Монте-Карло моделирование уравнений агрегаций
Малоранговый метод
Метод принятия-отклонения

Дьяченко Р.Р.

Университет Сириус, ВМК МГУ

Август - Сентябрь, 2022

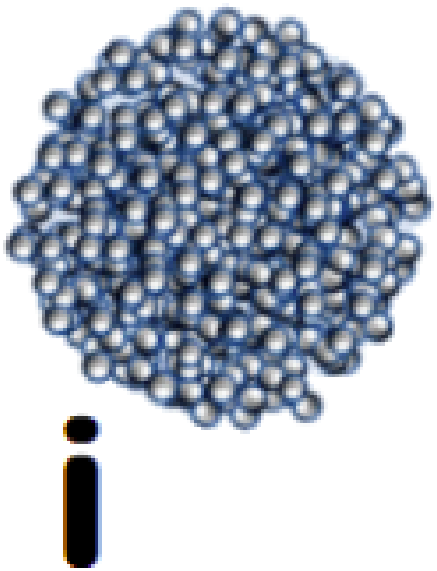
Университет Сириус

Contents

Что исследуем?

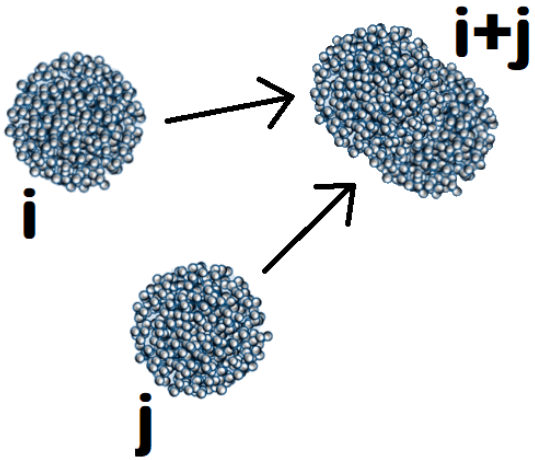
Объект изучения

Частицы, обладающие **размером**

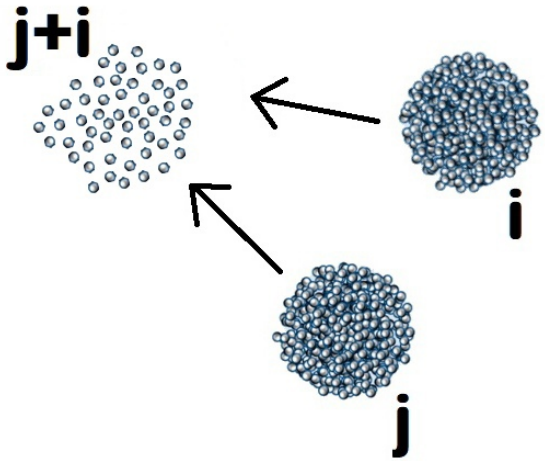


Типы взаимодействия частиц

Агрегация (aggregation)



Фрагментация (shattering)



Где встречается такие объекты??

Где встречается такие объекты?

Процессы агрегации и фрагментации широко представлены в природе и нередко являются частью технологических процессов. В качестве важных примеров таких процессов можно указать следующие:

1. Процессы обратимой полимеризации в растворах
2. Рост биологически важных биополимеров (например, прионов)
3. Образование протопланет в пылевых облаках в межзвездном пространстве
4. Распределение по размерам частиц в планетных кольцах, таких как кольца Сатурна [1]
5. Эволюция компьютерных сетей [2]
6. и др

В случае однородных систем, рассматриваемых в моёй работе, указанные выше процессы описываются системой уравнений Смолуховского.

[1] Cuzzi, J. N., Burns, J. A., Charnoz, S., Clark, R. N., Colwell, J. E., Dones, L., ... Weiss, J. (2010). An Evolving View of Saturn's Dynamic Rings. Science, 327(5972), 1470–1475.

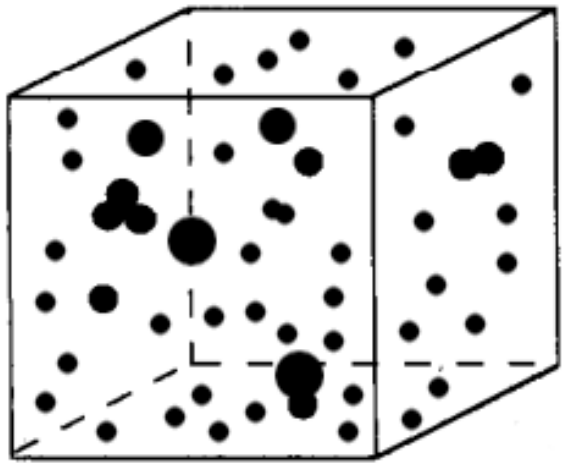
[2] W. Miura, H. Takayasu, and M. Takayasu. Effect of coagulation of nodes in an evolving complex network. Phys. Rev. Lett, 108:168701, 2012

Математическая формализация

Параметры рассматриваемой модели

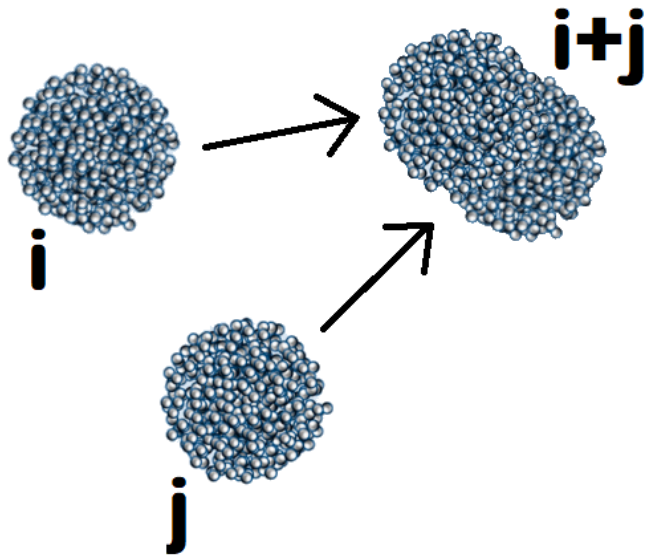
Основыне величины системы:

1. V - объём пространства взаимодейтвия
2. n_i - концентрация частиц размера i
3. N_i - количесвто частиц размера i в объёме V
4. t - момент времени, в который рассматривается система



Агрегация

$$\frac{dn_s(t)}{dt} = \frac{1}{2}\sum_{i+j=s} K_{ij} n_i(t) n_j(t) - \sum_{j=1}^{\infty} K_{sj} n_s(t) n_j(t).$$



M. Smoluchowski, Versuch Einer Mathematischen Theorie der Koagulationskinetik kolloider Losungen, Zeitschrift f. Physik. Chemie. XCII, 92 (1917) 129-168.

$$\frac{dn_s(t)}{dt} = \frac{1}{2}\sum_{i+j=s} K_{ij} n_i(t) n_j(t) - \sum_{j=1}^{\infty} K_{sj} n_s(t) n_j(t).$$

K_{ij}(**ядро** системы) – среднее число **столкновений** в единице **пространства** за единицу **времени** в системе с единичной **концентрацией частиц** размера– i, j

$$[K_{ij}] = \frac{\text{столкновения}}{\text{пространство} \cdot \text{время} \cdot \frac{\text{частица}}{\text{пространство}} \cdot \frac{\text{частица}}{\text{пространство}}}$$

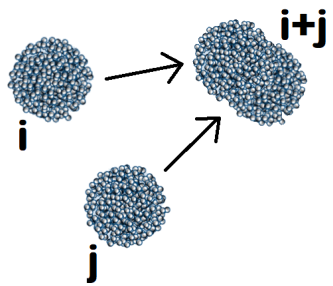
Отметим, что из физических соображений следует симметричность и неотрицательность коэффициентов K_{ij} , т.е. $K_{ij} = K_{ji} > 0$

$n_s(t)$ – концентрация частиц размера $-s$ в момент t

$$[n_s] = \frac{\text{частиц}}{\text{пространство}}$$

$$\frac{dn_s(t)}{dt} = \frac{1}{2}\sum_{i+j=s} K_{ij} n_i(t) n_j(t) - \sum_{j=1}^{\infty} K_{sj} n_s(t) n_j(t).$$

1. Первый член описывает скорость, с которой агрегаты размера s **формируются** из частиц с размерами i и j
2. Второй член описывают скорость **исчезновения** частиц размера s из-за слияний с частицами любого размера j

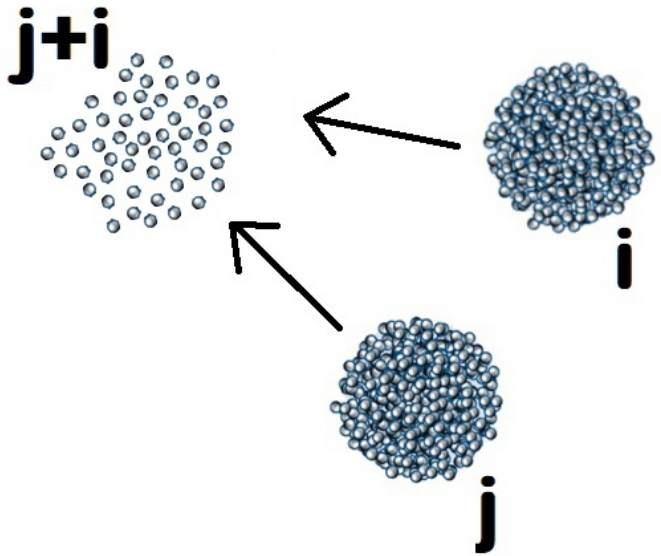


Фрагментация

$$\frac{dn_s}{dt} = \frac{1}{2}\sum_{i+j=s} K_{ij} n_i n_j - \sum_{j=1}^{\infty} K_{sj} n_s n_j - \lambda \sum_{j=1}^{\infty} K_{sj} n_s n_j,\ s \neq 1$$

$$\frac{dn_1}{dt} = \frac{\lambda}{2}\sum_{i,j \geqslant 2} (i+j) K_{ij} n_i n_j - \sum_{j=1}^{\infty} K_{1j} n_1 n_j + \lambda \sum_{j=2}^{\infty} j K_{1j} n_1 n_j,\ s = 1$$

λ – интенсивность фрагментации (безразмерная)

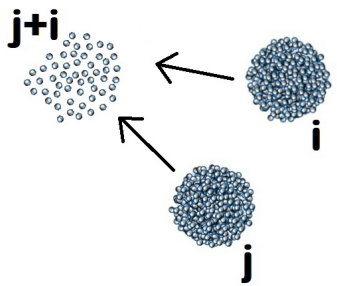


Фрагментация

$$\frac{dn_s}{dt} = \frac{1}{2}\sum_{i+j=s} K_{ij}n_i n_j - \sum_{j=1}^{\infty} K_{sj}n_s n_j - \lambda\sum_{j=1}^{\infty} K_{sj}n_s n_j,\ s \neq 1$$

$$\frac{dn_1}{dt} = \frac{\lambda}{2}\sum_{i,j\geqslant 2} (i+j)K_{ij}n_i n_j - \sum_{j=1}^{\infty} K_{1j}n_1 n_j + \lambda\sum_{j=2}^{\infty} jK_{1j}n_1 n_j,\ s = 1$$

1. Частота столкновений с агрегацией частиц пропорциональна частоте столкновений с фрагментацией. В результате получается, что ядра: K_{ij} и A_{ij} (*ядро фрагментации*) отличаются сомножителем $\lambda > 0$ т.е. $A_{ij} = \lambda K_{ij}$. (*Верно для широкого круга параметров системы)



Точное решение дифференциального уравнения

Поиск решения

$$\frac{dn_s}{dt} = \frac{1}{2}\sum_{i+j=s} K_{ij} n_i n_j - \sum_{j=1}^{\infty} K_{sj} n_s n_j - \lambda \sum_{j=1}^{\infty} K_{sj} n_s n_j,\ s \neq 1$$

$$\frac{dn_1}{dt} = \frac{\lambda}{2}\sum_{i,j\geqslant 2} (i+j) K_{ij} n_i n_j - \sum_{j=1}^{\infty} K_{1j} n_1 n_j + \lambda \sum_{j=2}^{\infty} j K_{1j} n_1 n_j,\ s = 1$$

Вводя начальные условия $n_1(t=0) = n_{10}, ..., n_s(t=0) = n_{s0}, ...$, получаем задачу Коши для бесконечной системы обыкновенных дифференциальных уравнений.

На данный момент известно мало теоретических решений для такой системы.

Достаточное условие, накладываемое на ядро для сохранения момента системы $\sum_{i=1}^{\infty} i n_i(t)$ при $t > 0$

Утверждение:

Первый момент постоянен при $t > 0$, если $\exists \lim_{i,j\to\infty} \frac{K_{ij}}{i+j}$ Иначе момент не сохраняется, либо сохраняется на конечном отрезке.

Известные решения

Начальные условия: $n_s(0) = \delta_{s1}$.

$$\frac{dn_s}{dt} = \frac{1}{2}\sum_{i+j=s} K_{ij} n_i n_j - \sum_{j=1}^{\infty} K_{sj} n_s n_j.$$

$K_{ij} = 2:$ $\quad n_s = \frac{1}{(1+t)^2} e^{-(s-1)\ln(1+1/t)}.$

$K_{ij} = i + j:$ $\quad n_s = \frac{s^{s-1}}{s!} e^{-t} \left(1 - e^{-t}\right)^{s-1} e^{-s\left(1-e^{-t}\right)}.$

Для уравнений с фрагментацией:

$K_{ij} = 1:$ $\quad n_s(t \to \infty) \to \lambda \frac{(2s-3)!!}{s!} \frac{(1/2+\lambda)^{s-1}}{(1+\lambda)^{2s-1}}, \quad n(t) = \frac{2\lambda}{2\lambda+1-e^{-\lambda t}}.$

Что же делать в других случаях???

Численные методы решения

Упрощение математической модели

Аппроксимируем исходную бесконечную систему дифференциальных уравнений конечной системой:

$$\frac{dn_s}{dt} = \frac{1}{2}\sum_{i+j=s} K_{ij} n_i n_j - \sum_{j=1}^{\mathbf{M}} K_{sj} n_s n_j.$$

Конечно-разностная схема

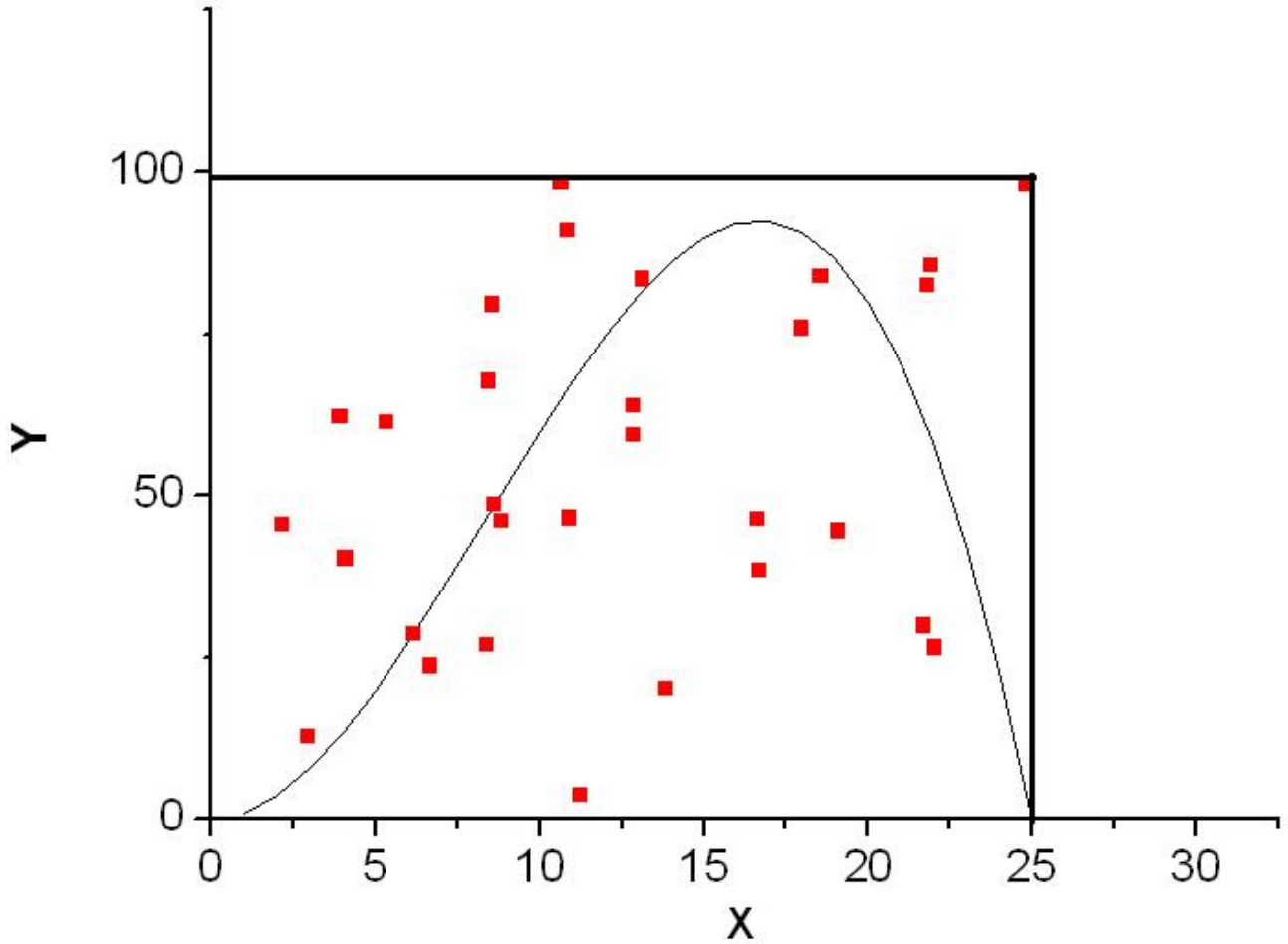
$$\frac{dn_s}{dt} = \frac{1}{2}\sum_{i+j=s} K_{ij} n_i n_j - \sum_{j=1}^{M} K_{sj} n_s n_j.$$

Сложность каждого шага по времени $O(M^2)$.
Если использовать малоранговую аппроксимацию K_{ij} ранга R, то $O\left(MR\log M\right)$.
С. А. Матвеев, Е. Е. Тыртышников, А. П. Смирнов, Н. В. Бриллиантов, Быстрый метод решения уравнений агрегационно-фрагментационной кинетики типа уравнений Смолуховского, Выч. мет. программирование, 15:1 (2014), 1–8

Можно запустить 10^6 частиц и уже получить точность порядка 0.1%.

Методы Монте-Карло

Простейшая идея методов Монте-Карло



Основная идея Монте-Карло

1. Генерируем случайное число $r \in (0, 1)$
2. Выберем две случайные частицы из нашей системы
3. IF $r < p_{ij}$:
   Добавляем частицу $i + j$ в память;
   Делаем шаг по времени;
   Обновляем вероятности системы;
4. ELSE переход на шаг №1

Моменты реализации

1. Генерируем случайное число $r \in (0, 1)$
2. Выберем две случайные частицы из нашей системы **(Как выбирать?)**
3. IF $r < p_{ij}$: **(Как считать вероятности столкновения частиц?)**
   Добавляем частицу $i + j$ в память; **(Как эффективно хранить частицы?)**
   Делаем шаг по времени; **(Как считать шаг по времени?)**
   Обновляем вероятности системы; **(Как быстро пересчитывать вероятности?)**
4. ELSE переход на шаг №1

Отвечая на эти вопросы, мы получим два алгоритма

Модификация до метода "принятия-отклонения"

1. **Как выбирать?**
   Выбираем пары равновероятно: $P_{ij}^0 = \frac{1}{2} N_i N_j / \left(\frac{1}{2} N^2\right)$ (N - общее число частиц)
2. **Как считать вероятности столкновения частиц?**
   Добавим шаг отклонения: после выбора пары производим столкновение с вероятностью принятия $P_{\text{acc}} = K_{ij} / \max\limits_{i,j} K_{ij}$.
3. **Как эффективно хранить частицы?**
   Будем хранить их в виде вектора размеров.
4. **Как быстро пересчитывать вероятности?**
   Необходима процедура обновления величины $\max\limits_{i,j} K_{ij}$ на каждом шаге алгоритма.
5. **Как считать шаг по времени?**

Как считать шаг по времени?

Вспомним:

K_{ij} – среднее число **столкновений** в единице **пространства** за единицу **времени** в системе с единичной **концентрацией частиц** размера – i, j

$$[K_{ij}] = \frac{\text{столкновения}}{\text{пространство} \cdot \text{время} \cdot \frac{\text{частица}}{\text{пространство}} \cdot \frac{\text{частица}}{\text{пространство}}}$$

Как считать шаг по времени?

Вспомним:

K_{ij} – среднее число **столкновений** в единице **пространства** за единицу **времени** в системе с единичной **концентрацией частиц** размера – i, j

$$[K_{ij}] = \frac{\text{столкновения}}{\text{пространство} \cdot \text{время} \cdot \frac{\text{частица}}{\text{пространство}} \cdot \frac{\text{частица}}{\text{пространство}}}$$

Тогда число **столкновений** частиц размера i и j в **пространстве** V за **время** Δt:

$$K_{ij} \cdot n_i \cdot n_j \cdot V \cdot \Delta t$$

А общее число столкновений в системе:

$$\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} K_{ij} n_i n_j V \Delta t$$

Как считать шаг по времени?

Зная общее число столкновений в системе за время - Δt :

$$\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} K_{ij} n_i n_j V \Delta t$$

Мы можем найти сколько время необходимо для одного столкновения τ_{coll}:

$$\frac{1}{\tau_{\text{coll}}} = \frac{1}{2}\sum_{i=1}^{\infty}\sum_{j=1}^{\infty} K_{i,j} n_i n_j V = \frac{1}{2}\sum_{i=1}^{\infty}\sum_{j=1}^{\infty} \frac{K_{i,j}}{V} N_i N_j = \frac{1}{2V} N(N-1)\left\langle K_{i,j}\right\rangle$$
$$= \frac{K_{\max} P_{acc} \cdot N(N-1)}{2V}$$

Из-за "принятий-отклонений"частота столкновений в нашем алгоритме уменьшается по сравнению с теоретическим значением, поэтому необходимо компенсировать это, уменьшенив шаг по времени:

$$\tau = \tau_{coll} \cdot P_{acc} = \frac{2V}{N(N-1) \cdot K_{max}}$$

Метод принятия-отклонения[1]

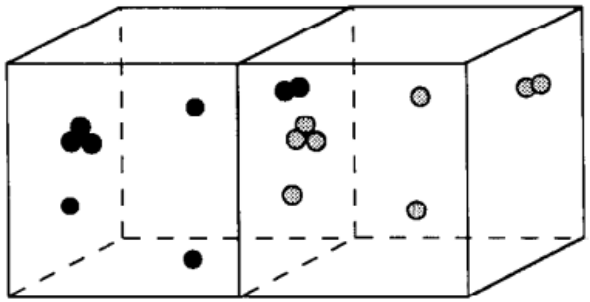
1. **Как выбирать?**
   Выбираем пары равновероятно: $P_{ij}^0 = \frac{1}{2} N_i N_j / \left(\frac{1}{2} N^2\right)$ (N - общее число частиц)
2. **Как считать вероятности столкновения частиц?**
   Добавим шаг отклонения: после выбора пары производим столкновение с вероятностью принятия $P_{\text{acc}} = K_{ij} / \max\limits_{i,j} K_{ij}$.
3. **Как эффективно хранить частицы?**
   Будем хранить их в виде вектора размеров.
4. **Как быстро пересчитывать вероятности?**
   Необходима процедура обновления величины $\max\limits_{i,j} K_{ij}$ на каждом шаге алгоритма.
5. **Как считать шаг по времени?**

$$\tau = \frac{2V}{N(N-1) \cdot K_{max}}$$

[1] Garcia, A. L., van den Broeck, C., Aertsens, M., Serneels, R. (1987). A Monte Carlo simulation of coagulation. Physica A: Statistical Mechanics and Its Applications, 143(3), 535–546.

Проблема деградации модели

После каждого успешного события частиц становится меньше, поэтому качество выборки становится хуже, следовательно, точность приближения снижается.
Наша среда - пространственно однородна, поэтому можно воспользоваться следующим трюком.[1] В момент, когда остаётся менее половины частиц, необходимо удвоить удвоить объём и количество частиц. Таким образом, мы получим те же концентрации, но меньший шаг по времени.



[1] Liffman, K. (1992). A direct simulation Monte-Carlo method for cluster coagulation. Journal of Computational Physics, 100(1), 116-127.

Псевдокод

Инициализация переменных и массива размеров кластеров s(k) из N = N(t=0) элементов.
while curtime < maxtime **do**
  repeat
    curtime += $\frac{2V}{K_{\max} N(N-1)}$
    v = random_integer(1, N)
    repeat
      l = random_integer(1, N)
    until v $\neq$ l
    i = s(v)
    j = s(l)
  until $K_{\max} \cdot \mathrm{rand}(0,1] \leqslant K_{ij}$
  v,l = min(v,l), max(v,l)
  s(v) = s(v) + s(l)
  s(l) = s(N)
  s(N) = 1
  N −= 1
  if N <= N(t=0)/2 **then**
    Скопировать размеры s(i+N) = s(i), i=$\overline{1, N}$.
    N *= 2
    V *= 2
  end if
  if s(i) > $v_{\max}$ **then**
    $v_{\max}$ = s(v)
    Обновление $K_{\max}(v_{\max})$
  end if
end while

Моменты реализации

1. Генерируем случайное число $r \in (0, 1)$
2. Выберем две случайные частицы из нашей системы **(Как выбирать?)**
3. IF $r < p_{ij}$: **(Как считать вероятности столкновения частиц?)**
   Добавляем частицу $i + j$ в память; **(Как эффективно хранить частицы?)**
   Делаем шаг по времени; **(Как считать шаг по времени?)**
   Обновляем вероятности системы; **(Как быстро пересчитывать вероятности?)**
4. ELSE переход на шаг №1

Малоранговое разложение матрицы (вспоминаем)

Пусть $B \in \mathbb{R}^{M \times N}$ – матрица $M \times N$ с вещественными элементами B_{ij}.

Ранг матрицы B – минимальное значение R, для которого B может быть выражена как сумма внешних произведений:

$$B = \sum_{r=1}^{R} B^r = \sum_{r=1}^{R} u^r \cdot (v^r)^T, \quad u^k \in \mathbb{R}^{M \times 1}, \quad v^k \in \mathbb{R}^{N \times 1}, \quad k = 1 \dots R$$

$$B_{ij} = \sum_{r=1}^{R} {u_i}^r ({v_j}^r)^T, \quad k = 1 \dots R$$

Когда $R < min(N, M)$, выражение называется малоранговым разложением матрицы B.

Малоранговый метод. Как выбирать частицы?

Ядро системы может быть представлено в виде $K_{ij} = \sum_{r=1}^{R} \widetilde{u_i}^r \widetilde{v_j}^r$, тогда при ${u_i}^r = N_i \widetilde{u_i}^r$, ${v_j}^r = N_j \widetilde{v_j}^r$:

$$K_{ij} N_i N_j = \sum_{r=1}^{R} {u_i}^r {v_j}^r$$

Для $R = 1$: $K_{ij} N_i N_j = u_i v_j$
В таком случае выбор размеров $i,\ j \in 1 \dots M$ (M – максимальная масса частицы) можно провести независимо $P_{ij} = p_i p_j$, так как вероятность того, что первая частица имеет массу i равна:

$$p_i = \frac{\frac{\Delta t}{V} \sum_{j=1}^{M} K_{ij} N_i N_j}{\frac{\Delta t}{V} \sum_{i,\, j=1}^{M} K_{ij} N_i N_j} = \frac{\sum_{j=1}^{M} u_i v_j}{\sum_{i,\, j=1}^{M} u_i v_j} = \frac{u_i \left(\sum_{j=1}^{M} v_j \right)}{\sum_{i=1}^{M} u_i \left(\sum_{j=1}^{M} v_j \right)} = \frac{u_i}{\sum_{i=1}^{M} u_i}$$

p_i зависит только от компонент вектора $\vec{u}$. Аналогично вычисляется p_j.

Малоранговый метод. Как выбирать частицы?

Если ранг больше 1, мы можем представить матрицу с элементами $B_{ij} = K_{ij} N_i N_j$ как сумму ядер ранга 1 и рассматривать их отдельно, как будто мы рассматриваем разные механизмы агрегации.

Чтобы определить, какое ядро использовать, мы можем сравнить общие скорости агрегации и выбрать r-ый член вида $B^r = u^r \cdot (v^r)^T$ с вероятностью:

$$p_r = \frac{\sum\limits_{i,\, j=1}^{M} B_{ij}^r}{\sum\limits_{r=1}^{R} \sum\limits_{i,\, j=1}^{M} B_{ij}^r} = \frac{\sum\limits_{i,\, j=1}^{M} {u_i}^r {v_j}^r}{\sum\limits_{r=1}^{R} \sum\limits_{i,\, j=1}^{M} {u_i}^r {v_j}^r} = \frac{\sum\limits_{i=1}^{M} {u_i}^r \sum\limits_{j=1}^{M} {v_j}^r}{\sum\limits_{r=1}^{R} \left(\sum\limits_{i=1}^{M} {u_i}^r \sum\limits_{j=1}^{M} {v_j}^r \right)}$$

Малоранговый метод. Как выбирать частицы?

Выбрав компоненту r, элемент предятавляется в виде:

$$K_{ij}N_iN_j = u_i^r v_j^r.$$

Выбор размеров i и j между 1 и $M = s_{\max}$ можно провести независимо:

$$P_{ij} = P_iP_j, \quad P_i = \frac{u_i}{\sum\limits_s u_s}, \quad P_j = \frac{v_j}{\sum\limits_s v_s}.$$

Как происходит выбор:

1. Генерируется число $x = \mathrm{rand}\,(0,1] \cdot \sum\limits_k u_k$.
2. Выбирается промежуток так, что

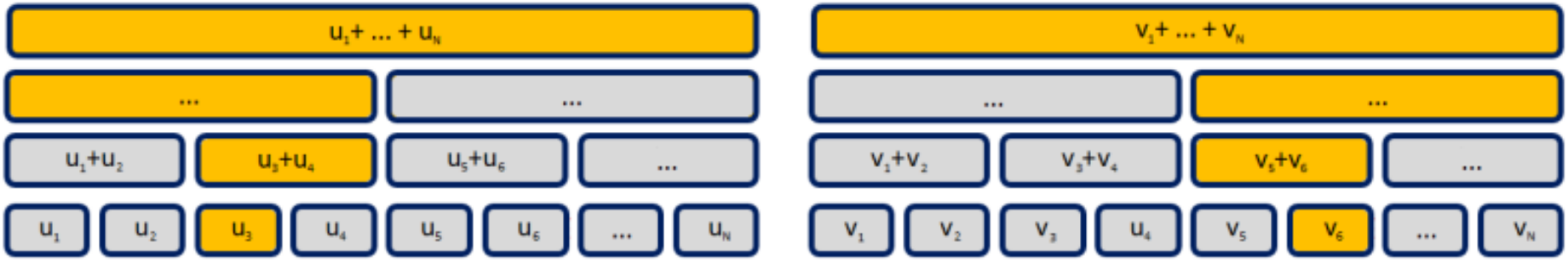
$$\sum_{k=1}^{i-1} u_k < x \leqslant \sum_{k=1}^{i} u_k.$$

3. Вероятность попасть в этот промежуток равна отношению его длины u_k к общей сумме длин $\sum\limits_k u_k$ – то, что нужно!

Линейный поиск займет $O(N)$ операций. Хотим бинарный.

Как сделать быстрый (бинарный) поиск?

Деревья отрезков:



Дерево отрезков – это двоичное дерево, которое содержит частичные суммы элементов массива. В вершине такого дерева содержится сумма всех элементов массива.
Поиск и модификация элементов в дереве отрезков занимают $O(\log M)$, где $M = s_{\max}$

Модификация до малорангового метода

1. **Как эффективно хранить частицы?**
   Необходимо хранить $2R$ деревьев отрезков. Длина каждого равна $2s_{max}$, $s_{\max}$ – максимальный размер
2. **Как быстро пересчитывать вероятности?**
   Модификация занимает $O(\log M)$, где $M = s_{\max}$
3. **Как выбирать пару частиц?**
   Поиск в дереве отрезков занимает $O(\log M)$, где $M = s_{\max}$
4. **Как считать вероятности столкновения частиц?**
   Поиск в дереве отрезков занимает $O(\log M)$, где $M = s_{\max}$
5. **Как считать шаг по времени?**

Малоранговый метод. Как считать шаг по времени?

$$\tau = \frac{\Delta t}{\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} K_{ij} n_i n_j V \Delta t} = \frac{1}{\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} K_{ij} n_i n_j V}$$

$$\frac{1}{\tau} = \frac{1}{2V}\sum_{i=1}^{N}\sum_{j=1}^{N} K_{ij} N_i N_j = \frac{1}{2V}\sum_{i,j=1}^{N}\sum_{k=1}^{R} u_i^k v_j^k =$$

$$| \; R = rank(B), \; \{B\}_{ij} = K_{ij} N_i N_j \; , B = \sum_{k=1}^{R} u_k v_k^T \; |$$

$$= \frac{1}{2V}\sum_{k=1}^{R}(\sum_{i=1}^{N} u_i^k \cdot \sum_{j=1}^{N} v_j^k)$$

Таким образом, шаг по времени:

$$\tau = \frac{2V}{\sum_{k=1}^{R}(\sum_{i=1}^{N} u_i^k \cdot \sum_{j=1}^{N} v_j^k)}$$

Модификация до малорангового метода

1. **Как эффективно хранить частицы?**
   Необходимо хранить $2R$ деревьев отрезков, длина каждого равна $2s_{max}$, $s_{\max}$ – максимальный размер
2. **Как быстро пересчитывать вероятности?**
   Обновление занимает $O(\log M)$, где $M = s_{\max}$
3. **Как выбирать пару частиц?**
   Поиск в дереве отрезков занимает $O(\log M)$, где $M = s_{\max}$
4. **Как считать вероятности столкновения частиц?**
   Поиск в дереве отрезков занимает $O(\log M)$, где $M = s_{\max}$
5. **Как считать шаг по времени?**

$$\tau = \frac{2V}{\sum\limits_{k=1}^{R}(\sum\limits_{i=1}^{N} u_i^k \cdot \sum\limits_{j=1}^{N} v_j^k)}$$

Таким образом, нужно посчитать сумму произведений вершин деревьев отрезков. Сложность подсчёта шага по времени: $O(R)$

Общая версия выбора размеров

1: **repeat**
2: z = $\mathrm{rand}(0,1)\sum_{r=1}^{R} u^{r,t}(1)v^{r,t}(1)$
3: {Обновление времени:}
4: curtime += $2V/\sum_{r=1}^{R} u^{r,t}(1)v^{r,t}(1)$
5: {Выбор компоненты:}
6: **for** k = 1 **to** R−1 **do**
7: z −= $u^{r,t}(1)v^{r,t}(1)$
8: **if** z <= 0 **then**
9: **break**
10: **end if**
11: **end for**
12: {Выбор размеров:}
13: i = **Find**$(u^{t,k})$
14: j = **Find**$(v^{t,k})$
15: {Отклонение столкновения "с собой":}
16: **if** (i == j) **and** $(N_i \cdot \mathrm{rand}(0,1] \leqslant 1)$ **then**
17: **continue**
18: **end if**
19: {Отклонение столкновения, если частиц нет:}
20: **if** $(N_i == 0)$ **or** $(N_j == 0)$ **then**
21: **continue**
22: **end if**
23: **until** i и j не выбраны

Тестирование алгоритмов

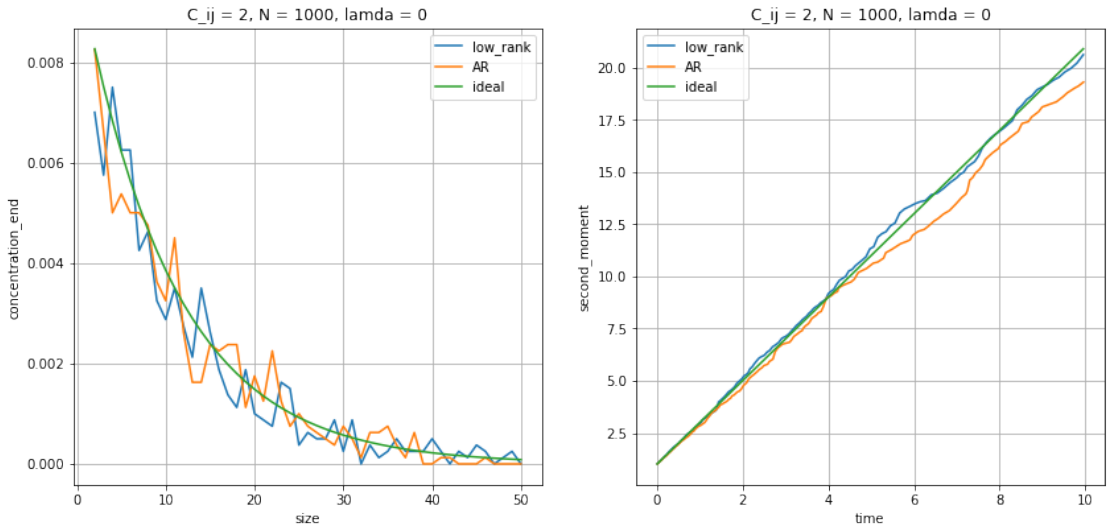
Описание тестов

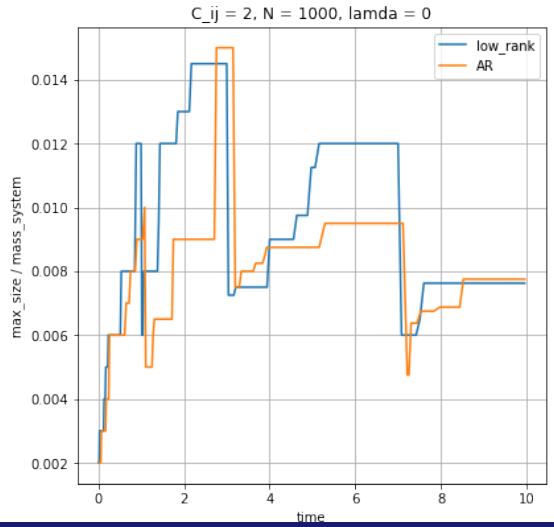
Мной были реализованы оба этих алгоритма на C++[1].
Мы будем сравнивать малоранговый метод с методом принятия-отклонения. Поскольку в конечном итоге все три метода приводят к одинаковым вероятностям, пропорциональным K_{ij} , а все остальные шаги те же, то они имеют одинаковую точность. Единственная разница во времени вычисления, поэтому мы сосредоточимся на ней.

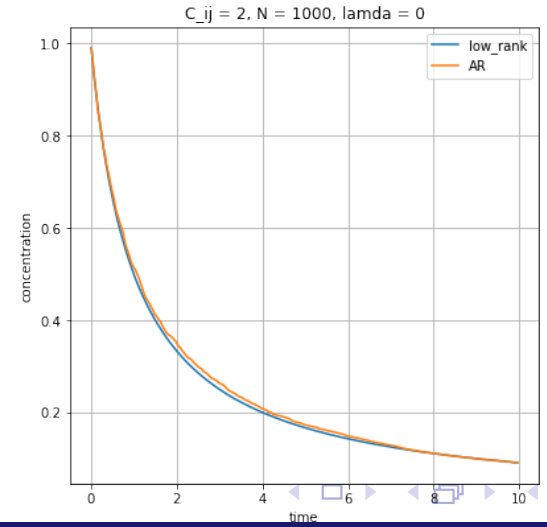
[1] https://github.com/DrEternity/Low-rank-Monte-Carlo-simulation-for-aggregation-fragmentation-kinetics

$K_{ij} = 2$, $N = 1000$, **maxtime** $= 10$

Малоранговый метод: 0.005 сек.
Принятие-отклонение: 0.3 сек.

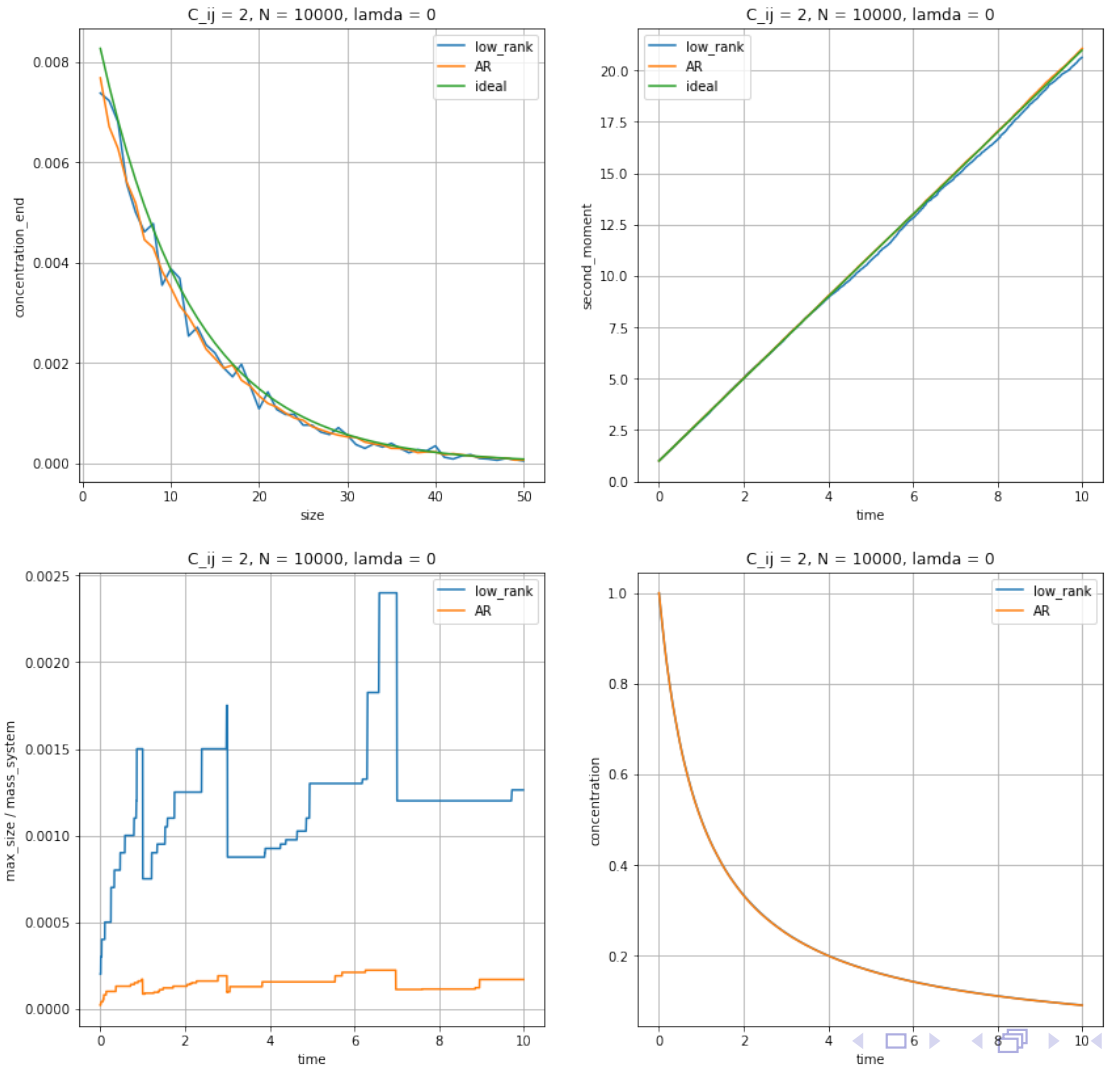






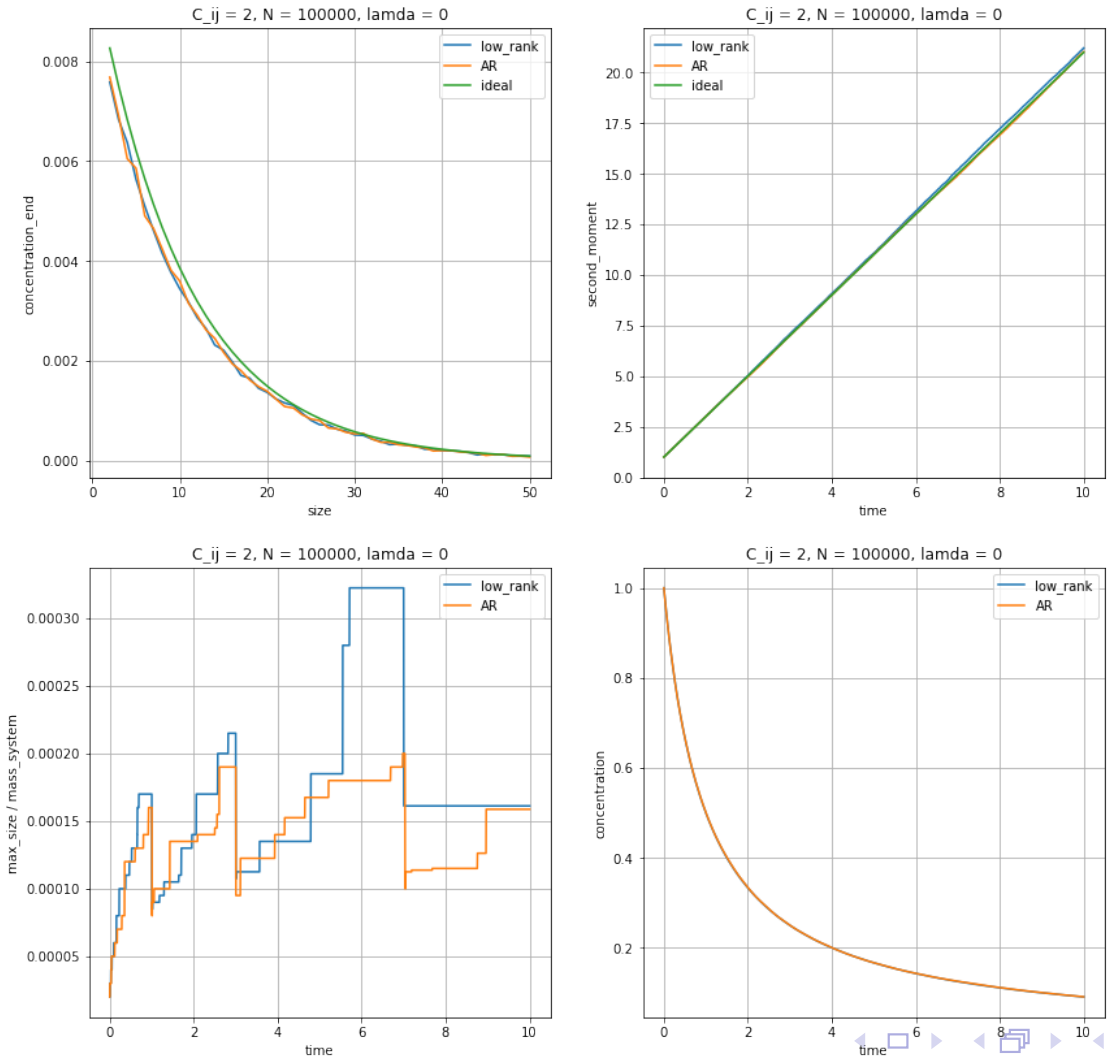
$K_{ij} = 2,\ N = 10000,\ \mathbf{maxtime} = 10$

Малоранговый метод: 0.025 сек.
Принятие-отклонение: 5.23 сек.



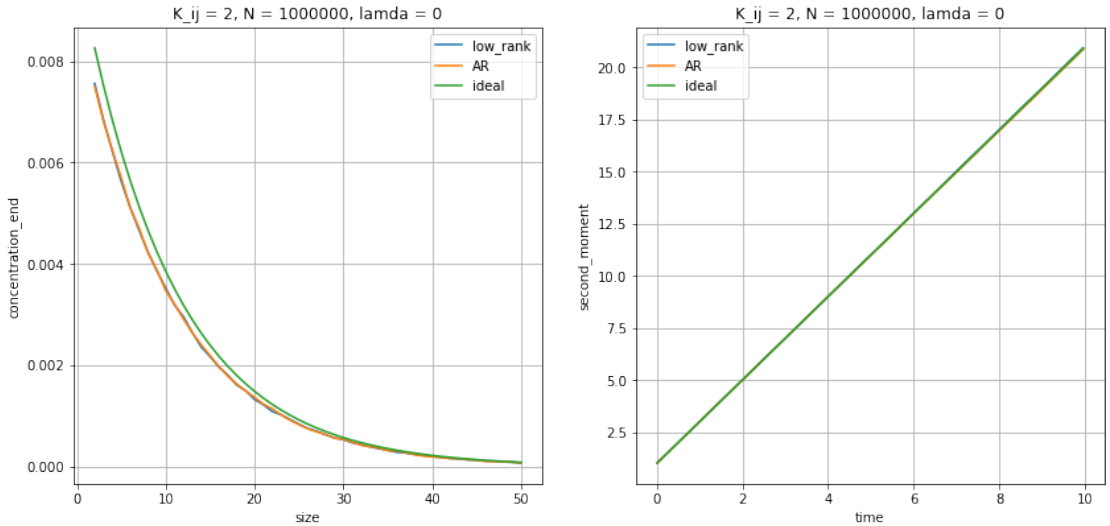
$K_{ij} = 2$, $N = 100000$, **maxtime** $= 10$

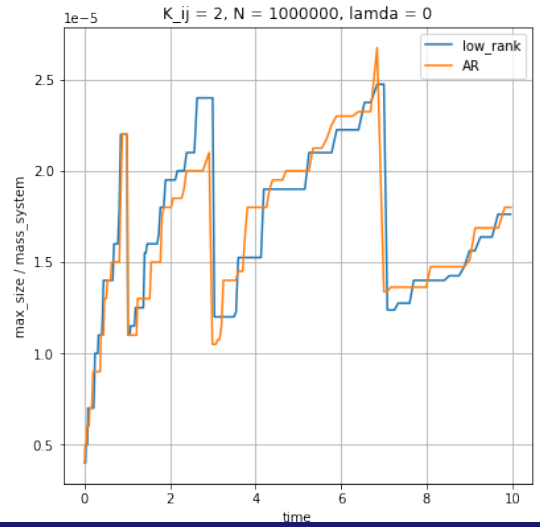
Малоранговый метод: 0.25 сек.
Принятие-отклонение: 70.59 сек.

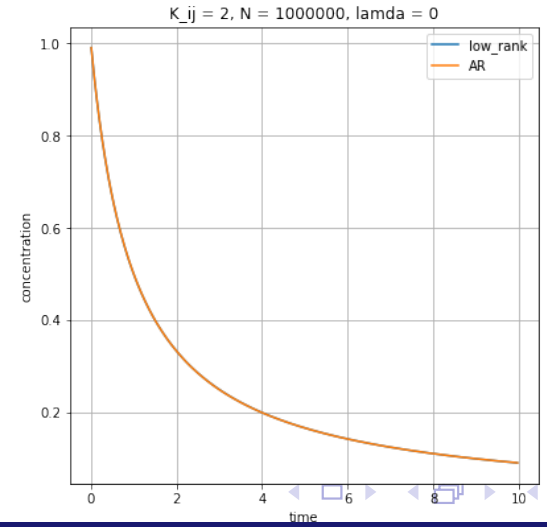


$K_{ij} = 2,\ N = 1000000,$ **maxtime** $= 10$

Малоранговый метод: 2.6 сек.
Принятие-отклонение: 1249.67 сек.

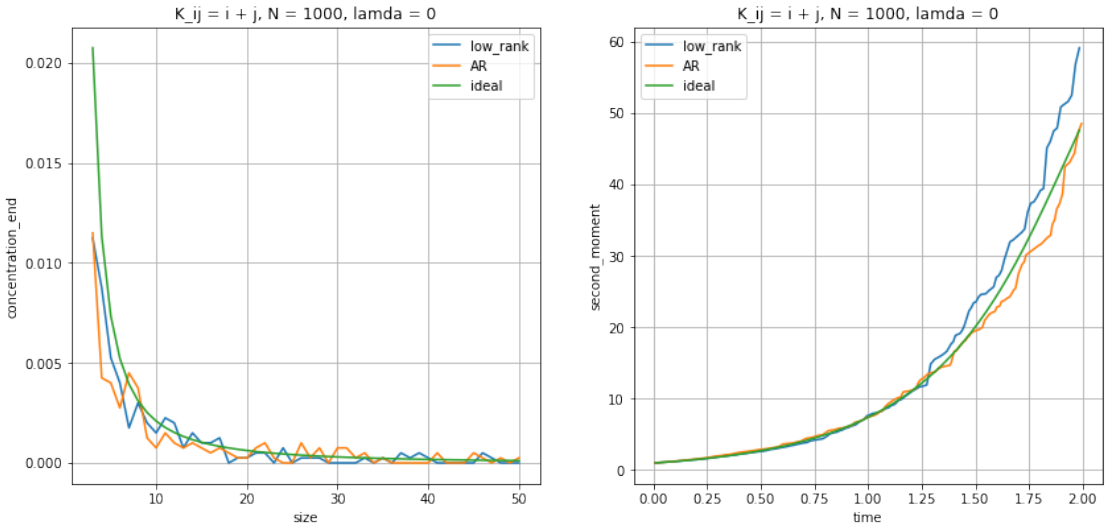


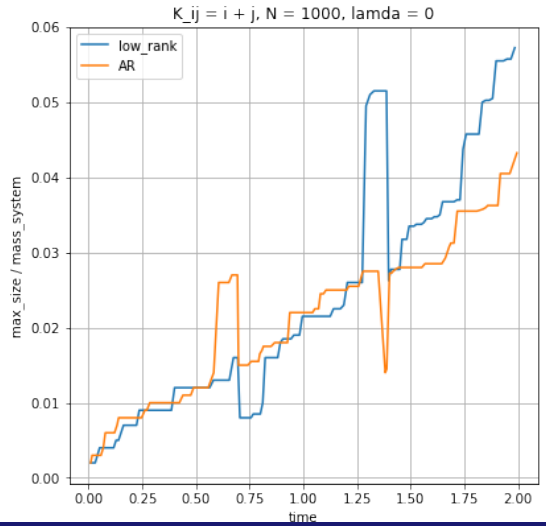


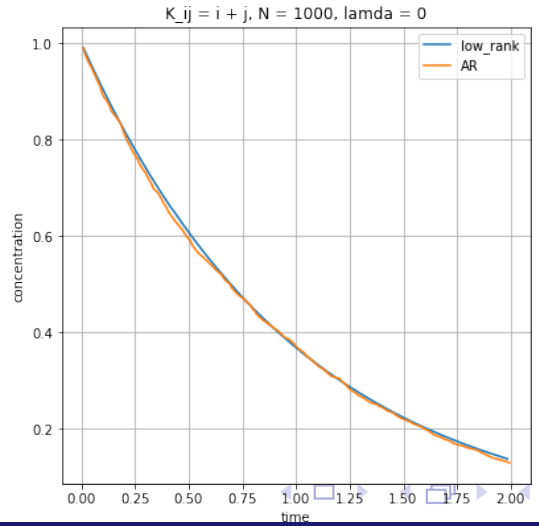


$K_{ij} = i + j$, $N = 1000$, **maxtime** $= 2$

Малоранговый метод: 0.004 сек.
Принятие-отклонение: 0.29 сек.

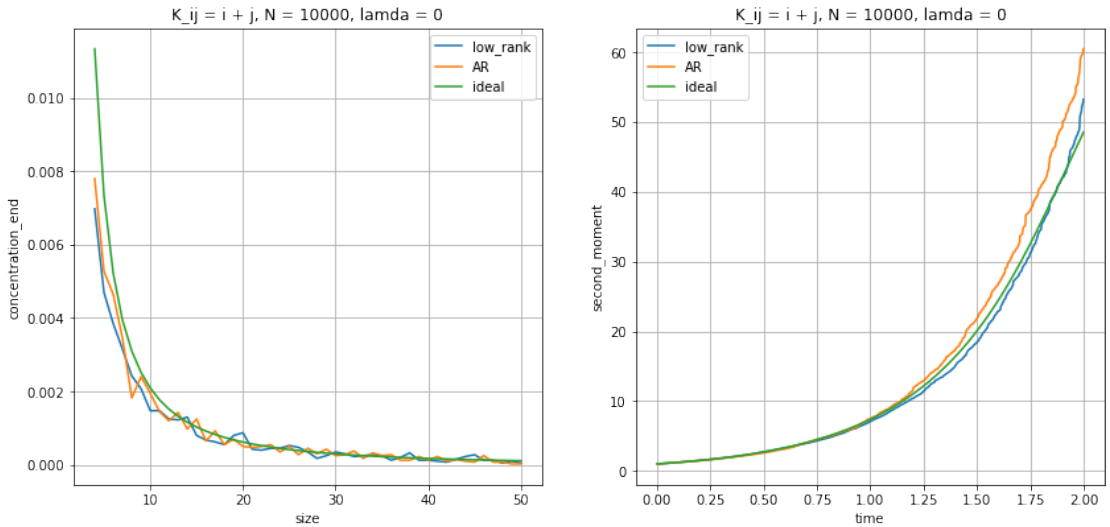


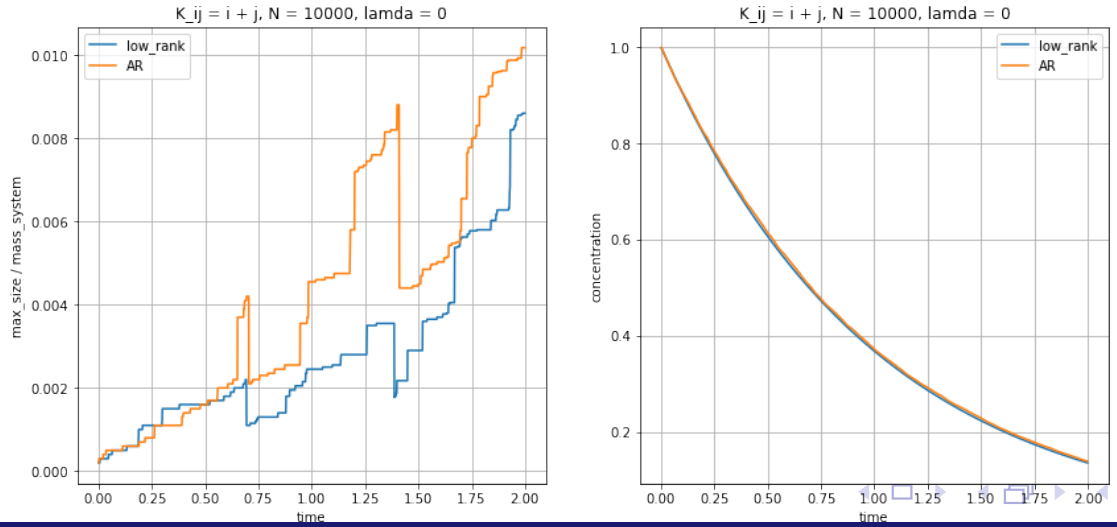




$K_{ij} = i + j$, $N = 10000$, **maxtime** $= 2$

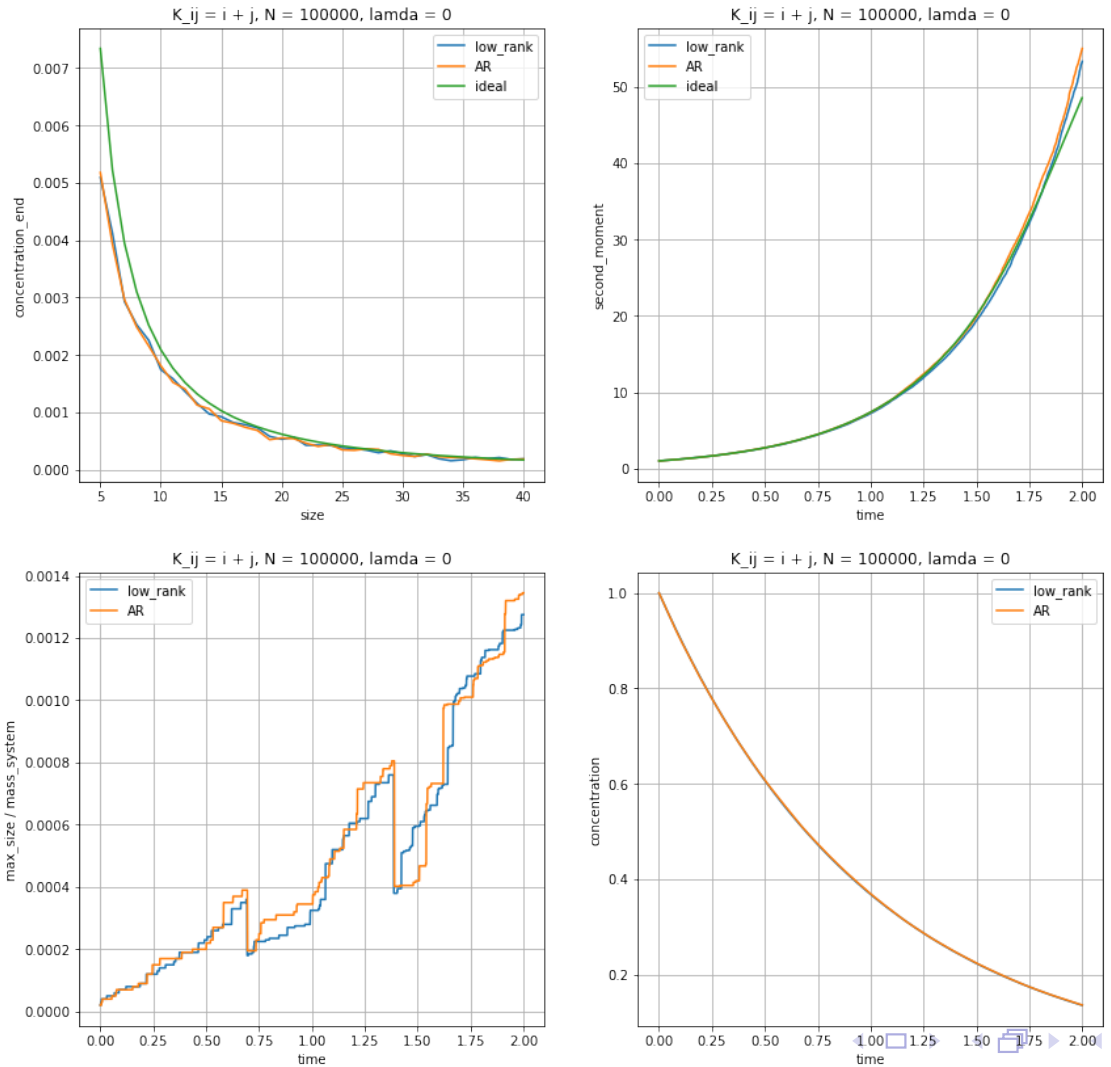
Малоранговый метод: 0.035 сек.
Принятие-отклонение: 7.2 сек.





$K_{ij} = i + j$, $N = 100000$, **maxtime** $= 2$

Малоранговый метод: 0.37 сек.
Принятие-отклонение: 210.59 сек.



Дополнительное тестирование малорангового метода

$\{K_{ij} = i + j;\ N = 10^7;\ t = 1\}$ - Столкновений: $0,76 \cdot 10^7$ Время работы: 1.629
$\{K_{ij} = i + j;\ N = 10^7;\ t = 2\}$ - Столкновений: $1,5 \cdot 10^7$ Время работы: 3.84858
$\{K_{ij} = i + j;\ N = 10^7;\ t = 3\}$ - Столкновений: $2,2 \cdot 10^7$ Время работы: 7.08634
$\{K_{ij} = i + j;\ N = 10^7;\ t = 4\}$ - Столкновений: $2,9 \cdot 10^7$ Время работы: 6.12887
$\{K_{ij} = i + j;\ N = 10^8;\ t = 1\}$ - Столкновений: $7,6 \cdot 10^7$ Время работы: 9.459
$\{K_{ij} = i + j;\ N = 10^8;\ t = 2\}$ - Столкновений: $14 \cdot 10^7$ Время работы: 28.72
$\{K_{ij} = i + j;\ N = 10^8;\ t = 3\}$ - Столкновений: $22 \cdot 10^7$ Время работы: 44.44

Осцилляции

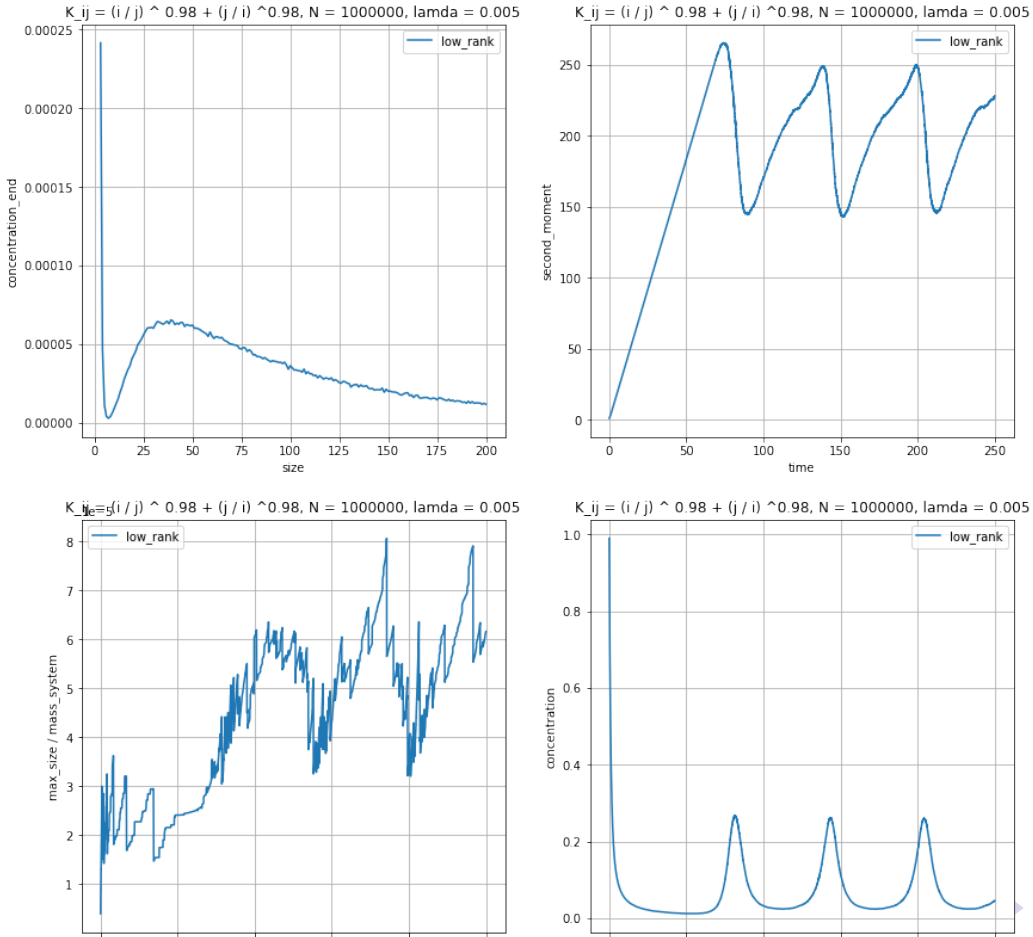
В противоположность простому релаксационному поведению, ожидаемому для уравнений, подобных Смолуховскому, недавно был зарегистрирован удивительный режим колебаний для кинетики агрегации-фрагментации. Было замечено, что кинетические уравнения с ядрами вида:

$$K_{ij} = (\frac{i}{j})^a + (\frac{j}{i})^a$$

приводят к устойчивым колебаниям для $a > 0,5$ и $0 < \lambda < \lambda_{crit}$
Такие колебания были обнаружены численно с помощью эффективной реализации конечно-разностной схемы интегрирования по времени Рунге-Кутты второго порядка.
Дополнительная независимая проверка этого явления крайне желательна. Здесь мы покажем, что Малоранговый метод Монте-Карло демонстрируют те же колебания плотностей, что и в статье.

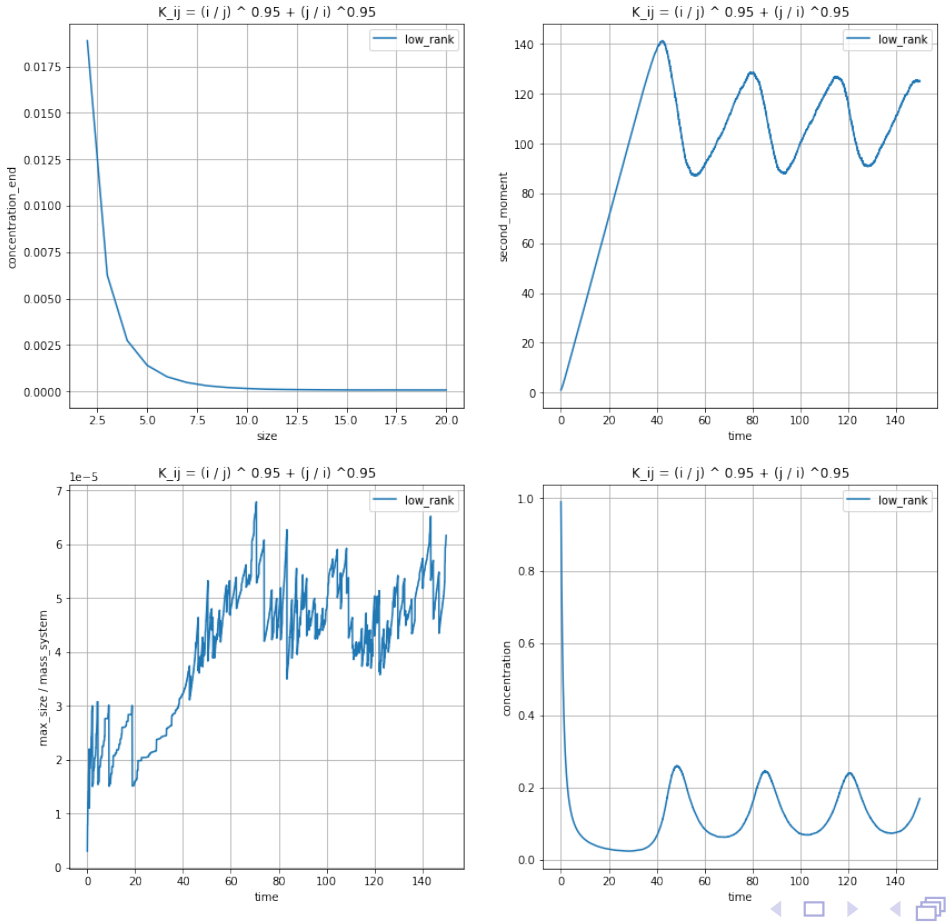
Осцилляции

Проведя численное моделирование:
$N = 10^6,\ K_{ij} = (\frac{i}{j})^{0.98} + (\frac{j}{i})^{0.98},\ \lambda = 0.005$
Получаем следущий результат:
Время: 768 сек. Соударений: 209951719



Осцилляции

Проведя численное моделирование:
$N = 10^6,\ K_{ij} = (\frac{i}{j})^{0.95} + (\frac{j}{i})^{0.95},\ \lambda = 0.01.$
Получаем следущий результат:
Время: 1177 сек. Соударений: 302575699



Итог тестов

1. Малоранговый метод показывает высокую скорость работы на ядрах разных типов по сравнению с методом принятия-отклонения
2. Колебательные решения для специального ядра являются надежными и стабильными по отношению к стохастическим ошибкам и флуктуациям, присущим методам Монте-Карло.

Заключение

Заключение

1. Изучена новая модель взаимодействия объектов в природе и её приложения
2. Осознана математическая формализация в виде системы дифференциальных уравнений
3. Разобраны теоретические аспекты существования решений
4. Реализованы численные методы Монте-Карло
5. Произведено сравнение временных затрат алгоритмов

Благодарю за внимание!