

Лекция 2

Основы метода Монте-Карло

Лю Шисян

Кафедра теплофизики (Э6)

https://sxliu.site/teaching/

План лекции

1. Основы метода Монте-Карло
2. Интегрирование методом Монте-Карло
3. Случайное блуждание и диффузия
4. Теплоперенос поперёк плёнки

Репозиторий кода

`https://github.com/lyushisyan/computational-methods-course`

1. **Основы метода Монте-Карло**
2. Интегрирование методом Монте-Карло
3. Случайное блуждание и диффузия
4. Теплоперенос поперёк плёнки

От иглы Бюффона к методу Монте-Карло

Что такое метод Монте-Карло?

Метод Монте-Карло — это способ решения задач с помощью случайных чисел и многократных испытаний. По результатам этих испытаний можно получить приближённое решение задачи.

1777 Бюффон предложил бросать иглу на лист с параллельными линиями. По частоте пересечений можно оценить число π.

1946 Станислав Улам задумался, как узнать вероятность выигрыша в пасьянсе. Его идея — много раз разложить карты и подсчитать долю выигрышей.

1948 Улам и фон Нейман применили случайные испытания на компьютере ENIAC для моделирования движения нейтронов.

1949 Метрополис и Улам опубликовали статью «The Monte Carlo Method». Название связано с казино Монте-Карло.

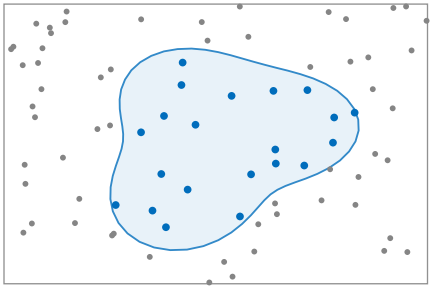
Идея метода Монте-Карло

На карте показан участок площадью 100 га. Как узнать площадь озера, если его берег имеет сложную форму?

Случайным образом выберем 80 точек на всём участке.
Посчитаем, сколько точек попало в озеро.

В нашем примере в озеро попала 21 точка из 80.

$$100 \text{ га} \times \frac{21}{80} \approx 26 \text{ га}.$$



21 точка в воде 59 точек на суше

Доля точек, попавших в озеро, позволяет оценить долю площади, которую оно занимает.

Почему доля попаданий равна среднему?

Выбор одной случайной точки — это **испытание**. Все 80 точек вместе образуют **выборку**.

Для каждой точки запишем результат:

Точка в воде Запишем 1 (21 точка).

Точка на суше Запишем 0 (59 точек).

Теперь найдём **среднее значение** всех результатов:

$$\bar{x} = \frac{21 \cdot 1 + 59 \cdot 0}{80} = \frac{21}{80} = 0{,}2625.$$

Среднее значение нулей и единиц равно доле попаданий.

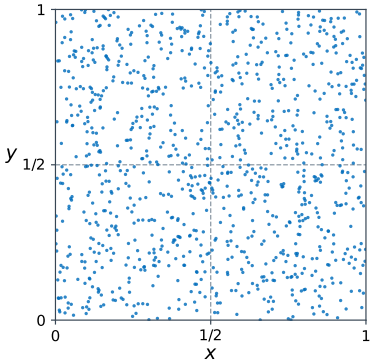
В Python результат проверки записывается как `True` или `False`. При вычислении среднего они соответствуют 1 и 0.

Как получить случайные точки в квадрате

Для каждой точки случайным образом выбираем две координаты x и y от 0 до 1.

```python
import numpy as np

rng = np.random.default_rng(42)
x = rng.random(1000)
y = rng.random(1000)
```



- `default_rng(42)` — создаёт генератор случайных чисел.
- `random(1000)` — создаёт 1000 случайных чисел от 0 до 1.
- `x[i], y[i]` — координаты одной точки.

Равномерное распределение: вероятность попасть в любую из четырёх равных частей квадрата одинакова и равна $1/4$.

Проверим метод на примере числа π

Возьмём квадрат со стороной 1 и поместим в него четверть круга радиуса 1.

Площади этих фигур:

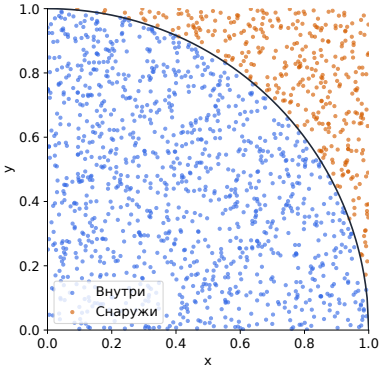
$$S_{\text{квадрата}} = 1, \qquad S_{\text{круга}} = \frac{\pi}{4}.$$

Равномерно распределим случайные точки по квадрату.
Вероятность попасть в четверть круга равна отношению площадей:

$$P = \frac{\pi/4}{1} = \frac{\pi}{4}.$$

Значит, число π можно оценить по доле попаданий:

$$\widehat{\pi} = 4\frac{N_{\text{внутри}}}{N}$$



Точки внутри четверти круга и за её пределами.

Как определить, попала ли точка внутрь?

Центр круга находится в точке $(0, 0)$, а радиус равен 1.

Точка (x, y) находится внутри круга или на его границе, если

$$x^2 + y^2 \leq 1$$

Рассмотрим три примера:

x	y	$x^2 + y^2$	Результат
0,2	0,3	0,13	Внутри: `True`
0,8	0,8	1,28	Снаружи: `False`
0,6	0,8	1,00	На границе: `True`

Из трёх точек две попали внутрь круга или на его границу.

В Python эту проверку можно выполнить сразу для всех случайных точек.

Оценка π по 100 000 точкам

```python
import numpy as np

rng = np.random.default_rng(42)
N = 100000
x = rng.random(N)
y = rng.random(N)

hit = x**2 + y**2 <= 1
pi_mc = 4 * hit.mean()

print(hit.sum(), pi_mc)
```

Что делает программа?

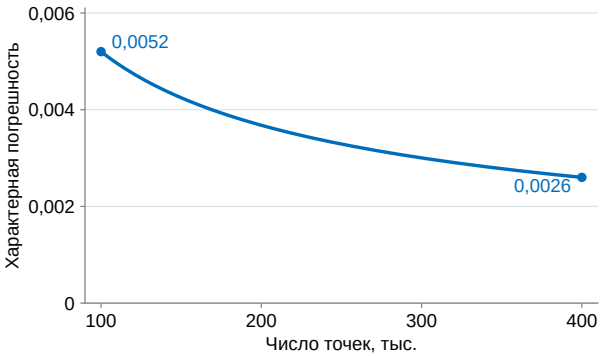
- Создаёт $100\,000$ случайных точек.
- Проверяет, какие точки попали в четверть круга.
- Вычисляет долю попаданий и умножает её на 4.

Результат:

$$\widehat{\pi} = 4\frac{78\,444}{100\,000} = 3{,}13776.$$

Точное значение $\pi \approx 3{,}14159$. Метод Монте-Карло даёт близкую оценку.

Как число точек влияет на точность?



Чем больше случайных точек, тем точнее обычно получается оценка числа π.

Для независимых случайных точек характерная погрешность уменьшается по закону

$$\varepsilon \propto \frac{1}{\sqrt{N}}$$

Например, если увеличить число точек в 4 раза, характерная погрешность уменьшится в 2 раза.

Важно: отдельная оценка может стать менее точной даже при увеличении N.

1. Основы метода Монте-Карло

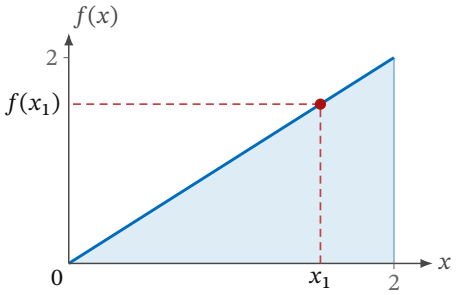
2. Интегрирование методом Монте-Карло

3. Случайное блуждание и диффузия

4. Теплоперенос поперёк плёнки

От случайных точек к интегралу

В предыдущем примере мы записывали для каждой точки 1 или 0.
Теперь будем записывать значение функции $f(x_i)$ в случайной точке x_i.



Возьмём простую функцию:

$$f(x) = x, \qquad x \in [0, 2].$$

Для каждой случайной точки x_i найдём высоту $f(x_i)$.

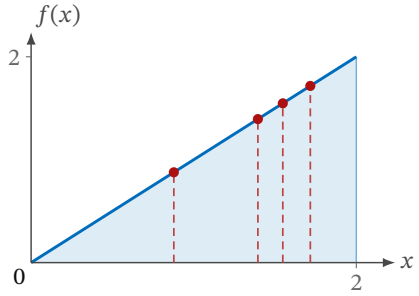
Площадь под графиком — это площадь треугольника:

$$S = \frac{1}{2} \cdot 2 \cdot 2 = 2.$$

Наша задача — оценить эту площадь с помощью случайных точек.

Средняя высота по четырём точкам

Выберем четыре случайные точки на отрезке $[0, 2]$ и найдём высоту $f(x_i)$ в каждой из них.



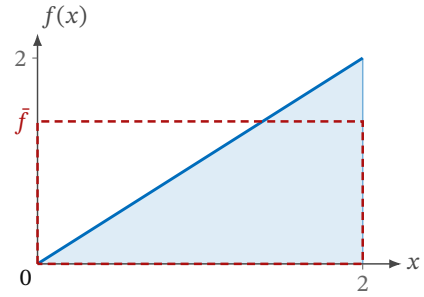
Точка	x_i	$f(x_i)$
1	1,5479	1,5479
2	0,8778	0,8778
3	1,7172	1,7172
4	1,3947	1,3947

Поскольку $f(x) = x$, высота равна координате.

Найдём **среднюю высоту**:

$$\bar{f} = \frac{1{,}5479 + 0{,}8778 + 1{,}7172 + 1{,}3947}{4} \approx 1{,}3844$$

От средней высоты к площади



Мы нашли среднюю высоту четырёх точек:

$$\bar{f} \approx 1{,}3844.$$

Умножим её на ширину отрезка:

$$\hat{I} = 2\bar{f} \approx 2{,}7688.$$

Точная площадь равна 2. Наша оценка получилась больше, потому что большинство точек попало в правую часть отрезка.

Для любой функции на отрезке $[a, b]$ используем тот же принцип:

$$\int_a^b f(x)\,dx \approx (b-a)\frac{1}{N}\sum_{i=1}^{N} f(x_i)$$

Интеграл ≈ длина отрезка × среднее значение функции.

Тот же треугольник в Python

Теперь возьмём $100\,000$ случайных точек и вычислим площадь тем же способом.

```python
import numpy as np

rng = np.random.default_rng(42)
a, b = 0, 2
N = 100000

x = rng.uniform(a, b, N)
heights = x
I_mc = (b - a) * heights.mean()

print(f"{I_mc:.6f}")
```

Что делает программа?

- `uniform(a, b, N)` — выбирает N случайных точек на $[a, b]$.
- `heights.mean()` — находит среднюю высоту.
- `(b-a)` — ширина отрезка.

Результат:

$$\hat{I} \approx 2{,}0025.$$

Точная площадь равна 2.

Чем больше случайных точек, тем ближе обычно оценка к точному значению.

Как вычислить интеграл от другой функции?

Теперь попробуем вычислить интеграл от более сложной функции:

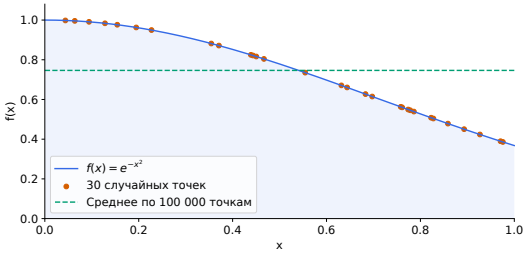
$$I = \int_0^1 e^{-x^2}\, dx.$$

```python
import numpy as np

rng = np.random.default_rng(42)
a, b = 0, 1
N = 100000
x = rng.uniform(a, b, N)

heights = np.exp(-x**2)
I_mc = (b - a) * heights.mean()

print(f"{I_mc:.6f}")
```

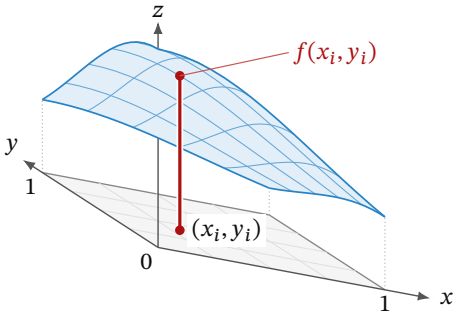


Точки показывают значения функции. Пунктир — среднюю высоту по всем 100 000 точкам.

Результат: $\hat{I} \approx 0{,}746396$. **Точное значение:** $I \approx 0{,}746824$.

Интеграл как объём под поверхностью

В одномерном случае мы умножали среднюю высоту на длину отрезка.
Теперь рассмотрим функцию двух переменных.



Возьмём функцию

$$f(x, y) = e^{-(x^2+y^2)}.$$

Выберем случайные точки (x_i, y_i) в единичном квадрате.
Для каждой точки найдём высоту $f(x_i, y_i)$.

Объём под поверхностью можно оценить по средней высоте.

$$\int_0^1 \int_0^1 f(x, y)\, dx\, dy \approx S \frac{1}{N} \sum_{i=1}^{N} f(x_i, y_i)$$

Площадь основания $S = 1$, поэтому интеграл равен примерно средней высоте.

Двумерный интеграл по 200 000 точкам

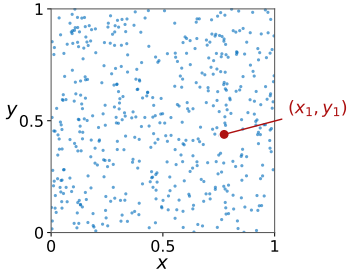
Равномерно выберем 200 000 случайных точек в единичном квадрате и вычислим значения функции $f(x, y) = e^{-(x^2+y^2)}$.

```python
import numpy as np

rng = np.random.default_rng(42)
N = 200000
points = rng.random((N, 2))

r2 = np.sum(points**2, axis=1)
heights = np.exp(-r2)
I_mc = heights.mean()

print(f"{I_mc:.5f}")
```



Первые 500 точек. Первая точка выделена красным.

`points` — массив из N строк и двух столбцов: (x_i, y_i).
`axis=1` — суммируем квадраты координат в каждой строке.

Результат: $\hat{I}_2 \approx 0{,}55787$. Площадь основания равна 1.

От двумерного интеграла к шестимерному

Теперь рассмотрим интеграл от функции шести переменных:

$$f(\mathbf{x}) = e^{-(x_1^2+x_2^2+\cdots+x_6^2)}, \qquad x_i \in [0,1].$$

```python
import numpy as np

rng = np.random.default_rng(42)
N = 200000
points = rng.random((N, 6))

r2 = np.sum(points**2, axis=1)
heights = np.exp(-r2)
I_mc = heights.mean()

print(f"{I_mc:.5f}")
```

`points` содержит $200\,000$ строк и 6 столбцов.

Каждая строка задаёт одну точку в шестимерном пространстве.

Объём области равен $1^6 = 1$, поэтому достаточно найти среднее значение функции.

Результат:

$$\hat{I}_6 \approx 0{,}17360.$$

Преимущество метода: на сетке со 100 точками по каждой оси потребовалось бы $100^6 = 10^{12}$ точек. Здесь мы используем только $200\,000$ случайных точек.

1. Основы метода Монте-Карло
2. Интегрирование методом Монте-Карло
3. **Случайное блуждание и диффузия**
4. Теплоперенос поперёк плёнки

Как построить случайную траекторию?

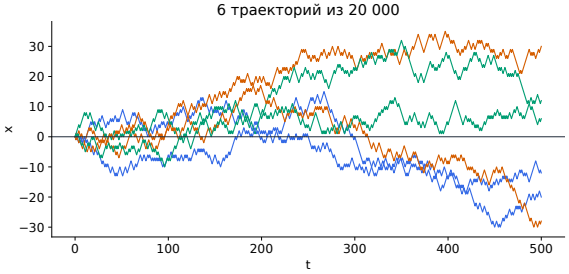
Представим частицу, которая движется вдоль одной прямой.

Начальная координата:

$$x_0 = 0.$$

Через каждый интервал времени Δt частица делает один шаг длиной ℓ:

$$x_{n+1} = x_n \pm \ell.$$



Шесть случайных траекторий из одной начальной точки.

- Вправо — вероятность $1/2$.
- Влево — вероятность $1/2$.
- Каждый шаг выбирается независимо.

Даже при одинаковых условиях частицы могут пройти разные пути.

Что происходит после двух шагов?

Пусть $\ell = 1$. После двух шагов возможны четыре пути, каждый с вероятностью $1/4$.

Путь	Координата x_2	Квадрат смещения x_2^2
$\leftarrow\leftarrow$	-2	4
$\leftarrow\rightarrow$	0	0
$\rightarrow\leftarrow$	0	0
$\rightarrow\rightarrow$	2	4

Средняя координата:

$$\langle x_2 \rangle = \frac{-2+0+0+2}{4} = 0.$$

Средний квадрат смещения (MSD):

$$\mathrm{MSD} = \langle x_2^2 \rangle = \frac{4+0+0+4}{4} = 2.$$

Средняя координата равна нулю, но частицы уже разошлись от начальной точки.

Почему растёт средний квадрат смещения?

После первого шага вправо частица находится в точке $x = 1$, поэтому $x^2 = 1$.

Шаг влево: $x = 0$, изменение x^2 равно -1.

Шаг вправо: $x = 2$, изменение x^2 равно $+3$.

$$\text{Среднее изменение} = \frac{-1 + 3}{2} = 1.$$

Для любого положения x и длины шага ℓ:

$$\frac{(x - \ell)^2 + (x + \ell)^2}{2} = x^2 + \ell^2.$$

Каждый шаг увеличивает средний квадрат смещения на ℓ^2.

$$\mathrm{MSD}(n) = \langle x_n^2 \rangle = n\ell^2, \qquad x_0 = 0.$$

Как найти коэффициент диффузии?

За время t частица делает $n = t/\Delta t$ шагов. Поэтому

$$\mathrm{MSD}(t) = n\ell^2 = \frac{\ell^2}{\Delta t}\, t.$$

Для одномерной диффузии:

$$\mathrm{MSD}(t) = 2Dt.$$

Сравнивая две формулы, получаем коэффициент диффузии:

$$D = \frac{\ell^2}{2\Delta t}.$$

Например, при $\ell = 1$ и $\Delta t = 1$:

$$D = 0{,}5.$$

Проверка: после 500 шагов ожидаем $\mathrm{MSD} = 500$.

Моделируем 20 000 траекторий в Python

```python
import numpy as np

rng = np.random.default_rng(42)
M, N = 20000, 500
ell, dt = 1.0, 1.0

steps = rng.choice([-ell, ell], size=(M, N))
x = np.cumsum(steps, axis=1)
msd = (x**2).mean(axis=0)

t = np.arange(1, N + 1) * dt
D_mc = msd[-1] / (2 * t[-1])
print(f"{D_mc:.5f}")
```

Основные шаги:

`choice` — случайно выбираем шаги влево или вправо.

`cumsum` — находим положение после каждого шага.

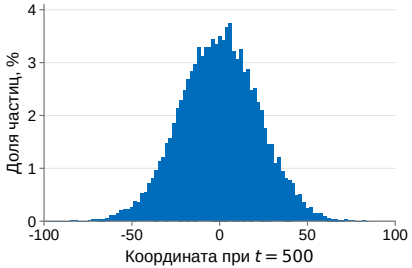
`mean(axis=0)` — вычисляем MSD по всем частицам.

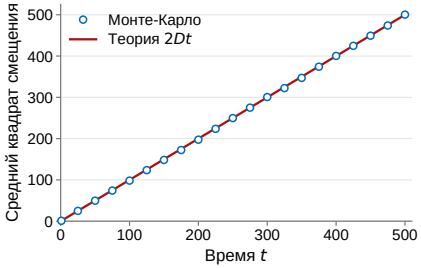
Результат:

$$\widehat{D} \approx 0{,}50032.$$

В массиве $20\,000$ **строк (частицы) и** 500 **столбцов (шаги).**

Что показало моделирование?





Распределение частиц
После 500 шагов большинство частиц находится около нуля, но некоторые ушли далеко от начальной точки.

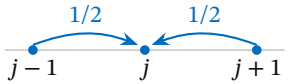
Средний квадрат смещения
MSD растёт почти линейно со временем:

$$\mathrm{MSD}(500) \approx 500{,}32.$$

Результат: $\widehat{D} \approx 0{,}50032$, **Теория:** $D = 0{,}5$.

От случайных шагов к уравнению диффузии

Пусть p_j^n — вероятность найти частицу в узле j после n шагов.



В точку j частица может прийти слева или справа:

$$p_j^{n+1} = \frac{p_{j-1}^n + p_{j+1}^n}{2}.$$

Вычтем p_j^n и разделим на время одного шага:

$$\frac{p_j^{n+1} - p_j^n}{\Delta t} = \underbrace{\frac{\ell^2}{2\Delta t}}_{D} \frac{p_{j+1}^n - 2p_j^n + p_{j-1}^n}{\ell^2}.$$

При $\ell, \Delta t \to 0$ и постоянном D получаем:

$$\frac{\partial \rho}{\partial t} = D \frac{\partial^2 \rho}{\partial x^2}.$$

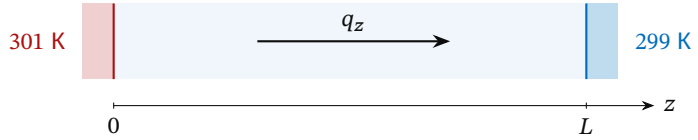
Случайное движение частиц приводит к уравнению диффузии.

1. Основы метода Монте-Карло
2. Интегрирование методом Монте-Карло
3. Случайное блуждание и диффузия
4. **Теплоперенос поперёк плёнки**

Как тепло проходит через кремниевую плёнку?

В кремнии тепло переносится главным образом фононами — колебаниями кристаллической решётки.

Рассмотрим плёнку толщиной $L = 200$ нм. Слева поддерживаем 301 К, справа — 299 К.



Температура меняется только вдоль оси z, поэтому рассмотрим одномерную задачу.

Наша задача: найти температуру $T(z)$ и тепловой поток q_z через плёнку.

Как движется фононный пакет?

В отличие от случайного блуждания, фононный пакет движется с постоянной скоростью до следующего рассеяния.

Время между рассеяниями выбирается случайно.

Объёмная теплоёмкость	$C = 1{,}6077 \cdot 10^6$ Дж/(м^3 · К)
Скорость пакета	$\upsilon = 6400$ м/с $= 6{,}4$ нм/пс
Средняя длина свободного пробега	$\Lambda = 43{,}7$ нм

В **серой модели** все фононные пакеты имеют одинаковую скорость и среднее время рассеяния.

$$\tau = \frac{\Lambda}{\upsilon} \approx 6{,}83 \text{ пс.}$$

Толщина плёнки $L = 200$ **нм — примерно** $4{,}6$ **средних свободных пробега.**

Зачем нужны пакеты со знаком?

Вместо всей энергии будем моделировать только её изменение относительно $T_0 = 300$ К.

При 300 К встречные потоки энергии компенсируют друг друга.

Термостат	По сравнению с 300 К	Знак пакета
Слева, 301 К	Энергии больше	+
Справа, 299 К	Энергии меньше	−

Положительный пакет — избыток энергии.
Отрицательный пакет — недостаток энергии.

Знак показывает **изменение энергии**, а не направление движения.

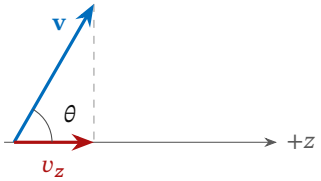
При рассеянии направление может измениться, но знак остаётся прежним.

Как выбрать направление движения?

После рассеяния фононный пакет может двигаться в любом направлении с одинаковой вероятностью.

Нас интересует скорость вдоль толщины плёнки:

$$v_z = v\cos\theta = v\mu, \qquad \mu = \cos\theta.$$



Для равномерного распределения направлений величина μ равномерно распределена от -1 до 1.

$$\mu = 2u - 1, \qquad u \in (0,1).$$

Например, при $\mu = 0{,}5$ получаем $v_z = 3{,}2$ нм/пс.

Как пакеты выходят с поверхности?

Внутри материала все направления равновероятны. Но через поверхность чаще проходят пакеты, движущиеся ближе к нормали.

1. Направление движения

Для $m = |\mu|$ распределение направлений выходящего потока:

$$p(m) = 2m, \qquad 0 \le m \le 1.$$

Для случайного $u \in (0, 1)$ получаем:

$$\mu_L = +\sqrt{u}, \qquad \mu_R = -\sqrt{u}.$$

2. Энергия пакета

Мощность источника относительно $T_0 = 300$ К:

$$J_b = \frac{Cv}{4}(T_b - T_0).$$

Если с каждой границы запускаем N_b траекторий, вес одной траектории:

$$w_b = \frac{J_b}{N_b}.$$

Как выбрать время до рассеяния?

В нашей модели вероятность рассеяния за короткое время dt равна dt/τ.

Вероятность не рассеяться за время t:

$$S(t) = e^{-t/\tau}.$$

Чтобы получить случайное время полёта, выбираем $u \in (0, 1)$:

$$t_{\text{sc}} = -\tau \ln u.$$

Например, при $u = 0{,}5$ и $\tau = 6{,}83$ пс:

$$t_{\text{sc}} \approx 4{,}73 \text{ пс}.$$

Важно: пакет может достичь границы плёнки раньше, чем произойдёт рассеяние.

Что произойдёт раньше: рассеяние или выход?

У пакета есть два возможных события: рассеяние внутри плёнки или выход через границу.

Время до ближайшей границы:

$$t_{\text{гр}} = \begin{cases} (L - z)/(v\mu), & \mu > 0, \\ z/(v|\mu|), & \mu < 0, \\ \infty, & \mu = 0. \end{cases}$$

Выбираем событие, которое произойдёт раньше:

$$h = \min(t_{\text{sc}}, t_{\text{гр}}), \qquad z' = z + v\mu h.$$

- $t_{\text{sc}} < t_{\text{гр}}$: пакет рассеивается, выбираем новое направление.
- $t_{\text{гр}} \leq t_{\text{sc}}$: пакет достигает границы и поглощается.

Пример: два полёта у холодной границы

Пакет находится в точке $z = 160$ нм. До холодной поверхности осталось 40 нм.

Начало	μ	t_{sc}	$t_{\text{гр}}$	Результат
160 нм	+0,50	5 пс	12,5 пс	Рассеяние при 176 нм
176 нм	+0,75	8 пс	5 пс	Выход при 200 нм

Первый полёт: рассеяние происходит раньше.

$$z' = 160 + 6{,}4 \cdot 0{,}5 \cdot 5 = 176 \text{ нм}.$$

Второй полёт: пакет достигает границы раньше.

$$z' = 176 + 6{,}4 \cdot 0{,}75 \cdot 5 = 200 \text{ нм}.$$

Пакет достиг холодного термостата. Его положительный знак сохранился.

Один полёт в Python

Для каждого пакета определяем, что произойдёт раньше: рассеяние или выход через границу.

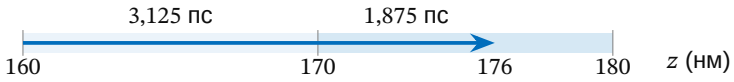
```
free_time = rng.exponential(RELAXATION_TIME, size=z.size)
exits = wall_time <= free_time
duration = np.minimum(free_time, wall_time)
end = np.where(exits, wall, np.clip(z + velocity * duration, 0.0, length))
add_residence_time(dwell, z, end, duration, cell_width)
```

- `exponential` — случайное время до рассеяния.
- `minimum` — выбираем меньшее из двух времён.
- `where` — определяем, где закончился полёт.
- `add_residence_time` — записываем время в каждой ячейке.

Один полёт — две ячейки

Разделим плёнку на 20 ячеек по 10 нм.

Пакет летит от 160 до 176 нм со скоростью $v_z = 3{,}2$ нм/пс.



Время пребывания в каждой ячейке:

$$t_1 = \frac{10}{3{,}2} = 3{,}125 \text{ пс}, \qquad t_2 = \frac{6}{3{,}2} = 1{,}875 \text{ пс}.$$

Общее время полёта:

$$t_1 + t_2 = 5 \text{ пс}.$$

Граница ячейки нужна только для расчёта. При её пересечении пакет продолжает движение.

Как получить температуру из времени пребывания?

В стационарном режиме фононные пакеты непрерывно поступают из термостатов.

Чем дольше пакет находится в ячейке, тем больше его вклад в среднюю энергию этой ячейки.

Изменение энергии на единицу площади:

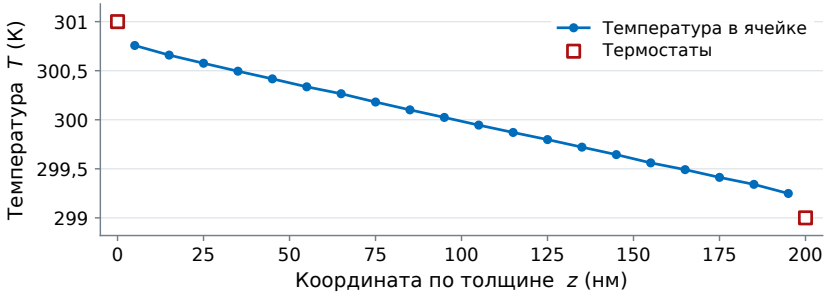
$$\sum_p w_p t_{pi} = C\,\Delta z\,(T_i - T_0).$$

Отсюда получаем температуру ячейки:

$$T_i = T_0 + \frac{\sum_p w_p t_{pi}}{C\,\Delta z}.$$

Положительные пакеты увеличивают энергию, отрицательные — уменьшают.

Как выглядит профиль температуры?



Квадраты — заданные температуры термостатов: 301 и 299 К.

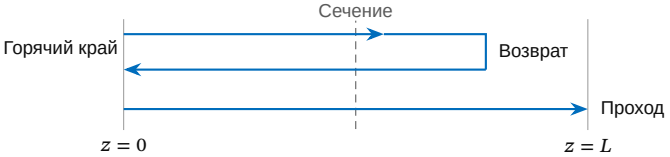
Синие точки — средние температуры внутри ячеек плёнки.

Температура уменьшается от горячей поверхности к холодной.

Как посчитать тепловой поток через сечение?

Выберем сечение внутри плёнки и будем считать пересечения фононных пакетов.

- Слева направо — записываем $+1$.
- Справа налево — записываем -1.



Если пакет вернулся на исходную сторону, встречные пересечения компенсируют друг друга.

Для каждого пересечения учитываем также вес пакета w_p:

$$q_z = \sum_p w_p \left(N_p^+ - N_p^-\right).$$

Тепловой поток определяется направлением пересечения и знаком веса пакета.

Тепловой поток по числу прошедших пакетов

Запустим по 50 000 пакетов с каждой стороны и посчитаем, сколько из них достигло противоположной границы.

1. Доля прошедших пакетов

$$p_{LR} = \frac{11\,105}{50\,000} = 0{,}2221, \qquad p_{RL} = \frac{11\,100}{50\,000} = 0{,}2220.$$

2. Мощность источников относительно 300 К

$$J_L = +2{,}57232 \cdot 10^9, \qquad J_R = -2{,}57232 \cdot 10^9 \quad \text{Вт/м}^2.$$

3. Итоговый тепловой поток

$$q_z = J_L p_{LR} - J_R p_{RL} \approx 1\,142{,}4 \text{ МВт/м}^2.$$

Оба источника дают положительный вклад в поток от горячей стороны к холодной.

Эффективная теплопроводность плёнки

Мы нашли тепловой поток q_z. Теперь рассчитаем эффективную теплопроводность плёнки:

$$\kappa_{\text{eff}} = \frac{q_z L}{T_L - T_R}.$$

При $L = 200$ нм и $T_L - T_R = 2$ К получаем:

$$\kappa_{\text{eff}} \approx 114{,}2 \ \frac{\text{Вт}}{\text{м} \cdot \text{К}}.$$

Для объёмного кремния в той же серой модели:

$$\kappa_{\text{bulk}} = \frac{C v \Lambda}{3} \approx 149{,}9 \ \frac{\text{Вт}}{\text{м} \cdot \text{К}}.$$

Теплопроводность тонкой плёнки ниже из-за влияния границ.

Итоги лекции

В методе Монте-Карло важно не только получить случайные числа, но и выбрать, **что именно нужно усреднять**.

Задача	**Что усредняем?**
Площадь и вероятность	Попадания случайных точек (0 или 1).
Интегрирование	Значения функции в случайных точках.
Диффузия	Квадраты смещений частиц.
Теплоперенос	Время пребывания и пересечения сечения с учётом веса пакетов.

Случайные испытания ⟶ статистика ⟶ оценка результата.