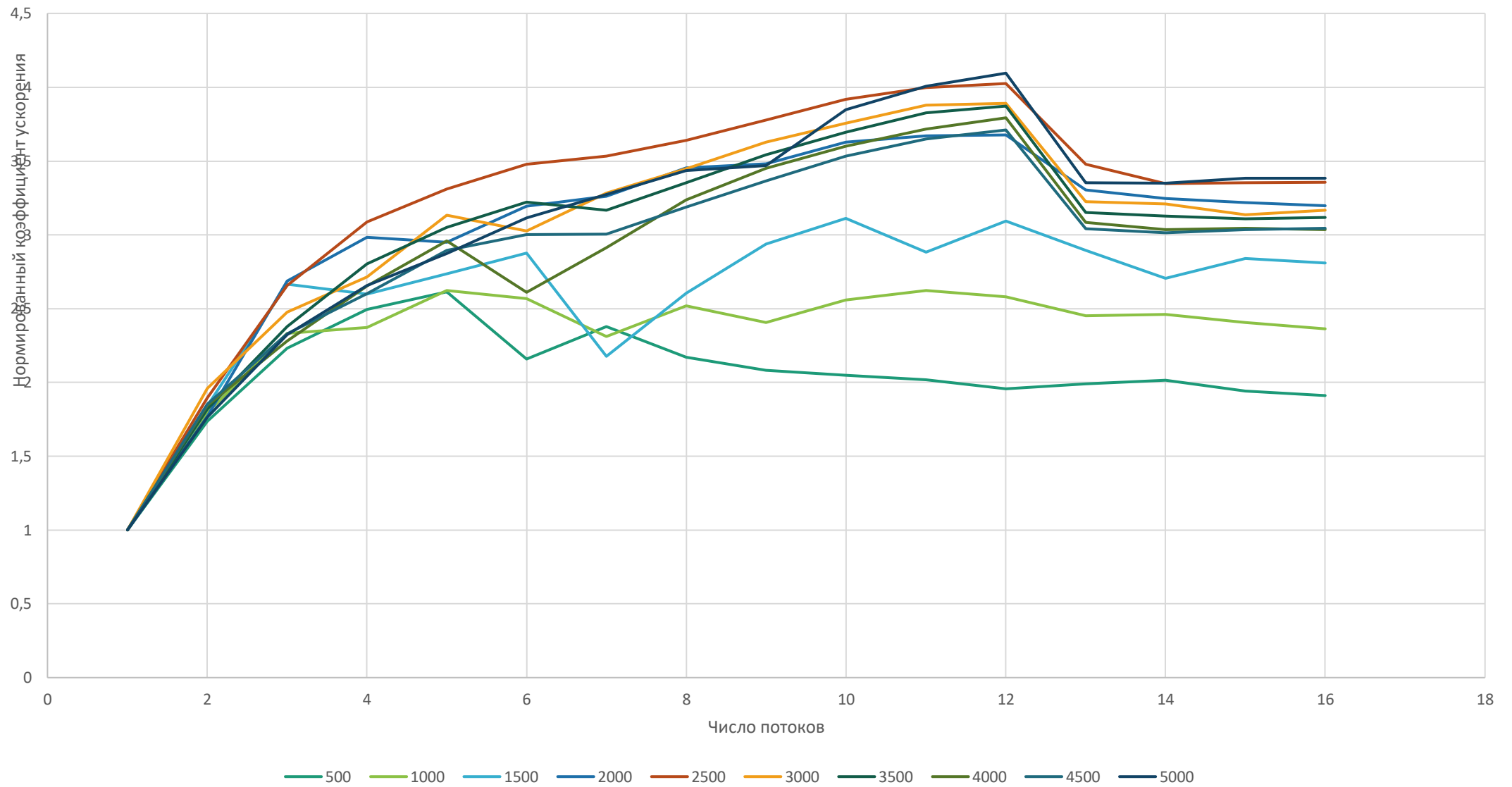
Ускорение Алгоритма №2 на CUDA по сравнению с алгоритмом на CPU



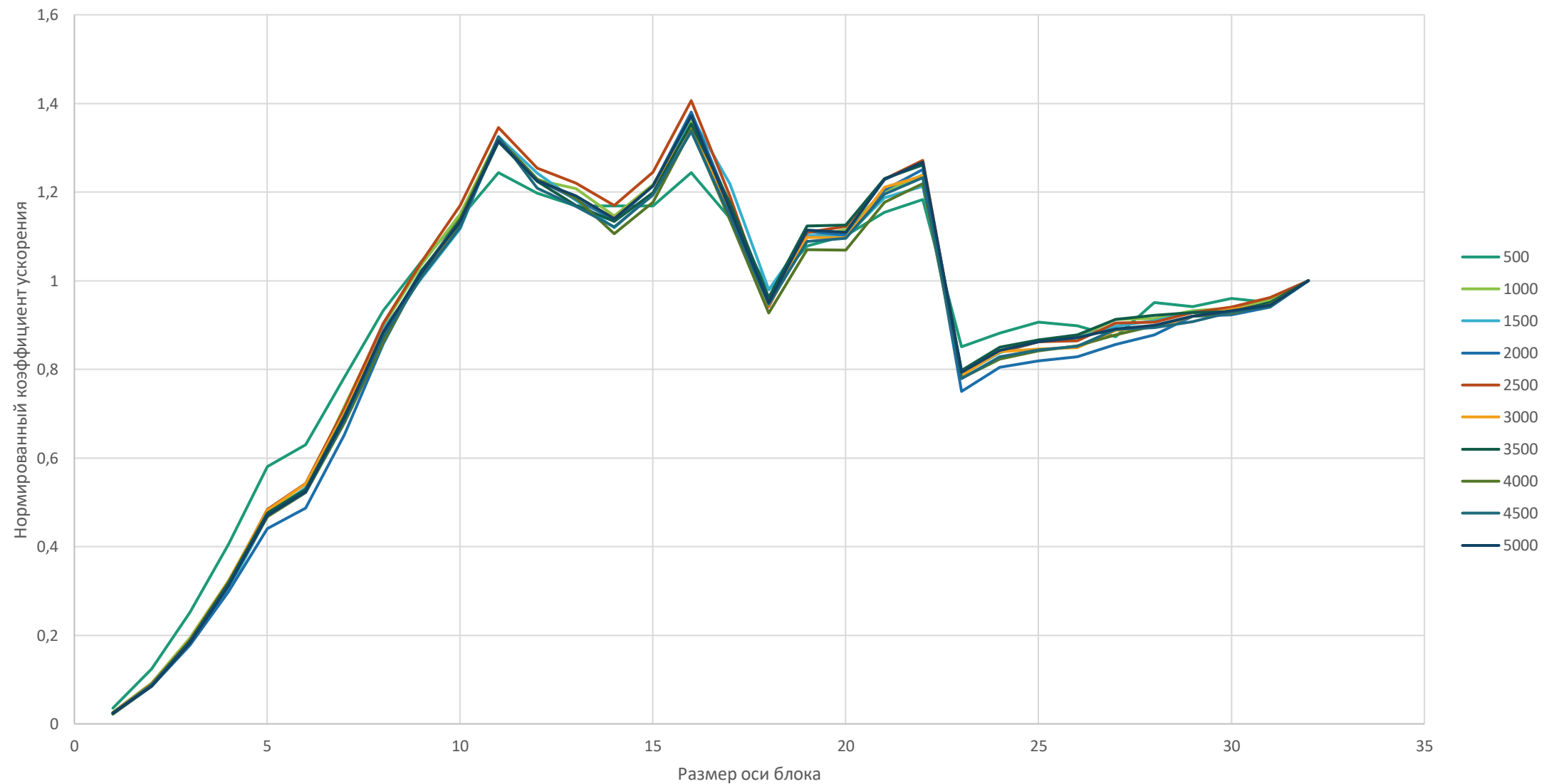
Ускорение при разном числе потоков



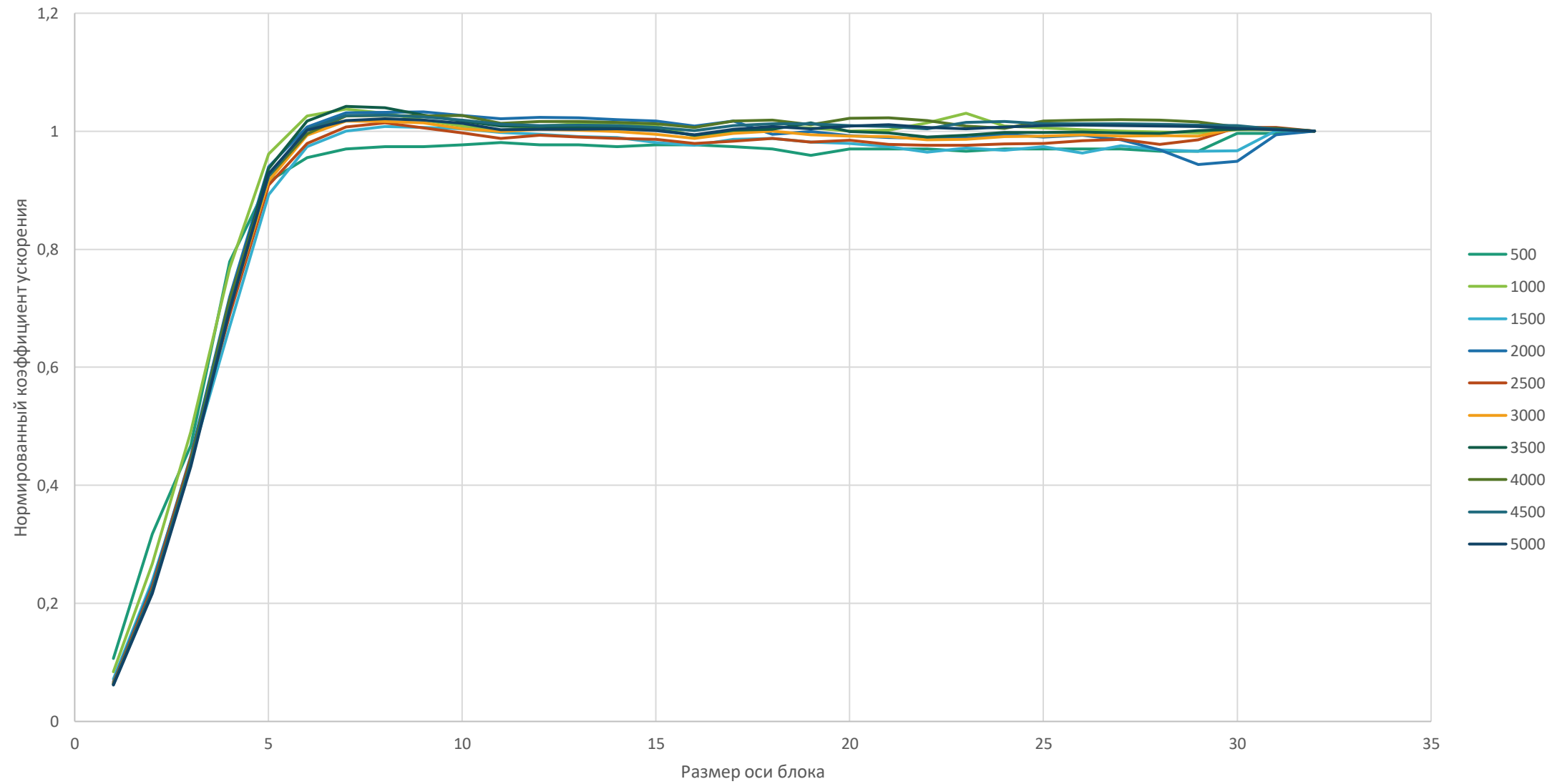
Ускорение при разном числе потоков алгоритма №2



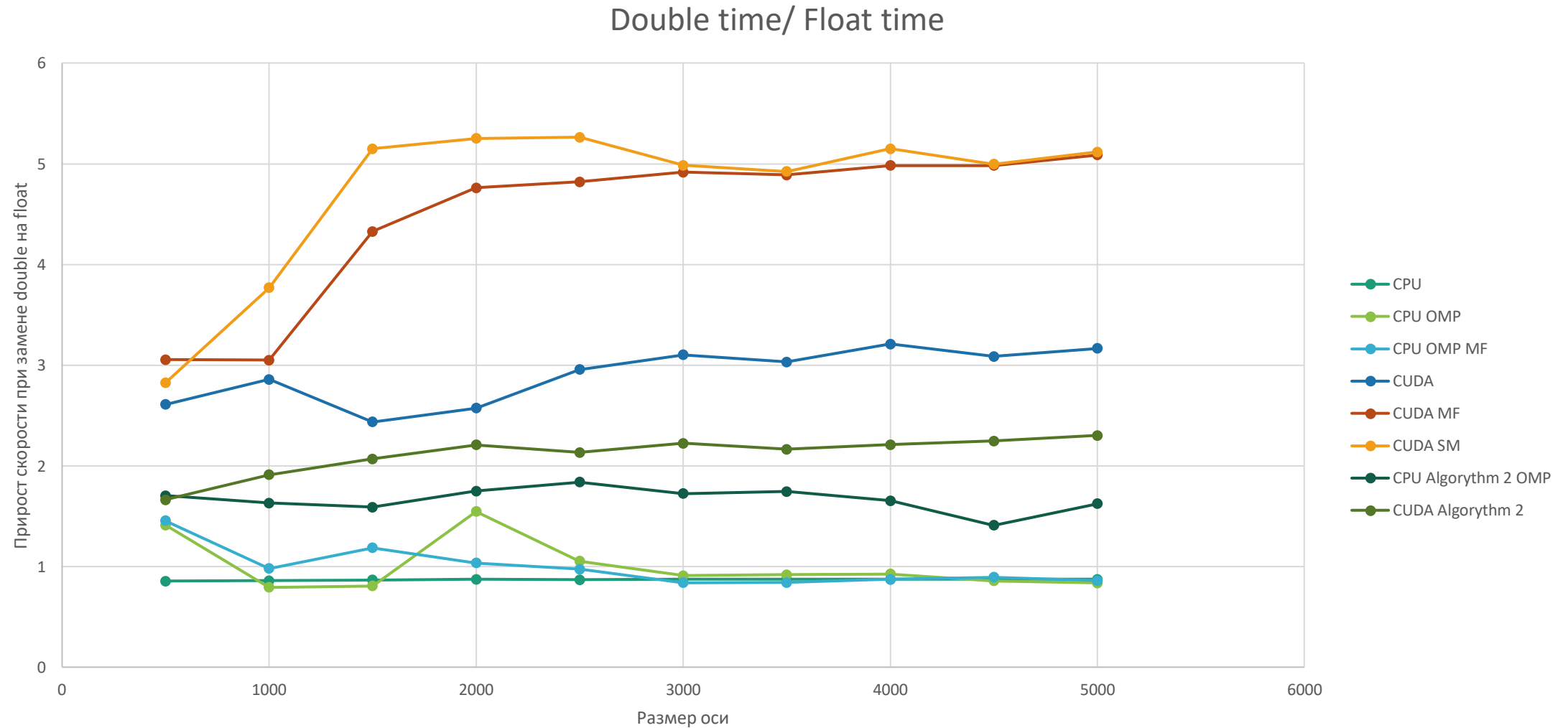
CUDA сравнение производительности при разном размере блоков ядра



CUDA Алгоритм №2 сравнение производительности при разном размере блоков ядра



Сравнение производительности алгоритмов, работающих с типом данных float и double



Заключение

- Задачи, поставленные в рамках научно-исследовательской практики, были выполнены.
- Проведен подробный сравнительный анализ производительности и найдены неочевидные зависимости.
- В дальнейшие планы входит использование полученных данных о производительности написанных методов для создания прототипа аудио системы реального времени. В рамках работы было изучено готовое решение PlaneVerb в виде плагина для среды Unity3D, что облегчит дальнейшее написание подобного плагина, проводящего асинхронные вычисления на видеокарте и использующее результаты этих вычислений для аурализации звука.

Код в репозитории:

<https://github.com/zarond/NIP>