Решеточные газы, решеточное уравнение Больцмана

Отчёт по третьему этапу группового проекта

Содержание

# 1 Введение

## 1.1 Цель проекта

Разработать и проанализировать модель на основе решеточного уравнения Больцмана для описания течений газа.

## 1.2 Задачи третьего этапа проекта

1. Реализовать и описать программный алгоритм решения задачи.

## 1.3 Актуальность

Моделирование газовых потоков и жидкостей традиционными методами требует значительных вычислительных ресурсов. В связи с этим, методы решеточных газов (LGA) и решеточного уравнения Больцмана (LBE) становятся все более актуальными. Они позволяют упростить вычисления, сохраняя при этом физическую достоверность, и находят применение в различных областях, от гидродинамики до биофизики. В данном докладе мы рассмотрим основные алгоритмы и модели, используемые для решения задач с применением LGA и LBE.

# 2 Основная часть

## 2.1 Модель HPP (Hardy–Pomeau–Pazzis)

Модель HPP (Hardy-Pomeau-Pazzis) — это базовая модель решеточных газов (LGA), используемая для моделирования гидродинамических явлений на микроскопическом уровне. Она представляет собой дискретную систему, где пространство и время дискретизованы, а частицы двигаются по узлам квадратной решетки.

### 2.1.1 Основные характеристики модели HPP:

1. **Решетка**: используется двумерная квадратная решетка, где узлы расположены на одинаковом расстоянии друг от друга.
2. **Частицы**: в каждом узле решетки могут находиться частицы единичной массы. Каждая частица может двигаться в одном из четырех направлений: вверх, вниз, вправо или влево.
3. **Скорость**: все частицы имеют одинаковую скорость, направленную к соседнему узлу. Расстояние между узлами () и шаг времени () выбираются так, чтобы частица могла переместиться в соседний узел за один временной шаг.
4. **Принцип исключения**: в каждом узле может находиться не более одной частицы, движущейся в заданном направлении.
5. **Этапы эволюции**:
   * **Распространение (Streaming)**: частицы перемещаются в соседние узлы в соответствии со своими скоростями. За один шаг времени частица переходит в соседний узел в направлении своего движения.
   * **Столкновения (Collision)**: в узлах происходят столкновения частиц, при которых сохраняются количество частиц и полный импульс.
6. **Правила столкновений**: столкновения происходят таким образом, чтобы выполнялись законы сохранения. В модели HPP нетривиальные столкновения происходят, когда две частицы движутся навстречу друг другу (почти “лоб в лоб”). После столкновения частицы меняют направления движения на 90 градусов. Во всех остальных случаях столкновения считаются несущественными, и частицы продолжают двигаться в прежних направлениях.
7. **Кодирование состояний**: состояние каждого узла решетки кодируется битами. Поскольку имеется четыре возможных направления движения, для кодирования состояния узла требуется четыре бита. Каждый бит соответствует одному из направлений: 0 — нет частицы, 1 — есть частица, движущаяся в этом направлении. Например, если частицы движутся вправо и вверх, состояние узла кодируется как 0101 в двоичном формате [2]

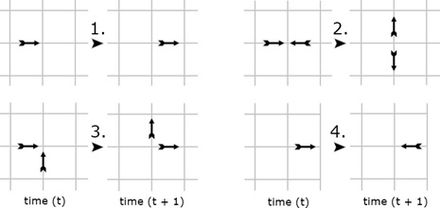


Figure 1: Примеры перемещений частиц в модели HPP

### 2.1.2 Математическое описание:

Обозначим возможные направления скорости как $. Тогда:

Основные операции для работы с состояниями узлов:

1. **Добавление частицы**: добавление к состоянию частицы с направлением скорости :
2. **Проверка наличия частицы**: проверка, есть ли в состоянии частица с направлением скорости :

* Если результат не равен 0, то частица с направлением присутствует в узле.

### 2.1.3 Недостатки модели HPP:

1. **Отсутствие симметрии**: квадратная решетка с четырьмя направлениями скорости недостаточно симметрична, что приводит к анизотропии в макроскопических свойствах.
2. **Нефизичное поведение**: модель HPP неточно описывает гидродинамические свойства жидкостей и газов.

Для устранения этих недостатков были разработаны более совершенные модели, такие как FHP (Frisch-Hasslacher-Pomeau) на треугольных решетках и модели с добавлением покоящихся частиц. [4]

## 2.2 Описание задания

Реализуйте модель HPP. Задайте периодические граничные условия. Это просто сделать, добавив по одному ряду узлов с каждой стороны области (фиктивные узлы). Перед шагом распространения необходимо скопировать значения левого ряда физических узлов в правый фиктивный ряд. Тогда частицы, вылетая из левой границы области налево, появятся на ее правой границе. С другими границами поступают также. Вначале возьмите одну единственную частицу и проверьте правильность всех граничных условий. Затем убедитесь, что для двух частиц их столкновения “почти лоб в лоб” и под прямым углом происходят верно. Для любого числа частиц должны сохраняться их полное число и полный импульс.

## 2.3 Описание программного кода

Код для модели HPP мы реализовали на языке Julia.

### 2.3.1 Подключение библиотек и настройка визуализации

using Plots  
gr()

* Подключается пакет Plots для построения графиков.
* Выбирается бэкенд gr() для отрисовки.

### 2.3.2 Константы и параметры модели

const Nx, Ny = 10, 10  
const dx = [1, 0, -1, 0]  
const dy = [0, 1, 0, -1]  
const dir\_colors = [:red, :blue, :green, :purple]

* Nx, Ny - размеры внутренней области решетки (без фиктивных узлов).
* dx, dy - массивы смещений по x и y для четырёх направлений движения частиц: вправо, вверх, влево, вниз.
* dir\_colors - цвета для визуализации направлений частиц.

### 2.3.3 Создание пустой сетки

function create\_grid()  
 zeros(Bool, Nx+2, Ny+2, 4)  
end

* Создаётся булев массив размером (Nx+2, Ny+2, 4).
* Nx+2 и Ny+2 - учитывают фиктивные узлы по краям (по одному ряду с каждой стороны).
* Последнее измерение 4 - количество направлений движения частиц.
* Все значения изначально false (нет частиц).

### 2.3.4 Добавление частицы в сетку

function add\_particle!(grid, x, y, d)  
 @assert 1 ≤ x ≤ Nx && 1 ≤ y ≤ Ny "Particle must be inside physical domain"  
 @assert 1 ≤ d ≤ 4 "Direction must be between 1 and 4"  
 grid[x+1, y+1, d] = true  
end

* Добавляет частицу в позицию (x, y) с направлением d.
* x+1, y+1 - сдвиг на 1 из-за фиктивных узлов.
* Проверяется корректность координат и направления.

### 2.3.5 Применение периодических граничных условий

function apply\_periodic\_boundaries!(grid)  
 for d in 1:4  
 # левая фиктивная = правая физическая  
 grid[1, 2:Ny+1, d] .= grid[Nx+1, 2:Ny+1, d]   
 # правая фиктивная = левая физическая  
 grid[Nx+2, 2:Ny+1, d] .= grid[2, 2:Ny+1, d]   
 # нижняя фиктивная = верхняя физическая  
 grid[2:Nx+1, 1, d] .= grid[2:Nx+1, Ny+1, d]   
 # верхняя фиктивная = нижняя физическая  
 grid[2:Nx+1, Ny+2, d] .= grid[2:Nx+1, 2, d]   
 end  
end

* Для каждого направления d копирует значения с противоположных физических границ в фиктивные узлы.
* Обеспечивает периодичность: частицы, выходящие с одной границы, появляются с противоположной.

### 2.3.6 Обработка столкновений частиц

function collide!(grid)  
 for x in 2:Nx+1, y in 2:Ny+1  
 right, up, left, down = grid[x,y,1],   
 grid[x,y,2],   
 grid[x,y,3],   
 grid[x,y,4]  
 if right && left && !up && !down  
 grid[x,y,1] = false  
 grid[x,y,3] = false  
 grid[x,y,2] = true  
 grid[x,y,4] = true  
 elseif up && down && !right && !left  
 grid[x,y,2] = false  
 grid[x,y,4] = false  
 grid[x,y,1] = true  
 grid[x,y,3] = true  
 end  
 end  
end

* Проходит по всем физическим узлам.
* Проверяет столкновения:
  + Лобовое: если есть частицы вправо и влево, меняет их направления на вверх и вниз.
  + Под прямым углом: если есть частицы вверх и вниз, меняет направления на вправо и влево.
* Другие случаи столкновений не обрабатываются (частицы проходят без изменений).

### 2.3.7 Распространение частиц

function propagate!(grid)  
 new\_grid = zeros(Bool, size(grid))  
 for x in 2:Nx+1, y in 2:Ny+1, d in 1:4  
 if grid[x, y, d]  
 nx, ny = x + dx[d], y + dy[d]  
 new\_grid[nx, ny, d] = true  
 end  
 end  
 return new\_grid  
end

* Создаёт новую пустую сетку.
* Для каждой частицы вычисляет новую позицию, смещая по направлению d.
* Записывает частицу в новую позицию.
* Возвращает обновлённую сетку.

### 2.3.8 Подсчёт числа частиц

function count\_particles(grid)  
 sum(grid[2:Nx+1, 2:Ny+1, :])  
end

* Считает общее количество частиц во всех направлениях во всех физических узлах.

### 2.3.9 Вычисление суммарного импульса

function calculate\_momentum(grid)  
 px, py = 0, 0  
 for x in 2:Nx+1, y in 2:Ny+1  
 # движение вправо минус влево  
 px += grid[x,y,1] - grid[x,y,3]  
 # движение вверх минус вниз  
 py += grid[x,y,2] - grid[x,y,4]  
 end  
 return (px, py)  
end

* Суммирует по всем узлам разницу частиц, движущихся в противоположных направлениях, по осям X и Y.
* Возвращает вектор импульса.

### 2.3.10 Визуализация состояния решетки

function plot\_grid(grid, step)  
 p = heatmap(0:Nx+1, 0:Ny+1, zeros(Nx+2, Ny+2),  
 c=:white, aspect\_ratio=1, legend=false,  
 xlims=(0.5, Nx+1.5), ylims=(0.5, Ny+1.5),  
 title="HPP Model (Step $step)")  
  
 plot!(p, [0.5, Nx+1.5, Nx+1.5, 0.5, 0.5],   
 [0.5, 0.5, Ny+1.5, Ny+1.5, 0.5],  
 color=:black, linewidth=2, label="")  
  
 for x in 2:Nx+1, y in 2:Ny+1, d in 1:4  
 if grid[x, y, d]  
 quiver!(p, [x], [y], quiver=([dx[d]\*0.4],   
 [dy[d]\*0.4]),  
 color=dir\_colors[d], lw=2, arrow=true)  
 end  
 end  
  
 return p  
end

* Создаёт белый холст с размерами решетки.
* Рисует чёрную рамку вокруг области.
* Для каждой частицы рисует стрелку в направлении движения с цветом, соответствующим направлению.
* Возвращает объект графика.

### 2.3.11 Запуск всех тестов

function run\_all\_tests()  
 println("Running single particle test...")  
 test\_single\_particle()  
  
 println("Running head-on collision test...")  
 test\_head\_on\_collision()  
  
 println("Running right angle collision test...")  
 test\_right\_angle\_collision()  
  
 println("All tests completed! Check generated GIFs.")  
end  
  
run\_all\_tests()

* **test\_single\_particle()** - одна частица в центре движется вправо, проверяется распространение и периодичность (рис. [[2](#fig:002)]-[[4](#fig:004)]).

# Тест 1: Одна частица  
function test\_single\_particle()  
 grid = create\_grid()  
 # Частица в центре, движется вправо  
 add\_particle!(grid, Nx÷2, Ny÷2, 1)   
   
 anim = @animate for step in 1:20  
 apply\_periodic\_boundaries!(grid)  
 collide!(grid)  
 p = plot\_grid(grid, step)  
 grid = propagate!(grid)  
   
 n = count\_particles(grid)  
 px, py = calculate\_momentum(grid)  
 annotate!(p, 0.5, Ny+1.2, text("Particles: $n", :left))  
 annotate!(p, 0.5, Ny+0.8, text("Momentum: ($px, $py)", :left))  
   
 p  
 end  
   
 gif(anim, "hpp\_single\_particle.gif", fps=2)  
end

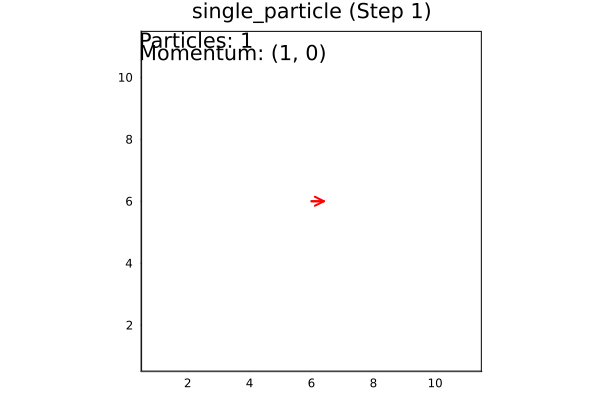


Figure 2: Запуск теста №1. Одна частица в центре, движется вправо. Шаг №1

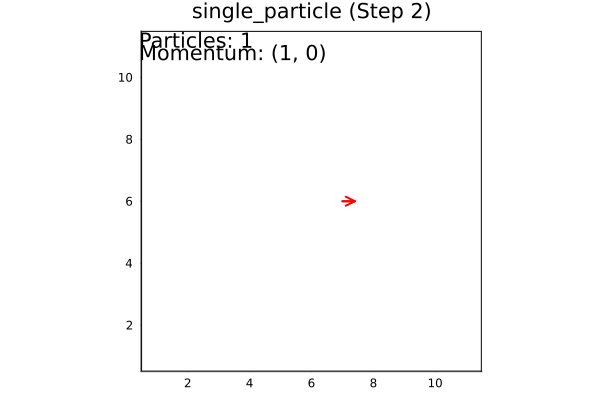


Figure 3: Запуск теста №1. Одна частица в центре, движется вправо. Шаг №2

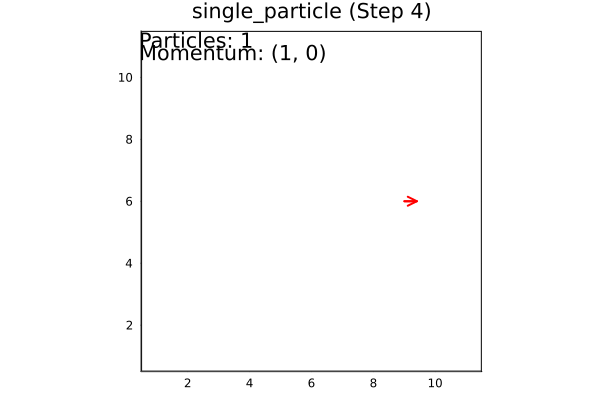


Figure 4: Запуск теста №1. Одна частица в центре, движется вправо. Шаг №4

* **test\_head\_on\_collision()** - две частицы движутся навстречу, проверяется лобовое столкновение (рис. [[5](#fig:005)]-[[7](#fig:007)]).

# Тест 2: Две частицы (лобовое столкновение)  
function test\_head\_on\_collision()  
 grid = create\_grid()  
 add\_particle!(grid, 4, 5, 1) # →  
 add\_particle!(grid, 6, 5, 3) # ←  
   
 anim = @animate for step in 1:10  
 apply\_periodic\_boundaries!(grid)  
 collide!(grid)  
 p = plot\_grid(grid, step)  
 grid = propagate!(grid)  
   
 n = count\_particles(grid)  
 px, py = calculate\_momentum(grid)  
 annotate!(p, 0.5, Ny+1.2, text("Particles: $n", :left))  
 annotate!(p, 0.5, Ny+0.8, text("Momentum: ($px, $py)", :left))  
   
 p  
 end  
   
 gif(anim, "hpp\_head\_on.gif", fps=1)  
end

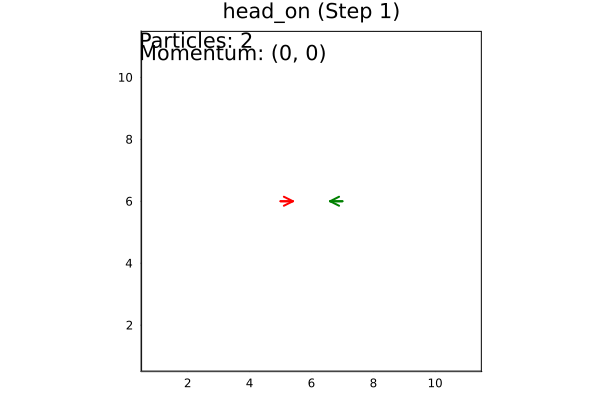


Figure 5: Запуск теста №2. Две частицы движутся навстречу, лобовое столкновение. Шаг №1

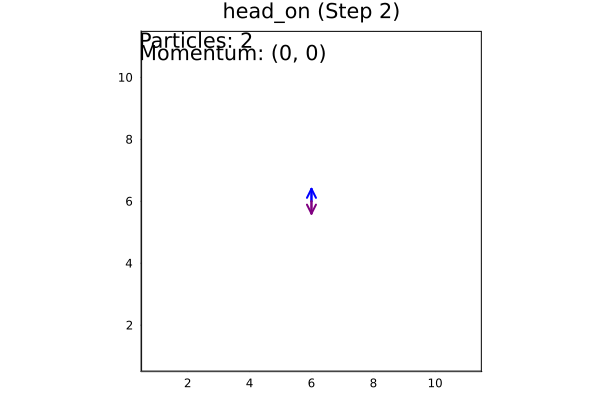


Figure 6: Запуск теста №2. Две частицы движутся навстречу, лобовое столкновение. Шаг №2

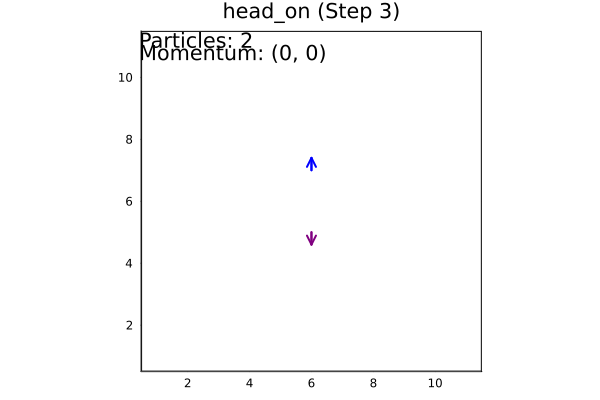


Figure 7: Запуск теста №2. Две частицы движутся навстречу, лобовое столкновение. Шаг №3

* **test\_right\_angle\_collision()** - четыре частицы движутся навстречу под прямым углом, проверяется корректность столкновений (рис. [[8](#fig:008)]-[[10](#fig:010)]).

# Тест 3: Четыре частицы (столкновение под прямым углом)  
function test\_right\_angle\_collision()  
 grid = create\_grid()  
 add\_particle!(grid, 5, 4, 2) # ↑  
 add\_particle!(grid, 5, 6, 4) # ↓  
 add\_particle!(grid, 4, 5, 1) # →  
 add\_particle!(grid, 6, 5, 3) # ←  
   
 anim = @animate for step in 1:10  
 apply\_periodic\_boundaries!(grid)  
 collide!(grid)  
 p = plot\_grid(grid, step)  
 grid = propagate!(grid)  
   
 n = count\_particles(grid)  
 px, py = calculate\_momentum(grid)  
 annotate!(p, 0.5, Ny+1.2, text("Particles: $n", :left))  
 annotate!(p, 0.5, Ny+0.8, text("Momentum: ($px, $py)", :left))  
   
 p  
 end  
   
 gif(anim, "hpp\_right\_angle.gif", fps=1)  
end

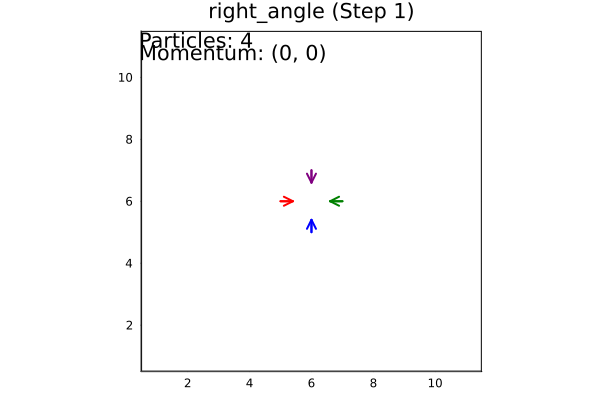


Figure 8: Запуск теста №3. Четыре частицы движутся навстречу под прямым углом. Шаг №1

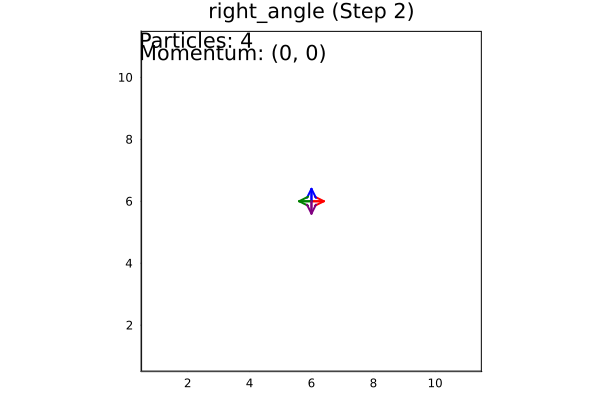


Figure 9: Запуск теста №3. Четыре частицы движутся навстречу под прямым углом. Шаг №2

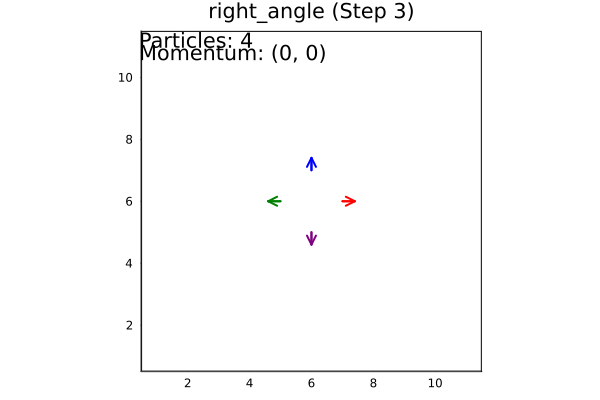


Figure 10: Запуск теста №3. Четыре частицы движутся навстречу под прямым углом. Шаг №3

В каждом тесте:

* Создаётся сетка.
* Добавляются частицы.
* В цикле на каждом шаге применяются граничные условия, столкновения, визуализация и распространение.
* Считается число частиц и импульс.
* Создаётся анимация и сохраняется в GIF.

# 3 Заключительная часть

## 3.1 Заключение

Модели решеточных газов и решеточное уравнение Больцмана представляют собой эффективные инструменты для моделирования газовых потоков. В данной части проекта мы рассмотрели простую базовую модель .

Реализовали двумерную модель решеточного газа HPP с четырьмя направлениями движения, периодическими граничными условиями, обработкой столкновений и визуализацией. Тесты демонстрируют корректность работы модели и сохранение физических величин (число частиц и импульс).

## 3.2 Выводы

Во время выполнения третьего этапа группового проекта мы описали и реализовали модель HPP - базовую модель решеточных газов (LGA), которая может быть использована для моделирования решеточного уравнения Больцмана.

# 4 Список литературы

1. Медведев Д.А.и.др. Моделирование физических процессов и явлений на ПК: Учеб. пособие. Новосибирск: Новосибирский государственный университет, 2010. С. 101.

2. Чащин Г.С. [Метод решёточных уравнений Больцмана: моделирование изотермических низкоскоростных течений](https://doi.org/10.20948/prepr-2021-99): 99. Институт прикладной математики им. М.В. Келдыша РАН, 2021.

3. Hardy J., Pomeau Y., Pazzis O. de. [Time evolution of a two-dimensional classical lattice system](https://doi.org/10.1103/PhysRevLett.31.276) // Physical Review Letters. 1973. Т. 31, № 5. С. 276–279.

4. Succi S. The Lattice Boltzmann Equation for Fluid Dynamics and Beyond. Oxford University Press, 2001.