

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

**«Дальневосточный федеральный университет»**

(ДВФУ)

Институт наукоемких технологий и передовых материалов

Департамент теоретической физики и интеллектуальных технологий

Пацук Никита Денисович

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

бакалаврская работа

**СТАТИСТИКА ГАЗА ТВЕРДЫХ СФЕР**

по направлению подготовки (специальности) 14.03.02 Физика атомного ядра и частиц, профиль «Ядерные физика и технологии»

Владивосток

2022

|  |  |
| --- | --- |
| В материалах данной выпускной  квалификационной работы не содержатся  сведения, составляющие государственную  тайну, и сведения, подлежащие экспортному  контролю  Уполномоченный по экспортному контролю  \_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  *подпись И.О. Фамилия*  «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_ г. | Автор работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_,  *подпись*  группа \_\_\_Б8118-14.03.02фаяч\_\_\_\_\_\_\_    «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_\_\_ г.  Руководитель ВКР \_\_\_\_профессор\_\_\_\_\_  \_\_\_\_\_департамента ТФиИТ\_\_\_\_\_\_\_\_\_\_\_  должность, ученое звание  \_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_К.В. Нефедев\_\_\_\_  *подпись И.О. Фамилия*  «\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_ г.  Консультант(ы)\*  \_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  *подпись И.О. Фамилия*  «\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_ г. |
|  | Назначен рецензент\* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  *ученое звание*  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  *фамилия, имя, отчество* |
| Защищена в ГЭК с оценкой  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Секретарь ГЭК  \_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  *подпись И.О. Фамилия*  «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_ г. | **«Допустить к защите»**  Заведующий кафедрой /  директор Департамента \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  *ученая степень, ученое звание*  \_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  *подпись И.О. Фамилия*  «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_ г. |

\*При наличии. Количество строк должно соответствовать количеству консультантов, назначенных обучающемуся.

**Аннотация**

Получены результаты исследования эволюции системы абсолютно упругих частиц при переходе от неравновесного состояния к равновесному. Проведены численные расчеты с применением технологии параллельных вычислений CUDA. В ходе численных расчетов показано существование необратимости перехода системы к термодинамическому равновесию. Разработанные авторские параллельные алгоритмы реализованы в многопоточном коде, исполняемом на GPU, которые позволяют получать детальное описание промежуточных неравновесных термодинамических состояний газа упругих частиц. Произведены расчеты системы, состоящей из двух противоположно направленных потоков частиц для изучения явления турбулентности при массопереносе в газе упругих частиц. Рассмотрены эффекты на границе раздела двух потоков и динамика изменения параметров системы при возникновении вихревых образований.

В работе содержится введение, 3 раздела, заключение, список литературы и одно приложение. Текст работы составляет 41 стр, включает 19 рисунков, одну таблицу.

**Abstract**

The results of the study of the evolution of a system of absolutely elastic particles during the transition from a nonequilibrium state to an equilibrium state are obtained. Numerical calculations were carried out using CUDA parallel computing technology. In the course of numerical calculations, the existence of the irreversibility of the transition of the system to thermodynamic equilibrium is shown. The developed author's parallel algorithms are implemented in multithreaded code executed on the GPU, which allow to obtain a detailed description of the intermediate nonequilibrium thermodynamic states of a gas of elastic particles. Calculations of a system consisting of two oppositely directed particle flows have been performed to study the phenomenon of turbulence during mass transfer of elastic particles in a gas. The effects at the interface of two flows and the dynamics of changes in system parameters during the occurrence of vortex formations are considered.

The work contains an introduction, 3 sections, a conclusion, a list of references and one appendix. The text of the work is 41 pages, includes 19 figures, one table.

Оглавление

[Введение 6](#_Toc106913674)

[1 Описание макроскопических систем 9](#_Toc106913675)

[1.1. Предмет и методы термодинамики и статистической физики 9](#_Toc106913676)

[1.2. Природа возникновения турбулентности 10](#_Toc106913677)

[1.3. Характеристики турбулентности 12](#_Toc106913678)

[2 Обзор моделей макроскопических систем 14](#_Toc106913679)

[2.1. Метод прямого моделирования Монте-Карло (DSMC) 14](#_Toc106913680)

[2.2. Прямое численное моделирования (DNS) 15](#_Toc106913681)

[2.3. Метод моделирования крупных вихрей (LES) 16](#_Toc106913682)

[2.4. Приложения моделей 18](#_Toc106913683)

[3 Реализация метода частиц с расчетом кванта времени 19](#_Toc106913684)

[3.1. Описание компьютерной модели 19](#_Toc106913685)

[3.1.1. Характеристики частиц 19](#_Toc106913686)

[3.1.2. Границы пространства 19](#_Toc106913687)

[3.1.3. Распределение импульсов при соударении частиц 20](#_Toc106913688)

[3.1.4. Расчет времени до столкновения пары частиц в пространстве 22](#_Toc106913689)

[3.1.5. Расчет времени до столкновения частицы с твердой стенкой 23](#_Toc106913690)

[3.1.6. Расчет времени до столкновения двух частиц с учетом периодических условий. 23](#_Toc106913691)

[3.1.7. Эволюция системы и квант времени 23](#_Toc106913692)

[3.1.8. Программный алгоритм 24](#_Toc106913693)

[3.2. Результаты численного расчета 26](#_Toc106913694)

[3.2.1. Планковские единицы 26](#_Toc106913695)

[3.2.2. Расчетные формулы 27](#_Toc106913696)

[3.2.3. Переход системы к равновесию 31](#_Toc106913697)

[3.2.4. Исследование турбулентности в системе упругих частиц 38](#_Toc106913698)

[Заключение 48](#_Toc106913699)

[Список литературы 49](#_Toc106913700)

[Приложение А 51](#_Toc106913701)

# **Введение**

Плазма – четвертое агрегатное состояние вещества, представляющая собой ионизированный газ, который был открыт лишь в XX веке [1]. Изучение плазмы привело к идее управляемого термоядерного синтеза (УТС) – синтез более тяжёлых атомных ядер из более лёгких с целью получения энергии. Количество вырабатываемой энергии при термоядерном синтезе колоссальное – в четыре раза больше, чем при реакциях деления ядер.

Сейчас научно-технический прогресс в области УТС требует от человечества все более точные модели поведения плазмы. Исследования поведения плазмы при различных условиях приблизят человека к освоению технологий

При создании моделей взаимодействия элементарных частиц (протонов, нейтронов, электронов) возникает вопрос о геометрических характеристиках объектов. Важно понимать, является ли элементарная частица материальной точкой, сферическим объектом или не имеет геометрии вовсе. Так, при рассмотрении упругого рассеяния налетающего быстрого протона на неподвижный протон в камере Вильсона возникают два трека под углом в 90 градусов, что говорит о центральном соударении [2]. Центральное соударение говорит о наличии размеров у частиц. Решение задачи столкновения протона с ядром атома гелия также подразумевает наличие геометрических размеров объектов. Однако в элементарная частица, как выяснилось, представляет из себя квантовомеханическую систему без четких границ в пространстве – составляющие её кварки расположены в пространстве в соответствии со своей волновой функцией [3]. В различных моделях частиц размеры объектам задаются условно. Так, размер протона можно определить как радиус твердого кора ядерных сил, электрический или магнитный радиус. Таким образом, проблема задания элементарным частицам размера остается актуальной и в настоящее время.

В настоящей работе представлена модель частного случая системы абсолютно упругих частиц (АУЧ) в двумерном пространстве с заданием круглой формы объектам. Рассматривается явления необратимого перехода системы АУЧ к термодинамическому равновесию и возникновение турбулентного движения.

Описанный частный случай модели частиц реализован путем создания программного кода с использованием технологии параллельных вычислений. Разбиение задачи на множество параллельных вычислительных потоков является эффективной реализацией алгоритма по сравнению с последовательным выполнением программы.

Результаты вычислений в рамках настоящей научно-исследовательской работы внесут вклад в развитие молекулярно-кинетической теории газов конечного числа частиц, турбулентности и плазмы – далеких от исчерпывающего теоретического толкования областей физики в настоящее время.

Разработанный программный код будет полезен для дальнейшего изучения физики жидкостей и газов, разработки специального прикладного программного обеспечения для медицинской, теплотехнической и других отраслей.

Для настоящей научно-исследовательской работы сформированы цели:

* Разработка программного комплекса для исследований в областях статистической физики, физики турбулентности и потенциального использования в прикладных задачах медицины, теплотехники.
* Исследование перехода к термодинамическому равновесию системы АУЧ.
* Исследование возникновения явления турбулентности в системе АУЧ.

Сформулированы задачи для достижения поставленных в работе целей:

* Разработка эффективного алгоритма расчета эволюции системы конечного статистически значимого количества АУЧ.
* Установление параметров распределений плотности вероятности системы АУЧ с течением времени до момента наступления термодинамического равновесия.
* Определение траектории изменения термодинамических величин в процессе эволюции системы к термодинамическому равновесию.
* Моделирование двух противоположных по направлению потоков АУЧ с для получения турбулентного движения и изучения эффектов на границе между потоками.

Объектом исследования выступает двумерная система частиц, обладающих размерами, массой, движущихся по закону равномерного движения, взаимодействующих посредством абсолютно упругого удара.

Предметом исследования является изучение траектории изменения термодинамических и других характеристик системы частиц в процессе ее эволюции.

# **1 Описание макроскопических систем**

* 1. **Предмет и методы термодинамики и статистической физики**

Через призму как классической, так и квантовой механики макроскопические системы, которые обладают огромным числом степеней свободы (к примеру, имеющих порядка 1020 частиц в 1 см3 воздуха) чрезвычайно сложны, тем не менее, такие сложные системы неплохо описываются довольно небольшим количеством макроскопических параметров. На практике все отличительные признаки газов определяют с помощью знания объема, массы и температуры этого газа. Состояния макроскопических систем, которые описываются макроскопическими характеристиками, называют макроскопическими или термодинамическими состояниями. Легко предположить, что эти макросостояния описываются огромным числом механически заданных (например, в классической механике, определением всех обобщенных координат и скоростей) микросостояний. Количество разрешенных микросостояний замкнутой системы есть статистический вес макроскопического состояния. Данная физическая характеристика имеет большую важность, так как с помощью этой величины возможно определить энтропию системы.

Измеряемые в макросостояниях физические характеристики не что иное, как результат усреднения значений этих величин в разрешенных микросостояниях. Для того, чтобы фактически провести усреднение, необходимо иметь данные о распределении вероятностей микросостояний. Чаще всего распределение микросостояний для замкнутой равновесной системы постулируется (основная статистическая гипотеза – микроканоническое распределение), в то время иные равновесные распределения выводятся на его фундаменте. Расчет физических величин используя усреднение их значений в микросостояниях образует фундамент статистического метода изучения макросистем. Термодинамический метод основывается на выводе общих законов между макроскопическими величинами на основе экспериментальных данных, без апелляции к атомно-молекулярному строению вещества. Термодинамический (феноменологический) метод содержит в себе больше общности; статистический метод дает возможность дальше углубиться в сущность явлений.

Центральная проблема статистической механики состоит в том, что любая изолированная макроскопическая система с течением времени переходит в состояние термодинамического равновесия, которое характеризуется постоянством макроскопических параметров с последующим течением времени. Процесс установления равновесия в системе называется релаксацией, время этого процесса − временем релаксации. Спектр возможных времен релаксации огромен, ~ 10−12 ÷ 108 сек. Равновесие говорит о том, что и отдельные макроскопические части системы (подсистемы) так же пребывают в состоянии внутреннего равновесия, а также в равновесии друг с другом, то есть нет никаких потоков энергии и частиц от одной подсистемы к любой другой. Частицы также двигаются и соударяются в равновесном состоянии, что приводит к возникновению непрерывных флуктуаций – небольшие кратковременные отклонения системы от полного равновесия (отклонения от средних значений).

Локальное (неполное) равновесное состояние говорит о том, что система делится на подсистемы, которые находятся в состоянии внутреннего равновесия, но нет равновесия между подсистемами. Если изолировать подсистемы, то изменения в системе заканчиваются. Количество независимых макроскопических параметров, которые характеризуют систему, растет по мере отклонения от полного равновесия. Однако в процессе релаксации количество независимых макроскопических параметров уменьшается (происходит так называемое сокращение описания). Таким образом, разрушаются препятствия на разрешенные микросостояния, и статистический вес системы увеличивается. Теперь достаточно оснований, чтобы заявить, что макроскопические состояния − это равновесные или локально-равновесные состояния макроскопических систем [4].

* 1. **Природа возникновения турбулентности**

Физическая причина, следствием которой станет появление турбулентности, является некоторая неустойчивость, образующаяся в наблюдаемом течении (например, вблизи стенок обтекаемого тела) [5]. До появления турбулентности часто возникают упорядоченные нестационарные образования (например, струи, слои смешивания) [6].

С точки зрения математики возникновение турбулентности обусловлено преобладанием дестабилизирующих конвективных членов над стабилизирующими вязкими членами в уравнении баланса импульса [5]. Таким образом, уравнения Навье-Стокса, описывающие динамику жидкости и газа, теряют свою устойчивость. При этом происходит быстрое накопление возмущений определенного вида.

С другой стороны, потерю устойчивости можно рассматривать как следствие второго начала термодинамики: в природе не бывает слишком «упорядоченных» течений.

Устойчивость ламинарных потоков зависит от большого количества параметров, таких как уровень турбулентности внешнего потока, градиентов давления или температур и т. п. (примеры: пограничный слой при наличии градиента давления, струя с внешним возмущением) [6]. Примечательно, что при утрате устойчивости колебания только начинают свое развитие, течение становится полностью турбулентным гораздо позднее (ниже по потоку), когда амплитуды колебаний достигнут определенного уровня. Отсутствие линейности газодинамических систем и описывающих их уравнений Навье-Стокса приводит к ограниченности возмущений амплитуды. В то время, как амплитуда возмущений достигает насыщения, говорят, что течение стало «развитым турбулентным» [6].

Прикладное интерес в большинстве случаев представляют средние характеристики, например коэффициент потерь в турбине, подъемная сила крыла, сопротивление автомобиля. Но также существуют задачи, в которых большой интерес представляют именно пульсации, например задачи аэроакустики и аэроупругости.

Благодаря этому факту большое развитие получила теория турбулентности, основанная на математической статистике. Также используются экспериментальные данные в изучении турбулентности. Такой метод является полуэмпирическим.

* 1. **Характеристики турбулентности**

Даже если сводить причину турбулентности к возникновению случайных пульсаций, природа явления по-прежнему будет иметь сложный характер. Турбулентные пульсации возникают по причине неустойчивости потока в общем, и их размер определяется равноденствиям сил разного происхождения. Стоит заметить, что якобы «случайный процесс», однако, подчиняется некоторым закономерностям, извлечение которых проводится разными подходами.

На практике применяют термин «турбулентного вихря», а само явление турбулентности описывают как набор вихрей разного масштаба. Вихрь в несжимаемой жидкости удовлетворяет уравнению неразрывности с различным распределением угловой скорости. То есть движение несжимаемой жидкости можно представить в виде суперпозиции вихрей [6].

Наибольший размер вихрей достигает порядка *L* – характерного линейного масштаба рассматриваемой задачи. Примером того может быть пограничный слой или профиль под углом атаки. Чаще всего наиболее крупные вихри двигаются весьма упорядочено (движение жидкости за цилиндром). Структуры такого характера носят название когерентных [6].

Наименьшие по масштабам вихри диссипируют напрямую в тепловую энергию. Размер этих вихрей описывается колмогоровским масштабом

|  |  |
| --- | --- |
|  | (1) |

где – местная скорость диссипации на единицу массы, – кинематическая вязкость [6]. Вихри «среднего» размера переносят больше всего энергии.

В теории турбулентности используют и другие термины, такие как «характерное время жизни» и «характерный размер» турбулентного вихря. В случае, когда не ясно, какие вихри подразумеваются, понятия выше относятся к «средним» вихрям с наибольшей энергией.

Большое внимание уделяется выделением спектральных характеристик турбулентности. В этом случае изучают частотный спектр случайных пульсаций. Обычно, выделяют энергетические спектры, в которых кинетическая энергия турбулентности раскладывается в спектр.

Размер вихрей имеет связь с частотой колебаний вихря во времени. Самые большие вихри, двигаясь в пространстве, характеризуются колебаниями с длинным периодом (низкочастотные). Относительно небольшим вихрям, наоборот, сопутствуют высокочастотные колебания.

В состоянии равновесия, в то время как приток энергии от осредненного движения к турбулентному равняется энергетической передачи от турбулентности в тепловую энергию, максимальное значение турбулентных пульсаций устанавливается на определенной величине. Если амплитуда пульсаций случайно возрастает, то это приводит к уменьшению притока энергии от осредненного течения, однако уменьшение амплитуды приводит к возрастанию диссипации энергии в тепло. Описанные факторы приводят к уменьшению пульсаций. Рассуждая в обратную сторону, придем к тому же результату. Обобщив сказанное, придем к выводу, что в равновесном состоянии турбулентность статистически устойчива [6].

# **2 Обзор моделей макроскопических систем**

* 1. **Метод прямого моделирования Монте-Карло (DSMC)**

В основе метода лежит использование вероятностного подхода с целью нахождения решения уравнения Больцмана, удовлетворяющего потокам жидкости, которые имеют конечное числом Кнудсена (*Kn)*.

Данный метод прямого моделирования был сформирован профессором Грэмом Бёрдом. Берд является заслуженным профессором аэронавтики в сиднейском университете [7] [8]. DSMC представляет собой численный метод моделирования потоков разреженного газа, длина свободного пробега частицы которого имеет тот же порядок (возможно больше), что и репрезентативная физическая шкала длины (*Kn* > 1). В сверх- и гиперзвуковых течениях разрежение описывается параметром Цзяня, который пропорционален произведению *Kn* на число Маха *M* или , где Re – число Рейнольдса [9]. В таких разреженных течениях неточными способны оказаться уравнения Навье-Стокса. Метод DSMC был дополнен, что допустило моделировать сплошные течения (*Kn* < 1), также результаты работы модели можно было сравнить с решениями Навье-Стокса.

Метод DSMC воспроизводит потоки вещества с помощью имитационных частиц, которые представляют из себя большое количество настоящих молекул, которые моделируются вероятностным способом для того, чтобы решить уравнения Больцмана. Молекулы двигаются в пространстве физически точно, потому как непосредственно связаны с физическими часами системы. Таким образом возможно моделировать нестационарные характеристики течений. Столкновения между молекулами и молекул с поверхностью вычисляются с помощью использования вероятностных феноменологических моделей. Обобщенные модели молекул содержат в себе модель Hard Sphere (твердая сфера), модель Variable Hard Sphere (переменная твердая сфера) и модель Variable Soft Sphere (переменная мягкая сфера). Фундаментальная идея метода DSMC заключается в том, что моменты движения частиц и моменты их столкновений могут быть поделены на небольшие отрезки времени, которые меньше среднего времени до столкновения. Вариации моделей соударений содержаться в [10]. Ныне DSMC используется для решения таких задач, как оценка аэродинамических характеристик при входе в атмосферу космического корабля, моделирование микроэлектромеханических систем (МЭМС).

* 1. **Прямое численное моделирования (DNS)**

Этот подход заключается в решении трехмерных нестационарных уравнений Навье-Стокса. При этом используются пространственные сетки и шаги интегрирования делятся по времени, которых будет достаточно для разрешения каждой существенной неоднородности. Не сложно понять, что данный подход является максимально строгим, так как он основывается только на одной, вполне обоснованной гипотезе о применимости уравнений Навье-Стокса для описания турбулентных потоков вещества. Также, не менее очевиден и тот факт, что для его численной реализации важно использовать очень небольшие сетки, количество узлов которых должно быстро увеличиваться с ростом числа Рейнольдса.

В компетенции этого подхода важно разрешить наиболее мелкие вихри, размер которых равен колмогоровскому масштабу согласно формуле (1). То, как зависит количество узлов сетки в одном из направлений от числа Рейнольдсона выражено в следующей зависимости

|  |  |
| --- | --- |
|  | (2) |

Так как методу DNS необходим трехмерный нестационарный расчет, то расходы на вычисления растут пропорционально числу Рейнольдсона в степени 3. То есть, если числа Рейнольдса увеличить в 2 раза, то стоимость увеличивается примерно на порядок.

Подобные строгие требования несколько облегчаются, если использовать высокоточные спектральные методы численного интегрирования уравнений Навье-Стокса, которые зачастую применяются для метода DNS. Тем не менее, такие методы не могут быть применимы к расчету потоков вещества со сложной геометрией. Описанные трудности говорят о том, что практически DNS применяется исключительно для расчета простых турбулентных течений при довольно небольших (порядка единиц на 103 и ниже) числах Рейнольдса. При этом главной задачей вычислений является не конкретноо получение данных о характеристиках осредненного течения (они, как правило, известны), а извлечение детальной информации о структуре турбулентности, а также нахождение отдельных членов, входящих некоторые модели турбулентности [11-13].

Легко понять, что в ближайшем будущем использование прямого численного моделирования для разрешения практических задач является недоступным.

* 1. **Метод моделирования крупных вихрей (LES)**

Способ расчётов LES образовался в начале 80-х годов 20-го века. Принцип LES заключается в том, что в отличие от «глобального» осреднения уравнений Навье-Стокса применяется их «фильтрация» исключительно от коротковолновых (длины волн того же порядка или меньших размеров расчетной сетки) турбулентных неоднородностей.

Процесс фильтрации какой-либо функции заключается в ее домножении на так называемую «фильтрационную» функцию, которая содержит некоторый характерный линейный масштаб Δ, и в дальнейшем интегрировании найденного произведения по всему рассматриваемому объему *V*. В итоге получим, что отфильтрованные и актуальные значения функции определяются выражениями:

|  |  |
| --- | --- |
|  | (3) |

где - функция фильтра, – координата наблюдаемой точки потока, – актуальное значение функции, а , - ее отфильтрованное и пульсационное значения соответственно.

Заменив основные переменные в уравнении Навье-Стокса на сумму соответствующих отфильтрованных и пульсационных величин и применив операции фильтрации к полученным уравнениям, получим систему уравнений похожую на уравнения Рейнольдса. Но физическое значение описанных двух систем довольно отличное. Заострим внимание на том, что процесс фильтрации фактически то же само, что осреднение функции по объемам с характерным размером Δ3, в итоге этого вся информация о турбулентных структурах с размерами меньшими Δ (то есть о пульсационных или подсеточных составляющих ) исчезает, длинноволновые структуры в свою очередь (отфильтрованные составляющие ) практически не изменяются.

Учитывая это, воздействие отфильтрованных («подсеточных») структур на длинноволновые структуры турбулентного потока, разрешаемые в рамках метода LES «точно», описывается при помощи полуэмпирических моделей, которые весьма похожи по своей сути на традиционные модели.

Основное достоинство LES заключается в том, что с помощью относительной однородной и изотропной структуры турбулентности малых масштабов описание ее характеристик с помощью подсеточной модели выходит весьма более точным, нежели создания модели полного спектра турбулентных колебаний. Главной причиной являются то крупные структуры, которые зачастую бывают не настолько случайными, в такой ситуации не до конца оправдано использовать статистические моменты, чтобы описывать их свойства. Важно помнить, что непосредственно эти величины моделируются, используя такие подходы.

Необратимой платой за перечисленные выше примечательные достоинства LES подхода является существенное возрастание вычислительных затрат. Однако по ясным причинам (часть спектра мелкого масштаба моделируется, а не вычисляется «точно») расчетные ресурсы, требуемые для осуществления LES выходят намного меньшими, чем для DNS.

За последние годы распространился подход, называемый MILES. В рамках этого подхода подсеточная модель не применяется, ее «диссипативные» функции выполняет сеточная диссипация. При этом подходу необходим тщательный подбор расчетных сеток и применяемых схем [12-14].

* 1. **Приложения моделей**

Развитие моделей, симулирующих большой спектр течении, даст множеству прикладных дисциплин мощный инструмент получения практически важных характеристик.

Так, станет возможно моделировать течение крови в сосудах с большой точностью, предупреждать образование тромбов, бляшек, нервизмов с разными параметрами сосудов: измененные углы, разные условия адгезии стенок сосудов, учет притяжения между тромбоцитами.

В теплотехнической области, например, станет возможным улучшить модели теории горения за счет развития моделей турбулентных течений, реализованных рассматриваемым методом частиц.

Вопросы внешней аэродинамики также требуют всё точных моделей.

В задачах управляемого термоядерного синтеза теоретическое исследование плазмы не будет эффективным без моделирования физики процесса.

Обобщив написанное, придем к выводу, что практическая нужда вынуждает разрабатывать всё более эффективные, быстрые и не затратные (относительно вычислительных ресурсов) методов расчета разного рода течений. Всё это привело к тому, что в нынешнее время моделирование в области жидкостей и газов стало быстро и эффективно растущим направлением в области численного моделирования.

# **3 Реализация метода частиц с расчетом кванта времени**

* 1. **Описание компьютерной модели**
     1. Характеристики частиц

Реализация метода заключается в моделировании конечного статистически значимого количества твердых тел в двумерном пространстве. Частицы в модели обладают следующими характеристиками:

* Имеют форму окружности с радиусом .
* Электрически нейтральны.
* Масса нормирована на единицу (.
* Имеют координаты и векторы скорости .

Тела двигаться в пространстве по закону равномерного прямолинейного движения. Так, новые координаты определяются как приращение к текущей координате величины

|  |  |
| --- | --- |
| , | (4) |

которая является произведением величины текущего вектора скорости частицы и кванта времени системы .

Двигаясь, частица способна столкнутся с другим телом. Соударения считаются абсолютно упругими, то есть полная кинетическая энергия системы сохраняется после соударения частиц.

* + 1. Границы пространства

Пространство в модели ограничивается квадратной областью, середина которой расположена в декартовых координатах в точке . Размер стороны области обозначим за . Область имеет стенки с разными характеристиками. В зависимости от рассматриваемой задачи стенки могут быть

* абсолютно упругими (твердыми),
* с периодическими граничными условиями.

Соударение с абсолютно упругой стенкой приводит к изменению направления движения частицы. Если взаимодействие -й частицы происходит с вертикальной стенкой, то горизонтальная компонента вектора скорости меняет знак на противоположный. Аналогичное изменение вектора скорости происходит и при соударении тела с горизонтальной стенкой – меняет знак вертикальная составляющая.

Пролетая через стенку с периодическими граничными условиями, тело будет появляться с противоположной стороны области. Система является замкнутой самой на себя. Таким образом, ни одна частица не покинет систему.

* + 1. Распределение импульсов при соударении частиц

Для начала рассмотрим центральные удары абсолютно упругих частиц. В этом случае векторы скорости двух частиц и до удара и направлены вдоль прямой, соединяющей их центры.Если массы двух абсолютно упругих частиц одинаковы, то при их центральном столкновении они просто обмениваются скоростями [15]. Таким образом, скорости частиц и после центрального удара

|  |  |
| --- | --- |
|  | (5.1) |
|  | (5.2) |

Рассмотрим теперь нецентральный удар твердых упругих шаров. При таком столкновении в момент удара начальные скорости частиц не совпадают по направлению с линией центров. При нецентральном столкновении гладких идеально упругих шаров их тангенциальные скорости не изменяются. Нормальные же скорости изменяются так же, как и скорости при центральном ударе. Таким образом, получим нормальные компоненты скоростей частиц и после нецентрального соударения:

|  |  |
| --- | --- |
|  | (6.1) |
|  | (6.2) |

Рассмотрим в декартовой системе координат с осями и столкновение частиц и , которые имеют координаты, соответственно, и векторы скоростей ) и ).

Повернем систему координат на угол , который образован прямой, соединяющей центры частиц и осью . Таким образом,

|  |  |
| --- | --- |
|  | (7.1) |
|  | (7.2) |

После поворота перейдем в систему с осями и (рисунок 1).

|  |
| --- |
|  |
| Рисунок – 1. Система и |

Компоненты скоростей и частиц и в системе будут иметь координаты, соответственно, , и , , величина которых определяется формулами, выражающими в аналитической геометрии собственное вращение в прямоугольных декартовых координатах [16]:

|  |  |
| --- | --- |
|  | (8.1) |
|  | (8.2) |
|  | (8.3) |
|  | (8.4) |

При нецентральном ударе тангенциальные скорости () не изменяются, частицы обмениваются нормальными компонентами () согласно формулам (5.1) и (5.2). Таким образом, векторы скоростей частиц в системе принимают вид и .

Применим для компонент векторов и в системе формулы поворота координатной системы, перейдя обратно в систему и получив новые векторы скоростей , с компонентами, соответственно,

|  |  |
| --- | --- |
|  | (9.1) |
|  | (9.2) |
|  | (9.3) |
|  | (9.4) |

* + 1. Расчет времени до столкновения пары частиц в пространстве

Для того чтобы определить время до столкновения двух частиц и , необходимо решить систему уравнений

|  |  |
| --- | --- |
|  | (10) |

где , и , – координаты частиц и , соответственно, при контакте друг с другом, , и , – текущие координаты частиц и , соответственно, , и , – компоненты вектора скорости частиц и , соответственно, до соударения.

Система (10) содержит уравнения движения для компонент координаты двух частиц и условие, при котором наступает контакт двух тел – расстояние между их центрами должно равняться удвоенному радиусу частиц.

Частицы не столкнутся, если система (10) не имеет решений или все корни отрицательные. Столкновение возможно только при положительном корне. Если оба корня положительные, то отбирается наименьший из них.

Таким образом, наименьшее положительное значение, полученное при решении системы (10) является временем до столкновения двух частиц.

* + 1. Расчет времени до столкновения частицы с твердой стенкой

Расчет времени до соударения -й частицы c одной из горизонтальных стенок задается следующим уравнением

|  |  |
| --- | --- |
|  | (11) |

В числителе рассчитывается расстояние, которое частица должна преодолеть до столкновения со стенкой с учетом размера частицы, направления движения и текущей координаты. Знаменатель есть модуль скорости по оси .

Аналогично формулируется формула расчета времени до столкновения частицы с вертикальной стенкой.

* + 1. Расчет времени до столкновения двух частиц с учетом периодических условий.

Положим, что в системе есть две вертикальные стенки с периодическими граничными условиями и пара частиц и . При расчете времени до столкновения частиц необходимо учитывать замкнутость системы саму на себя. Для этого частице создается две копии и , которые удалены по оси от настоящей на расстояния и соответсвенно.

Далее решается система (10) для следующих пар частиц: и , и , и . В итоге для одной из пар система (10) может иметь положительные решения. Если для трех пар частиц не нашлось неотрицательного решения, то частицы не столкнутся.

Аналогичные рассуждения справедливы и для горизонтальных стенок с периодическими граничными условиями.

* + 1. Эволюция системы и квант времени

Система эволюционирует последовательно, каждый раз прибавляя к времени эволюции системы квант времени – время до первого столкновения либо пары частиц, либо частицы с твёрдой стенкой в текущем состоянии системы. Рассчитав квант времени , изменяются координаты частиц согласно формуле (4). Каждое столкновение приводит к перераспределению импульсов у частиц, что вынуждает снова пересчитывать квант времени.

* + 1. Программный алгоритм

Программный код, реализующий метод частиц с расчетом кванта времени написан на языке C++ с использованием программно-аппаратной архитектуры параллельных вычислений CUDA.

Выбор языка обусловлен его высокой производительностью и популярностью среди разработки ПО. Технология CUDA выбрана в целях высокой скорости вычислений за счет параллельного выполнения задачи на графическом процессоре (GPU), чего не в состоянии сделать центральный процессор (CPU), работающий последовательно. При проведении вычислений на GPU задача разделяется на множество потоков, выполняемых параллельно. Тандем GPU и CPU образует гибридную систему вычислений, которая обеспечивает эффективную работу алгоритма.

Расчеты проводились на базе ЦКП «Центр данных ДВО РАН» при помощи GPU ускорителя NVIDIA Tesla P100.

Листинг кода компьютерной программы представлен в приложении «А».

Далее описываются основные вычисления, которые удалось осуществить посредством разделения на множество независимых потоков.

Выясним, сколько существует неповторяющихся пар частиц в системе с заданным количеством частиц . Рассмотрим матрицу взаимодействий пар частиц и

|  |  |
| --- | --- |
|  | (12) |

Элементы матрицы исключаются, поскольку частицы не могут сталкиваться сами с собой. Число таких элементов равно . Взаимодействия и равнозначны. Таким образом, в рассматриваемой системе возможно

|  |  |
| --- | --- |
|  | (13) |

неповторяющихся пар частиц.

Разделим выполнение части программы, в которой рассчитывается время до первого столкновения двух частиц при текущем состоянии системы, на потоков, каждому потоку присваивается своя пара частиц и . Распределение номеров частиц по потокам представлено в таблице 1.

Таблица 1. Распределение номеров частиц по потокам.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | … |  |  |  | … |  |
|  | 0, 1 | 0, 2 | … | 0, | 1, 2 | 1, 3 | … |  |

Результатом работы каждого потока является время до столкновения частиц и в текущем состоянии системы.

Совокупность всех результатов обозначим как . Наименьшее значение из совокупности результатов будет являться временем до первого столкновения двух частиц в текущем состоянии системы.

Однако некоторая частица может столкнуться с твердой стенкой, при этом столкновение будет первым в текущем состоянии системы. Тогда нам необходимо найти квант времени, спустя который произойдет первое столкновение частицы со стенкой. Рассчитав квант времени для двух частиц и частицы со стенкой, выбираем наименьший с последующим приращением координат частиц согласно формуле (4).

Допустим, имеется две горизонтальных твердых стенки. Создадим потоков для всех частиц и в каждом потоке инициируем вычисления времени полета частицы до соударения с одной из стенок согласно формуле (11). Для оптимизации необходимо исключить из расчетов частицы, находящиеся далеко от стенки. В программном коде это реализовано проверкой расстояния частицы до стенки. Если оно превышает несколько средних свободных пробегов тел в системе, то частица исключается из расчета.

Если возникает необходимость найти наименьшее значение в массиве данных (например, минимальное время в системе до соударения одной из пар частиц), используется параллельная редукция – алгоритм, которой оптимизирует поиск минимального значения за счет разбиения массива на несколько независимых частей, в которых ведется одновременно.

Параллельному вычислению поддаются и координаты частиц. После расчета кванта времени нужно совершить приращение к координатам всех частиц согласно формуле (4). Для этого каждая частица получает приращение координаты в отдельном потоке, что ускоряет работу программы в сравнении с последовательным изменением координаты каждой частицы в цикле.

* 1. **Результаты численного расчета**
     1. Планковские единицы

Планковские единицы – это система единиц измерения, опирающаяся на основные фундаментальные константы. В 1901 году была введена Максом Планком [17].

Система единиц Макса Планка распространилась недостаточно широко, так как единицы измерения, входящие в эту систему, практически не применимы. Тем не мене, система Планка успешно используется в теоретической физике по причине простоты использования уравнений, которые освобождаются от множества лишних коэффициентов [18].

В настоящее время планковская система единиц – система единиц, основу которой составляют фундаментальные физические постоянные [19]:

* – постоянная Больцмана
* – скорость света
* – постоянная Дирака
* – гравитационная постоянная

В планковской системе фундаментальные постоянные нормированы на единицу: , , , . Коэффициент пропорциональности в законе Кулоне также равен единице [20].

Результаты вычислений в текущей главе представлены в планковских единицах измерения. Необходимость введения таких единиц измерения связано с определением связи между температурой и энергией.

Введем обозначения для этих единиц:

* планковская длина – пд
* планковское время – пв
* планковская масса – пм
* планковская энергия – пэ
* планковская температура – пт
  + 1. Расчетные формулы

Средняя кинетическая энергия хаотичного поступательного движения частиц пропорциональна половине произведению абсолютной температуры и постоянной Больцмана на каждую степень свободы [21].

|  |  |
| --- | --- |
|  | (14) |

где

|  |  |
| --- | --- |
|  | (15) |

Число степеней свободы показывает минимальное количество независимых переменных (обобщённых координат), необходимых для полного описания состояния механической системы [21].

Учитывая, что пространство двумерное, система обладает двумя степенями свободы (). Также учтем, что постоянная Больцмана , поскольку используется планковская система единиц. Тогда получим выражение для температуры рассчитываемой системы согласно формуле (14):

|  |  |
| --- | --- |
|  | (16) |

Концентрация двумерной системы определяется как

|  |  |
| --- | --- |
|  | (17) |

где – количество частиц, – площадь двумерной области системы [21].

Плотность численно равна концентрации, поскольку массы всех частицы равны единице, тогда

|  |  |
| --- | --- |
|  | (18) |

Средняя длина свободного пробега частицы в системе определяется как произведение средней скорости теплового движения и среднего времени между двумя столкновениями [21]:

|  |  |
| --- | --- |
|  | (19) |

Частицы представляют собой твердые окружности одного размера, взаимодействующие между собой только при соударении. Пусть – эффективное сечение частицы, то есть полное поперечное сечение рассеяния, характеризующее столкновение между двумя молекулами. Площадь, в которую не может проникнуть центр любой другой частицы . Здесь – диаметр частицы.

За одну секунду частица проходила путь, равный средней арифметической скорости . За ту же секунду частица претерпевает столкновений [21]. Следовательно,

|  |  |
| --- | --- |
|  | (20) |

Найдем число столкновений . Вероятность столкновения трех и более частиц бесконечно мала. Положим, что все частицы остановились, кроме одной. Ее траектория будет представлять собой ломанную линию. Сталкиваться частица будет только с частицы, центры которых лежат внутри прямоугольника шириной .

Путь, который пройдет частица за одну секунду, равен длине прямоугольника . Умножим площадь прямоугольника на число частиц в квадратной области , получим среднее число столкновений в одну секунду:

|  |  |
| --- | --- |
|  | (21) |

В действительности, все частицы движутся (и в сторону, и навстречу друг другу), поэтому число соударений определяется средней скоростью движения частиц относительно друг друга [21].

По закону сложения случайных величин

|  |  |
| --- | --- |
|  | (22) |

Таким образом получим

|  |  |
| --- | --- |
|  | (23) |

Найдем выражение для коэффициента вязкости в двумерной системе частиц. Пусть слой 1 лежит под линией на расстоянии средней длины свободного пробега частиц . Тогда частицы, летящие из слоя 1 по направлению к линии , достигнут ее без столкновений. Число частиц , пролетающих через линию за время из слоя 1:

|  |  |
| --- | --- |
|  | (24) |

где – средняя скорость теплового движения частиц, – число частиц в единице площади (концентрация). Множитель 1/4 учитывает равную вероятность движения частиц обоих направлений каждой из координатных осей.

Такое же количество частиц из слоя 2 пересекают линию в противоположном направлении. Но, поскольку скорости и слоев 1 и 2 различны, то в прямом и обратном направлениях будут перенесены различные импульсы и направленного движения:

|  |  |
| --- | --- |
|  | (25.1) |
|  | (25.2) |

В результате этих двух переносов импульса, происходящих в противоположных направлениях, через линию будет перенесен от слоя 2 к слою 1 импульс

|  |  |
| --- | --- |
|  | (26) |

Разность абсолютных величин скоростей , можно выразить через градиент скорости . Градиент скорости есть величина, численно равная изменению величины скорости движения слоев на единицу длины в направлении, перпендикулярном линии слоя. Для нашего случая градиент скорости равен , где .

Тогда разность равна градиенту скорости, умноженному на расстояние между слоями 1 и 2, то есть на :

|  |  |
| --- | --- |
|  | (27) |

Следовательно формула (26) принимает вид

|  |  |
| --- | --- |
|  | (28) |

Заметим, что есть плотность газа , получим

|  |  |
| --- | --- |
|  | (29) |

Сила внутреннего трения , действующая со стороны более медленного слоя 1 на более быстрый слой 2, согласно второму знакому Ньютона равна приращению импульса слоя 2 в единицу времени

|  |  |
| --- | --- |
|  | (30) |

Введем обозначение

|  |  |
| --- | --- |
|  | (31) |

тогда

|  |  |
| --- | --- |
|  | (32) |

Сила внутреннего трения, возникающего при макроскопических движениях в газе или жидкости, прямо пропорциональна градиенту скорости. Коэффициент пропорциональности η носит название коэффициента внутреннего трения, или динамического коэффициента вязкости. Коэффициент вязкости (или динамическая вязкость) численно равна силе внутреннего терния, действующей на единицу площади.

Введем теоретическую функцию распределения плотности вероятности Максвелла-Больцмана для состояния термодинамического равновесия системы частиц [21]. Распределение по скоростям строится согласно функции

|  |  |
| --- | --- |
|  | (34) |

* + 1. Переход системы к равновесию

В исследовании явления необратимости перехода системы в термодинамическое равновесие была численно рассчитана система частиц со следующими параметрами:

* Количество частиц
* Размер стороны квадратной области пд
* Начальный модуль скорости пд / пс со случайной ориентацией вектора
* Температура системы пэ
* Концентрация (плотность) частиц / пд2 (пм / пд2)
* Средняя длина пробега пд
* Динамическая вязкость пм / (пд\*пс)
* Частицы равномерно расположены в области

Система частиц эволюционировала в течение времени пс с фиксацией состояний системы в разных моментах времени. Гистограммы состояний представлены на рисунке 2.

|  |
| --- |
|  |
| Рисунок 2 – Гистограммы состояний системы в разные моменты времени |

По оси абсцисс отложен модуль скорости частиц , по оси ординат – количество частиц в диапазоне скоростей . Каждая гистограмма отмечена цветом, которому соответствует определённый момент времени, указанный на рисунке.

Найдем время релаксации системы по эволюции вида кривой функции распределения. Стационарность распределения будет говорить о наступлении в системе термодинамического равновесия согласно разделу 1.1.

Для численного построения кривой распределения использована библиотека seaborn. Был построен график оценки плотность ядра. График оценки плотности ядра (KDE) – это метод визуализации распределения наблюдений в наборе данных, аналогичный гистограмме. KDE представляет данные с использованием непрерывной кривой плотности вероятности в одном или нескольких измерениях [21].

По сравнению с гистограммой KDE может создавать график, который менее загроможден и более интерпретируем, особенно при рисовании нескольких распределений.

На рисунке 3 представлена эволюция распределения вероятностей.

|  |
| --- |
|  |
| Рисунок 3 – Эволюция состояния системы, описываемое кривой распределения |

Из рисунка 3 можно наблюдать стационарность распределения спустя некоторое время. Для определения этого времени рассмотрим состояние в промежутке времени от близкого к равновесному и конечному времени эволюции. Результаты представлены на рисунке 4.

|  |
| --- |
|  |
| Рисунок 4 – Распределения вероятностей в промежутке времени от близкого к равновесию до конечного времени эволюции |

Из рисунка 4 видно, что начиная с времени пс, система перешла в состояние термодинамического равновесия. Таким образом, пс. Далее наблюдается дрейф кривой распределению в небольшом диапазоне, что говорит о флуктуациях состояния – небольших отклонениях от среднего значения.

В результате построения кривой распределения по скоростям, удалось рассчитать эволюцию моды скорости (рисунок 5).

|  |
| --- |
|  |
| Рисунок 5 – Эволюция моды скорости частиц |

По характеру изменения моды скорости определена точка, в которой система частиц перешла в равновесное состояние. После момента времени пс мода скорости начала колебаться после монотонного уменьшения. Найденное время релаксации по графику изменения моды скорости с течением времени удовлетворяет времени , найденному из эволюции кривой распределения по скоростям (рисунок 4).

На рисунке 6 представлено изменение максимальной скорости частиц в системе с течением времени.

|  |
| --- |
|  |
| Рисунок 6 – Изменение максимальной скорости частиц в системе |

Тенденция изменения максимальной скорости в системе имеет схожий характер с изменением моды скорости. Обе скорости меняются монотонно до определенного момента времени, после которого как максимальная, так и наивероятнейшая скорость претерпевают колебания.

Сравним численную функцию распределения вероятностей скоростей в состоянии термодинамического равновесия и функцию распределения Максвелла-Больцмана. Для сравнения приведем численные кривые распределений всех состояний и кривую теоретической функции после момента наступления в системе термодинамического равновесия (рисунок 7).

|  |
| --- |
|  |
| Рисунок 7 – Численные кривые распределений вероятности в разные моменты времени в состоянии термодинамического равновесия и теоретическое распределение Максвелла-Больцмана (пунктиром), построенное по формуле (34) |

Как видно из рисунка 7, численные кривые после наступления равновесия в системе располагаются согласно кривой распределения Максвелла-Больцмана, что говорит об удовлетворении эксперимента теоретическим ожиданиям.

Выводы:

* Было проверено одно из наиболее важных положений термодинамики – любая замкнутая макросистема с течением времени приходит в состояние равновесия, в котором физические величины, характеризующие систему, не меняются во времени, и остается в этом состоянии неопределенно долго. Система частиц, где преобладают нецентральные абсолютно упругие соударения, со случайно заданными направлениями скоростей в итоге перешла в состояние термодинамического равновесия. Это характеризуется соответствием кривых распределения вероятностей в состоянии равновесия, определенных численно, с теоретической кривой распределения Максвелла-Больцмана.
* Установлены параметры, при которых система может являться равновесной. Так, до момента наступления равновесия, мода и максимальное значение скорости изменяются монотонно: мода уменьшается, максимальное значение возрастает. В момент наступления равновесия обе величины начинают хаотически колебаться. Наступление колебаний моды и максимального значения скорости соответствует моменту времени, при котором кривые распределения вероятностей с последующим течением времени начинают располагаться по теоретической кривой распределения Максвелла-Больцмана.
* Для рассматриваемой системы частиц определено время релаксации. Система перешла в состояние термодинамического равновесия спустя время пс.
  + 1. Исследование турбулентности в системе упругих частиц

В исследовании явления турбулентности в системе абсолютно упругих частиц была рассчитана система частиц со следующими параметрами:

* Количество частиц
* Размер стороны квадратной области пд
* Начальная скорость, модуль которой равен 100 пд / пс со случайной ориентацией вектора
* К начальной скорости частиц добавляется -я компонента размером в 1000 пд / пс. Половине частиц добавляется -1000, другой половине +1000. Таким образом были созданы два противоположных потока.
* Температура системы пэ
* Концентрация (плотность) пм / пд2 (частиц / пд2)
* Средняя длина пробега пд
* Динамическая вязкость пм / (пд\*пс)
* Частицы равномерно расположены в области

Область системы была ограничена двумя горизонтальными твердыми стенками и двумя вертикальными с периодическими граничными условиями.

В модели были рассчитаны два противоположных потока частиц, интерес представлял граничный слой между потоками. Вопрос состоял в том, какие эффекты будут возникать на границе между двумя потоками, возникнет ли турбулентность, появятся ли устойчивые вихри.

В начальный момент времени система выглядела следующим образом (рисунок 8).

|  |
| --- |
|  |
| Рисунок 8 – Система двух потоков в начальный момент времени |

На рисунке 8 красным цветом отмечены частицы первого потока, который движется преимущественно вправо. Синие частицы принадлежат потоку, движущемуся преимущественно влево. На протяжении эволюции системы цвет частиц не менялся. Это сделано в целях детального наблюдения за взаимодействием двух потоков. В частности, получится четко увидеть перемешивание потоков.

Эволюция системы двух потоков сопровождалась изменением ее состояния. Система эволюционировала на протяжении времени пс с фиксацией состояния в различные моменты времени. Численно были построены кривые распределения вероятностей для каждого состояния системы (рисунок 9).

|  |
| --- |
|  |
| Рисунок 9 – Определенные численно распределения вероятностей для каждого состояния системы и распределение Максвелла-Больцмана |

Так, система приблизилась к термодинамическому равновесию (рисунок 10).

|  |
| --- |
|  |
| Рисунок 10 – Распределение вероятностей, определенное численно (синяя кривая) в момент последней фиксации состояния системы ( пс), и кривая распределения Максвелла-Больцмана (пунктирная линия) |

Рассматривая эффекты на границе потоков, были замечены области с низкой концентрацией частиц. Для отслеживания концентрации частиц в разных областях системы, область была разделена на 64 ячейки. Минимальная концентрация частиц была замечена в момент времени пс и составила частиц / (пд\*пд). Данная концентрация была обнаружена на границе двух потоков. Карта распределения концентраций в момент времени пс представлена на рисунке 11.

|  |
| --- |
|  |
| Рисунок 11 – Карта распределения концентраций в системе в момент ее наименьшего значения в одной из ячеек |

В процессе эволюции в термодинамическом состоянии близкому к равновесному, концентрация частиц в каждой ячейке близка к первоначальной средней концентрации по всем ячейкам. Карта распределения плотностей в конечном состоянии, близком к равновесному представлена на рисунке 12.

|  |
| --- |
|  |
| Рисунок 12 – Карта распределения концентраций в системе в конечный момент времени пс |

В области на границе раздела двух потоков рассчитаны численные кривые распределения вероятностей для каждого состояния системы (рисунок 13).

|  |
| --- |
|  |
| Рисунок 13 – Эволюция состояния области с границей раздела двух потоков |

Как видно из рисунка 13, в области на границе раздела двух потоков состояние системы также приближается к состоянию равновесия. Численные кривые несколько не совпадают с кривой распределения Максвелла-Больцмана, однако их форма удовлетворяет теоретической кривой. Рассмотрим график изменения средней кинетической энергии в области (рисунок 14).

|  |
| --- |
|  |
| Рисунок 14 – Изменение средней кинетической энергии в области раздела потоков |

Внушительное уменьшение средней кинетической энергии связано с потерей вещества из области , о чем свидетельствует распределение концентраций в системе в момент времени пс (рисунок 11). После момента времени пс, в области происходят затухающие колебания (пульсации) средней кинетической энергии и наблюдаются характерные структуры на границе между потоками (рисунок 15).

|  |
| --- |
|  |
| Рисунок 15 – Образование вихрей в системе |

После затухания пульсаций средней кинетической энергии ( пс), на границе раздела двух потоков перестают образовываться структуры, система стремится к хаосу (рисунок 16).

|  |
| --- |
|  |
| Рисунок 16 – Граница раздела потоков после прекращения пульсаций средней кинетической энергии |

Рассмотрим области , которая расположена выше области , и . Для областей и были построены кривые распределения вероятностей (рисунок 17).

|  |  |
| --- | --- |
|  |  |
| Рисунок 17 – Эволюция состояний в областях и | |

Из рисунка 17 видно, что области и приблизились к состоянию равновесия, причем эволюция состояний этих областей идентичны.

На рисунке 18 приведено изменение средней кинетической энергии областей и .

|  |
| --- |
|  |
| Рисунок 18 – Изменения средней кинетической энергии областей и |

Изменение средних кинетических энергии двух крайних областей происходит идентично в первые моменты времени, имеют место пульсации. Затухание пульсаций средней кинетической энергии в областях и происходит в момент времени пс, как и в области , после чего средние кинетические энергии крайних областей значительно меняются относительно своего состояния до затухания пульсаций.

В конечном наблюдаемом состоянии система частиц изображена на рисунке 19.

|  |
| --- |
|  |
| Рисунок 19 – Система двух потоков в конечный момент времени |

Выводы:

* Численно рассчитанная система двух противоположных по направлению потоков приблизилась к состоянию термодинамического равновесия. Эволюция данной системы характеризуется разрушением потоков, начиная с их границы раздела, смешиванием течений (рисунок 19). Возрастание хаотичности движения разрушает направленные течения, что приводит к состоянию, близкому к равновесному.
* На границе раздела потоков в области наблюдается большая потеря концентрации. Это приводит к увеличению концентраций в крайних областях, прижимая вещество к стенкам. Потоки налетают друг на друга на границе раздела после столкновения со стенками. Это приводит к образованию вихревых структур на границе раздела потоков. Концентрация выравнивается во всей системе с течением времени.
* По характеру изменения средней кинетической энергии в трех областях были замечены ее пульсации. Отрезок времени, при котором происходили резкие колебания средней кинетической энергии во всех областях, совпадает с вихревым движением частиц системы. После затухания пульсаций вихревое движение перестает существовать.

# **Заключение**

В ходе научно-исследовательской работы был разработан алгоритм расчета эволюции системы абсолютно упругих частиц. Алгоритм был реализован посредством написания компьютерной программы с использованием эффективной технологии параллельных расчетов CUDA.

Исследована необратимость перехода системы частиц в термодинамическое равновесие. Результаты выполнения компьютерной программы предоставили информацию о поведении параметров системы до перехода в равновесное состояние и их изменение после наступления равновесия.

Исследована система двух противоположно направленных потоков абсолютно упругих частиц. Наблюдалось уменьшение концентрации в области границы раздела двух потоков. Уменьшение концентрации вызывало уменьшение средней кинетической энергии в области близ границы раздела потоков. Далее были обнаружены пульсации средней кинетической энергии, которые характеризовались на визуализации как турбулентные образования.

Причиной необратимости перехода от неравновесного состояния до равновесного, которое сопровождается изменением распределения частиц по скоростям, является принцип минимума свободной энергии. Переход от неравновесного состояния к равновесному приводит к увеличению дисперсии по скоростям, что приводит к увеличению энтропии.

# **Список литературы**

1. Langmuir I. Oscillations in ionized gases / I. Langmuir // Proceedings of the National Academy of Sciences. – 1928. – Т. 14. – № 8. – С. 627–637.
2. Бутиков Е.И. Физика для углубленного изучения 1. Механика: учебник/ Е.И. Бутиков, А.С. Кондратьев. – М.: Физматлит, 2004. – 350 с.;
3. Физическая энциклопедия: в 5 т. / гл. ред. А. М. Прохоров. – М.: Большая российская энциклопедия, 1994. – Т. 4. – 704 с.
4. Аминов Л.К. Термодинамика и статистическая физика. Конспекты лекций и задачи. – Казань: Казан. ун-т, 2015. – 180 с.
5. Гиневский А.С. Теория турбулентных струй и следов: учебное пособие. – М. Машиностроение, 1969. – 400 c.;
6. Фрост У. Турбулентность. Принципы и применения: учебное пособие/ У. Фрост, Т. Моулден. – М.: Мир, 1980. – 535 c.;
7. G. A. Bird. Molecular Gas Dynamics. – Oxford: Clarendon Press, 1976. – 238 с.;
8. G. A. Bird. Molecular Gas Dynamics and the Direct Simulation of Gas Flows. – Oxford: Clarendon Press, 1994. – 484 с.;
9. Tsien, Hsue-Shen. Superaerodynamics, Mechanics of Rarefied Gases / Tsien, Hsue-Shen // Journal of the Aeronautical Sciences. – 1946. – 13 (12): 653–64.
10. Roohi, E., Stefanov, S. Collision partner selection schemes in DSMC: From micro/nano flows to hypersonic flow / E. Roohi, , S. Stefanov // Physics Reports. – 2016. – 656 (1): 1–38.
11. Alfonsi G. On Direct Numerical Simulation of Turbulent Flows / G. Alfonsi // Applied Mechanics Reviews. – 2011. – 64 (2): 020802. – 33 c.
12. И.А. Белов, С.А. Исаев. Моделирование турбулентных течений. — СПб.: Балт. гос. техн. ун-т, 2001.
13. К.Н. Волков, В.Н. Емельянов. Моделирование крупных вихрей в расчетах турбулентных течений. — М.: ФИЗМАТЛИТ, 2008.
14. Bouffanais. R. Advances and challenges of applied large-eddy simulation / R. Bouffanais // Computers & Fluids. – 2010. – Т. 39. – № 5. – С. 735–738.
15. Д.В. Сивухин. Общий курс физики: в 2 т. Т 1. – М.: Наука, 1979. – 520 с.;
16. Ю.Г. Игнатьев. Аналитическая геометрия. Часть II. Аффинные и евклидовы пространства: учебное пособие/ Игнатьев Ю.Г. – Казань: ТГГПУ, 2013. – 188 с.;
17. Х.Г. Шёпф. Макс Планк Об элементарных квантах материи и электричества // От Кирхгофа до Планка. — М.: Мир, 1981. – 192 c.;
18. Л. А. Сена. Единицы физических величин и их размерности. – М.: Наука, 1968. – 306 с.;
19. А. Г. Чертов. Единицы физических величин. – М.: «Высшая школа», 1977. – 287 с.;
20. К. А. Томилин. Фундаментальные физические постоянные в историческом и методологическом аспектах. – Москва: Физматлит, 2006. – 368 с.;
21. Д.В. Сивухин. Общий курс физики: в 2 т. Т 2. – М.: Наука, 1990. – 591 с.;
22. Seaborn.kdeplot: Seaborn. [Электронный ресурс]. – URL: https://seaborn.pydata.org/generated/seaborn.kdeplot.html

# **Приложение А**

**(обязательное)**

**Листинг кода компьютерной программы**

#include <iostream>

#include <stdio.h>

#include <math.h>

#include <chrono>

#include <fstream>

#define PI 3.141592653589793

#define DOUBLE\_MAX 1.7976931348623157E+308

#define blocks 16380

#define blockSize 512

#define EPS 1E-12

#define FREE\_PATH 2.03646753

#define CUDA\_CHECK\_ERROR(err) \

if ((err) != cudaSuccess) { \

printf("Cuda error: %s\n", cudaGetErrorString(err)); \

printf("Error in file: %s, line: %i\n", \_\_FILE\_\_, \_\_LINE\_\_); \

}

using namespace std;

class Timer

{

private:

using clock\_t = std::chrono::high\_resolution\_clock;

using second\_t = std::chrono::duration<double, std::ratio<1> >;

std::chrono::time\_point<clock\_t> m\_beg;

public:

Timer() : m\_beg(clock\_t::now())

{

}

void reset()

{

m\_beg = clock\_t::now();

}

double elapsed() const

{

return std::chrono::duration\_cast<second\_t>(clock\_t::now() - m\_beg).count();

}

};

\_\_device\_\_ double search\_D\_t(double x1, double y1, double x2, double y2, double vx1, double vy1, double vx2, double vy2, double R) {

double A, B, C, D;

double t1, t2;

A = (vx1 - vx2)\*(vx1 - vx2) + (vy1 - vy2)\*(vy1 - vy2);

B = 2\*((x1-x2)\*(vx1 - vx2) + (y1-y2)\*(vy1 - vy2));

C = (x1-x2)\*(x1-x2) + (y1-y2)\*(y1-y2) - 4\*R\*R;

D = B\*B - 4\*A\*C;

if (D < 0) {return DOUBLE\_MAX;}

t1 = (-B - (double)sqrt(D)) / 2 / A;

t2 = (-B + (double)sqrt(D)) / 2 / A;

if (abs(t1) < EPS) {

t1 = 0;

}

if (abs(t2) < EPS) {

t2 = 0;

}

if (t1 < 0 && t2 > 0) {return t2;}

if (t1 > 0 && t2 < 0) {return t1;}

if (t1 > 0 && t2 > 0 && t1 < t2) {return t1;}

if (t1 > 0 && t2 > 0 && t2 < t1) {return t2;}

return DOUBLE\_MAX;

}

\_\_device\_\_ double search\_r(double x1, double y1, double x2, double y2) {

return sqrt((x1-x2)\*(x1-x2)+(y1-y2)\*(y1-y2));

}

\_\_global\_\_ void kernel\_pwa(double \*x, double \*y, double \*vx, double \*vy, int \*m, int \*l, double \*tmin0, double limit, double R) {

int index = blockIdx.x \* blockDim.x + threadIdx.x;

double t;

int mm, ll;

mm = m[index];

ll = l[index];

for(int i = 0; i < 5; i++) {

switch(i) {

case 0: t = search\_D\_t(x[mm], y[mm], x[ll], y[ll], vx[mm], vy[mm], vx[ll], vy[ll],R); break;

case 1: t = search\_D\_t(x[mm]-2\*limit, y[mm], x[ll], y[ll], vx[mm], vy[mm], vx[ll], vy[ll],R); break;

case 2: t = search\_D\_t(x[mm]+2\*limit, y[mm], x[ll], y[ll], vx[mm], vy[mm], vx[ll], vy[ll],R); break;

case 3: t = search\_D\_t(x[mm], y[mm]+2\*limit, x[ll], y[ll], vx[mm], vy[mm], vx[ll], vy[ll],R); break;

case 4: t = search\_D\_t(x[mm], y[mm]-2\*limit, x[ll], y[ll], vx[mm], vy[mm], vx[ll], vy[ll],R); break;

}

if (t != DOUBLE\_MAX) {

break;

}

}

tmin0[index] = t;

\_\_syncthreads();

}

\_\_global\_\_ void kernel\_pwv(double \*x, double \*y, double \*vx, double \*vy, int \*m, int \*l, double \*tmin0, double limit, double R) {

// periodic walls vertical

int index = blockIdx.x \* blockDim.x + threadIdx.x;

double t;

int mm, ll;

mm = m[index];

ll = l[index];

for(int i = 0; i < 3; i++) {

switch(i) {

case 0: t = search\_D\_t(x[mm], y[mm], x[ll], y[ll], vx[mm], vy[mm], vx[ll], vy[ll],R); break;

case 1: t = search\_D\_t(x[mm]-2\*limit, y[mm], x[ll], y[ll], vx[mm], vy[mm], vx[ll], vy[ll],R); break;

case 2: t = search\_D\_t(x[mm]+2\*limit, y[mm], x[ll], y[ll], vx[mm], vy[mm], vx[ll], vy[ll],R); break;

}

if (t != DOUBLE\_MAX) {

break;

}

}

tmin0[index] = t;

\_\_syncthreads();

}

\_\_global\_\_ void kernel\_hwh(int R, double \*y, double \*vy, double \*time, double limit) {

int index = blockIdx.x \* blockDim.x + threadIdx.x;

if (limit - abs(y[index]) > 10\*FREE\_PATH)

time[index] = DOUBLE\_MAX;

} else {

time[index] = ( limit - R - y[index] \* vy[index] / abs(vy[index]) ) / abs(vy[index]);

}

\_\_syncthreads();

}

\_\_device\_\_ double atomicMin\_double(double\* address, double val)

{

unsigned long long int\* address\_as\_ull = (unsigned long long int\*) address;

unsigned long long int old = \*address\_as\_ull, assumed;

do {

assumed = old;

old = atomicCAS(address\_as\_ull, assumed,

\_\_double\_as\_longlong(fmin(val, \_\_longlong\_as\_double(assumed))));

} while (assumed != old);

return \_\_longlong\_as\_double(old);

}

\_\_global\_\_ void cuda\_minimum(double\* in, unsigned int size, double\* out)

{

\_\_shared\_\_ double buf [blockSize];

unsigned int index = blockIdx.x \* blockDim.x + threadIdx.x;

if (index < size)

buf[threadIdx.x] = in[index];

else

buf[threadIdx.x] = in[0];

index += blocks \* blockSize;

while (index < size) {

buf[threadIdx.x] = fmin(buf[threadIdx.x], in[index]);

index += blocks \* blockSize;

}

\_\_syncthreads();

for (int s = blockSize / 2; s > 0; s >>= 1) {

if (threadIdx.x < s)

buf[threadIdx.x] = min(buf[threadIdx.x], buf[threadIdx.x + s]);

\_\_syncthreads();

}

if (threadIdx.x == 0) {

atomicMin\_double(out, buf[0]);

}

\_\_syncthreads();

}

\_\_global\_\_ void search\_min\_index(double \*times, double min\_t, int \*min\_index) {

int index = blockIdx.x \* blockDim.x + threadIdx.x;

if (times[index] == min\_t) {

min\_index[0] = index;

}

}

\_\_global\_\_ void dXdY\_limit(double \*x, double \*y, double \*vx, double \*vy, double time, double lim) {

int index = blockIdx.x \* blockDim.x + threadIdx.x;

x[index] = x[index] + vx[index]\*time;

y[index] = y[index] + vy[index]\*time;

if(x[index] < -lim)

{x[index]=x[index]+2\*lim;}

if(x[index] > lim)

{x[index]=x[index]-2\*lim;}

if(y[index] < -lim)

{y[index]=y[index]+2\*lim;}

if(x[index] > lim)

{y[index]=y[index]-2\*lim;}

\_\_syncthreads();

}

\_\_device\_\_ void dVX\_dVY(double \*x, double \*y, double \*vx, double \*vy, int p1, int p2, double R) {

double vx1\_, vx2\_, vy1\_, vy2\_;

double cosa = (x[p2]-x[p1])/2/R;

double sina = (y[p2]-y[p1])/2/R;

vx1\_ = vx[p1]\*cosa+vy[p1]\*sina;

vy1\_ = -vx[p1]\*sina+vy[p1]\*cosa;

vx2\_ = vx[p2]\*cosa+vy[p2]\*sina;

vy2\_ = -vx[p2]\*sina+vy[p2]\*cosa;

vx[p1] = vx2\_\*cosa-vy1\_\*sina;

vy[p1] = vx2\_\*sina+vy1\_\*cosa;

vx[p2] = vx1\_\*cosa-vy2\_\*sina;

vy[p2] = vx1\_\*sina+vy2\_\*cosa;

}

\_\_host\_\_ double search\_V(double vx, double vy) {

return sqrt(vx\*vx + vy\*vy);

}

\_\_global\_\_ void dist\_impulses\_pwv(double \*x, double \*y, double \*vx, double \*vy, int \*min\_i, int \*m, int \*l, double lim, double R) {

double cor\_r;

int min;

int sample = 0;

min = min\_i[0];

for (int k = 0; k < 3; k++) {

switch(k) {

case 0: cor\_r = search\_r(x[m[min]], y[m[min]], x[l[min]], y[l[min]]); break;

case 1: cor\_r = search\_r(x[m[min]] - 2\*lim, y[m[min]], x[l[min]], y[l[min]]); break;

case 2: cor\_r = search\_r(x[m[min]] + 2\*lim, y[m[min]], x[l[min]], y[l[min]]); break;

}

if (abs(cor\_r - 2\*R) < EPS) {

sample = k;

break;

}

}

switch(sample) {

case 0: break;

case 1: x[m[min]] = x[m[min]] - 2\*lim; break;

case 2: x[m[min]] = x[m[min]] + 2\*lim; break;

}

dVX\_dVY(x, y, vx, vy, m[min], l[min], R);

switch(sample) {

case 0: break;

case 1: x[m[min]] = x[m[min]] + 2\*lim; break;

case 2: x[m[min]] = x[m[min]] - 2\*lim; break;

}

}

\_\_global\_\_ void dist\_impulses\_pwa(double \*x, double \*y, double \*vx, double \*vy, int \*min\_i, int \*m, int \*l, double lim, double R) {

double cor\_r;

int min;

int sample = 0;

min = min\_i[0];

for (int k = 0; k < 5; k++) {

switch(k) {

case 0: cor\_r = search\_r(x[m[min]], y[m[min]], x[l[min]], y[l[min]]); break;

case 1: cor\_r = search\_r(x[m[min]] - 2\*lim, y[m[min]], x[l[min]], y[l[min]]); break;

case 2: cor\_r = search\_r(x[m[min]] + 2\*lim, y[m[min]], x[l[min]], y[l[min]]); break;

case 3: cor\_r = search\_r(x[m[min]], y[m[min]] + 2\*lim, x[l[min]], y[l[min]]); break;

case 4: cor\_r = search\_r(x[m[min]], y[m[min]] - 2\*lim, x[l[min]], y[l[min]]); break;

}

if (abs(cor\_r - 2\*R) < EPS) {

sample = k;

break;

}

}

switch(sample) {

case 0: break;

case 1: x[m[min]] = x[m[min]] - 2\*lim; break;

case 2: x[m[min]] = x[m[min]] + 2\*lim; break;

case 3: y[m[min]] = y[m[min]] + 2\*lim; break;

case 4: y[m[min]] = y[m[min]] - 2\*lim; break;

}

dVX\_dVY(x, y, vx, vy, m[min], l[min], R);

switch(sample) {

case 0: break;

case 1: x[m[min]] = x[m[min]] + 2\*lim; break;

case 2: x[m[min]] = x[m[min]] - 2\*lim; break;

case 3: y[m[min]] = y[m[min]] - 2\*lim; break;

case 4: y[m[min]] = y[m[min]] + 2\*lim; break;

}

}

\_\_global\_\_ void dist\_impulses\_hwh(double \*x, double \*y, double \*vx, double \*vy, int \*min\_i) {

int min = min\_i[0];

vy[min] = -vy[min];

}

\_\_host\_\_ void init\_threads (int N, int threads, int \*m, int \*l) {

int k, step;

k = 0;

step = N - 1;

for (int i = 0; i < threads; i++) {

if (step == 0) {

k++;

step = N - 1 - k;

}

m[i] = k;

l[i] = N - step;

step--;

}

}

\_\_host\_\_ bool readFile(double \*array, string filename, int N) {

setlocale(LC\_ALL, "rus");

ifstream file (filename);

if(!file) {

return 0;

} else {

for (int i = 0; i < N; i++) {

file >> array[i];

}

return 1;

}

}

int main() {

printf("\n//------------------------------//\n\n");

int preset = 1; // 0 - труба и два потока, 1 - 4 стенки с периодическими условиями и случайные скорости

int N = 4096; // количество частиц

double R = 1; // радиус частицы (ПЛАНКОВСКИХ МЕТРОВ)

double vmod0 = 100; // начальный модуль скорости (ПЛАНКОВСКИХ ДЛИН В ПЛАНКОВСКУЮ СЕКУНДУ)

int boost = 1000; // для создания двух потоков (ПЛАНКОВСКИХ ДЛИН В ПЛАНКОВСКУЮ СЕКУНДУ)

int cells = 4096; // корень должен давать целое

double cell\_size = 2.4; // размер клетки для инициализации частиц (ПЛАНКОВСКИХ ДЛИН)

int seed = 900;

if (2\*R>cell\_size) {

printf("Частица не влезает в ячейку\n");

return 0;

}

srand(seed);

string file\_x("//home//npacuk//FINAL//x.txt"); // пути к файлам с координатами и скоростями

string file\_y("//home//npacuk//FINAL//y.txt");

string file\_vx("//home//npacuk//FINAL//vx.txt");

string file\_vy("//home//npacuk//FINAL//vy.txt");

Timer t;

switch(preset) {

case 0:

printf("\nДве горизонтальных твердых стенки");

printf("\nДве вертикальных периодических стенки");

printf("\nДва противоположных потока\n\n");

break;

case 1:

printf("\n4 периодических стенки");

printf("\nСлучайная ориентация векторов скорости\n");

break;

}

bool flag; // выбор: новый запуск/продолжение

printf("%i частиц\n\n", N);

printf("Новый запуск? (yes-0 / no-1)? ");

scanf("%i", &flag);

printf("\n");

int iter;

printf("Введите количество итераций кода iter: ");

scanf("%i", &iter);

printf("\n");

int coll[iter];

bool flag1;

printf("Указать одно значение столкновений для всех итераций - 0\n");

printf("Указать разные количества столкновений за итерации - 1\n");

flag1 = 0;

scanf("%i", &flag1);

if (flag1 == 0) {

printf("Укажите одно количество столкновений для всех итераций: ");

scanf("%i", &coll[0]);

for (int i = 1; i < iter; i++) {

coll[i] = coll[0];

}

} else {

printf("Введите количества соударений на каждую итерацию кода colli: \n");

for (int i = 0; i < iter; i++) {

printf("iter %i/%i:", i+1, iter);

scanf("%i", &coll[i]);

printf("\n");

}

}

double \*x, \*y, \*vx, \*vy;

double sumE0, sumE1; // для контроля сохранения энергии

double system\_time;

double \*out;

int \*m;

int \*l;

double border;

int threads;

double tmin, tmin2;

double \*dev\_x, \*dev\_y, \*dev\_vx, \*dev\_vy;

double \*dev\_tmin0, \*dev\_tmin02, \*dev\_out, \*dev\_out2;

int \*dev\_min\_index, \*dev\_min\_index2, \*dev\_m, \*dev\_l;

threads = (N \* N - N) / 2; // количество нитей

double dSIZE = threads \* sizeof(double);

int iSIZE = threads \* sizeof(int);

cudaMalloc((void \*\*) &dev\_x, dSIZE);

cudaMalloc((void \*\*) &dev\_y, dSIZE);

cudaMalloc((void \*\*) &dev\_vx, dSIZE);

cudaMalloc((void \*\*) &dev\_vy, dSIZE);

cudaMalloc((void \*\*) &dev\_m, iSIZE);

cudaMalloc((void \*\*) &dev\_l, iSIZE);

cudaMalloc((void \*\*) &dev\_tmin0, dSIZE);

cudaMalloc((void \*\*) &dev\_tmin02, dSIZE);

cudaMalloc((void \*\*) &dev\_out, dSIZE);

cudaMalloc((void \*\*) &dev\_out2, dSIZE);

cudaMalloc((void \*\*) &dev\_min\_index, iSIZE);

cudaMalloc((void \*\*) &dev\_min\_index2, iSIZE);

x = (double \*) malloc(dSIZE);

y = (double \*) malloc(dSIZE);

vx = (double \*) malloc(dSIZE);

vy = (double \*) malloc(dSIZE);

m = (int \*) malloc(iSIZE);

l = (int \*) malloc(iSIZE);

out = (double \*) malloc(dSIZE);

init\_threads(N, threads, m, l);

int a = sqrt(cells);

border = a\*cell\_size/2.;

sumE0 = 0;

sumE1 = 0;

int np = 0; // number particle

switch (flag) {

case 0:

for (int i = 0; i < a; i++) {

for(int k = 0; k < a; k++) {

x[np] = - a\*cell\_size/2 + cell\_size/2 + i\*cell\_size;

y[np] = - a\*cell\_size/2 + cell\_size/2 + k\*cell\_size;

np++;

}

}

for (int i = 0; i < N; i++) {

vx[i] = ((vmod0 \* (double) rand() / RAND\_MAX) \* 2) - vmod0;

if (rand() % 2 == 0) {

vy[i] = sqrt(vmod0 \* vmod0 - vx[i] \* vx[i]);

} else {

vy[i] = -(sqrt(vmod0 \* vmod0 - vx[i] \* vx[i]));

}

}

if (preset == 0) {

for (int i = 0; i < N; i++) {

if (y[i] > 0) {

vx[i] = vx[i] + boost;

} else {

vx[i] = vx[i] - boost;

}

}

}

break;

case 1:

if (readFile(x, file\_x, N) == 1) {printf("Координаты x считаны\t+\n");}

else {printf("Ошибка:\tКоординаты x не считаны!!!\n"); return 0;}

if (readFile(y, file\_y, N) == 1) {printf("Координаты y считаны\t+\n");}

else {printf("Ошибка:\tКоординаты y не считаны!!!\n"); return 0;}

if (readFile(vx, file\_vx, N) == 1) {printf("Скорости vx считаны\t+\n");}

else {printf("Ошибка:\tСкорости x не считаны!!!\n"); return 0;}

if (readFile(vy, file\_vy, N) == 1) {printf("Скорости vy считаны\t+\n");}

else {printf("Ошибка:\tСкорости x не считаны!!!\n\n"); return 0;}

break;

}

for (int i = 0; i < N; i++) {

sumE0 += vx[i]\*vx[i] + vy[i]\*vy[i];

}

sumE0 = sumE0/2

printf("Начальная энергия: %.16f\n", sumE0);

if (flag == 0) {

FILE\* f0;

f0 = fopen("info.csv", "w");

fprintf(f0,"N,R,vmod0,boost,cell\_size,cells,seed,threads,blocks,blockSize\n");

fprintf(f0,"%i,%f,%f,%i,%f,%i,%i,%i,%i,%i\n",N,R,vmod0,boost,cell\_size,cells,seed,threads,blocks,blockSize);

fclose(f0);

FILE\* f1;

f1 = fopen("log.csv", "w");

fprintf(f1,"x,y,vx,vy,V,coll,evol\_time,prog\_time\n");

for (int i = 0; i < N; i++) {

fprintf(f1,"%.16f,%.16f,%.16f,%.16f,%.16f", x[i], y[i], vx[i], vy[i], search\_V(vx[i], vy[i]));

if (i == 0) {

fprintf(f1,",0,0,0");

} fprintf(f1,"\n");

} fclose(f1);

}

t.reset();

CUDA\_CHECK\_ERROR(cudaMemcpy(dev\_m, m, iSIZE, cudaMemcpyHostToDevice));

CUDA\_CHECK\_ERROR(cudaMemcpy(dev\_l, l, iSIZE, cudaMemcpyHostToDevice));

CUDA\_CHECK\_ERROR(cudaMemcpy(dev\_x, x, dSIZE, cudaMemcpyHostToDevice));

CUDA\_CHECK\_ERROR(cudaMemcpy(dev\_y, y, dSIZE, cudaMemcpyHostToDevice));

CUDA\_CHECK\_ERROR(cudaMemcpy(dev\_vx, vx, dSIZE, cudaMemcpyHostToDevice));

CUDA\_CHECK\_ERROR(cudaMemcpy(dev\_vy, vy, dSIZE, cudaMemcpyHostToDevice));

FILE\* f3;

switch(preset) {

case 0: for (int j = 0; j < iter; j++) {

system\_time = 0;

for (int i = 0; i < coll[j]; i++) {

out[0] = DOUBLE\_MAX;

kernel\_pwv <<<threads/blockSize, blockSize>>>(dev\_x, dev\_y, dev\_vx, dev\_vy, dev\_m, dev\_l, dev\_tmin0, border, R);

CUDA\_CHECK\_ERROR(cudaMemcpy(dev\_out, out, dSIZE, cudaMemcpyHostToDevice));

cuda\_minimum <<<blocks, blockSize>>>(dev\_tmin0, threads, dev\_out);

CUDA\_CHECK\_ERROR(cudaMemcpy(out, dev\_out, dSIZE, cudaMemcpyDeviceToHost));

tmin = out[0];

out[0] = DOUBLE\_MAX;

kernel\_hwh <<<N/blockSize, blockSize>>> (R, dev\_y, dev\_vy, dev\_tmin02, border);

CUDA\_CHECK\_ERROR(cudaMemcpy(dev\_out2, out, dSIZE, cudaMemcpyHostToDevice));

cuda\_minimum <<<blocks, blockSize>>>(dev\_tmin02, N, dev\_out2);

CUDA\_CHECK\_ERROR(cudaMemcpy(out, dev\_out2, dSIZE, cudaMemcpyDeviceToHost));

tmin2 = out[0];

if (tmin < tmin2) {

system\_time += tmin;

search\_min\_index <<<blocks, blockSize>>> (dev\_tmin0, tmin, dev\_min\_index);

dXdY\_limit <<<N/blockSize, blockSize>>>(dev\_x, dev\_y, dev\_vx, dev\_vy, tmin, border);

dist\_impulses\_pwv <<<1,1>>> (dev\_x, dev\_y, dev\_vx, dev\_vy, dev\_min\_index, dev\_m, dev\_l, border, R);

} else {

system\_time += tmin2;

search\_min\_index <<<N/blockSize, blockSize>>> (dev\_tmin02, tmin2, dev\_min\_index2);

dXdY\_limit <<<N/blockSize, blockSize>>>(dev\_x, dev\_y, dev\_vx, dev\_vy, tmin2, border);

dist\_impulses\_hwh <<<1,1>>> (dev\_x, dev\_y, dev\_vx, dev\_vy, dev\_min\_index2);

}

}

CUDA\_CHECK\_ERROR(cudaMemcpy(x, dev\_x, dSIZE, cudaMemcpyDeviceToHost));

CUDA\_CHECK\_ERROR(cudaMemcpy(y, dev\_y, dSIZE, cudaMemcpyDeviceToHost));

CUDA\_CHECK\_ERROR(cudaMemcpy(vx, dev\_vx, dSIZE, cudaMemcpyDeviceToHost));

CUDA\_CHECK\_ERROR(cudaMemcpy(vy, dev\_vy, dSIZE, cudaMemcpyDeviceToHost));

sumE1 = 0;

for (int i = 0; i < N; i++) {

sumE1 += vx[i]\*vx[i] + vy[i]\*vy[i];

}

sumE1 = sumE1/2.;

if (abs(sumE0/sumE1 - 1.) > 0.1) {

printf("\n!!! Ошибка в законе сохранения энергии !!!\n");

break;

} else {

FILE\* f21;

f21 = fopen("x.txt", "w");

for(int i = 0; i < N; i++) {

fprintf(f21, "%.16f\n", x[i]);

} fclose(f21);

FILE\* f22;

f22 = fopen("y.txt", "w");

for(int i = 0; i < N; i++) {

fprintf(f22, "%.16f\n", y[i]);

} fclose(f22);

FILE\* f23;

f23 = fopen("vx.txt", "w");

for(int i = 0; i < N; i++) {

fprintf(f23, "%.16f\n", vx[i]);

} fclose(f23);

FILE\* f24;

f24 = fopen("vy.txt", "w");

for(int i = 0; i < N; i++) {

fprintf(f24, "%.16f\n", vy[i]);

} fclose(f24);

f3 = fopen("log.csv", "a");

for (int i = 0; i < N; i++) {

fprintf(f3,"%.16f,%.16f,%.16f,%.16f,%.16f", x[i], y[i], vx[i], vy[i], search\_V(vx[i], vy[i]));

if (i == 0) {

fprintf(f3,",%i,%.16f,%.6f", coll[j], system\_time, t.elapsed()/60.);

}

fprintf(f3,"\n");

} fclose(f3);

printf("%i/%i готово, прошло %.5f min\n", j+1, iter, t.elapsed()/60.);

}

} break;

case 1: for (int j = 0; j < iter; j++) {

system\_time = 0;

for (int i = 0; i < coll[j]; i++) {

out[0] = DOUBLE\_MAX;

kernel\_pwa <<<threads/blockSize, blockSize>>>(dev\_x, dev\_y, dev\_vx, dev\_vy, dev\_m, dev\_l, dev\_tmin0, border, R);

CUDA\_CHECK\_ERROR(cudaMemcpy(dev\_out, out, dSIZE, cudaMemcpyHostToDevice));

cuda\_minimum <<<blocks, blockSize>>>(dev\_tmin0, threads, dev\_out);

CUDA\_CHECK\_ERROR(cudaMemcpy(out, dev\_out, dSIZE, cudaMemcpyDeviceToHost));

tmin = out[0];

system\_time += tmin;

search\_min\_index <<<blocks, blockSize>>> (dev\_tmin0, tmin, dev\_min\_index);

dXdY\_limit <<<N/blockSize, blockSize>>>(dev\_x, dev\_y, dev\_vx, dev\_vy, tmin, border);

dist\_impulses\_pwa <<<1,1>>> (dev\_x, dev\_y, dev\_vx, dev\_vy, dev\_min\_index, dev\_m, dev\_l, border, R);

}

CUDA\_CHECK\_ERROR(cudaMemcpy(x, dev\_x, dSIZE, cudaMemcpyDeviceToHost));

CUDA\_CHECK\_ERROR(cudaMemcpy(y, dev\_y, dSIZE, cudaMemcpyDeviceToHost));

CUDA\_CHECK\_ERROR(cudaMemcpy(vx, dev\_vx, dSIZE, cudaMemcpyDeviceToHost));

CUDA\_CHECK\_ERROR(cudaMemcpy(vy, dev\_vy, dSIZE, cudaMemcpyDeviceToHost));

sumE1 = 0;

for (int i = 0; i < N; i++) {

sumE1 += vx[i]\*vx[i] + vy[i]\*vy[i];

}

sumE1 = sumE1/2.;

if (abs(sumE0/sumE1 - 1.) > 0.1) {

printf("\n!!! Ошибка в законе сохранения энергии !!!\n");

break;

} else {

FILE\* f21;

f21 = fopen("x.txt", "w");

for(int i = 0; i < N; i++) {

fprintf(f21, "%.16f\n", x[i]);

} fclose(f21);

FILE\* f22;

f22 = fopen("y.txt", "w");

for(int i = 0; i < N; i++) {

fprintf(f22, "%.16f\n", y[i]);

} fclose(f22);

FILE\* f23;

f23 = fopen("vx.txt", "w");

for(int i = 0; i < N; i++) {

fprintf(f23, "%.16f\n", vx[i]);

} fclose(f23);

FILE\* f24;

f24 = fopen("vy.txt", "w");

for(int i = 0; i < N; i++) {

fprintf(f24, "%.16f\n", vy[i]);

} fclose(f24);

f3 = fopen("log.csv", "a");

for (int i = 0; i < N; i++) {

fprintf(f3,"%.16f,%.16f,%.16f,%.16f,%.16f", x[i], y[i], vx[i], vy[i], search\_V(vx[i], vy[i]));

if (i == 0) {

fprintf(f3,",%i,%.16f,%.6f", coll[j], system\_time, t.elapsed()/60.);

}

fprintf(f3,"\n");

} fclose(f3);

printf("%i/%i готово, прошло %.5f min\n", j+1, iter, t.elapsed()/60.);

}

} break;

}

printf("\nКонечная энергия: %.16f\n", sumE1);

printf("Время исполнения программы (минут): %.5f\n", t.elapsed()/60.);

printf("Время эволюции системы: %.5f\n", system\_time);

free(x);

free(y);

free(vx);

free(vy);

free(m);

free(l);

cudaFree(dev\_x);

cudaFree(dev\_y);

cudaFree(dev\_vx);

cudaFree(dev\_vy);

cudaFree(m);

cudaFree(l);

return 0;

}