ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ · ЛЕКЦИЯ 4

Численное решение дифференциальных уравнений

ОДУ, методы Эйлера и Рунге–Кутты, УЧП и Метод конечных разностей

Лю Шисян

Кафедра теплофизики (Э6)
МГТУ им. Н.Э. Баумана
Москва / 2026

Место этой лекции в курсе

Лекция 4 — Численное решение дифференциальных уравнений

Дифференциальные уравнения описывают эволюцию и пространственное распределение физических величин; численные методы позволяют получать их приближённые решения.

После лекции вы сможете:

- решать начальные задачи для обыкновенных дифференциальных уравнений;
- сравнивать методы Эйлера и Рунге–Кутты;
- дискретизировать производные методом конечных разностей;
- строить явную схему для одномерного уравнения теплопроводности;
- оценивать влияние шага по времени, граничных условий и устойчивости.

План

TOC

1

Обыкновенные дифференциальные уравнения

От алгебраической задачи к динамике

В лекции 3

Мы искали отдельные неизвестные:

$$f(x) = 0,$$

или набор неизвестных:

$$A\mathbf{x} = \mathbf{b}.$$

Результатом было число или конечный вектор.

Теперь

Неизвестная величина изменяется во времени:

$$\frac{dy}{dt} = f(t, y)$$

Чтобы определить конкретное решение, задаём начальное состояние:

$$y(t_0) = y_0$$

Тогда ОДУ описывает, как состояние изменяется после момента t_0.

Пример: закон охлаждения Ньютона

Физическая модель

Тело с температурой $T(t)$ охлаждается в окружающей среде с постоянной T_∞. Скорость охлаждения пропорциональна разности температур:

$$\frac{dT}{dt} = -k\,(T - T_\infty)$$

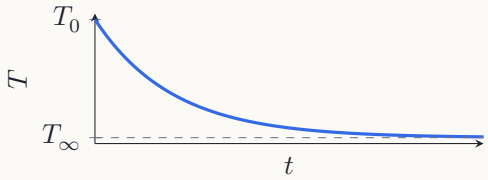
где $k > 0$ — коэффициент охлаждения.

Начальное условие:

$$T(0) = T_0$$

Для этого примера решение известно

$$T(t) = T_\infty + (T_0 - T_\infty)e^{-kt}.$$



Зачем нужен численный метод?

Здесь аналитическое решение известно, поэтому пример удобно использовать для проверки алгоритма. Для более сложных моделей явного решения обычно уже нет.

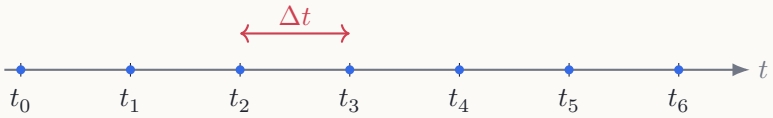
Дискретизация времени

От непрерывного времени к сетке

Непрерывную ось времени заменяем конечным набором узлов:

$$t_n = t_0 + n\Delta t, \qquad n = 0, 1, 2, \dots, N$$

где Δt — шаг по времени.



Численная задача

Вычисляем последовательность: $y_0,\ y_1,\ y_2,\ \dots,\ y_N$, где $y_n \approx y(t_n)$.

Численный метод должен определить, как перейти от y_n к y_{n+1}.

2

Метод Эйлера

Метод Эйлера

Исходная задача

Рассмотрим начальную задачу

$$\frac{dy}{dt} = f(t, y), \qquad y(t_0) = y_0.$$

На временной сетке известны

$$t_n, \qquad y_n \approx y(t_n).$$

Необходимо определить

$$y_{n+1}.$$

Метод Эйлера

Производную в точке t_n заменим разностью вперёд:

$$\left.\frac{dy}{dt}\right|_{t_n} \approx \frac{y_{n+1} - y_n}{\Delta t}.$$

Подставляем в ОДУ:

$$\frac{y_{n+1} - y_n}{\Delta t} = f(t_n, y_n).$$

Отсюда

$$\boxed{y_{n+1} = y_n + \Delta t\, f(t_n, y_n)}$$

Новое состояние = текущее состояние + шаг $\times$ текущая скорость изменения.

Геометрический смысл метода Эйлера

В точке

$$(t_n, y_n)$$

производная

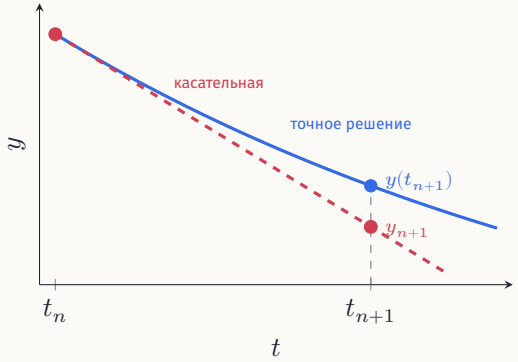
$$f(t_n, y_n)$$

задаёт наклон касательной.
Метод Эйлера предполагает, что на коротком интервале

$$[t_n, t_{n+1}]$$

решение можно заменить отрезком этой касательной.

$$y_{n+1} = y_n + \Delta t\, f(t_n, y_n)$$



Чем длиннее шаг Δt, тем сильнее касательная может отклониться от истинной траектории.

Метод Эйлера для охлаждения

Для закона охлаждения Ньютона

$$\frac{dT}{dt} = -k(T - T_\infty)$$

метод Эйлера даёт

$$\boxed{T_{n+1} = T_n - k\Delta t(T_n - T_\infty)}$$

Параметры

Возьмём

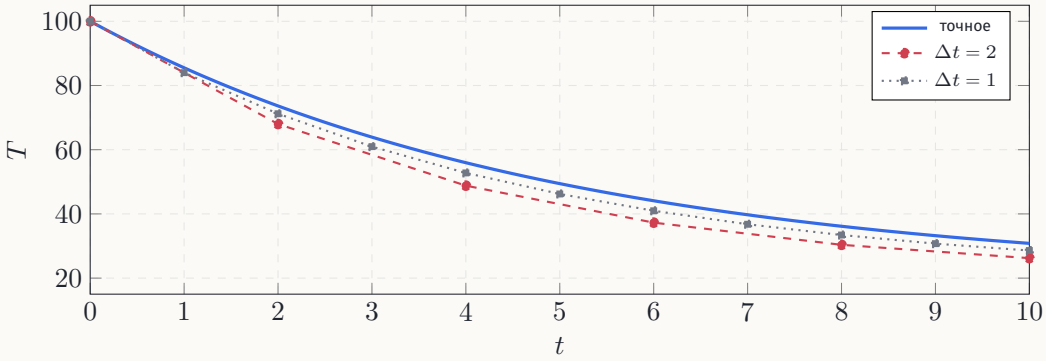
$$T_0 = 100, \qquad T_\infty = 20,$$

$$k = 0.2, \qquad \Delta t = 0.5.$$

```python
import numpy as np

k = 0.2
T_inf = 20.0

dt = 0.5
t_end = 20.0

t = np.arange(
    0, t_end + dt, dt
)

T = np.zeros_like(t)
T[0] = 100.0

for n in range(len(t) - 1):
    T[n+1] = (
        T[n]
        - k*dt*(T[n] - T_inf)
    )
```

Один цикл по времени последовательно строит $T_1, T_2, \ldots, T_N$.

Влияние шага Δt



Шаг Δt — компромисс между точностью и вычислительной стоимостью.

Устойчивость метода Эйлера

Рассмотрим простую модель затухания:

$$\frac{dy}{dt} = -\lambda y, \qquad \lambda > 0.$$

Метод Эйлера даёт:

$$y_{n+1} = (1 - \lambda \Delta t) y_n$$

Обозначим $r = \lambda \Delta t$.

$0 < r < 1$

$$0 < 1 - r < 1$$

Решение **монотонно затухает**:

$$y_n \to 0.$$

$1 < r < 2$

$$-1 < 1 - r < 0$$

Решение меняет знак, но амплитуда уменьшается:

$$|y_n| \to 0.$$

Устойчиво, но осциллирует.

$r > 2$

$$|1 - r| > 1$$

Амплитуда растёт:

$$|y_n| \to \infty.$$

Численная неустойчивость.

3

Методы Рунге–Кутты

Зачем нужны методы Рунге–Кутты?

Ограничение метода Эйлера

Метод Эйлера использует только наклон в начале шага:

$$k_1 = f(t_n, y_n).$$

Затем:

$$y_{n+1} = y_n + h\,k_1.$$

Если наклон внутри шага заметно меняется, такое приближение может быть неточным.

Идея Рунге–Кутты

Вместо одного наклона оцениваем **несколько наклонов внутри шага** и объединяем их.

Методы Рунге–Кутты повышают точность, используя дополнительную информацию внутри одного шага.

Классический метод RK4

Для задачи

$$\frac{dy}{dt} = f(t, y)$$

на одном шаге h вычисляются четыре наклона.

$$k_1 = f(t_n, y_n),$$

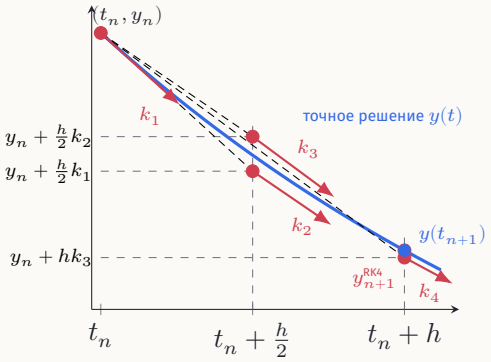
$$k_2 = f\left(t_n + \frac{h}{2},\, y_n + \frac{h}{2}k_1\right),$$

$$k_3 = f\left(t_n + \frac{h}{2},\, y_n + \frac{h}{2}k_2\right),$$

$$k_4 = f(t_n + h,\, y_n + hk_3).$$

После этого:

$$y_{n+1} = y_n + \frac{h}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right)$$



RK4 использует информацию в начале, середине и конце шага, поэтому обычно намного точнее метода Эйлера.

RK4 на Python: от алгоритма к задаче

Один шаг RK4

```python
def rk4_step(f, t, y, h):

    k1 = f(t, y)

    k2 = f( t + h/2, y + h*k1/2 )

    k3 = f( t + h/2, y + h*k2/2 )

    k4 = f( t + h, y + h*k3 )

    return y + h/6 * ( k1 + 2*k2 + 2*k3 + k4 )
```

Закон охлаждения Ньютона

```python
import numpy as np

k = 0.2
T_inf = 20.0

def rhs(t, T):
    return -k * (T - T_inf)

h = 1.0
t = np.arange(0, 21, h)

T = np.zeros_like(t)
T[0] = 100.0

for n in range(len(t) - 1):
    T[n+1] = rk4_step( rhs, t[n], T[n], h )
```

Функция `rk4_step` реализует общий алгоритм, а `rhs` задаёт конкретную физическую модель.

Решение ОДУ с помощью SciPy

```python
import numpy as np
from scipy.integrate import solve_ivp

k = 0.2
T_inf = 20.0

def rhs(t, T):
    return -k * (T - T_inf)

sol = solve_ivp(
    rhs,
    t_span=(0, 20),
    y0=[100.0],
    method="RK45",
    rtol=1e-7,
    atol=1e-9
)

t = sol.t
T = sol.y[0]
```

Основные параметры `solve_ivp`

- `rhs` — правая часть ОДУ;
- `t_span` — интервал времени;
- `y0` — начальное состояние;
- `method="RK45"` — метод интегрирования;
- `rtol` — относительный допуск;
- `atol` — абсолютный допуск.

Мы задаём модель и требуемую точность, а интегратор автоматически выбирает шаги по времени.

Системы ОДУ

Пример: двухтемпературная модель

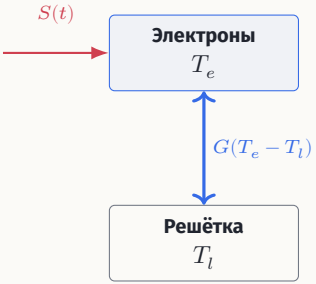
После быстрого возбуждения электроны и кристаллическая решётка могут иметь разные температуры:

$$C_e \frac{dT_e}{dt} = -G(T_e - T_l) + S(t)$$

$$C_l \frac{dT_l}{dt} = G(T_e - T_l)$$

где

- T_e — температура электронов;
- T_l — температура решётки;
- G — коэффициент связи;
- $S(t)$ — внешний источник энергии.



$$\mathbf{y} = \begin{bmatrix} T_e \\ T_l \end{bmatrix}, \quad \frac{d\mathbf{y}}{dt} = \mathbf{f}(t, \mathbf{y})$$

Система ОДУ в SciPy: двухтемпературная модель

```python
import numpy as np
from scipy.integrate import solve_ivp
Ce = 1.0; Cl = 5.0; G  = 0.8

def S(t):
    return 200.0 * np.exp(-(t - 1.0)**2 / 0.1)

def rhs(t, y):
    Te, Tl = y
    q = G * (Te - Tl)
    dTe = (-q + S(t)) / Ce
    dTl =  q / Cl
    return [dTe, dTl]

sol = solve_ivp(
    rhs,
    t_span=(0, 10),
    y0=[300.0, 300.0],
    t_eval=np.linspace(0, 10, 200),
    method="RK45"
)

Te = sol.y[0]; Tl = sol.y[1]
```

Вектор состояния

Теперь неизвестная величина — не одно число, а два:

$$\mathbf{y} = \begin{bmatrix} T_e \\ T_l \end{bmatrix}.$$

Поэтому

`y0 = [300, 300].`

Внешнее возбуждение

Функция `S(t)` задаёт временной профиль внешнего источника энергии.
В примере используется короткий импульс, который сначала нагревает электронную подсистему.

4

От ОДУ к уравнениям в частных производных

Почему ОДУ уже недостаточно?

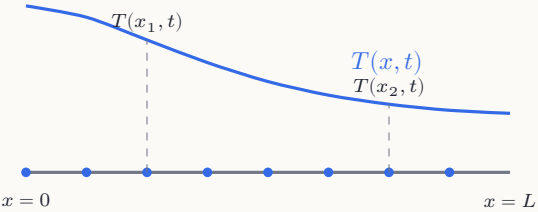
Если тело характеризуется одной температурой, $T = T(t)$, то достаточно ОДУ. В каждый момент времени вся система описывается одним значением температуры.

Распределённая система

В реальном теле температура может быть различной в разных точках:

$$T = T(x, t)$$

Теперь неизвестной является **температурное поле**.



В пространственно распределённой задаче появляются производные

$$\frac{\partial T}{\partial t}, \qquad \frac{\partial T}{\partial x}, \qquad \frac{\partial^2 T}{\partial x^2}.$$

ОДУ описывает изменение состояния во времени, а УЧП — изменение поля во времени и пространстве.

Уравнение теплопроводности

Для однородного одномерного тела без объёмного источника тепла:

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2}$$

где $\alpha = k/(\rho c_p)$ — температуропроводность.

Физический смысл

$$\frac{\partial T}{\partial t}$$

показывает, как температура меняется во времени.

$$\frac{\partial^2 T}{\partial x^2}$$

характеризует локальную кривизну температурного профиля.

Для решения необходимо задать

Начальное условие

$$T(x, 0) = T_0(x)$$

и **граничные условия** при $x = 0, \qquad x = L.$

Например:

$$T(0, t) = T_L, \qquad T(L, t) = T_R.$$

Температурные неоднородности вызывают перенос тепла и постепенно сглаживаются во времени.

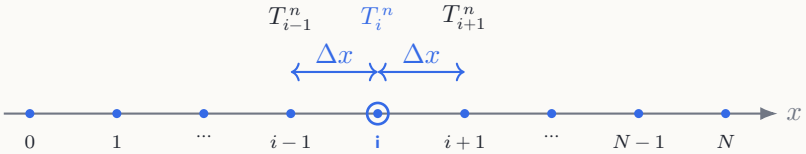
Пространственная сетка

Непрерывную область

$$0 \le x \le L$$

разбиваем на узлы:

$$x_i = i\Delta x, \qquad \Delta x = \frac{L}{N}, \qquad i = 0, 1, \dots, N$$



После дискретизации температурное поле заменяется конечным набором значений:

$$T_i^n \approx T(x_i, t_n)$$

где

i — номер пространственного узла, n — номер шага по времени.

5

Метод конечных разностей для теплопроводности

Аппроксимация производных

Производная по времени

Используем разность вперёд:

$$\left.\frac{\partial T}{\partial t}\right|_i^n \approx \frac{T_i^{n+1} - T_i^n}{\Delta t}$$

Используются два временных слоя:

$$n \quad \to \quad n+1.$$

Вторая производная по пространству

Используем центральную разность:

$$\left.\frac{\partial^2 T}{\partial x^2}\right|_i^n \approx \frac{T_{i+1}^n - 2T_i^n + T_{i-1}^n}{\Delta x^2}$$

Используются соседние узлы:

$$i-1, \qquad i, \qquad i+1.$$

После подстановки:

$$\frac{T_i^{n+1} - T_i^n}{\Delta t} = \alpha \frac{T_{i+1}^n - 2T_i^n + T_{i-1}^n}{\Delta x^2}$$

FTCS = Forward Time + Central Space

разность вперёд по времени $+$ центральная разность по пространству

Явная схема FTCS

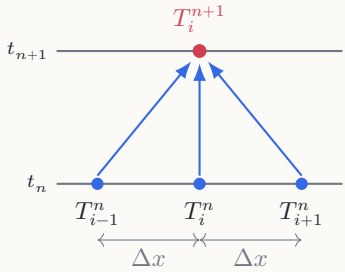
Из разностного уравнения выражаем температуру нового временного слоя:

$$T_i^{n+1} = T_i^n + \mathrm{Fo}\left(T_{i+1}^n - 2T_i^n + T_{i-1}^n\right)$$

где

$$\mathrm{Fo} = \frac{\alpha \Delta t}{\Delta x^2}$$

— сеточное число Фурье.



Почему схема явная?

Все величины в правой части $T_{i-1}^n, \quad T_i^n, \quad T_{i+1}^n$ уже известны.
Поэтому T_i^{n+1} вычисляется напрямую.

$$T_{i-1}^n,\ T_i^n,\ T_{i+1}^n \implies T_i^{n+1}$$

Новый временной слой строится последовательно из уже известного предыдущего слоя.

Устойчивость явной схемы

Для одномерного уравнения теплопроводности явная FTCS-схема устойчива при

$$\mathrm{Fo} = \frac{\alpha \Delta t}{\Delta x^2} \leq \frac{1}{2}$$

Следовательно,

$$\Delta t \leq \frac{\Delta x^2}{2\alpha}$$

Сгущаем пространственную сетку

Пусть

$$\Delta x_{\text{new}} = \frac{\Delta x}{10}.$$

Тогда

$$\Delta x^2_{\text{new}} = \frac{\Delta x^2}{100}.$$

Что происходит с шагом времени?

Для сохранения устойчивости необходимо

$$\Delta t_{\text{new}} \lesssim \frac{\Delta t}{100}$$

То есть расчёт потребует примерно в 100 раз больше временных шагов.

Начальные и граничные условия

Начальное условие

Температурное поле в начальный момент времени:

$$T_i^0 = T(x_i, 0)$$

Dirichlet

Задана температура:

$$T(0,t) = T_L.$$

Например:

$$T_L = 100^\circ\text{C}.$$

Neumann

Задан тепловой поток:

$$-k\frac{\partial T}{\partial x} = q.$$

Адиабатическая граница:

$$q = 0.$$

Robin

Конвективный теплообмен:

$$-k\frac{\partial T}{\partial x} = h(T - T_\infty).$$

Температура границы не задана заранее.

Уравнение, начальное условие и граничные условия вместе определяют физическую задачу.

Явная схема FTCS на Python

```
import numpy as np
alpha = 1.0e-4
L = 0.10
Nx = 51
dx = L / (Nx - 1)
Fo = 0.4
dt = Fo * dx**2 / alpha
T = np.full(Nx, 20.0)
T[0]  = 100.0
T[-1] = 20.0

Nt = 500
for n in range(Nt):
    T_new = T.copy()
    T_new[1:-1] = (
            T[1:-1]
            + Fo * (T[2:] - 2*T[1:-1] + T[:-2])
    )
    T_new[0]  = 100.0
    T_new[-1] = 20.0

    T = T_new
```

Параметры сетки

`Nx` — число узлов;

`dx` — пространственный шаг;

`dt` — шаг по времени;

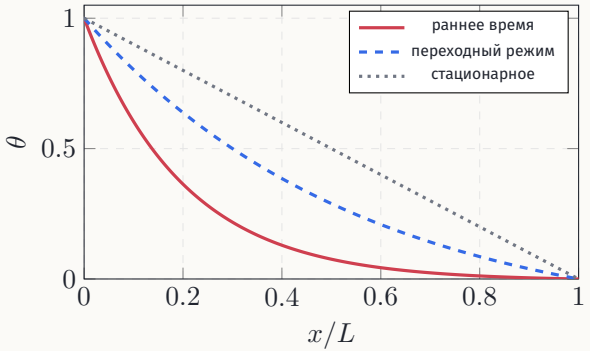
`Fo` — сеточное число Фурье.

Обновление

Срез `T[1:-1]` содержит внутренние узлы. Граничные температуры задаются отдельно после каждого шага.

Одна строка NumPy одновременно обновляет все внутренние узлы сетки.

Эволюция температурного профиля



Нормированная температура:

$$\theta = \frac{T - T_R}{T_L - T_R}.$$

Со временем

Температурный профиль постепенно сглаживается,

$$\frac{\partial T}{\partial t} \to 0,$$

и решение приближается к стационарному состоянию.

Численное решение показывает не только конечное состояние, но и весь процесс его формирования.

Явная и неявная схемы

Явная схема

Используются только известные значения слоя n:

$$T_i^{n+1} = \Phi\left(T_{i-1}^n, T_i^n, T_{i+1}^n\right).$$

Поэтому новый слой вычисляется напрямую.

Неявная схема

Пространственные производные берутся на новом слое:

$$\frac{T_i^{n+1} - T_i^n}{\Delta t} = \alpha \frac{T_{i+1}^{n+1} - 2T_i^{n+1} + T_{i-1}^{n+1}}{\Delta x^2}.$$

Все T_i^{n+1} связаны между собой.

Поэтому на каждом временном шаге возникает система:

$$A\mathbf{T}^{n+1} = \mathbf{b}$$

Связь с лекцией 3: решение УЧП снова приводит к системе линейных уравнений.

Неявная схема: как найти новый временной слой?

Для неявной схемы:

$$\frac{T_i^{n+1} - T_i^n}{\Delta t} = \alpha \frac{T_{i+1}^{n+1} - 2T_i^{n+1} + T_{i-1}^{n+1}}{\Delta x^2}.$$

Введём

$$\mathrm{Fo} = \frac{\alpha \Delta t}{\Delta x^2}.$$

После переноса всех неизвестных нового слоя в левую часть:

$$-\mathrm{Fo} T_{i-1}^{n+1} + (1 + 2\mathrm{Fo}) T_i^{n+1} - \mathrm{Fo} T_{i+1}^{n+1} = T_i^n$$

Для всех внутренних узлов получаем систему:

$$\underbrace{\begin{bmatrix} 1+2\mathrm{Fo} & -\mathrm{Fo} & 0 & \cdots \\ -\mathrm{Fo} & 1+2\mathrm{Fo} & -\mathrm{Fo} & \cdots \\ 0 & -\mathrm{Fo} & 1+2\mathrm{Fo} & \ddots \\ \vdots & \vdots & \ddots & \ddots \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} T_1^{n+1} \\ T_2^{n+1} \\ T_3^{n+1} \\ \vdots \end{bmatrix}}_{\mathbf{T}^{n+1}} = \underbrace{\begin{bmatrix} T_1^n \\ T_2^n \\ T_3^n \\ \vdots \end{bmatrix}}_{\mathbf{b}}$$

На каждом шаге времени

решаем систему

$$A\mathbf{T}^{n+1} = \mathbf{b}$$

Как учитывать граничные условия?

Рассмотрим граничные условия Дирихле:

$$T_0^{n+1} = T_L, \qquad T_N^{n+1} = T_R.$$

Для внутренних узлов:

$$-\mathrm{Fo} T_{i-1}^{n+1} + (1 + 2\mathrm{Fo}) T_i^{n+1} - \mathrm{Fo} T_{i+1}^{n+1} = T_i^n.$$

Для первого внутреннего узла:

$$(1 + 2\mathrm{Fo}) T_1^{n+1} - \mathrm{Fo} T_2^{n+1} = T_1^n + \mathrm{Fo} T_L.$$

Для последнего внутреннего узла:

$$-\mathrm{Fo} T_{N-2}^{n+1} + (1 + 2\mathrm{Fo}) T_{N-1}^{n+1} = T_{N-1}^n + \mathrm{Fo} T_R.$$

$$A \begin{bmatrix} T_1^{n+1} \\ T_2^{n+1} \\ \vdots \\ T_{N-1}^{n+1} \end{bmatrix} = \begin{bmatrix} T_1^n + \mathrm{Fo} T_L \\ T_2^n \\ \vdots \\ T_{N-1}^n + \mathrm{Fo} T_R \end{bmatrix}$$

Граничные температуры T_L, T_R уже известны. Поэтому они не входят в вектор неизвестных, а добавляются в правую часть системы.

Неявная схема на Python

```python
import numpy as np
alpha = 1.0e-4
L, Nx = 0.10, 51
dx = L / (Nx - 1)
Fo = 2.0
dt = Fo * dx**2 / alpha
TL, TR = 100.0, 20.0
T = np.full(Nx, 20.0)
T[0], T[-1] = TL, TR
Nint = Nx - 2
A = (
    np.diag((1 + 2*Fo) * np.ones(Nint))
    + np.diag(-Fo * np.ones(Nint-1),  1)
    + np.diag(-Fo * np.ones(Nint-1), -1)
)

for n in range(100):
    b = T[1:-1].copy()
    b[0]  += Fo * TL
    b[-1] += Fo * TR
    T[1:-1] = np.linalg.solve(A, b)
    T[0], T[-1] = TL, TR
```

Один временной шаг

1. Берём старый слой $\mathbf{T}^n$

2. Формируем правую часть **b**

3. Добавляем вклад границ

$$T_L, T_R$$

4. Решаем

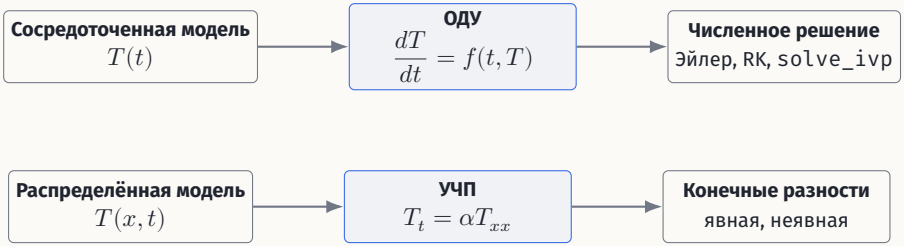
$$A\mathbf{T}^{n+1} = \mathbf{b}$$

Неявная схема заменяет простое обновление узлов решением линейной системы на каждом временном шаге.

6

Практика и итоги

Одна физика — разные математические модели



Выбор численного метода начинается с выбора математического уровня описания физической системы.

Один процесс — разные вычислительные подходы

Один и тот же процесс диффузии можно описывать разными вычислительными идеями.

Одна физическая задача может решаться детерминированно, стохастически и с помощью методов машинного обучения.

Практическое задание

Часть A: ОДУ

Для закона охлаждения Ньютона:

$$\frac{dT}{dt} = -k(T - T_\infty)$$

1. реализовать метод Эйлера;
2. использовать

$$\Delta t = 2,\ 1,\ 0.5,\ 0.25;$$

3. сравнить с точным решением;

Часть B: УЧП

Для одномерного уравнения теплопроводности:

$$T_t = \alpha T_{xx}$$

1. реализовать явную FTCS-схему;
2. выполнить расчёт при

$$\text{Fo} = 0.4 \quad \text{и} \quad \text{Fo} = 0.6;$$

3. сравнить устойчивое и неустойчивое решения;