**Государственное бюджетное общеобразовательное учреждение города Москвы «Школа № 1573»**

**«РАЗРАБОТКА ПРИЛОЖЕНИЯ «СИМУЛЯТОР ИДЕАЛЬНОГО ГАЗА»**

Участник:

учащаяся 10 «И» класса ГБОУ Школа

№1573 Лотошвили Арина Гурамовна

Руководитель**:**

Педагог ГБОУ Школа №1573

Назаров Никита Владимирович

­­­­

Содержание

[ВВЕДЕНИЕ 3](#_Toc159792552)

[Актуальность 3](#_Toc159792553)

[Цель и задачи 4](#_Toc159792554)

[Оснащение и оборудование 4](#_Toc159792555)

[Методика выполнения работы по созданию приложения для визуализации модели идеального газа 5](#_Toc159792556)

[Физические законы, описывающие свойства идеального газа и поведение его молекул 5](#_Toc159792557)

[Численное моделирование и преимущества Объектно-ориентированного программирования 6](#_Toc159792558)

[Столкновения 7](#_Toc159792559)

[Параметры и графики 11](#_Toc159792560)

[Разработка рабочего приложения для моделирования и изучения идеального газа. 15](#_Toc159792561)

[Проверка корректной работы симулятора. Выполнение закона сохранения энергии, установление теоретического распределения частиц по скоростям. 17](#_Toc159792562)

[Результаты и перспективы 22](#_Toc159792563)

[Список используемой литературы 24](#_Toc159792564)

# ВВЕДЕНИЕ

Идеальный газ – теоретическая модель, в которой пренебрегается взаимодействием между молекулами в газе, за исключением кратких моментов столкновений. Хорошим приближением к модели идеального газа является разреженный в достаточной степени любой реальный газ. Молекулы идеального газа в основном находятся в состоянии равномерного и прямолинейного движения. Все направления движения в отсутствие внешнего поля (или при его малости) равновероятны. Движение имеет хаотический характер: после каждого столкновения скорости и направления движения существенным образом меняются.

Такой газ легче подогнать под численное моделирование, а так как его свойства во многих случаях коррелируют со свойствами реальных газов, то есть при измерениях реального газа мы получаем тот же результат, что и у модели.

# Актуальность

В рамках изучения молекулярно-кинетической теории (МКТ) в школьном курсе физики не все свойства идеального газа рассматриваются детально. Например, распределение частиц по скоростям и энергиям (распределение Максвелла) изучается лишь на качественном уровне, а чаще всего - вовсе опускается. Несмотря на сложность рассмотрения данной темы с математической точки зрения в 10 классе, возможно получить количественные результаты с помощью численного моделирования идеального газа, основанного на законах механики. Также численное моделирование позволяет изучить флуктуации температуры, давления, количества частиц, которые в школьном курсе не рассматриваются и не могут быть рассмотрены в принципе, ввиду недостаточных математических компетенций для этого у школьников.

Модель идеального газа оказалась настолько универсальной, что физики применяют её не только для газов, подобных воздуху, но и для электронного газа в металле, для излучения электромагнитных волн и даже для звуковых колебаний в кристаллах.

Теория идеального газа позволяет оценить давление и температуру внутри звёзд, результаты таких оценок близки к полученным строгими расчётами.

# Цель и задачи

Цель проекта — разработать приложение «Симулятор идеального газа», которое может быть использовано в образовательных и научных целях для изучения свойств газов в рамках молекулярно-кинетической теории строения вещества.

Для достижения поставленной цели были сформулированы следующие задачи:

1. Изучить физические законы, описывающие свойства идеального газа;
2. Разработать механическую модель идеального газа и реализовать ее с помощью ООП
3. Изучить существующие подходы к численному моделированию идеального газа;
4. Разработать программу для моделирования, визуализации и вычисления физических характеристик идеального газа;
5. Проверить корректность работы программы;
6. Организовать функционал по сбору данных для построения графиков;
7. Создать рабочее приложение;
8. Реализовать удобный пользовательский интерфейс.

# Оснащение и оборудование

* ­­­Персональный компьютер (ноутбук) с установленным ПО
* Python 3.9 с интегрированной средой разработки Pycharm Community 2023.2.3
* Библиотеки Matplotlib, NumPy, Pygame, PyQt5­­­­­­­­­­­­­­

Было решено использовать язык программирования Python, так как он больше подходит для анализа и визуализации данных. Как раз-таки из-за необходимости использовать инструменты для визуализации я выбрала данный язык программирования и набор модулей языка программирования Python – pygame, который очень удобен для реализации моего проекта.

Pycharm был очень удобен из-за интеграции в него GitHub, его интерфейс легко позволял работать с репозиторием без использования сайта. Для моего проекта достаточно Pycharm Community.

Для построения графиков я использовала matplotlib, а для реализации векторов модуль Numpy.

PyQt выбран из-за своего огромного количества виджетов и удобной документации к ним. Отвечает за реализацию графического интерфейса.

# Методика выполнения работы по созданию приложения для визуализации модели идеального газа

## Физические законы, описывающие свойства идеального газа и поведение его молекул

Чтобы реализовать моделирование идеального газа, нужно изучить его свойства, они будут главными параметрами, описывающими движение частиц, которые нужно будет отобразить в коде.

Построена следующая механическая модель двумерного идеального газа:

* Частицы представляют собой упругие шары (круги). Их суммарный объем (площадь) много меньше объема (площади) сосуда;
* Частицы не взаимодействуют на расстоянии;
* При столкновениях частиц друг с другом происходит абсолютно упругий удар;
* При столкновениях частиц со стенками происходит абсолютно упругий удар.

Важно отметить, что моделирование решено делать в двухмерном пространстве, поэтом средний квадрат проекции скорости на одну из осей OX или OY будет равен половине среднего квадрата самой скорости.

Учитывая, что столкновения частиц между собой и удары о стенки (границы) экрана являются абсолютно упругими, скорости после соударения двух частиц можно описать следующей формулой (рисунок 1):

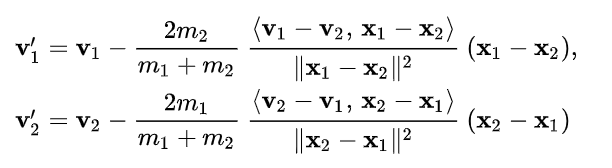


Рисунок 1 - Формула упругого удара двух сферических частиц

V’1, V’2 – конечные скорости начальных скоростей V1 и V2 соответственно. «Иксы» - радиус векторы соударяющихся частиц.

А при столкновении частиц со стенкой направление проекции вектора на ось, перпендикулярную данной стенке, меняется на противоположное, остальные параметры не изменяются (рисунок 2).

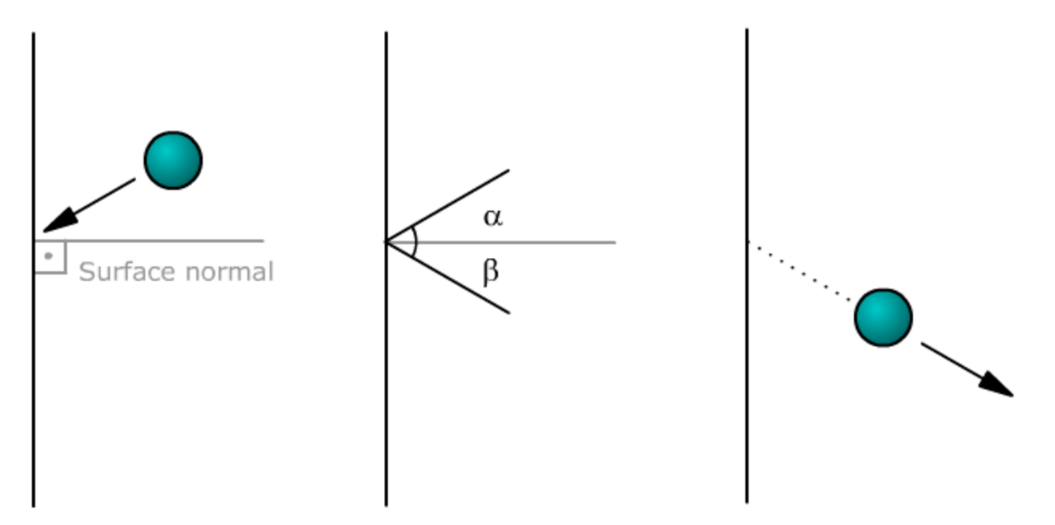


Рисунок 2 – упругое столкновение со стенкой

## Численное моделирование и преимущества Объектно-ориентированного программирования

**Численное моделирование** — это математическое представление физического (или иного) поведения, основанное на соответствующих гипотезах и упрощающих допущениях. Оно позволяет определить параметры технологических, физических процессов, которые не могут быть измерены в ходе натурных испытаний. Использование численных методов позволяет сократить стоимость исследований, повысить их скорость и выявить ряд аспектов, которые могут быть не выявлены при физическом моделировании.

**Объектно-ориентированное программирование (ООП)** ­- самая используемая парадигма программирования. Главным понятием ООП является понятие программного объекта, в данном случае частица, которая имеет свои свойства – масса, скорость, радиус-вектор, радиус и методы – передвижение. Так как эта концепция – то, как мы описываем объекты реальной жизни, ООП очень подходит для структурирования кода, его упрощения и читабельности. Язык Python – типичный представитель ООП-семейства, обладающий элегантной и мощной объектной моделью.

В своем проекте я реализовала класс частицы (программный объект) так (рисунок 3):

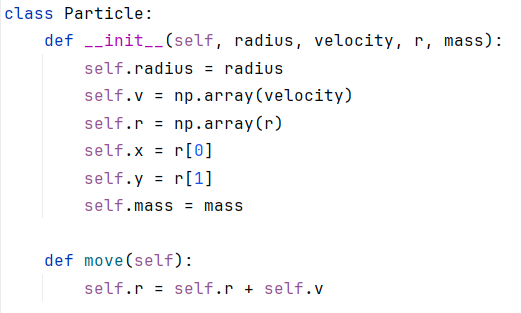


Рисунок 3 – реализация класса Частица

Свойствами моего объекта (класса) Частица являются радиус (radius), скорость (velocity), радиус-вектор (r) и масса (mass).

Метод (действия, которые может выполнять Частица) – передвижение (move). Этот метод реализован векторно: к радиус-вектору прибавляется вектор скорости.

## Столкновения

Столкновение со стенкой (рисунок 4) реализовано так, что при столкновении со стенкой, где координата x равна либо нулю (а так как координата принадлежит центру шарика, то к нулю нужно прибавить радиус), либо равна размеру окна (так же с учетом радиуса), умножая на -1, меняю направление проекции скорости на ось OX на противоположное. По оси OY аналогично.

Скорость умножается на 100 для того, чтобы перевести ее в м/с.

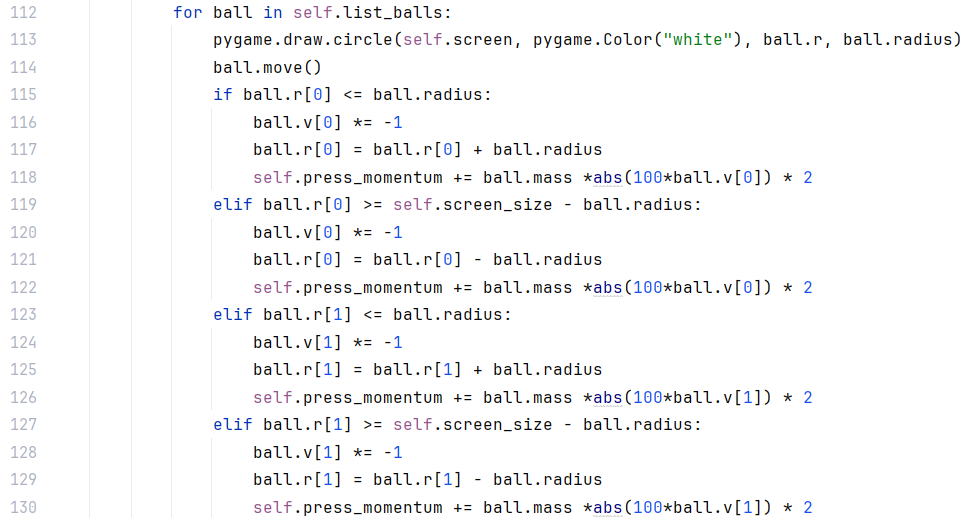


Рисунок 4 - функция "столкновение со стенкой"

Строчки 117, 121, 125, 129 сделаны для того, чтобы из-за дискретного времени частица не оказалась за пределами поля, ведь иначе она там застрянет (рисунок 5).



Рисунок 5 - застревание частиц

Соударение частиц (рисунок 6) обрабатывается с помощь вышеупомянутой формулы (рисунок 1), массу можно не писать, так как она одинакова и обращается в 1, также уменьшение обращений к объектам класса оптимизирует код. Наличие столкновение проверяется теоремой Пифагора.

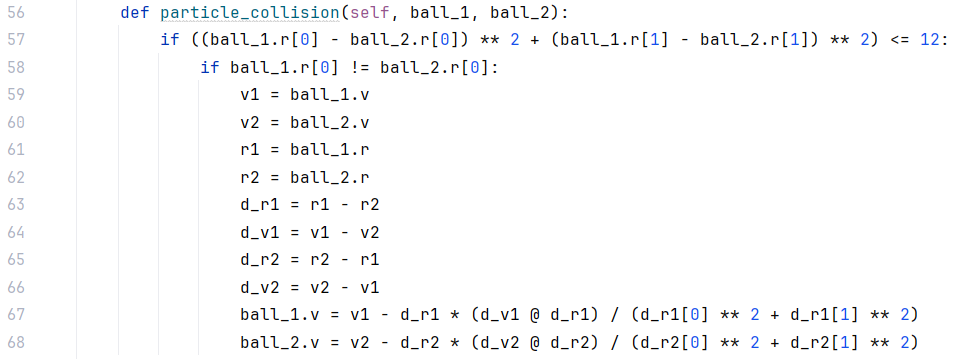


Рисунок 6 - обработка соударения частиц

Условие в 57 строчке нужно для предотвращения “слипания” частиц, ведь радиус их взаимодействия больше радиуса самих частиц.

Строчка 58 нужна для того, чтобы температура, подсчет которой будет в коде дальше, не уходила в тип «nun».

Как “Слипание”, так и застревание частиц в стенке, происходит из-за дискретности времени в моделировании, а коэффициент помогает перехватить в каких-то случаях вхождение одной частицы в другую, однако это не помогает полностью избавиться от этой проблемы (рисунок 7).



Рисунок 7 - "слипание" частиц

В игровом цикле, когда перебираются частицы (отбираются два шарика вложенными циклами) для проверки наличия и обработки столкновения, перебор всех частиц (алгоритм грубого перебора brute-force) будет очень неэффективным (приложение нормально работает максимум при 500 частицах). По этой причине было решено использовать алгоритм пространственного разбиения (space partioning). Его суть в том, что вызов функции обработки столкновений вызывается не для любой пары из всего пространства, а только для той пары, которая находится в данной части пространства. Конкретнее, поле разбивается на 4 части (квадрата), в каждой из которых пара частиц перебирается отдельно (рисунок 8).

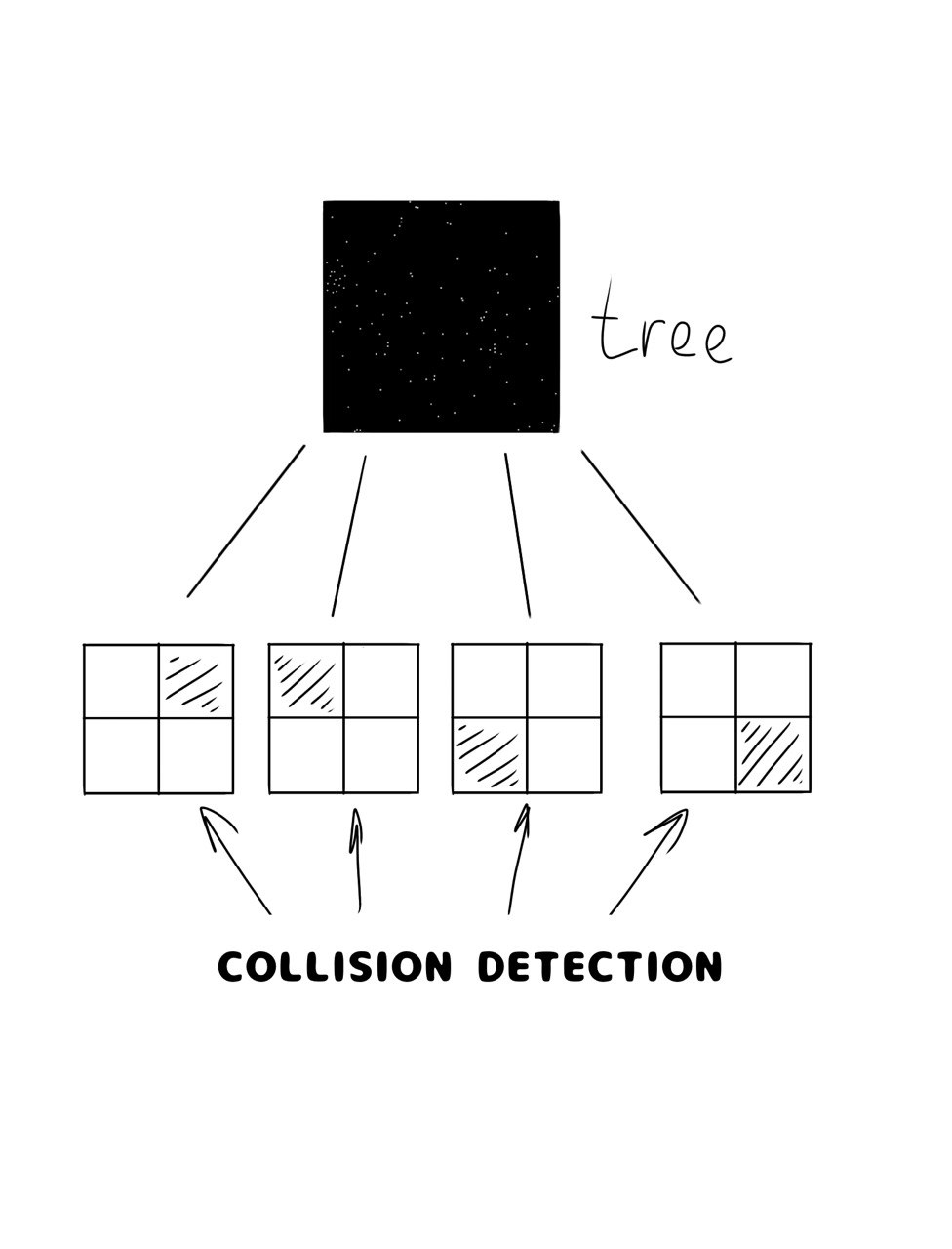


Рисунок 8 - схема перебора списка частиц для обработки их столкновения

Программно этот алгоритм реализован списком из четырех элементов — вложенных списков, в каждом из которых находятся частицы, координаты которого входят в область одного из четырех квадратов (рисунок 9). После в каждом вложенном списке отдельно обрабатываются пара частиц и их столкновение. С помощью этого алгоритма приложение неплохо обрабатывает и 1000 частиц.

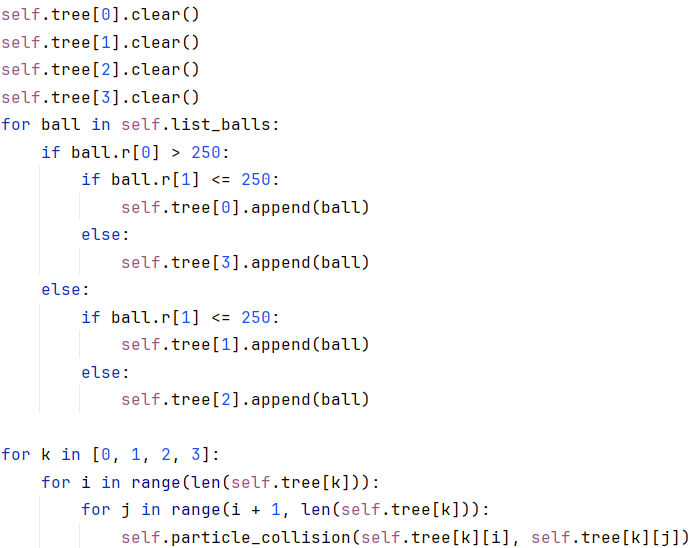


Рисунок 9 - программный алгоритм обработки частиц для проверки на соударения

Кроме этого алгоритма обработки коллизий, есть еще дерево квадрантов (QuadTree), но он не подходит, так как концентрация частиц в основном примерно одинаковая, а суть дерева квадрантов в том, что часть пространства с большей концентрацией частиц делится еще на части (рекурсивный алгоритм). [10], [11]

## Параметры и графики

1. Адаптация физических законов для случая двумерного идеального газа;

**Уравнение состояния идеального газа:** для трехмерного случая –

pV = νRT.

В итоге уравнение будет выглядеть как pA = νRT, где A – площадь.

Так как ν = N / NА, а постоянная Больцмана k равна отношению универсальной газовой постоянной R к числу Авогадро NА, то уравнение можно упростить до pA = kNT.

Концентрация будет равна отношению количества частиц к площади (длины стенки L в квадрате).

В двухмерном пространстве объем V превращается в площадь A. Давление p, в первом случае равное F/S (отношение приложенной силы к площади поверхности), во втором случае будет равно отношению силы к длине поверхности (F/L). (1)

1. Измерение давления, температуры, скорости и энергии.

**Давление.** По формуле (1) p = F/L, где F – средняя сила, с которой частицы действуют на стенку (и наоборот). Так как по второму закону Ньютона Δp = FΔt, где Δp – это изменение импульса, то давление p равно Δp / LΔt.

Рассмотрим ударение о стенку, перпендикулярную оси OX.

Изменение импульса по определению равен произведению массы на изменение скорости. Так как при ударении частицы о стенку модуль проекции скорости на ось OX не меняется, то изменение скорости ΔVx равна 2Vx. Отсюда следует, что давление у стенок, перпендикулярных оси OX находится по формуле p = **Σ**m \* 2Vx / LΔt. Аналогично для давления у стенок, перпендикулярных оси OY (p = **Σ**m \* 2Vy / LΔt).

**Скорость.** Среднеквадратичная скорость равна квадратному корню отношения суммы квадратов модулей скоростей всех частиц к количеству частиц.

**Энергия.** Энергия равна сумме всех кинетических энергий молекул. **(Eк = m \* V2 / 2)**

**Температура.** Из формулы **E = kT; T = E / k.**

1. Программный подсчет параметров;

**Давление.** 118, 122, 126, 130 строчки кода на рисунке 4 отвечают за подсчет импульса. А давление находится как:

self.press\_momentum / (4\*500\*0.2e-9\*50\*self.dt).

(self.dt – единица времени равная 0.2e-11, 0.2e-9 – это один пиксель, равный радиусу атома аргона.)

**Скорость.** (рисунок 10) Подсчет происходит в игровом цикле.



Рисунок 10 - подсчет среднеквадратичной скорости

**Температура.** (рисунок 11)

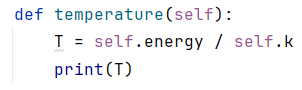
****

Рисунок 11 - подсчет энергии

**Энергия.** (рисунок 12)

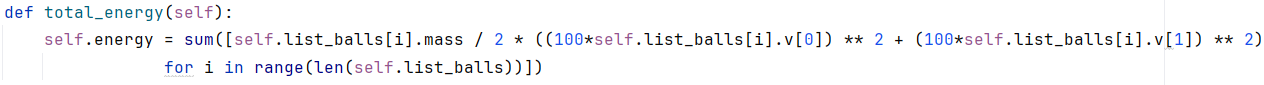


Рисунок 12 - подсчет энергии

1. Разработка программного инструмента для изучения распределения частиц газа по скоростям;

При соударении частиц их суммарная скорость не меняется, однако конечная скорость первой частицы при ударе о вторую не равняется начальной скорости второй, т.е. они не обмениваются скоростями, а они перераспределяются по формуле (рисунок 1).

В итоге получилась такая функция (рисунок 12)

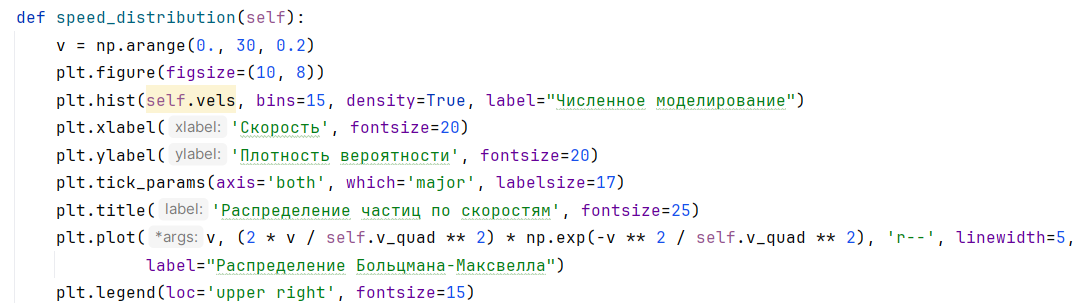


Рисунок 13 - функция графика распределения скоростей

При активации функция строит график, показанный на рисунке 14.

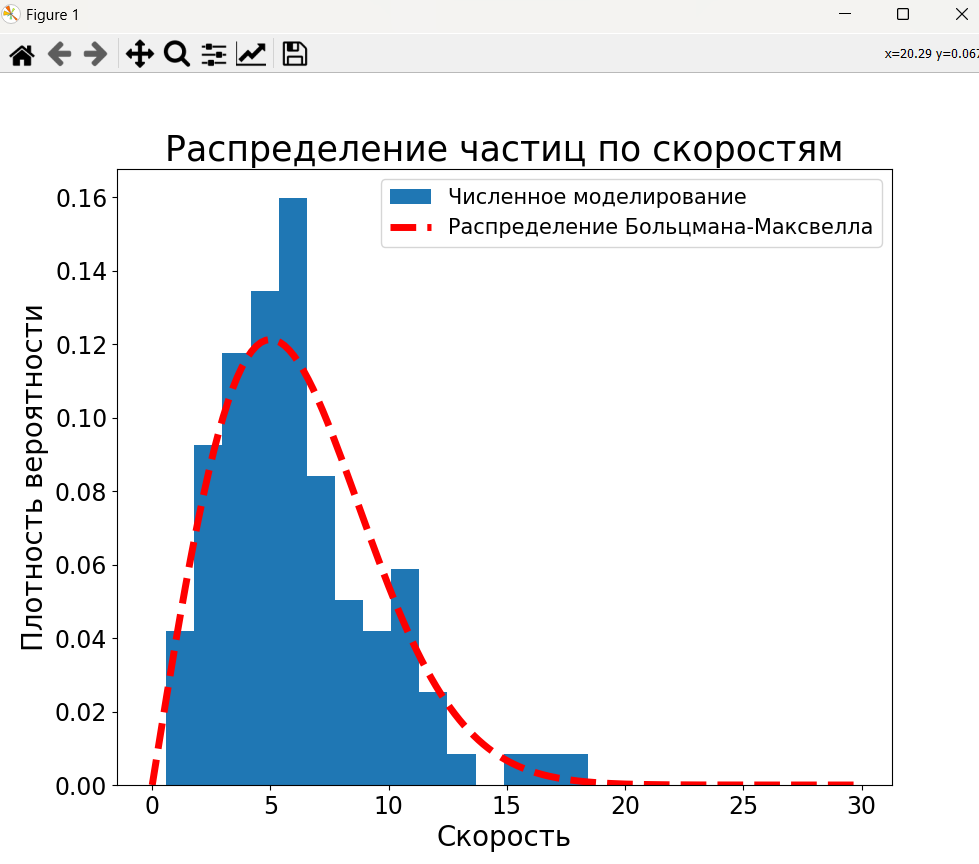


Рисунок 14 - распределение по скоростям

Ось OX (первый аргумент метода .hist()) содержит значения скоростей, а ось OY плотность вероятности — параметр, показывающий. Обновляя график можно получить приближенную к теоретическому распределению скоростей, распределению Максвелла (рисунок 13, пунктир). [4]

Формула, по которой высчитывается теоретическое распределение:

## Разработка рабочего приложения для моделирования и изучения идеального газа.

Графический интерфейс — так называется внешний вид программы — то, что видит пользователь и с чем он может взаимодействовать.

* Обзор интерфейса;

В своем приложении я организовала два интерактивных функционала:

1. Пользователь может вводить количество частиц (от 1 до 1000) и температуру, далее нажатием на кнопку «Начать моделирование» запустить отрисовку шариков и их движение (рисунок 15).

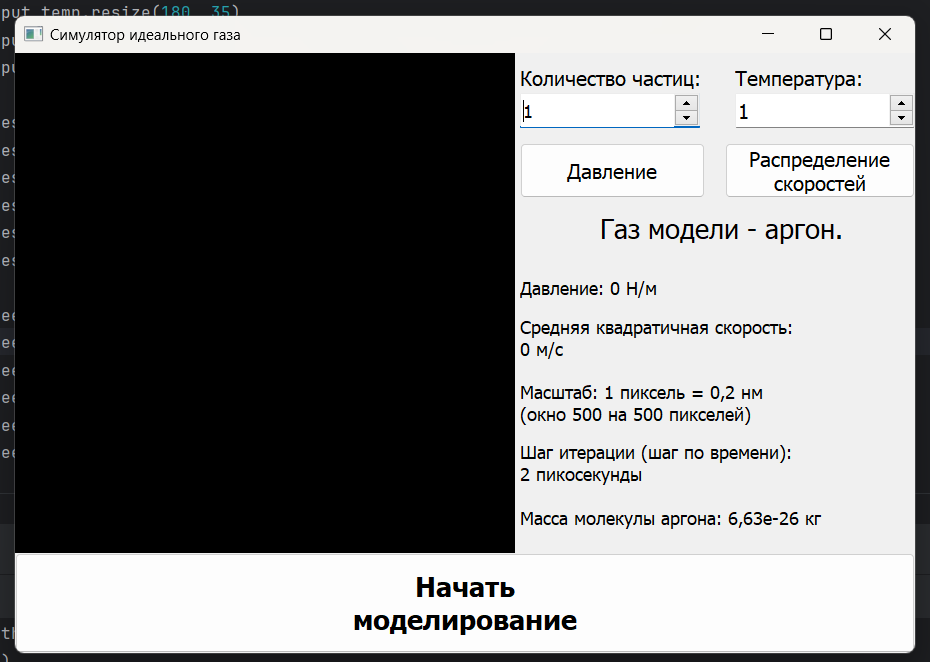


Рисунок 15 - запуск приложения, возможность пользователя ввести количество частиц

1. Нажимая на кнопки «Давление» и «Распределение скоростей», пользователь будет открывать соответствующие графики. График «Давления» показывает зависимость давления от времени. Оба графика обновляются в реальном времени, увидеть новых график пользователь может закрытием окна с графиком и повторным нажатием кнопки.

Кроме того, на «интерактивной поверхности» расположены **параметры модели**: давление (обновляется каждые 50 единиц времени), среднеквадратичная скорость, масштаб, временной шаг между двумя последовательным состояниями системы (шаг итерации), масса молекулы аргона.

Масштаб. Окно 500 на 500 пикселей (pxl). 1 pxl = 0,2 нм (нанометров). Отсюда следует, что реальный размер сосуда – 100 нм.

Шаг итерации. Временной шаг между двумя последовательным состояниями системы принимается за dt = 2 пс (пикосекунды). Тогда скорость v, задаваемая как число проходимых пикселей за временной шаг, связана с реальной скоростью так: 1 pxl / 1 вр.ш. = 0,2e-9 / 2e-12 =100 м/с.

Масса. Масса молекулы аргона m = 6,63e-26 кг.

* Программная реализация пользовательского интерфейса;

В итоговом коде получилось 3 класса: класс Частицы (рисунок 3), класс Игры (Pygame) и класс Игрового Виджета (PyQt).

Класс Игры отвечает за инициализацию «шариков», обработку столкновений частиц со стенками и друг с другом, построение графиков, основной игровой цикл, где отрисовываются все шарики и исполняются соударения частиц и их столкновения со стенками, а также каждые 50 единиц времени считывается давление для графика зависимости давления от времени («Давление»).

Класс Игрового Виджета (PyQt) отвечает за все кнопки, поле ввода количества частиц, отображения температуры и давления, итоговое отображение графиков при нажатии на кнопки, а также за совмещение Pygame и PyQt.

Сложность реализации приложения заключалась в последнем, ведь полноценно это сделать нельзя, также после придется еще больше оптимизировать код.

В итоге все получилось, и окно Pygame как бы отрисовывается на окне PyQt с помощью виджетов второй библиотеки: QImage и QPainter. Отвечающая за это функция класса Игрового Виджета изображена на рисунке 15.

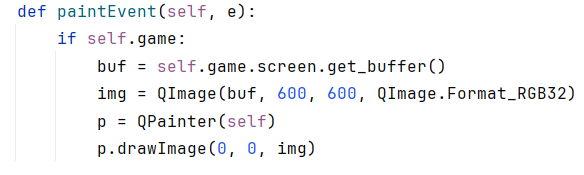


Рисунок 16 - отрисовка окна Pygame на окне PyQt (интеграция)

## Проверка корректной работы симулятора. Выполнение закона сохранения энергии, установление теоретического распределения частиц по скоростям.

Чтобы приложение прошло валидацию, нужно провести на модели несколько тестов: проверить закон сохранения энергии, флуктуацию давления (т. е. проверить, колеблются ли значения давления около одной прямой, параллельной горизонтальной оси времени.), проверить тезис: чем больше частиц, чем ближе график распределения скоростей к теоретическому, и выполнить лабораторную работу «Проверка закона p=nkT».

Начальные данные: единичный временной промежуток – 50, количество частиц – 100, начальная скорость – 5.

* Закон сохранения энергии

Для этого нам нужно построить график зависимости суммарной кинетической энергии молекул от времени

Для этого нужно воспользоваться функцией подсчета энергии (рисунок 13).

Было проведено несколько тестов, и при 100 частиц с начальной скоростью 5 было всегда такое значение (рисунок 17)

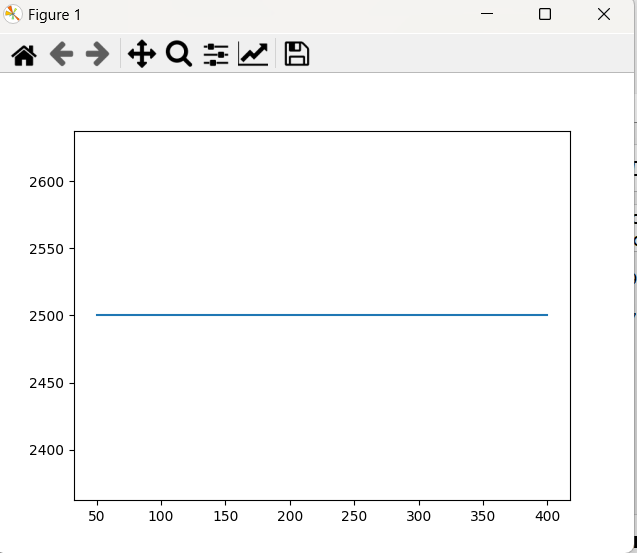


Рисунок 17 - сохранение энергии

Закон сохранения работает.

* Флуктуация давления

Колебания газа в равновесной системе должна выглядеть так, как на рисунке 18.

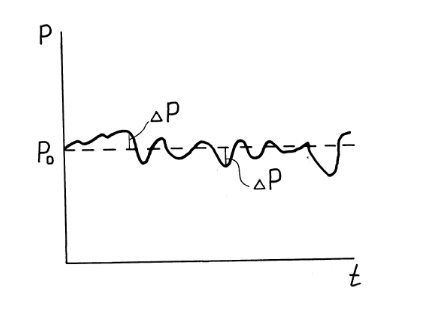


Рисунок 18 - флуктуация давления газа в равновесной системе

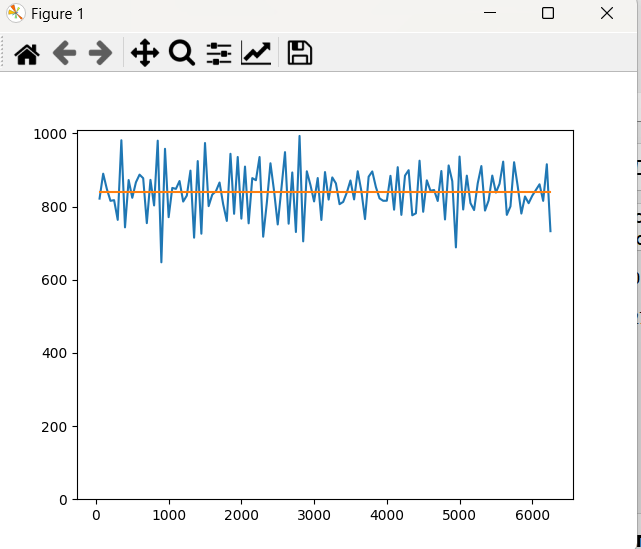


Рисунок 19 - давление после 6000+ единиц времени

Из рисунка 19 можно ясно понять, что колебания происходят около одного значения, а значит тезис верен.

* Зависимость распределения скоростей от количества частиц.

Получено, что независимо от начальных скоростей, со временем устанавливается распределение, похоже на теоретическое. Однако при увеличении количества частиц, схожесть с распределением Максвелла больше (рисунок 20).

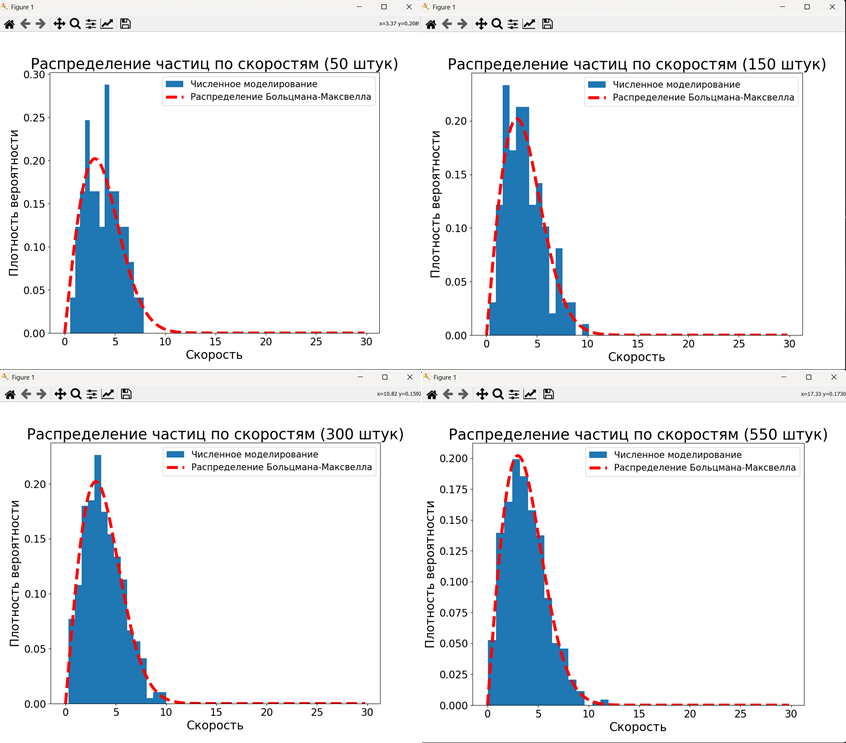


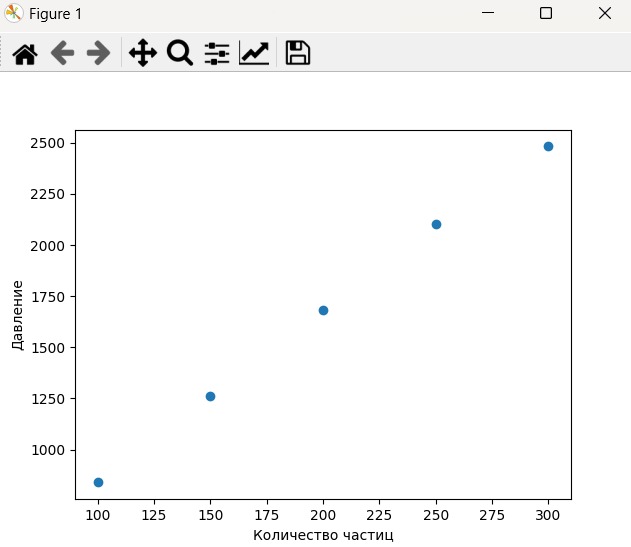
Рисунок 20 - зависимость распределения скоростей от количества частиц

* Лабораторная работа «Проверка закона p=nkT»

Для проверки закона p=nkT, нужно:

* проверить зависимость p от n, а так как концентрация n прямо пропорциональная количеству частиц (n = N / L^2), то нужно доказать, что p прямо пропорционально количеству молекул N;
* проверить зависимость p от T. Так как T по формуле E = kT прямо пропорциональна кинетической энергии E, а та в свою очередь по формуле E = m0V2/2 прямо пропорциональна средней квадратичной скорости, то можно просто проверить зависимость давления p от среднеквадратичной скорости (который постоянна, так как постоянна энергия (по доказанному)).

**Для первого** понадобится сначала выписать несколько значений давления при разном N и по среднему значению давления для каждого значения N сделать график (рисунок 21). Начальная скорость: 5, время: 800.



*Рисунок 21 - зависимость давления от количества частиц*

Проведя линейную линию тренда, видно, что точки ей соответствуют, а значит давление прямо пропорционально количеству частиц, а значит p ∝ n.

**Для второй проверки** нужно сделать подсчеты давления при разных начальных скоростях. Количество частиц постоянно: 200 шариков. (Давление — ось OY) (рисунок 22).

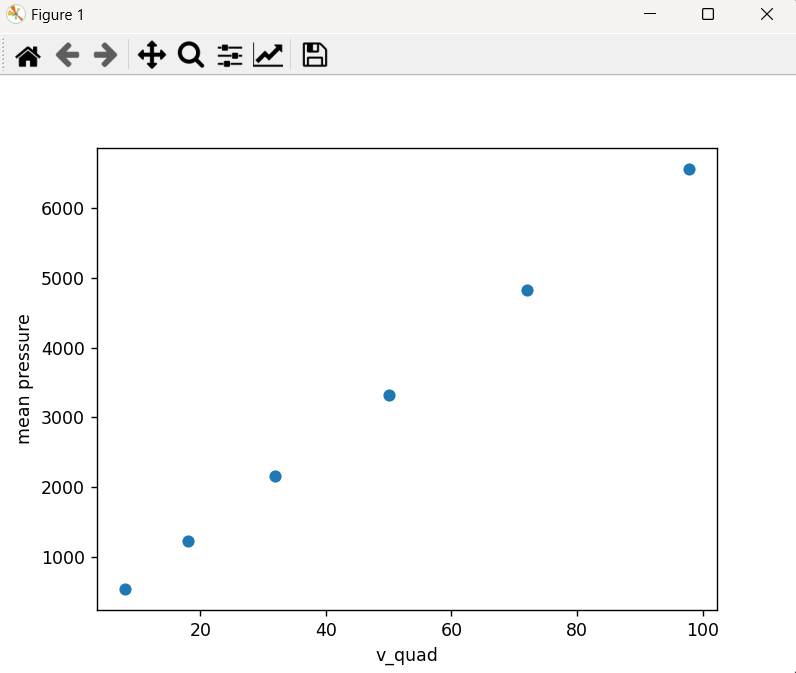


Рисунок 22 - зависимость давления от среднеквадратичной скорости

Здесь также точки соответствуют линейной линии тренда, а значит p ∝ V2 ∝ E ∝ T, p ∝ T.

По итогу, так как p ∝ T и p ∝ n, то **p=nkT** (k — константа).

# Результаты и перспективы

В ходе работы над проектом была создана модель идеального газа инертного газа аргона. Свойства модели соответствуют свойствам идеального газа.

Создано приложение (рисунок 23) с графическим дизайном для визуализации модели идеального газа (аргона). Приложение может строить графики, показывающие флуктуации давления и распределение скоростей (а также теоретическое распределение на нем). К тому же на части с кнопками отображается давление в Н/м, среднеквадратичная скорость, шаг итерации, масштаб и масса молекулы аргона.



Рисунок 23 - итоговое приложение

Также проведены тесты модели: проверена корректная работа закона сохранения энергии модели, наличие флуктуации давления около одной прямой, параллельной оси времени, проверена зависимость вида графика распределения скоростей от количества частиц, проведена лабораторная работа «Проверка закона p=nkT» и закон доказан в созданной системе.

Также планируется введение дополнительного функционала, например броуновского движения; оптимизация программы, чтобы моделировать 1500+ частиц.

Приложение планируется интегрировать в библиотеку МЭШ с сохранением интерактивных функций в качестве виртуальной лаборатории, чтобы старшеклассники на уроках физики могли взаимодействовать с моделью через школьные интерактивные доски. Сейчас же учитель может просто транслировать на доску приложение и через самостоятельное взаимодействие с моделью помогать ученикам лучше понять тему «Идеальный газ» и свойства газа в МКТ.

# Список используемой литературы

1. <https://hf.nsu.ru/image/lectures_2019_1-5.pdf> — lectures\_2019\_1-5.pdf (дата обращения 04.02.2024)
2. <https://www.sciencedirect.com/topics/engineering/numerical-modeling#:~:text=Численное%20моделирование%20-%20это%20математическое,использовался%20для%20анализа%20толщины%20древесины> — Numerical Modeling - an overview | ScienceDirect Topics (дата обращения 17.02.2024)
3. <https://pythonru.com/biblioteki/tipy-grafikov-v-matplotlib-plt3> — Как строить линейные графики, гистограммы и диаграммы в Matplotlib (дата обращения 17.02.2024)
4. <https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.hist.html> — matplotlib.pyplot.hist — Matplotlib 3.8.3 documentation (дата обращения 29.01.2024)
5. <http://fn.bmstu.ru/files/FN4/lec_2sem/2sem_lec_15.pdf> —2й\_семестр\_Лекция\_15 (дата обращения 15.02.2024)
6. <https://doc.qt.io/> — Qt Documentation | Home (дата обращения 17.02.2024)
7. <https://en.wikipedia.org/wiki/Elastic_collision> - Elastic collision - Wikipedia (дата обращения 14.01.2024)
8. <https://pythonru.com/biblioteki/tipy-grafikov-v-matplotlib-plt3> - Как строить линейные графики, гистограммы и диаграммы в Matplotlib (дата обращения 29.01.2024)
9. <https://studfile.net/preview/4415420/page:4/> - 1.3. Распределение Максвелла (18.02.2024)
10. <https://code.tutsplus.com/quick-tip-use-quadtrees-to-detect-likely-collisions-in-2d-space--gamedev-374t> — Quick Tip: Use Quadtrees to Detect Likely Collisions in 2D Space | Envato Tuts+ (дата обращения 01.02.2024)
11. <https://ru.wikipedia.org/wiki/Дерево_квадрантов> — Дерево квадрантов — Википедия (дата обращения 01.02.20204)