

# Лекция 2

# Основы метода Монте-Карло

Лю Шисян

Кафедра теплофизики (Э6)

# План лекции

| | |
|---|---|
| **2.1 Основы метода** | Свяжем точность оценки вероятности с числом точек. |
| **2.2 Интегрирование** | Найдём интеграл по значениям функции в случайных точках. |
| **2.3 Блуждание и диффузия** | Определим коэффициент диффузии по смещениям частиц. |
| **2.4 Теплоперенос в плёнке** | Рассчитаем профиль температуры в плёнке и тепловой поток между её поверхностями. |

**Репозиторий кода**

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

# 2.1 Основы метода Монте-Карло

Оценим вероятность по доле попаданий и разберёмся, как число случайных точек влияет на точность оценки.

# От иглы Бюффона к методу Монте-Карло

**1777** Бюффон связал вероятность пересечения иглой одной из параллельных линий с числом $\pi$. По доле пересечений при множестве бросков можно оценить $\pi$.

**1946** Раскладывая пасьянс, Станислав Улам задумался о шансах на выигрыш. Он предложил много раз перетасовать карты и подсчитать долю удачных партий.

**1948** Идею Улама и Джона фон Неймана применили на компьютере ENIAC: с помощью случайной выборки исследователи моделировали движение нейтронов.

**1949** Николас Метрополис и Улам опубликовали статью «Метод Монте-Карло». Название предложил Метрополис в честь Монте-Карло, знаменитого своим казино.

Источники Бюффон (1777) Статья 1949 года Воспоминания Метрополиса (1987)

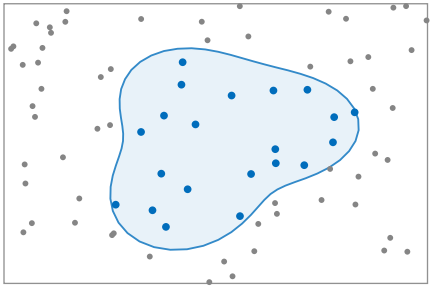
# Идея метода Монте-Карло

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

Равномерно выберем 80 точек на всём участке, каждую независимо от остальных. Тогда вероятность попасть в воду равна доле участка под озером.

В озеро попала **21 точка из 80**. Умножим долю попаданий на площадь участка.

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



21 точка в воде  59 точек на суше

Мы оценили площадь озера по результатам многих случайных испытаний. Это первый пример метода Монте-Карло.

# Почему доля попаданий равна среднему?

Выбор одной точки — одно **испытание**. Все 80 выбранных точек образуют **выборку**. Для каждой точки запишем, попала ли она в озеро.

**Точка в воде** Запишем 1. Таких точек 21.

**Точка на суше** Запишем 0. Таких точек 59.

Сложим все записи и разделим на число точек:

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

В сумме каждая единица учитывает одно попадание. Значит, среднее нулей и единиц даёт долю попаданий.

В Python результат проверки записывается как `True` или `False`. При подсчёте среднего это те же 1 и 0.

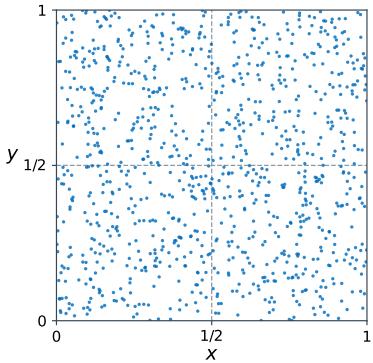
# Как получить случайную точку в квадрате

Две координаты выбираем независимо и равномерно на $[0, 1)$. Числа с одинаковым индексом в двух массивах задают одну точку.

```
import numpy as np

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

Первая точка `(x[0], y[0])` имеет координаты примерно $(0{,}774;\ 0{,}062)$.



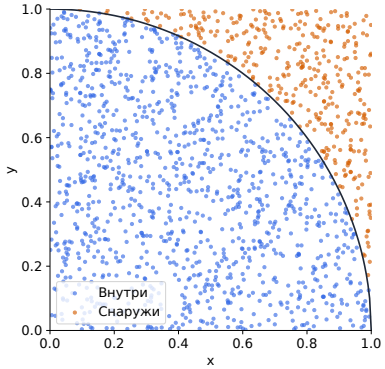

Площади четырёх ячеек равны. Вероятность попасть в каждую равна $1/4$. В нижнюю левую ячейку попала 251 точка.

# Проверим метод на фигуре известной площади

Поместим четверть круга радиуса 1 в квадрат со стороной 1. Площадь квадрата равна 1, всего круга — $\pi$, а четверти круга — $\pi/4$.

В четверть круга попадает в среднем доля $\pi/4$ всех точек. Значит, чтобы оценить $\pi$, умножим долю попаданий на 4.

$$\widehat{\pi} = 4\,\frac{\text{число попаданий}}{\text{число точек}}.$$




На рисунке первые 1500 точек выборки. Дуга отделяет точки внутри четверти круга от точек снаружи.

# Как определить, попала ли точка внутрь?

Центр круга — начало координат. Квадрат расстояния до центра равен $x^2 + y^2$. Для единичного круга точка внутри или на границе, если $x^2 + y^2 \leq 1$.

| $x$ | $y$ | $x^2 + y^2$ | Результат проверки |
|---|---|---|---|
| 0,2 | 0,3 | $0{,}04 + 0{,}09 = 0{,}13$ | Внутри: `True` |
| 0,8 | 0,8 | $0{,}64 + 0{,}64 = 1{,}28$ | Снаружи: `False` |
| 0,6 | 0,8 | $0{,}36 + 0{,}64 = 1$ | На границе: `True` |

Для этих трёх точек получили два попадания. В NumPy ту же проверку можно сразу применить ко всем парам координат.

# Оценка $\pi$ по 100 000 точкам

```python
import numpy as np

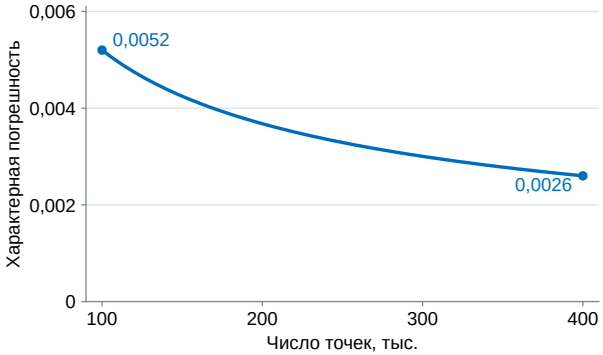
rng = np.random.default_rng(42)
N = 100000
x = rng.random(N)
y = rng.random(N)
hit = x**2 + y**2 <= 1
pi_mc = 4 * hit.mean()
print(hit.sum(), pi_mc)
```

В массиве `hit` хранится результат проверки для каждой точки. Команда `hit.sum()` считает попадания, а `hit.mean()` даёт их долю.

Из 100 000 точек внутрь попали 78 444. Получаем

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

# Как число точек влияет на точность?




По мере роста числа случайных точек $N$ доля попаданий приближается к $\pi/4$, а оценка числа $\pi$ становится точнее.

Для независимых точек характерная погрешность оценки убывает как $1/\sqrt{N}$.

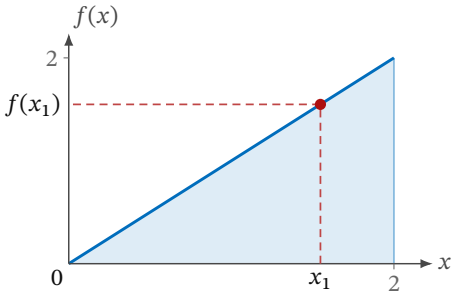
При переходе от 100 тысяч точек к 400 тысячам характерная погрешность уменьшается вдвое.

## 2.2 Интегрирование по выборке

Свяжем площадь под кривой со средней высотой функции. Затем применим тот же способ к интегралу по многомерной области.

# Что записать в выбранной точке?

В задаче об озере для каждой точки мы записывали 1 или 0. Теперь выберем на оси случайную координату $x_i$ и запишем значение функции $f(x_i)$.




Возьмём прямую $f(x) = x$ на отрезке $[0, 2]$. Длина вертикального отрезка от оси до прямой равна $f(x_i)$.

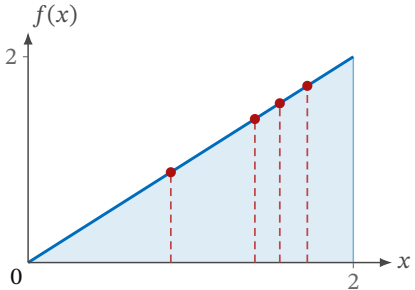
Под прямой лежит треугольник. Его площадь известна:

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

Оценим ту же площадь по нескольким измерениям высоты.

# Средняя высота по четырём точкам

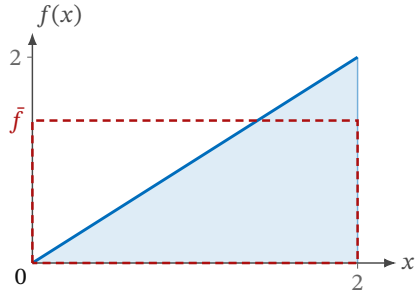
Выберем четыре координаты независимо и равномерно на $[0, 2]$. Поскольку $f(x) = x$, высота в каждой точке равна её координате.




| 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 |

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

# От средней высоты к площади




Сумма площадей узких полос одинаковой ширины равна общей ширине, умноженной на среднюю высоту полос. По четырём измерениям получаем прямоугольник площадью

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

Точная площадь равна $2$. Три из четырёх точек лежат правее середины, где прямая выше.

Для равномерной выборки на $[a, b]$ используем то же правило:

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

# Тот же треугольник в Python

Теперь возьмём $100\,000$ точек. Вычислим их среднюю высоту и умножим её на ширину отрезка, как при ручном расчёте.

```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}")
```

x хранит случайные координаты. Для прямой $f(x) = x$ они же задают высоты.

`heights.mean()` даёт среднюю высоту $1{,}00125$.

Ширина равна $2$. Программа выводит `2.002500`. Из геометрии мы получили $2$.

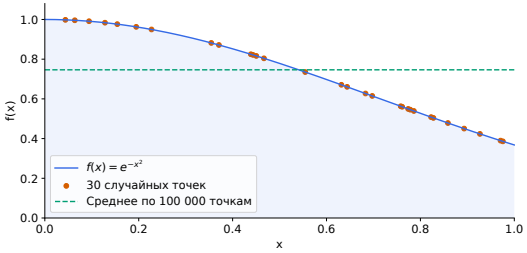
# Как вычислять высоты для другой кривой?

Возьмём $f(x) = e^{-x^2}$ на $[0, 1]$. Теперь для каждой координаты $x_i$ нужно вычислить высоту $e^{-x_i^2}$.

```python
import numpy as np

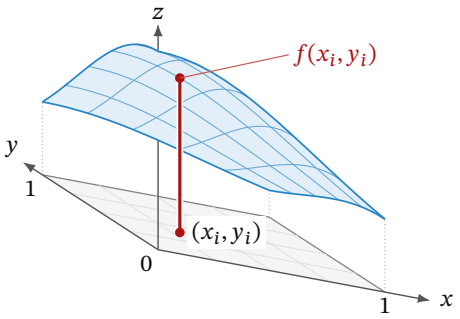
rng = np.random.default_rng(42)
a, b = 0, 1
N = 100000
x = rng.uniform(a, b, N)
heights = np.exp(-x**2)
I_mc = (b - a) * heights.mean()
print(f"{I_mc:.6f}")
```

При $N = 100\,000$ получаем $0{,}746396$. Значение интеграла с точностью до шестого знака равно $0{,}746824$.




На графике показаны первые 30 точек. Пунктир отмечает среднюю высоту по всем $100\,000$ точкам.

# Интеграл как объём под поверхностью




Для кривой $e^{-x^2}$ мы умножали среднюю высоту на длину отрезка. Над квадратом высоту задаёт функция двух координат

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

Выберем $x$ и $y$ независимо и равномерно на $[0, 1]$. Над точкой $(x, y)$ поднимемся до поверхности и запишем одну высоту $f(x, y)$.

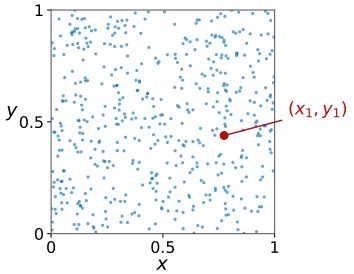
Объём под поверхностью оцениваем как **площадь основания, умноженную на среднюю высоту**. У единичного квадрата площадь равна 1.

# Двумерный интеграл по 200 000 точкам

Равномерно выберем $200\,000$ точек в единичном квадрате. Каждая строка массива задаёт точку $(x_i, y_i)$, для которой вычислим высоту $f_i = e^{-(x_i^2+y_i^2)}$.

```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)
print(f"{heights.mean():.5f}")
```




Первые 500 точек выборки. Первая точка выделена красным.

Параметр `axis=1` задаёт сумму квадратов по каждой строке. Затем `heights.mean()` усредняет все $200\,000$ высот. Площадь основания равна 1, поэтому $\hat{I}_2 \approx 0{,}55787$.

# Шесть координат у каждой точки

В единичном шестимерном кубе вычислим интеграл от $f(\mathbf{x}) = e^{-(x_1^2+\cdots+x_6^2)}$.

```
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)
print(f"{heights.mean():.5f}")
```

В `points` содержится $N$ строк. В каждой строке шесть случайных координат от 0 до 1.

Из каждой строки получаем одно значение функции. Объём куба равен $1^6 = 1$, поэтому среднее по $N$ значениям даёт оценку интеграла $0{,}17360$.

Для проверки возьмём одномерный интеграл со слайда 17. Функция распадается на шесть множителей, поэтому $I_6 = I_1^6 \approx (0{,}746824)^6 \approx 0{,}17350$.

## 2.3 Случайное блуждание и диффузия

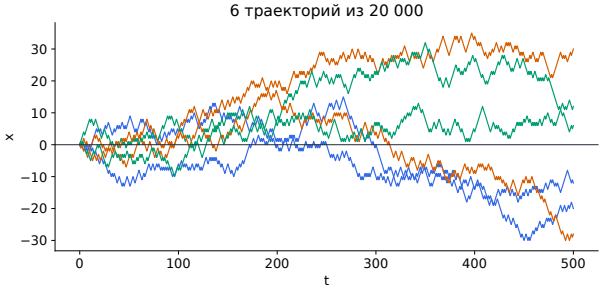
Проследим, как частицы расходятся от места старта. По их смещениям определим коэффициент диффузии.

# Как построить траекторию частицы

Все частицы стартуют в точке $x_0 = 0$.
Через каждый интервал времени $\Delta t$ частица делает шаг длиной $\ell$.

$$x_{n+1} = x_n + \ell \quad \text{или} \quad x_{n+1} = x_n - \ell.$$

Оба направления равновероятны.
Направление каждого следующего шага выбирается независимо от предыдущих.




Шесть траекторий с общим началом.

# Что измерять после двух шагов?

При $\ell = 1$ у каждого из четырёх путей вероятность $1/4$. Поскольку частицы вышли из нуля, конечная координата равна смещению от старта.

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

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

Средняя координата показывает положение центра облака. **Средний квадрат смещения** (MSD) показывает, насколько частицы разошлись от старта.

# Почему растёт средний квадрат смещения?

Возьмём два пути из предыдущей таблицы, которые начинаются шагом вправо. После первого шага частица стоит в $x = 1$, и квадрат её смещения равен $1$.

**Шаг обратно к нулю.** Квадрат смещения меняется с $1$ до $0$. Изменение равно $-1$.

**Ещё один шаг вправо.** Частица попадает в $x = 2$. Квадрат смещения меняется с $1$ до $4$. Изменение равно $+3$.

$$\text{Средняя прибавка} = \tfrac{1}{2}(-1) + \tfrac{1}{2}(+3) = 1.$$

Для частицы в любой точке $x$ при длине шага $\ell$ получаем

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

В среднем каждый шаг добавляет $\ell^2$. После $n$ шагов из нуля получаем $\langle x_n^2 \rangle = n\ell^2$.

# От числа шагов к коэффициенту диффузии

За время $t$ частица делает $n = t/\Delta t$ шагов. Подставим это число в закон роста среднего квадрата смещения. Получаем

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

В одном измерении эту зависимость записывают как $\mathrm{MSD}(t) = 2Dt$. Половина наклона прямой даёт **коэффициент диффузии**

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

При $\ell = 1$ и $\Delta t = 1$ ожидаем $D = 0{,}5$. Через 500 шагов средний квадрат смещения должен быть близок к $500$ при моделировании большого числа частиц.

# Как смоделировать 20 000 траекторий

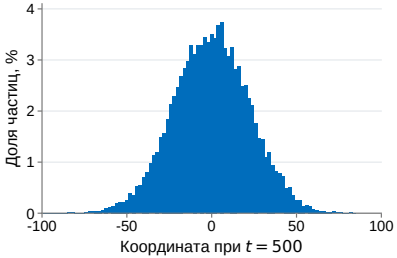
```python
import numpy as np

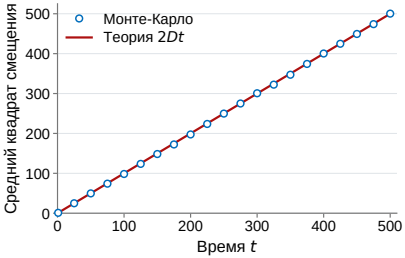
rng = np.random.default_rng(42)
M, N = 20000, 500
ell, dt = 1.0, 1.0
steps = rng.choice([-ell, ell], size=(M, N))
x = np.cumsum(steps, axis=1)
msd = (x**2).mean(axis=0)
t = np.arange(1, N + 1) * dt
D_mc = msd[-1] / (2 * t[-1])
print(f"{D_mc:.5f}")
```

Одна строка массива хранит шаги одной частицы. `cumsum` складывает их вдоль строки и даёт координаты.

Один столбец хранит координаты всех частиц в один момент. Усредняя их квадраты по столбцу, получаем MSD в этот момент.

# Что показывают 20 000 траекторий



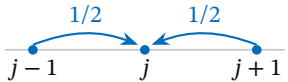




**После 500 шагов** большинство частиц оказалось около нуля. Далеко вправо или влево ушли немногие.

**За время наблюдения** MSD растёт почти по прямой. При $t = 500$ получаем $500{,}3224$, откуда $\widehat{D} \approx 0{,}50032$. Прогноз модели — $0{,}5$.

# Как возникает уравнение диффузии?

$p_j^n$ —вероятность найти частицу в узле $j$ после $n$ шагов. В этот узел попадает половина вероятности из каждого соседнего узла.




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

Вычтем $p_j^n$ и разделим на время шага $\Delta t$. Обозначим расстояние между узлами через $\ell$.

$$\underbrace{\frac{p_j^{n+1} - p_j^n}{\Delta t}}_{\text{изменение за единицу времени}} = \underbrace{\frac{\ell^2}{2\Delta t}}_{D} \underbrace{\frac{p_{j+1}^n - 2p_j^n + p_{j-1}^n}{\ell^2}}_{\text{дискретная кривизна}}.$$

Усредним распределение по соседним узлам. При $\ell, \Delta t \to 0$ и постоянном $D$ для гладкой плотности вероятности $\rho$ получаем

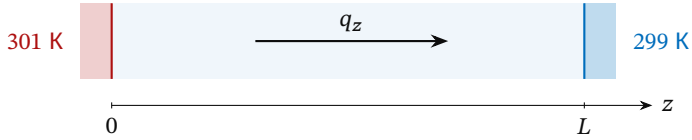
$$\frac{\partial \rho}{\partial t} = D\frac{\partial^2 \rho}{\partial x^2}.$$

# 2.4 Теплоперенос поперёк плёнки

По траекториям фононных пакетов найдём распределение температуры и определим, сколько тепла проходит через плёнку за секунду.

# Теплоперенос через кремниевую плёнку

Два термостата поддерживают на поверхностях плёнки температуры 301 и 299 К. Толщина $L = 200$ нм. Определим стационарный профиль $T(z)$ и плотность теплового потока $q_z$.




В серой модели у всех пакетов одинаковы скорость $v = 6{,}4$ нм/пс и средняя длина свободного пробега $\Lambda = 43{,}7$ нм. Среднее время между рассеяниями $\tau = \Lambda / v \approx 6{,}83$ пс.

# Что означает знак пакета

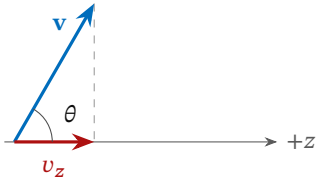
При одинаковой температуре $T_0 = 300$ К с обеих сторон в плёнку поступают равные встречные потоки энергии. Будем отсчитывать изменение мощности каждого источника от этого равновесия.

| Термостат | Что изменилось по сравнению с 300 К | Вес пакета |
|---|---|---|
| Левый, 301 К | Вправо поступает больше энергии | Положительный |
| Правый, 299 К | Влево поступает меньше энергии | Отрицательный |

Если с границы $b$ выпускаем $N_b$ пакетов, каждому присваиваем вес

$$w_b = J_b/N_b, \qquad J_b = \tfrac{1}{4} C v (T_b - T_0) \quad [\text{Вт/м}^2].$$

# Направление после рассеяния и при испускании




Глубина меняется со скоростью $v_z = v\mu$, где $\mu = \cos\theta$. При $\mu = 0{,}5$ за пикосекунду пакет продвинется по нормали на 3,2 нм.

**Внутри плёнки.** После рассеяния все направления равновероятны. Равномерно выбираем $u \in (0, 1)$ и берём $\mu = 2u - 1$.

**На поверхности.** Чем больше $|v_z|$, тем больше пакетов пересекает поверхность за то же время. При испускании слева берём $\mu = +\sqrt{u}$, справа $\mu = -\sqrt{u}$.

# Сколько длится один полёт

В однородном кремнии частота рассеяния постоянна. Вероятность пролететь время $t$ без рассеяния равна $e^{-t/\tau}$. Выбираем $u$ равномерно между нулём и единицей и получаем $t_{\text{sc}} = -\tau \ln u$.

До поверхности по ходу движения осталось время

$$t_{\text{гр}} = \begin{cases} (L - z)/v_z, & v_z > 0, \\ z/|v_z|, & v_z < 0, \\ \infty, & v_z = 0. \end{cases} \qquad h = \min(t_{\text{sc}}, t_{\text{гр}}), \qquad z' = z + v_z h.$$

Если рассеяние происходит раньше, в точке $z'$ выбираем новое направление. При достижении поверхности пакет поглощается термостатом.

# Два полёта одного пакета в расчёте

Пакет с горячей стороны дошёл до $z = 160$ нм. Посмотрим, как он пройдёт оставшиеся 40 нм до холодной поверхности.

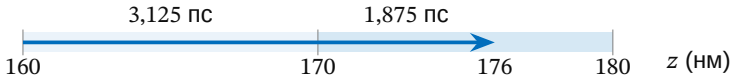
| Старт | $\mu$ | До рассеяния | До поверхности | Конец участка |
|---|---|---|---|---|
| 160 нм | +0,50 | 5 пс | $40/3{,}2 = 12{,}5$ пс | 176 нм, рассеяние |
| 176 нм | +0,75 | 8 пс | $24/4{,}8 = 5$ пс | 200 нм, поглощение |

```
free_time = rng.exponential(RELAXATION_TIME, size=z.size)
exits = wall_time <= free_time
duration = np.minimum(free_time, wall_time)
```

В обеих строках `duration` равно 5 пс. Только во второй строке `exits` принимает значение `True`.

# Время в ячейке определяет её температуру

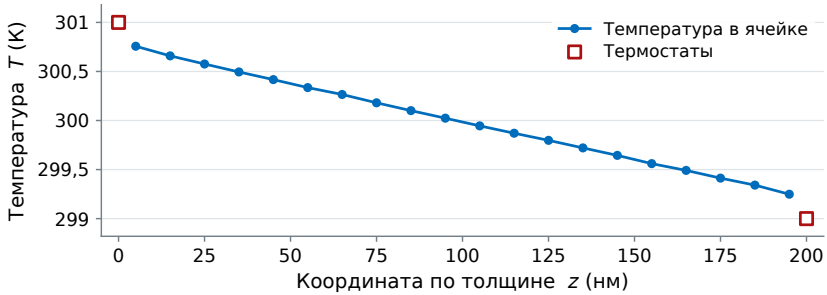
Разобьём плёнку на 20 ячеек толщиной $\Delta z = 10$ нм. На участке от 160 до 176 нм пакет летел со скоростью $v_z = 3{,}2$ нм/пс.




Первые 10 нм он пролетел за 3,125 пс, следующие 6 нм за 1,875 пс. Для каждого пакета $p$ складываем время всех посещений ячейки $i$:

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

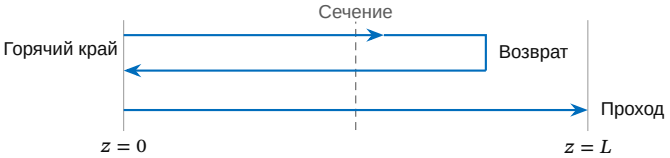
# Профиль температуры по толщине




Квадраты показывают заданные температуры термостатов. Синие точки — температуры ячеек. У первой и последней ячеек центры при 5 и 195 нм, а средние температуры равны 300,76 и 299,25 К.

# Поток через плоскость внутри плёнки

Выберем плоскость, параллельную поверхностям плёнки. Если пакет пересекает её слева направо, запишем $+1$, если справа налево, запишем $-1$.




При возврате на исходную сторону встречные пересечения сокращаются. При прохождении слева направо остаётся $+1$, при прохождении справа налево — $-1$. Каждое число умножаем на вес пакета.

# Тепловой поток и теплопроводность плёнки

Из $50\,000$ пакетов левого термостата до правого дошли $11\,105$. При запуске справа налево прошли $11\,100$ из $50\,000$. С учётом весов получаем

$$q_z = J_L \frac{11\,105}{50\,000} - J_R \frac{11\,100}{50\,000} \approx 1\,142{,}4 \text{ МВт/м}^2.$$

Поскольку $J_L > 0$ и $J_R < 0$, оба слагаемых дают поток вправо. При разности температур термостатов $2$ К и толщине $200$ нм

$$\kappa_{\text{eff}} = \frac{q_z L}{T_L - T_R} \approx 114{,}2 \ \frac{\text{Вт}}{\text{м} \cdot \text{К}}.$$

Для объёмного кремния та же серая модель даёт $C v \Lambda / 3 \approx 149{,}9$ Вт/(м • К).

# Итоги лекции

Сначала выбираем, что усреднять. Чем больше независимых наблюдений, тем точнее обычно оценка.

| Что хотим узнать | Что берём из выборки или траектории |
|---|---|
| Вероятность попадания | Для каждой случайной точки отмечаем попадание в область. Доля попаданий оценивает вероятность. |
| Интеграл | Среднее значение функции умножаем на объём области. |
| Коэффициент диффузии | Усредняем квадрат смещения. В одном измерении наклон графика $\mathrm{MSD}(t)$ равен $2D$. |
| Теплоперенос в плёнке | Температуру находим по времени пребывания пакетов в ячейке, поток — по стороне выхода. В обоих расчётах учитываем вес пакета. |