

# Неравновесная агрегация, фракталы

## Этап 3

---

Шалыгин Г. Э. Низамова А. А. Голощапова И. Б. Серегин Д. А. Пиняева А. А.

11 марта 2023

Российский университет дружбы народов, Москва, Россия

# Информация

---

- Голощапова Ирина Борисовна
- Низамова Альфия Айдаровна
- НФИ-02-20
- Российский университет дружбы народов

## **Вводная часть**

---

- У неравновесной агрегации есть модели.
- Для изучения строятся алгоритмы.
- Модели сложные, ресурсы ограничены, значит необходимы эффективные алгоритмы.
- Помимо теории, необходима эффективная реализация

- Математические модели неравновесной агрегации.
- Построение алгоритмов решения задачи моделирования процесса неравновесной агрегации.
- Програмная реализация алгоритмов

Цель:

- Реализовать алгоритмы моделирования неравновесной агрегации.

Задачи:

- Рассмотреть возможности языков для программной реализации алгоритмов
- Реализовать алгоритмы, описанные втором этапе

## Используемые методы

---



- Стандартные функции
  - `sin(x)`
  - `cos(x)`
  - `round(x)`, `floor(Int, x)`
  - `abs`
  - `sqrt`
  - `maximum`, `minimum`
- Макрос
  - `@time`
- Библиотеки
  - `Plots`

# Алгоритмы для DLA

---

# Генерация псевдослучайных чисел

```
next = 0
function rand()
    global next = (next * 1664525 + 1013904223) % 2^32
    return next / 2 ^ 32
end
```

## Генерация координат следующей частицы

```
function GetNextParticular(x_center, y_center, r)
    r = r
    angle = 2 * pi * rand()
    x = r * cos(angle) + x_center
    y = r * sin(angle) + y_center
    return round(x), round(y)
end
```

Расстояние между точками  $(x_1, y_1), (x_2, y_2)$

```
function dist(x1, y1, x2, y2)
    return sqrt((x2-x1)*(x2-x1) + (y2-y1)*(y2-y1))
end
```

Проверка того, что частица столкнулась с кластером

```
function check(x, y)
    for i in 1:n
        if abs(X[i] - x) + abs(Y[i] - y) == 1
            return true
        end
    end
    return false
end
```

## Блуждание частицы

```
function RandomWalk(x, y, i, r, xl, xr, yu, yd)
    step = 1; dx = [1, -1, 0, 0]; dy = [0, 0, 1, -1]
    while step < 500 && dist(x, y, (xl + xr) / 2, (yu + yd) / 2) < 4 * r
        if check(x, y)
            X[i] = x; Y[i] = y
            return true
        end
        j = floor(Int, 100 * rand()) % 4 + 1
        x = x + dx[j]
        y = y + dy[j]
        step += 1
    end
    return false
end
```

## Псевдокод модели DLA

```
function DLA(t)
  i = 1
  while i < t
    xl = minimum(X)
    xr = maximum(X)
    yu = minimum(Y)
    yd = maximum(Y)
    r = dist(xl, yd, xr, yu) / 2 + 3
    x, y = GetNextParticular((xr+xl)/2, (yu+yd)/2, r)
    ok = RandomWalk(x, y, i, r, xl, xr, yu, yd)
    if ok
      i += 1
    end
  end
end
```

```
DLA(n)
```

```
plot(X, Y, seriestype=:scatter, title="dla", label="cluster")
```



## Другие модели

---

- **Бессеточная:** добавляется выбор случайной длины шага.
- **Химически-ограниченная:** вводится условие прилипания.

```
function RandomWalk(x, y, i, r, xl, xr, yu, yd)
    <...>
    x = x + 2 * rand() - 1
    y = y + 2 * rand() - 1
    <...>
end

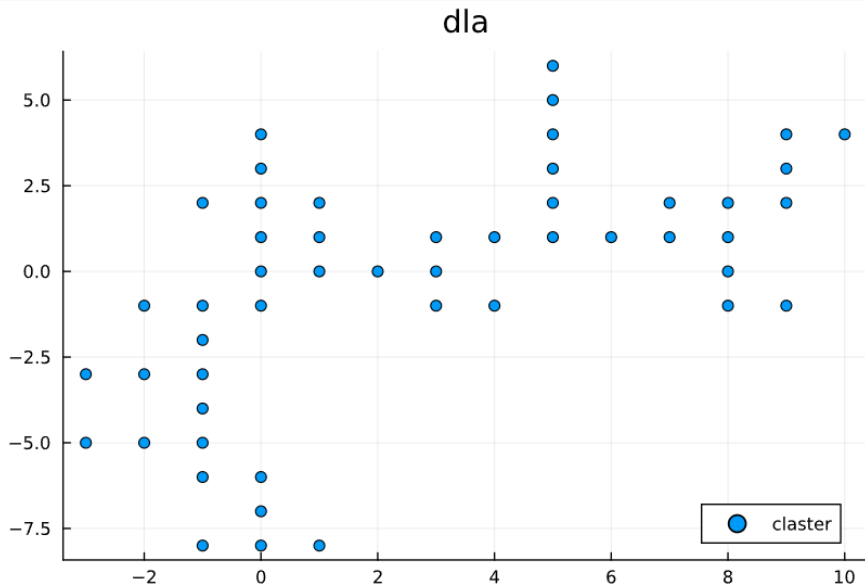
function check(x, y)
    <...>
    if abs(X[i] - x) + abs(Y[i] - y) <= 1
        return true
    <...>
end
```

```
function RandomWalk(x, y, i, r, xl, xr, yu, yd)
    <...>
    if check(x, y) and random() > 0.2
        X[i] = x
        Y[i] = y
    <...>
end

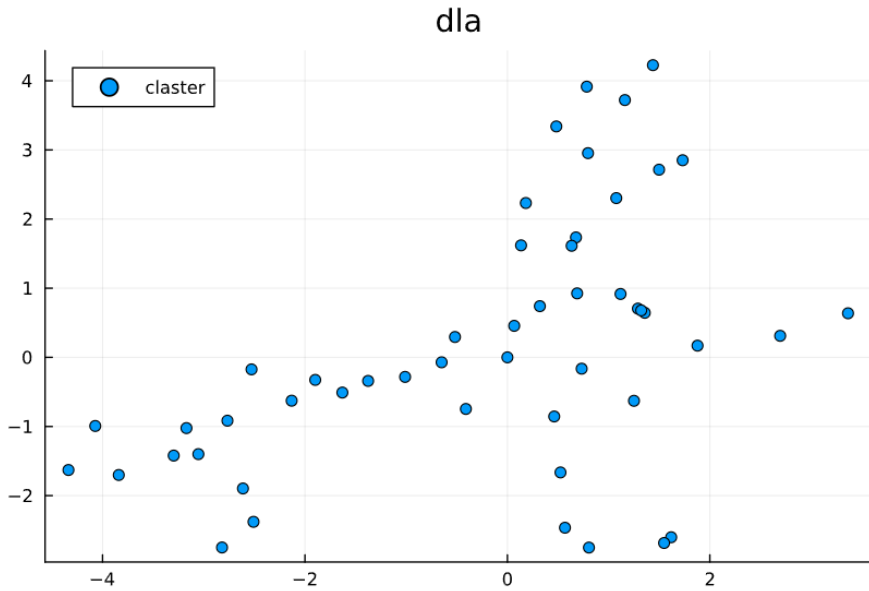
function RandomWalk()
    <...>
    x = x + dx[j]; y = y + dy[j]
    if fiels[x][y] == 1:
        x -= dx[i]; y -= dx[i]
    <...>
```

## Промежуточные результаты

---



# Бессеточная модель



## Итоги

---



- Реализованные алгоритмы позволяют провести вычислительные эксперименты.
- Возможно изучение разных моделей при минимальных изменениях кода.