

Федеральное государственное
автономное образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Институт космических и информационных технологий
Кафедра «Вычислительная техника»

**Параллельные вычислительные
системы**

Сквозной мини-проект.
Последовательная программа.
Симуляция жидкости.

Преподаватель

Студент КИ22-076, 032211509
номер группы, зачетной книжки

подпись, дата

подпись, дата

Сиротинина Н.Ю.

инициалы, фамилия

Осташов С. Е.

инициалы, фамилия

Красноярск - 2024

Содержание

1. Алгоритм	3
2. Измерения	3
3. Оптимизация	5
4. Результаты	6
Заключения	8
Приложение А	9
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	14

1. Алгоритм

Алгоритмы симуляции жидкости делятся на два типа: основанные на частицах и основанные на сетке. Я выбрал первый тип. Самый известный алгоритм данного типа - SPH [2]. Я выбрал алгоритм называемый Position Based Fluid (PBF) [3].

Данные алгоритмы основаны на параметре «плотность». Каждая частица вносит свой вклад в значение плотности вокруг нее. Ниже представлен рисунок функции (см. рис. 1), использующаяся для вычисления влияния частицы на значение плотности вокруг нее. Частица влияет на данный параметр в каждой точке пространства внутри круга, выделенного черными границами. Чем больше влияние тем более фиолетовый цвет мы имеем.

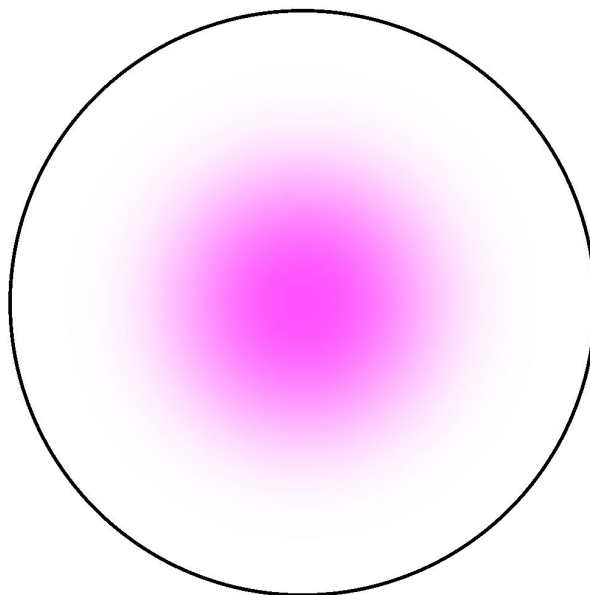


Рисунок 1 - Плотность

Каждая частица стремится сохранить константное значение плотности D_t . Поэтому они отталкиваются от тех точек в пространстве, где плотность больше D_t . И отталкивается в противном случае. Более подробно можно почитать в работе PBF [3].

2. Измерения

Я провел измерения последовательный программы для n количества частиц. В каждом измерении я считал сколько нужно времени алгоритму чтобы применить

силы, посчитать новое положение частиц и обновить его. Таким образом для каждого эксперимента я получал график, где x - новый фрейм, а y - время необходимое для его подсчета. Данные графики приведены в приложении А.

Среднее значение времени, занимаемое вычислениями алгоритма для каждого эксперимента можно увидеть ниже (см. рис. 2 и таблицу 1). Важно учитывать что данные измерения проводились при фиксированном размере пространства, выделенного для движения частиц. Если пространство увеличить, то время резко сократиться, так как частицы растекаются и меньше взаимодействуют друг с другом. Если же его наоборот уменьшить до время для вычисления резко увеличиться. Измерения проводились в пространстве размером 300x100. Данный параметр можно поменять в файле settings.hpp (BOXWIDTH, BOXHEIGHT).

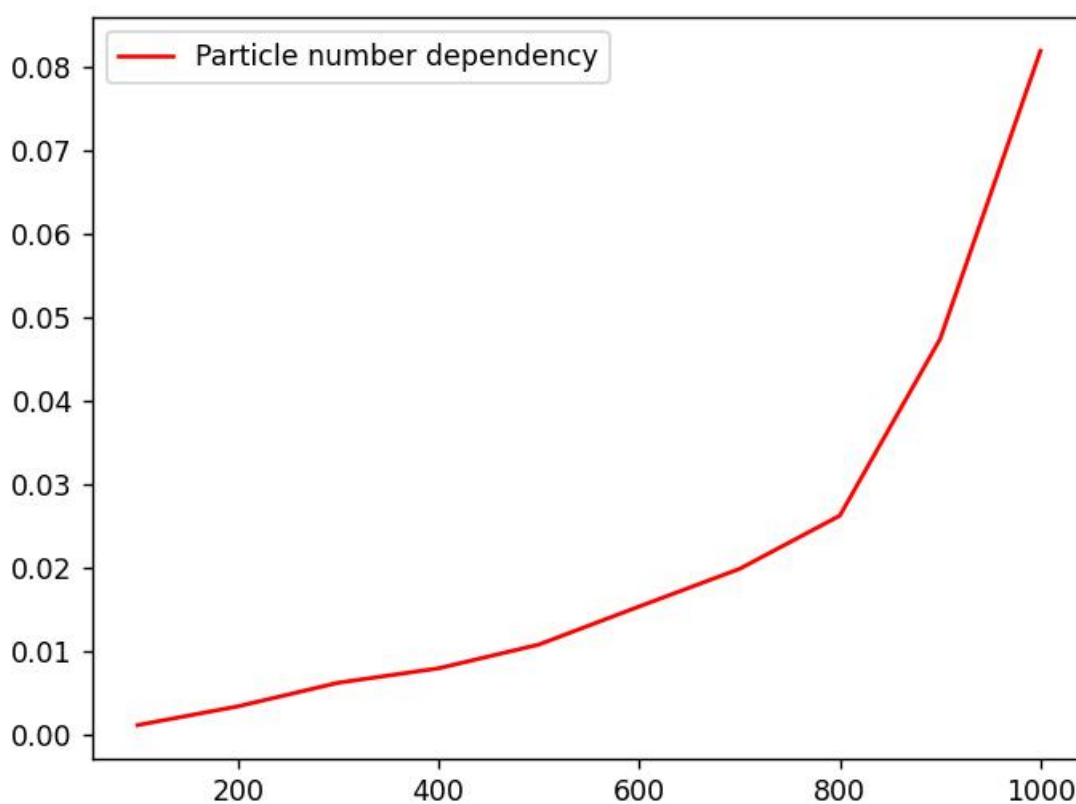


Рисунок 2 - Зависимость среднего времени от количества частиц

Таблица 1 - Зависимость среднего времени от количества частиц

Количество частиц	Время в секундах
100	0.00119412022
200	0.0034384875782881
300	0.006264027474747475
400	0.00798934
500	0.01083396244274809
600	0.015392861723280861
700	0.01988865497881356
800	0.02625939353891336
900	0.0474051626223092
1000	0.08187426836734694
20000	~76.1703

Нагрузку процессора при значении количества частиц равном 20000 можно посмотреть на рисунке 3. При этом значение в таблице 1 при данном количестве частиц не является средним, а только за первый фрейм.


Name	Status	43% CPU	93% Memory
>  Fluid		20.4%	59.5 MB

Рисунок 3 - Нагрузка процессора

Наблюдая за программой в диспетчере задач, я установил что значение нагрузки CPU колебалось с 15~25% на протяжении всего выполнения.

3. Оптимизация

Результаты измерений выше приведены в уже оптимизированном коде! До оптимизации программа могла справиться максимум с 100 частицами так, чтобы окно программы реагировало оперативно, позволяя закрыть ее сразу (без задержки). Смотря на данный результат я решил, что необходимо коду необходима оптимизация.

Для оптимизации я использовал встроенный в Visual Studio C++ 2022 Profiler. Результаты показали что дольше всего считалось значения плотности (функция calcPoly6). Чтобы решить данную проблему я избавился от встроенной функции pow. А также вынес код данной функции в header и поставил ей квалификатор inline, чтобы компилятор встраивал данную функцию. Также я объединил те функции и вычисления, где порядок не важен и проход выполняется по одним и тем же элементам. Результаты оптимизации можно увидеть на рисунке 4.

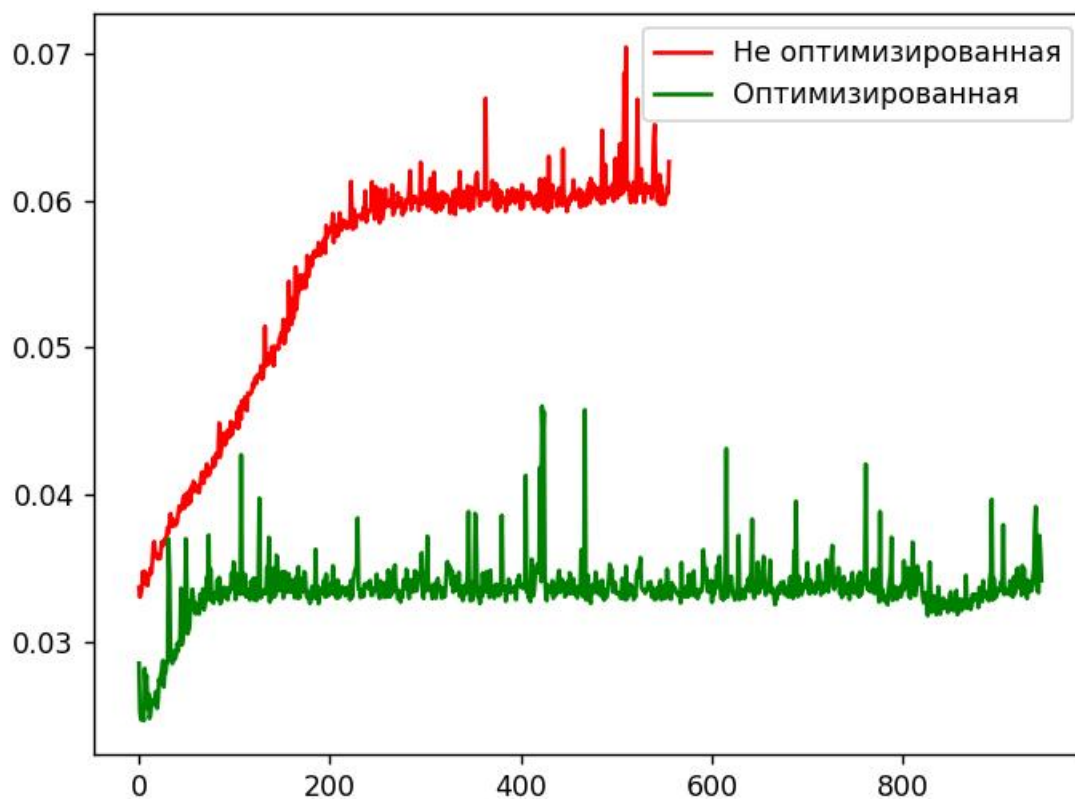


Рисунок 4 - Оптимизация

Неоптимизированная версия программа работает в среднем 1,64 раза медленнее оптимизированной.

4. Результаты

В результате я получил небольшую относительно быструю анимацию жидкости (рис.5). В данной программе и алгоритме есть и недочеты.

Данный алгоритм не рассчитан на взаимодействие жидкости с твердыми объектами, поэтому на их границе можно наблюдать аномалии. В моем случае это происходит там где заканчивается пространство коробки, в котором расположена жидкость.

Помимо этого есть проблемы с определением значение плотности. В некоторых местах, зачастую ближе ко дну, можно заметить что плотность намного не постоянная, в некоторых местах она аномально больше.

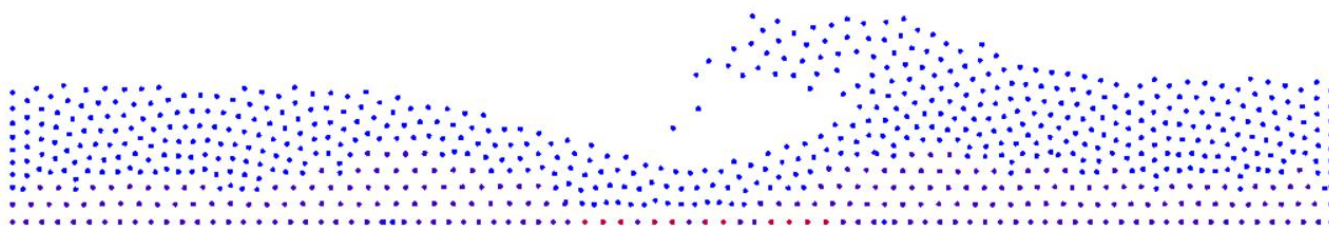


Рисунок 5 - Результат

Заключения

На данной стадии у меня получилось реализовать алгоритм Position Based Fluid на языке программирования C++.

Приложение А

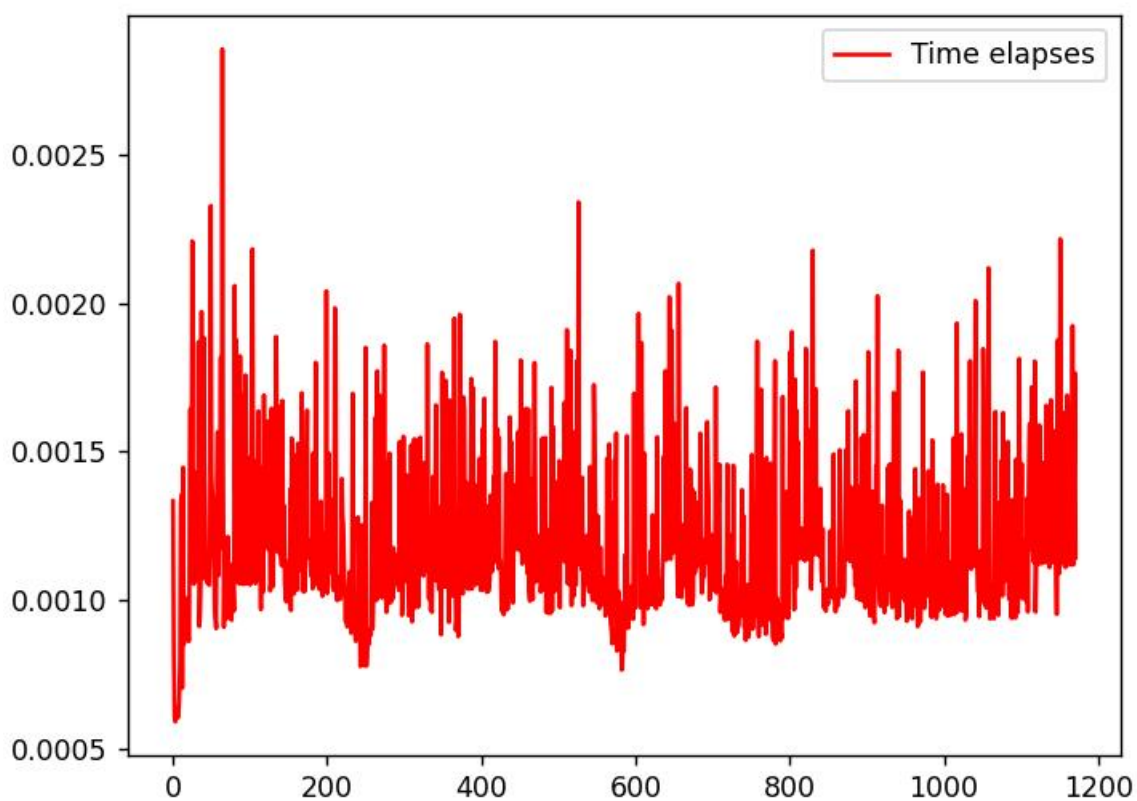


Рисунок 6 - 100 частиц

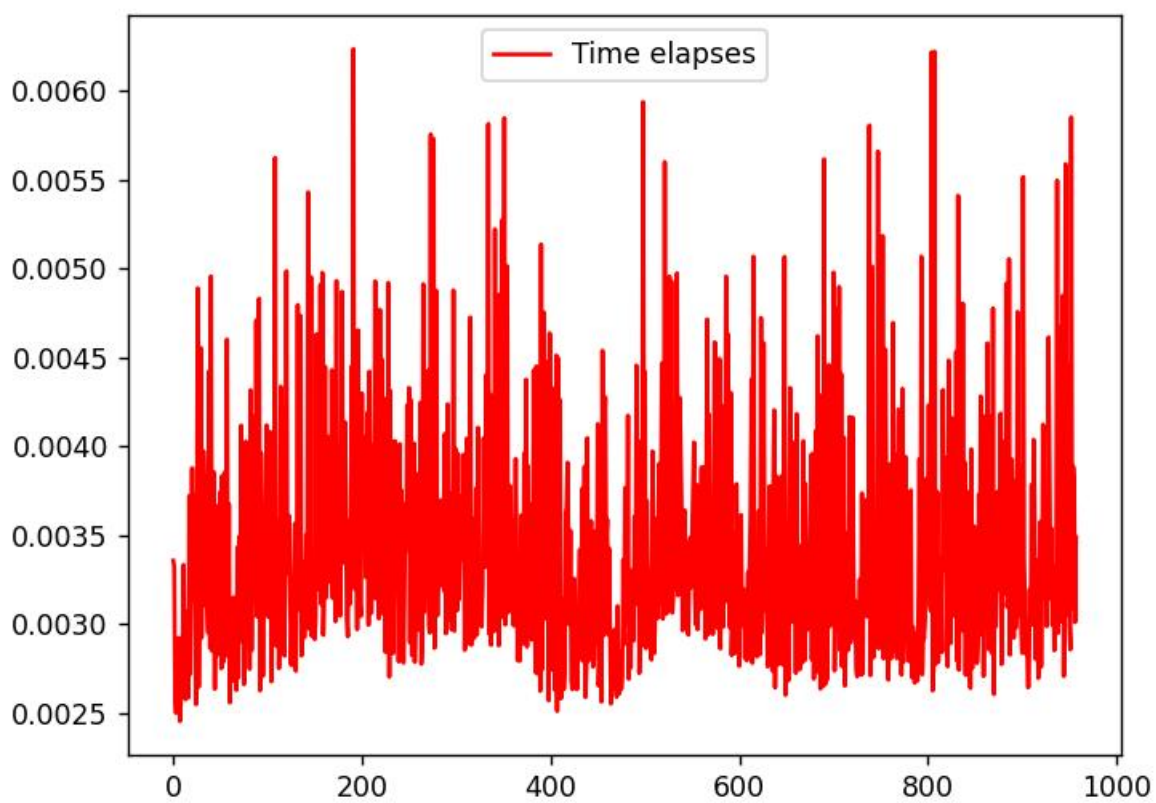


Рисунок 7 - 200 частиц

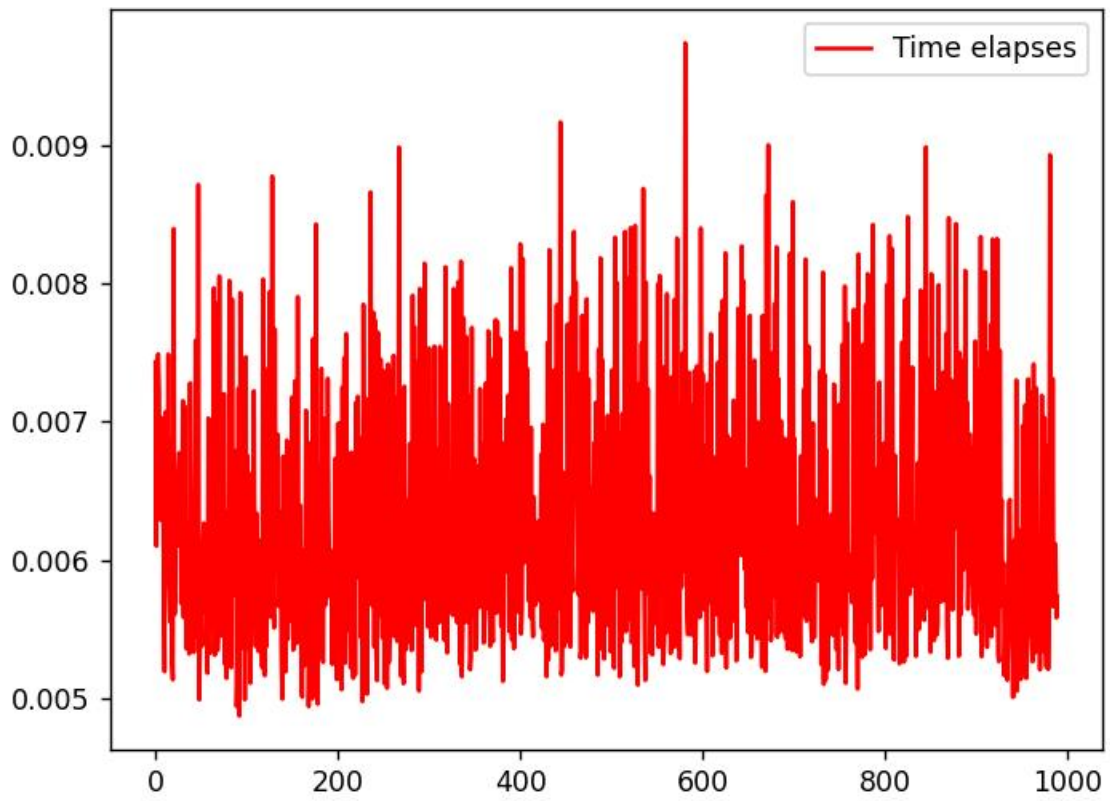


Рисунок 8 - 300 частиц

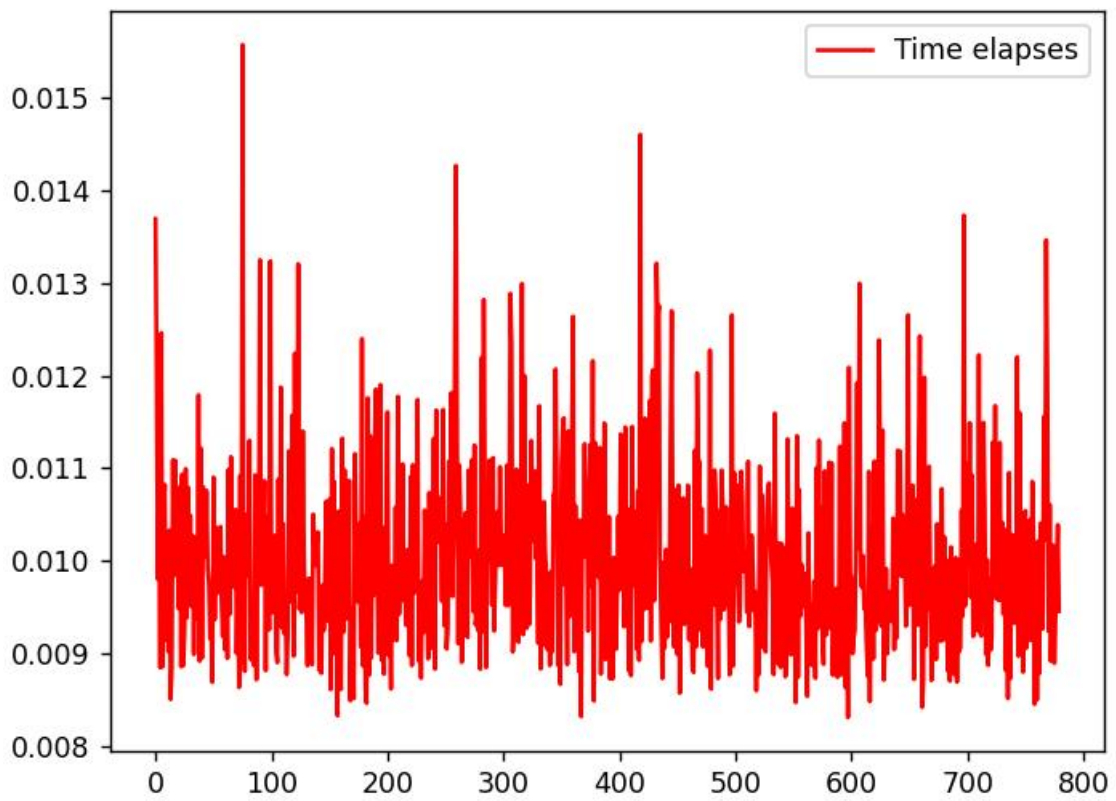


Рисунок 9 - 400 Частиц

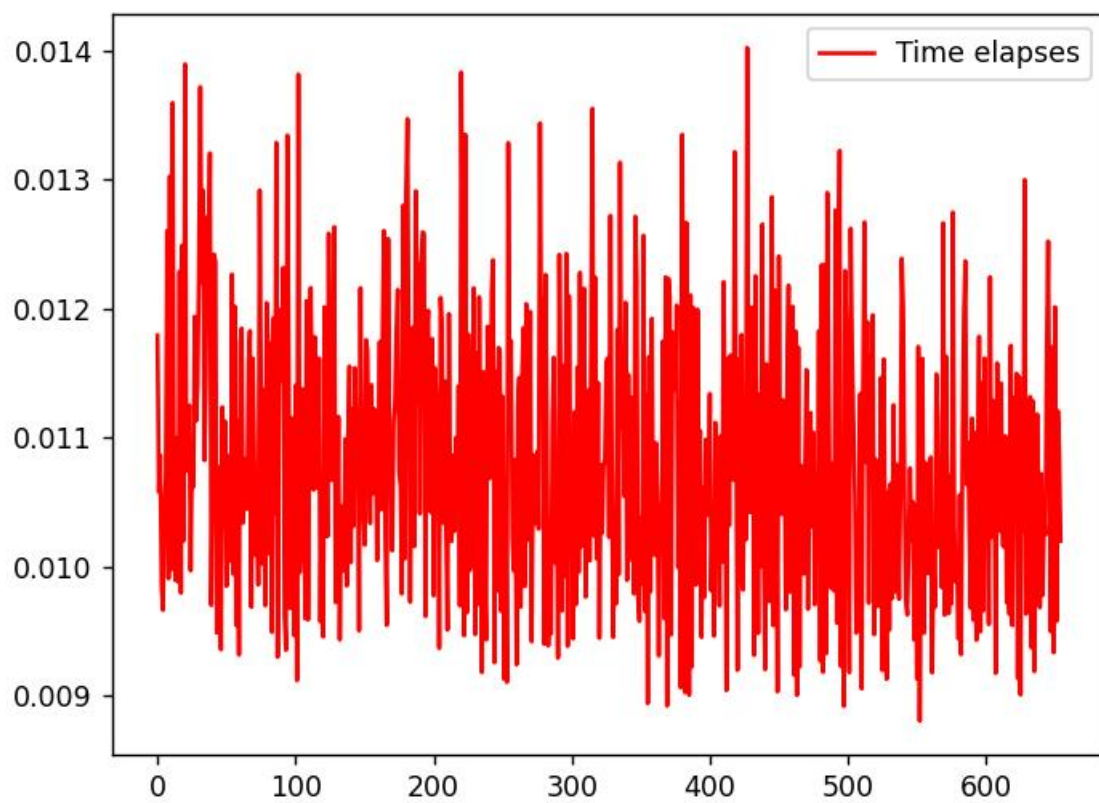


Рисунок 10 - 500 частиц

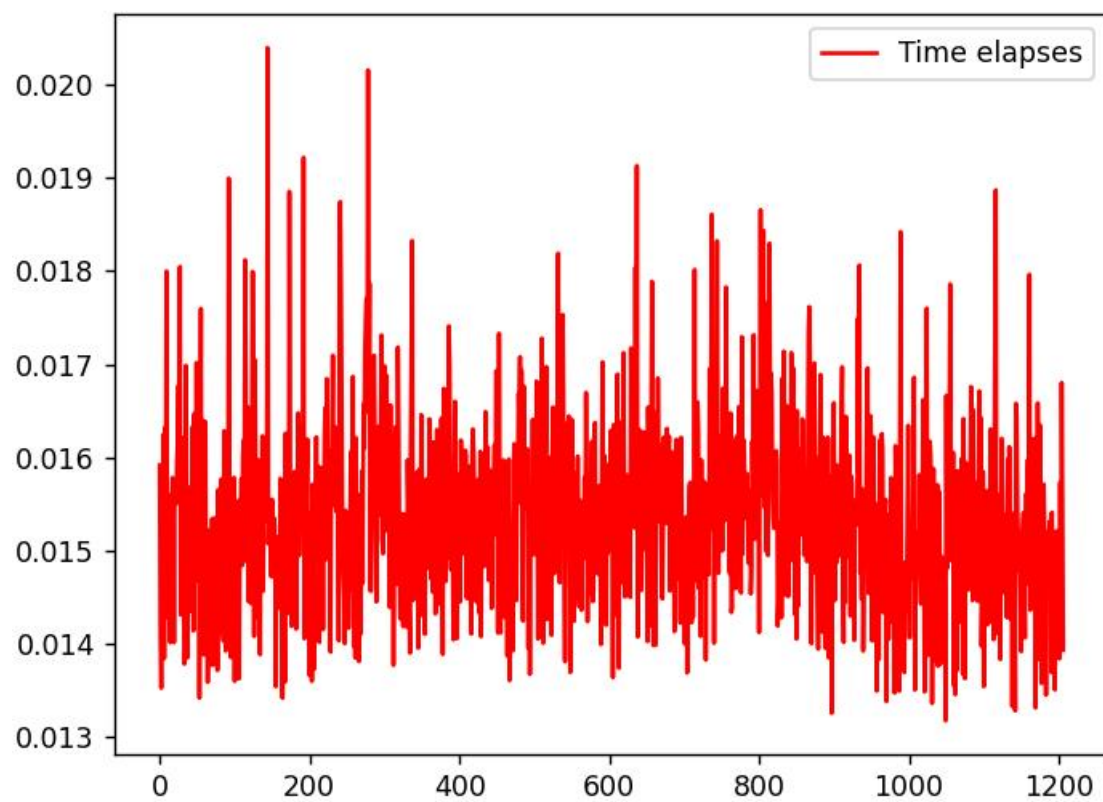


Рисунок 11 - 600 частиц

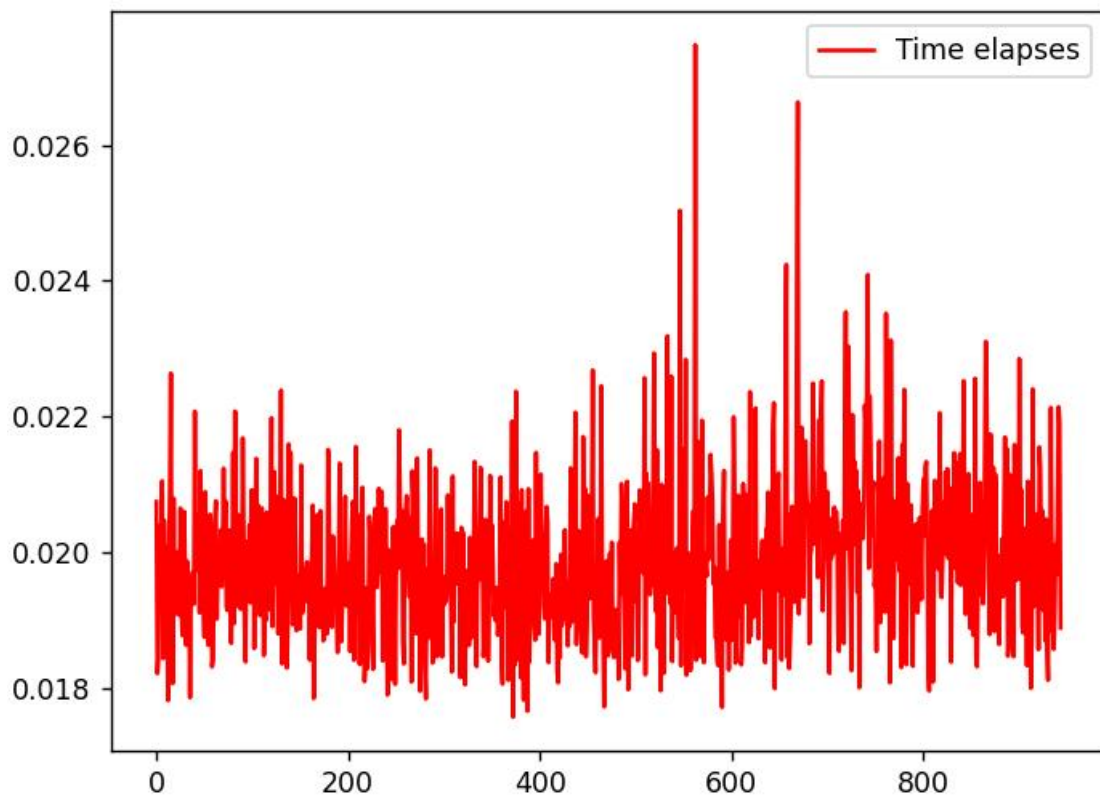


Рисунок 12 - 700 частиц

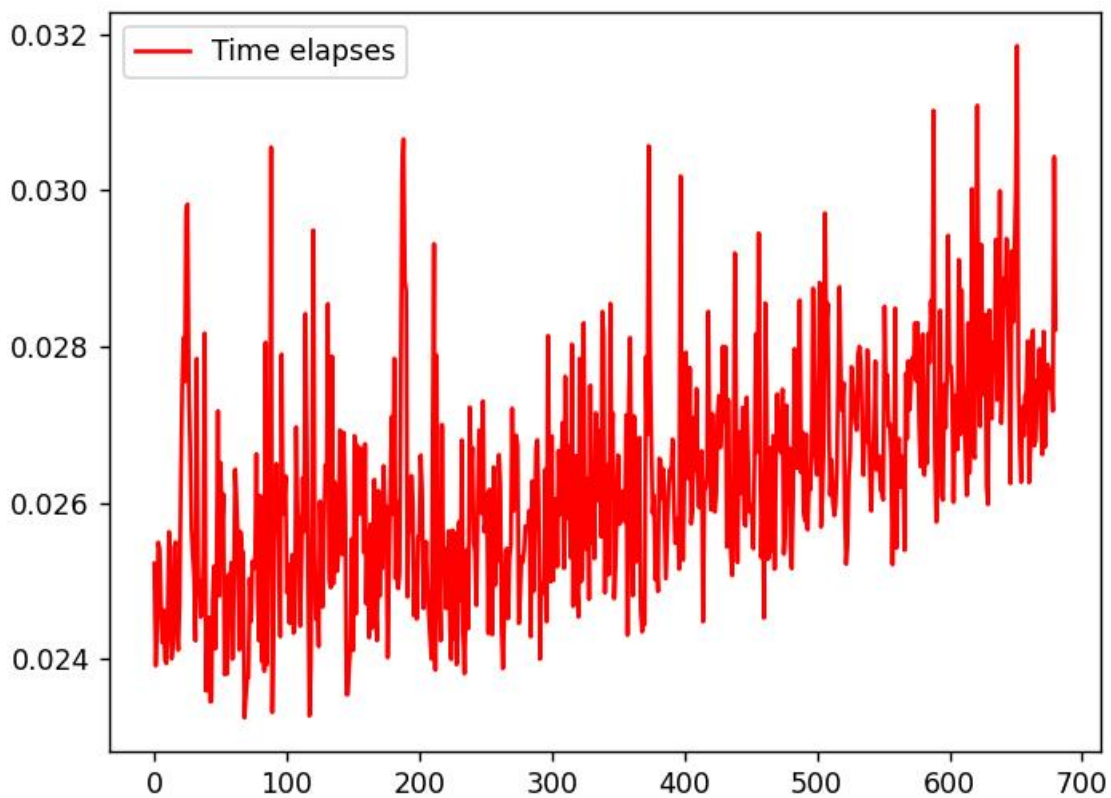


Рисунок 13 - 800 частиц

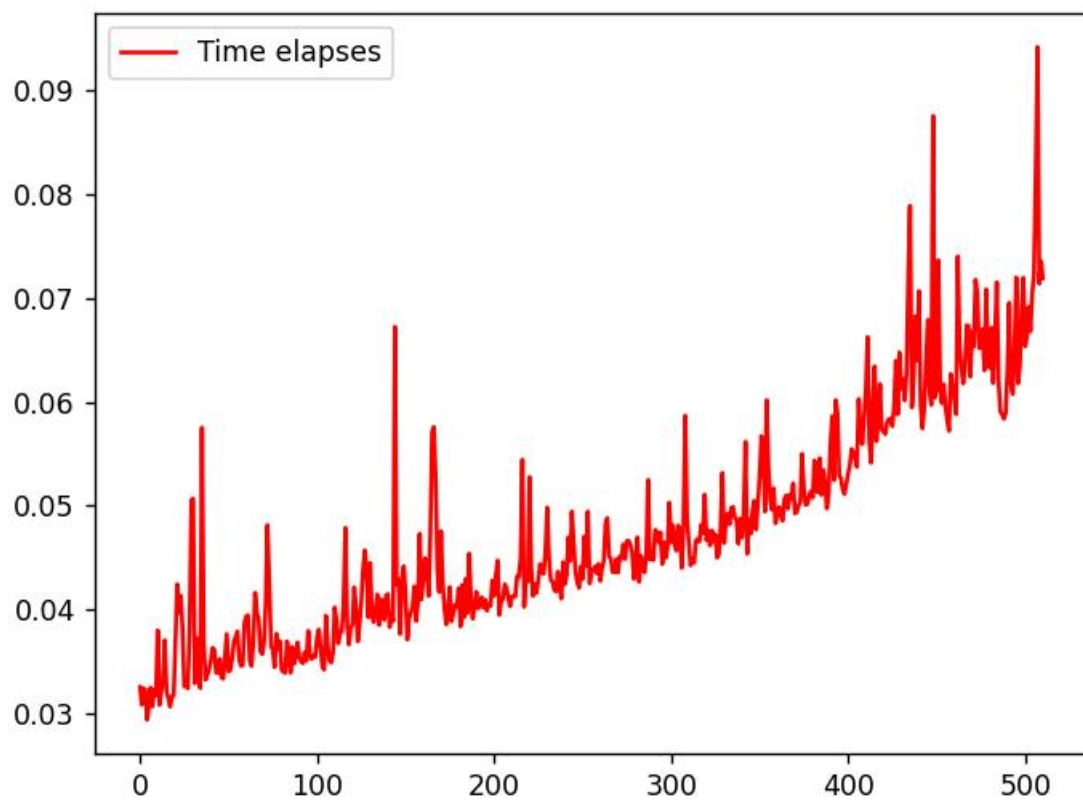


Рисунок 14 - 900 частиц

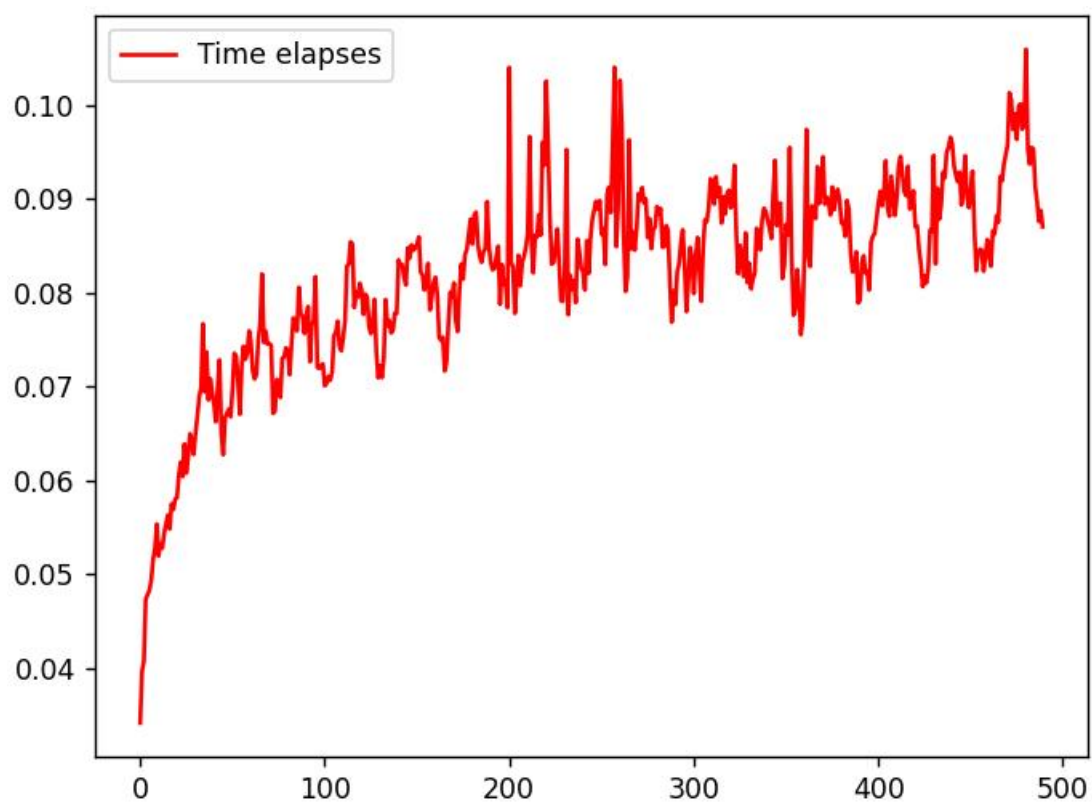


Рисунок 15 - 1000 частиц

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Система менеджмента качества. Общие требования к построению, изложению и оформлению документов учебной деятельности СТУ 7.5–07–2021 // Сибирский федеральный университет: [официальный сайт]. – URL: <http://polytech.sfu-kras.ru/students/documents/sfu-stu-7.5-07-291121-s-podp.pdf> (дата обращения: 19.09.2024).
2. Gingold, Robert A. Smoothed particle hydrodynamics: theory and application to non-spherical stars // Gingold, Robert A, Monaghan, Joseph J. (1977). Monthly Notices of the Royal Astronomical Society 181(3): 375-89. - URL: [10.1093/mnras/181.3.375](https://doi.org/10.1093/mnras/181.3.375) (дата обращения: 20.10.2024)
3. Miles Macklin. Position Based Fluids // Miles Macklin, Matthias Muller NVIDIA. ACM TOG 32(4). - 2013. [сайт статьи]. - URL: <https://dl.acm.org/doi/10.1145/2461912.2461984> (дата обращения: 19.10.2024)
4. Robert Bridson. Fluid Simulation // Robert Bridson, Matthias Muller-Fischer. SIGGRAPH 2007 Course Notes
5. Sebastian Lague. Coding Adventure: Simulation Fluids // Sebastian Lague. YouTube [сайт видео]. - URL: <https://youtu.be/rSKMYc1CQHE?si=JeHFDpMt7pbYPAjv> (дата обращения: 17.10.2024)
6. ShaderToy. - URL: <https://www.shadertoy.com/> (дата обращения: 19.11.2024)
7. Navier-Stokes Equation // ScienceDirect: [официальный сайт]. - URL: <https://www.sciencedirect.com/topics/physics-and-astronomy/navier-stokes-equation> (дата обращения: 19.10.2024)