

Шипилова Полина, 303 группа, Физический Факультет МГУ

# Решение одномерного уравнения переноса

Импортируем необходимые библиотеки

```
In [24]: import numpy as np

import plotly
import plotly.graph_objs as go

import matplotlib
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec

import warnings
from IPython.display import clear_output
```

## Постановка задачи

Используя схему бегущего счета и итерационные методы, решить задачу:

$$\begin{cases} \frac{\partial u}{\partial t} + \frac{1}{1+u^2} \frac{\partial u}{\partial x} = 0, & 0 < x \leq 1, \\ u(x, 0) = x, \\ u(0, t) = 0. \end{cases}$$

## Исследование задачи

### Исследование характеристик

Общий вид квазилинейного уравнения переноса

$$u_t + F(u)u_x = 0.$$

Уравнение характеристик имеет вид

$$dt = \frac{dx}{F(u(x, t))}, \quad u = \text{const.}$$

В нашем случае

$$dt = (1 + u^2) dx.$$

Преобразуем его

$$\int_{t_o}^t dt = \int_{x_o}^x (1 + u^2) dx,$$

$$t = (1 + u_0^2)(x - x_0) + t_0, u_0 = u(x_0, t_0).$$

Воспользуемся начальным и граничным условиями для получения двух семейств кривых:

$$1) \quad t_0 = 0 : \quad t = (1 + x_0^2)(x - x_0)$$

$$2) \quad x_0 = 0 : \quad t = x + t_0$$

Позже построим проекции характеристик в заданных областях и проверим пересечения кривых. В точках пересечения проекций его характеристик решение будет разрывным.

Уравнения характеристик

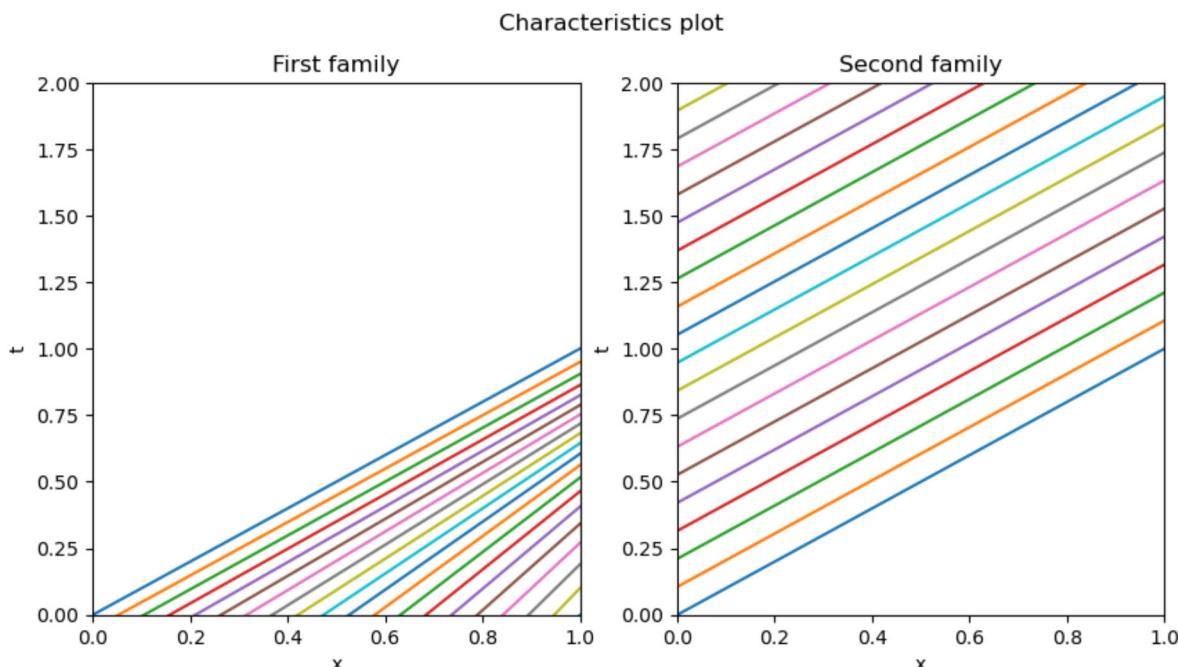
```
In [25]: def f_t_0(x, x0):
    return (1+x0**2)*(x-x0)

def f_x_0(x, t0):
    return x + t0
```

Построение характеристик

```
In [26]: x_start = 0  
x_end = 1  
t_start = 0  
t_end = 2  
how_many = 20
```

```
x = np.linspace(x_start, x_end, 100)  
  
gs = gridspec.GridSpec(1, 2)  
fig = plt.figure(figsize=(10,5))  
ax1 = fig.add_subplot(gs[0, 0]) # row 0, col 0  
  
fig.suptitle('Characteristics plot')  
ax1.set_xlim(x_start, x_end)  
ax1.set_ylim(t_start, t_end)  
ax1.set_title('First family')  
ax1.set_xlabel('x')  
ax1.set_ylabel('t')  
  
x_for_char = np.linspace(x_start, x_end, how_many)  
t_for_char = np.linspace(t_start, t_end, how_many)  
  
for x0 in x_for_char:  
    ax1.plot(x, f_t_0(x, x0))  
  
ax2 = fig.add_subplot(gs[0, 1])  
ax2.set_title('Second family')  
ax2.set_xlim(x_start, x_end)  
ax2.set_ylim(t_start, t_end)  
ax2.set_xlabel('x')  
ax2.set_ylabel('t')  
for t0 in t_for_char:  
    ax2.plot(x, f_x_0(x, t0))
```



Итак, в данной области  $x \in (0, 1]$  характеристики не пересекаются, а значит не происходит явления опрокидывания волны и профиль является однозначным. Во всей области решение будет представимо через разностную схему.

## Численное решение

### Сетка

Введем в области  $\Omega = \{(x, t) : 0 \leq x < 1, 0 < t < 2\}$  сетку с шагом  $h$  по  $x$  и шагом  $\tau$  по  $t$ :

$$\omega_{h,\tau} = \begin{cases} x_n = n \cdot h, & h = \frac{1}{N}, \quad n = \overline{0, N} \\ t_m = m \cdot \tau, & \tau = \frac{1}{M}, \quad m = \overline{0, M} \end{cases}$$

Выберем  $N = M = 100$ .

На  $\omega_{h,\tau}$  будем рассматривать сеточную функцию  $u_n^m = u(x_n, t_m)$ .

Поскольку рассматриваемое уравнение переноса является дифференциальным уравнением в частных производных первого порядка, то для его аппроксимации необходимо использовать шаблон, содержащий узлы сетки, принадлежащие как минимум двум временным и двум пространственным слоям.

### Шаблон

Приведём исходное уравнение к дивергентному виду  $u_t + (\tilde{F}(x))_x = 0$ .

Так как  $(\arctan x)' = \frac{1}{1+x^2}$ , то уравнение преобразуется к виду

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x}(\arctan u) = 0.$$

Соответственно,  $F_n^m = F(u_n^m) = \arctan(u_n^m)$ .

### Схема "Верхний уголок"

Разностная схема в этом случае имеет вид

$$\frac{1}{\tau}(u_{n+1}^{m+1} - u_n^m) + \frac{1}{h}(F(u_{n+1}^{m+1}) - F(u_n^m)) = 0$$

Погрешность данной схемы  $O(h + \tau)$ .

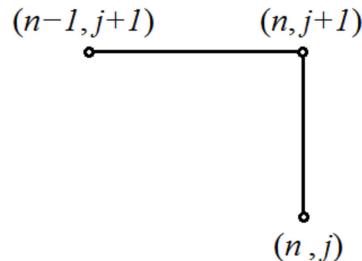


Схема "Квадрат"

Для рассматриваемой задачи будем использовать [четырехточечный шаблон](#). Он безусловно устойчив и аппроксимирует задачу как  $O(h^2 + \tau^2)$ .

Производная  $\frac{\partial u}{\partial t}$  аппроксимируется как среднее арифметическое односторонних разностных производных по  $t$ , взятых при  $x = x_i$  и  $x = x_{i+1}$ , а производная  $\frac{\partial u}{\partial x}$  — как среднее арифметическое разностных производных по  $x$ , взятых на слоях  $j$  и  $j+1$ .

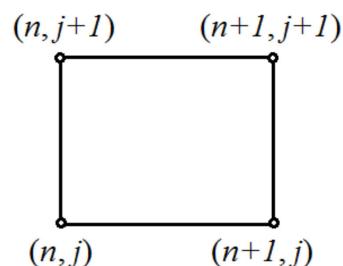
Таким образом, разностная схема задачи имеет вид

$$\frac{u_n^{m+1} - u_n^m + u_{n+1}^{m+1} - u_{n+1}^m}{2\tau} + \frac{F_{n+1}^{m+1} - F_n^{m+1} + F_{n+1}^m - F_n^m}{2h} = 0.$$

При помощи метода Ньютона найдем корень этого уравнения —  $u_{n+1}^{m+1}$  и перейдем к вычислению следующей точки.

Используем начальное и граничное условия

$$\begin{cases} u_n^0 = nh \\ u_0^m = 0 \end{cases}$$



Уравнение данной схемы - это уравнение относительно  $u_{n+1}^{m+1}$ :  $f(u_{n+1}^{m+1}) = 0$ .

Таким образом разностная аппроксимация уравнения в точке  $(x_n + 0,5h; t_m + 0,5\tau)$  имеет следующий вид

$$\left\{ \begin{array}{l} f(u_{n+1}^{m+1}) = \frac{u_n^{m+1} - u_n^m + u_{n+1}^{m+1} - u_{n+1}^m}{2\tau} - \frac{F_{n+1}^{m+1} - F_n^{m+1} + F_{n+1}^m - F_n^m}{2h} \\ f'(u_{n+1}^{m+1}) = \frac{1}{2\tau} - \frac{1}{1 + u_{n+1}^{m+1}} \cdot \frac{1}{2h} \\ u_n^0 = x_n \\ u_0^m = 0 \end{array} \right.$$

Полученную разностную схему будем решать с помощью метода бегущего счета. Считая известными значения  $t_m$  и последовательно вычисляя  $\Gamma_1^{m+1}, \Gamma_2^{m+1}, \dots, \Gamma_N^{m+1}$ , найдем  $t_{m+1}$ . Корни уравнения для каждого  $n$  будем искать методом Ньютона.

### Метод Ньютона

Суть метода Ньютона заключается в итерационной последовательности

$$u_{n+1}^{m+1(s+1)} = u_{n+1}^{m+1(s)} - \frac{f(u_{n+1}^{m+1(s)})}{f'(u_{n+1}^{m+1(s)})},$$

которая продолжается до тех пор, пока не будет достигнута необходимая точность  $\varepsilon$ :  $|u_{n+1}^{m+1(s+1)} - u_{n+1}^{m+1(s)}| \leq \varepsilon$ .

## Обоснование математической модели

### Обоснование аппроксимации

Обозначим  $C(u^m) = \arctan(u)$ . Тогда разностная схема примет следующий вид:

$$\frac{u_{n+1}^{m+1} - u_n^m}{2\tau} + C(u) \frac{u_{n+1}^{m+1} - u_n^{m+1}}{2h} + \frac{u_n^{m+1} - u_n^m}{2\tau} + C(u) \frac{u_{n+1}^m - u_n^m}{2h} = 0$$

Разложим  $u(x, y)$  по формуле Тейлора около центра ячейки  $\left(x_n + \frac{h}{2}; t_m + \frac{\tau}{2}\right)$ .

$$u_{n+1}^m = u_{n+1/2}^m + \frac{h}{2}y'_{n+1/2} + \frac{h^2}{2 \cdot 4}y''_{n+1/2} + \frac{h^3}{6 \cdot 8}y'''_{n+1/2} + O(h^4)$$

$$u_n^m = u_{n+1/2}^m - \frac{h}{2}y'_{n+1/2} + \frac{h^2}{2 \cdot 4}y''_{n+1/2} - \frac{h^3}{6 \cdot 8}y'''_{n+1/2} + O(h^4)$$

Тогда

$$\frac{u_{n+1}^m - u_n^m}{2h} = \frac{1}{2h} \left( hy'_{n+1/2} + \frac{h^3}{3 \cdot 8}y'''_{n+1/2} + O(h^4) \right) = y'_{n+1/2} + \frac{h^2}{3 \cdot 8}y'''_{n+1/2} + O(h^3)$$

Аналогично для приращения по временному индексу.

$$u_n^{m+1} = u_n^{m+1/2} + \frac{\tau}{2}y'^{m+1/2} + \frac{\tau^2}{2 \cdot 4}y''^{m+1/2} + \frac{\tau^3}{6 \cdot 8}y'''^{m+1/2} + O(\tau^4)$$

$$u_n^m = u_n^{m+1/2} - \frac{\tau}{2}y'^{m+1/2} + \frac{\tau^2}{2 \cdot 4}y''^{m+1/2} - \frac{\tau^3}{6 \cdot 8}y'''^{m+1/2} + O(\tau^4)$$

Тогда

$$\begin{aligned} \frac{u_n^{m+1/2} - u_n^m}{2\tau} &= \frac{1}{2\tau} \left( \tau y'^{m+1/2} + \frac{\tau^3}{3 \cdot 8} y'''^{m+1/2} + O(\tau^4) \right) = \\ &= y'^{m+1/2} + \frac{\tau^2}{3 \cdot 8} y'''^{m+1/2} + O(\tau^3) = y'^{m+1/2} + O(\tau^2) \end{aligned}$$

Следовательно, аппроксимация в узле  $\left(x_n + \frac{h}{2}; t_m + \frac{\tau}{2}\right)$  имеет порядок  $O(\tau^2 + h^2)$ .

## Обоснование устойчивости

Необходимое условие сходимости - спектральный критерий Неймана

Воспользуемся методом «замораживания». Зафиксируем коэффициент перед  $\frac{\partial u}{\partial x}$ . Пусть  $\gamma(u(x, t)) = \frac{1}{1+u^2}$ . Тогда

$$\frac{u_n^{m+1} - u_n^m}{\tau} + \gamma(x_n, t_m) \frac{u_{n+1}^m - u_n^m}{h} = 0$$

Фиксируем внутреннюю точку  $(x_0, t_0)$  и будем искать решение в виде  $u_n^m = \lambda^m e^{i\alpha n}$ .

Подставляя, получаем:

$$\begin{aligned}\frac{\lambda - 1}{\tau} &= \beta \frac{e^{i\alpha} - 1}{h} \\ \lambda(\alpha) &= 1 + \beta \frac{\tau(e^{i\alpha} - 1)}{h}\end{aligned}$$

Сделав замену  $r = \sqrt{\beta\tau h}$  получим  $\lambda(\alpha) = 1 - r + re^{i\alpha}$ .

Спектр  $\lambda(\alpha)$  - щокружность на комплексной плоскости радиусом  $r$  и имеющая центр в точке  $1 - r$ .

$$|\lambda(\alpha)| \leq 1 \Rightarrow r \leq 1 \Rightarrow \max \left[ \frac{\gamma\tau}{h} \right] \leq 1$$

$$\tau = h \Rightarrow \max[\gamma] \leq 1$$

Исследуем  $\gamma(u)$  на экстремумы

$$\begin{aligned}\frac{\partial \gamma}{\partial u} &= \frac{2u}{(1+u^2)^2} = 0 \\ u &= 0 \\ \gamma(0) &= 1 \leq 1\end{aligned}$$

Таким образом условие  $|\lambda(\alpha)| \leq 1$  выполнено для любых значений шага по времени и координате, следовательно, спектральный критерий Неймана также выполнен для любых  $\tau, h$ . Поэтому данная схема безусловно устойчива.

Достаточное условие сходимости - критерий Куранта

Представим разностную схему в виде

$$u_{n+1}^{m+1} \left(1 - \frac{C\tau}{h}\right) + u_n^{m+1} \left(1 + \frac{C\tau}{h}\right) = u_n^m \left(1 - \frac{C\tau}{h}\right) + u_{n+1}^m \left(1 + \frac{C\tau}{h}\right)$$

Оценим это равенство по норме

$$\|u^{m+1}\| \left(1 - \frac{C\tau}{h}\right) + \|u^{m+1}\| \left(1 + \frac{C\tau}{h}\right) \leq 2\|u^m\|,$$

$$\|u^{m+1}\| \leq \|u^m\| \leq \|u^{m-1}\| \leq \dots \leq \|\varrho^0\| = \|\varphi\|.$$

То есть критерий Куранта выполняется.

## Численный расчёт

Зададим:  $\varepsilon$  - точность в методе Ньютона,  $N$  - количество шагов по  $x$ ,  $M$  - количество шагов по  $t$ , а также границы нашей сетки.

```
In [27]: X_START = 0
X_END = 1
T_START = 0
T_END = 2
error = 0.001
i_max = 10000

def calculate_u(u, N, M, dt, dx):
    print('Calculating...')
    for n in range(1, N):
        clear_output(wait = True)
        print(f'Progress: {round(n/(N-1)*100)}%')
        for m in range(1, M):

            # Определение точек, на которых будет работать шаблон
            u11 = u[n-1][m-1]
            u21 = u[n][m-1]
            u12 = u[n-1][m]

            # Вычисление значения и в новой точке
            u_new = solve_newton(u21, u12, u11, i_max, dt, dx)
            u[n][m] = u_new
    print('Done!')
    return u

def solve_newton(u21, u12, u11, i_max, dt, dx):
    x = u11
    i = 0
    while np.abs(f(u12, u11, u21, x, dt, dx)) > error and i < i_max:
        x += (-f(u12, u11, u21, x, dt, dx) / f_div(x, dt, dx))
        i+=1
    return x
```

Конкретные функции

```
In [28]: def C(u):
    return (1) / (1 + u ** 2)

def C_integral(u):
    return np.arctan(u)

def f(u12, u11, u21, u22, dt, dx):
    return 0.5*(u12 - u11 + u22 - u21) / dt + 0.5*(C_integral(u22) - C_integral(u12))

def f_div(u22, dt, dx):
    return 0.5 / dt + 0.5*(C(u22)) / dx
```

Вычислим значение функции  $u(x, t)$  во всех точках сетки.

```
In [29]: def diff_scheme(N=10, M=10):
    x = np.linspace(X_START, X_END, N)
    t = np.linspace(T_START, T_END, M)
    dx = x[1] - x[0]
    dt = t[1] - t[0]

    u = np.empty((N, M))

    for n in np.arange(N):
        u[0][n] = dx * n
    for m in np.arange(M):
        u[m][0] = 0

    return calculate_u(u, N, M, dt, dx)
```

```
In [37]: scheme10 = diff_scheme(10, 10)
scheme20 = diff_scheme(20, 20)
scheme30 = diff_scheme(30, 30)
scheme50 = diff_scheme(50, 50)
scheme100 = diff_scheme(100, 100)
```

Progress: 100%  
Done!

Визуализация решения

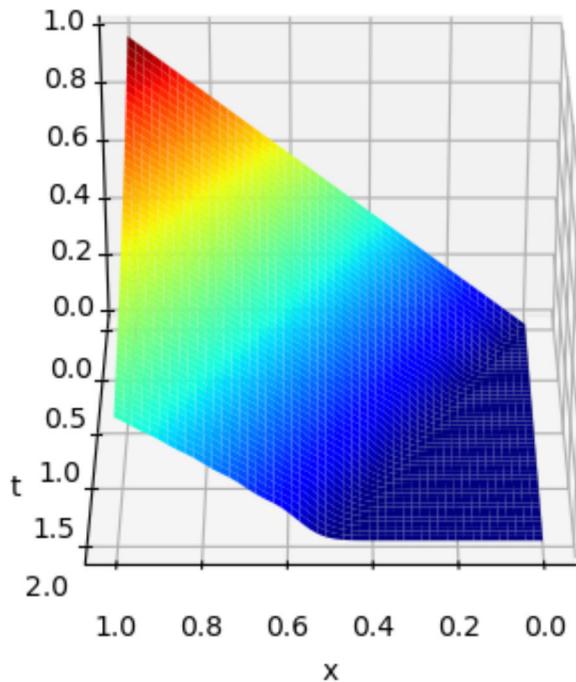
```
In [38]: from mpl_toolkits.mplot3d import axes3d

xn = np.linspace(X_START, X_END, num = 50)
tm = np.linspace(T_START, T_END, num = 50)

X, T = np.meshgrid(xn, tm)

fig1 = plt.figure()
o1 = fig1.add_subplot(111, projection = '3d')
p1 = o1.plot_surface(X, T, scheme50, rstride = 1, cstride = 1, cmap = 'jet')
o1.view_init(30, 90)
plt.title('Решение при N=M=50')
plt.xlabel('x')
plt.ylabel('t')
plt.show()
```

## Решение при $N=M=50$



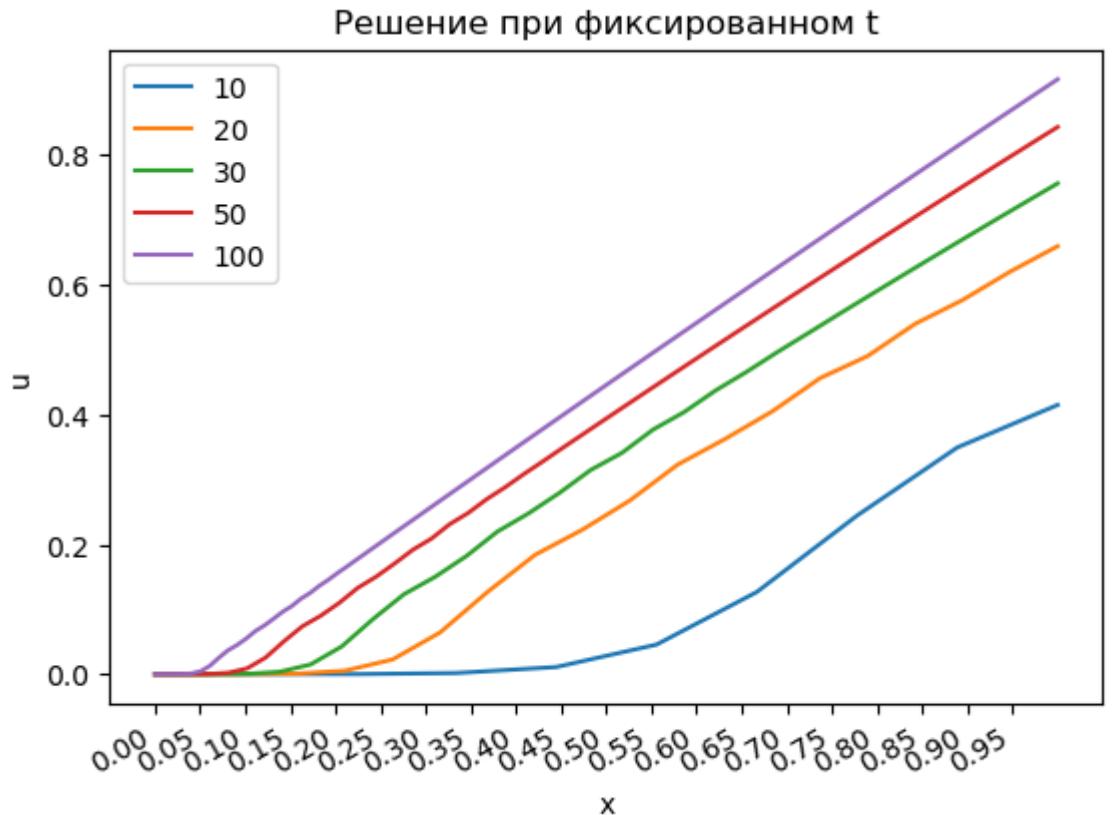
Решение при фиксированном  $t$

```
In [47]: fig2 = plt.figure()

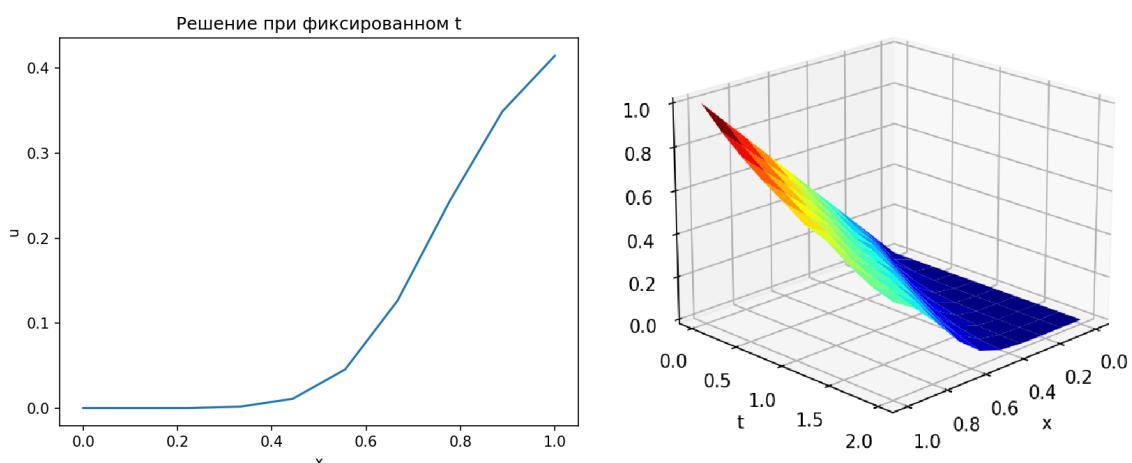
xn10 = np.linspace(X_START, X_END, num = 10)
xn20 = np.linspace(X_START, X_END, num = 20)
xn30 = np.linspace(X_START, X_END, num = 30)
xn50 = np.linspace(X_START, X_END, num = 50)
xn100 = np.linspace(X_START, X_END, num = 100)

plt.plot(xn10, scheme10[9], label='10')
plt.plot(xn20, scheme20[9], label='20')
plt.plot(xn30, scheme30[9], label='30')
plt.plot(xn50, scheme50[9], label='50')
plt.plot(xn100, scheme100[9], label='100')

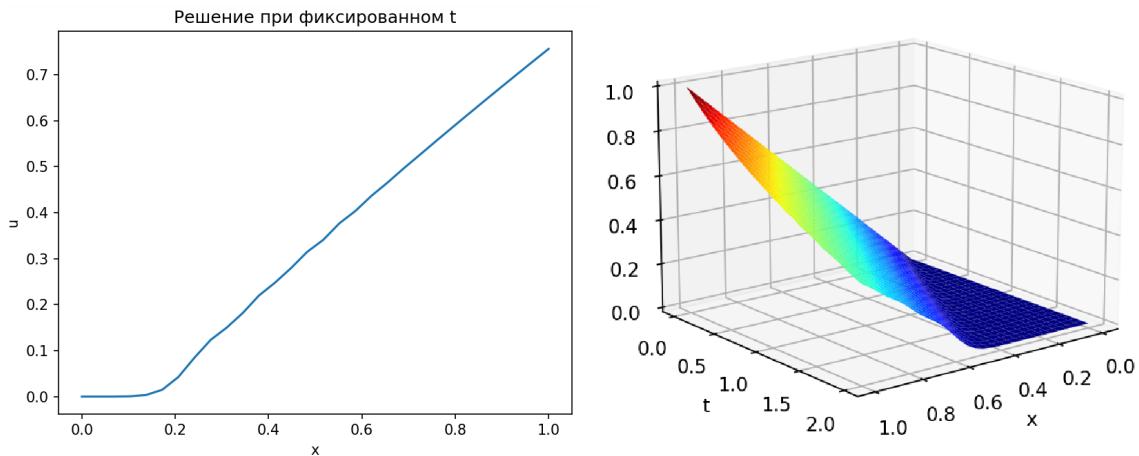
plt.title('Решение при фиксированном t')
plt.xlabel('x')
plt.ylabel('u')
plt.xticks(np.arange(0,1,0.05))
fig2.autofmt_xdate()
plt.legend()
plt.show()
```



Сетка  $N = M = 10$



Сетка  $N = M = 30$



Сетка  $N = M = 50$

