МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
имени М.В. Ломоносова

ФИЗИЧЕСКИЙ ФАКУЛЬТЕТ

КАФЕДРА ОБЩЕЙ ФИЗИКИ И ВОЛНОВЫХ ПРОЦЕССОВ

ЗАДАНИЕ №3

по курсу «Численные методы в физике»

$T_0 = 3.47$

Выполнил:
Делекторский Никита Юрьевич,
студент 425 группы

Преподаватель:
Шлёнов Святослав Александрович

Москва
2024

Оглавление

1. Постановка задачи

Численно решите уравнение теплопроводности:

$$\frac{\partial T}{\partial t} = \frac{\partial^2 T}{\partial x^2} \quad (1)$$

на отрезке $x = [0, 10]$ с граничными условиями $T(0) = T(10) = 0$ и интервале времени $t = [0, 100]$ с начальными условиями:

$$T(x, t = 0) = T_0 (x - x_0)^2 e^{-(x - x_0)^2} \quad (2)$$

где $x_0 = 5$, а T_0 вычисляется в соответствии с правилом*.

1) Воспользуйтесь **явной схемой** численного решения. Рассмотрите два варианта выбора шагов интегрирования dx по x и dt по t: 1) $dx = 0.1$, $dt = 0.01$; 2) $dx = 0.1$, $dt = 0.005$. Для каждого набора шагов интегрирования постройте графики $T(x)$ для шести моментов времени $t = 0, 0.1, 0.2, 0.3, 0.5, 1$.

2) Используйте **схему Кранка-Николсона** и метод прогонки, при этом может быть использована только одна библиотечная функция – для вычисления экспоненты. Рассмотрите те же варианты выбора шагов интегрирования, что и в случае явной схемы, и сравните полученные результаты.

3) **Выведите** условия устойчивости для использованных схем.

4) Постройте графики сеточной диффузии для использованных численных схем и шагов интегрирования в сравнении с диффузией, описываемой исходным уравнением.

5) Подготовьте отчет о выполненном задании в виде pdf-файла. В отчете следует отразить постановку задачи, методы ее решения, <u>текст написанной вами программы</u>, построенные графики и <u>вывод</u> условия устойчивости. На титульном листе рядом с ФИО в скобках укажите вычисленное значение T_0.

2. Явная схема решения

Зададим дискретные сеточные значения (x_j, t_n) непрерывных переменных (x, t):

$$x_j = (j-1)\Delta x, \quad j = 0,1,\dots,\frac{L}{\Delta x}+1, \tag{3}$$

$$t_n = n\Delta t, \quad n = 0,1,\dots,\frac{T}{\Delta t}+1, \tag{4}$$

где L и T – размеры пространственного и временного доменов соответственно. Тогда сеточное решение для температуры $T(x,t)$:

$$T(x,t) \to T_j^n(x_j, t_n) \tag{5}$$

с граничным условием:

$$T_0^n = T_L^n = 0 \tag{6}$$

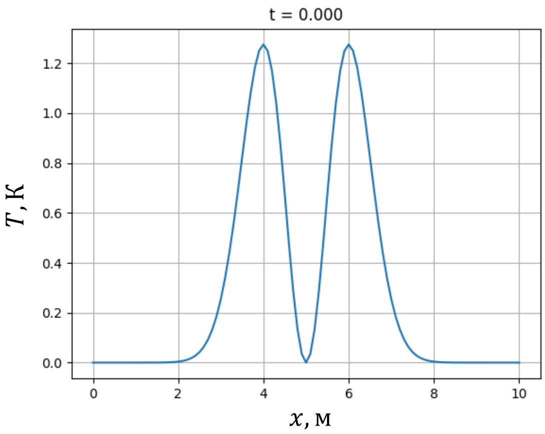
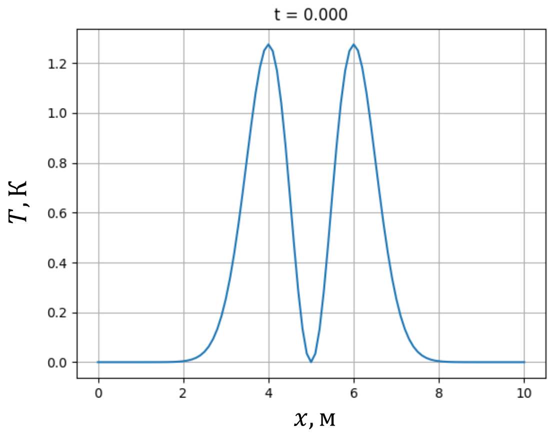
и начальным условием:

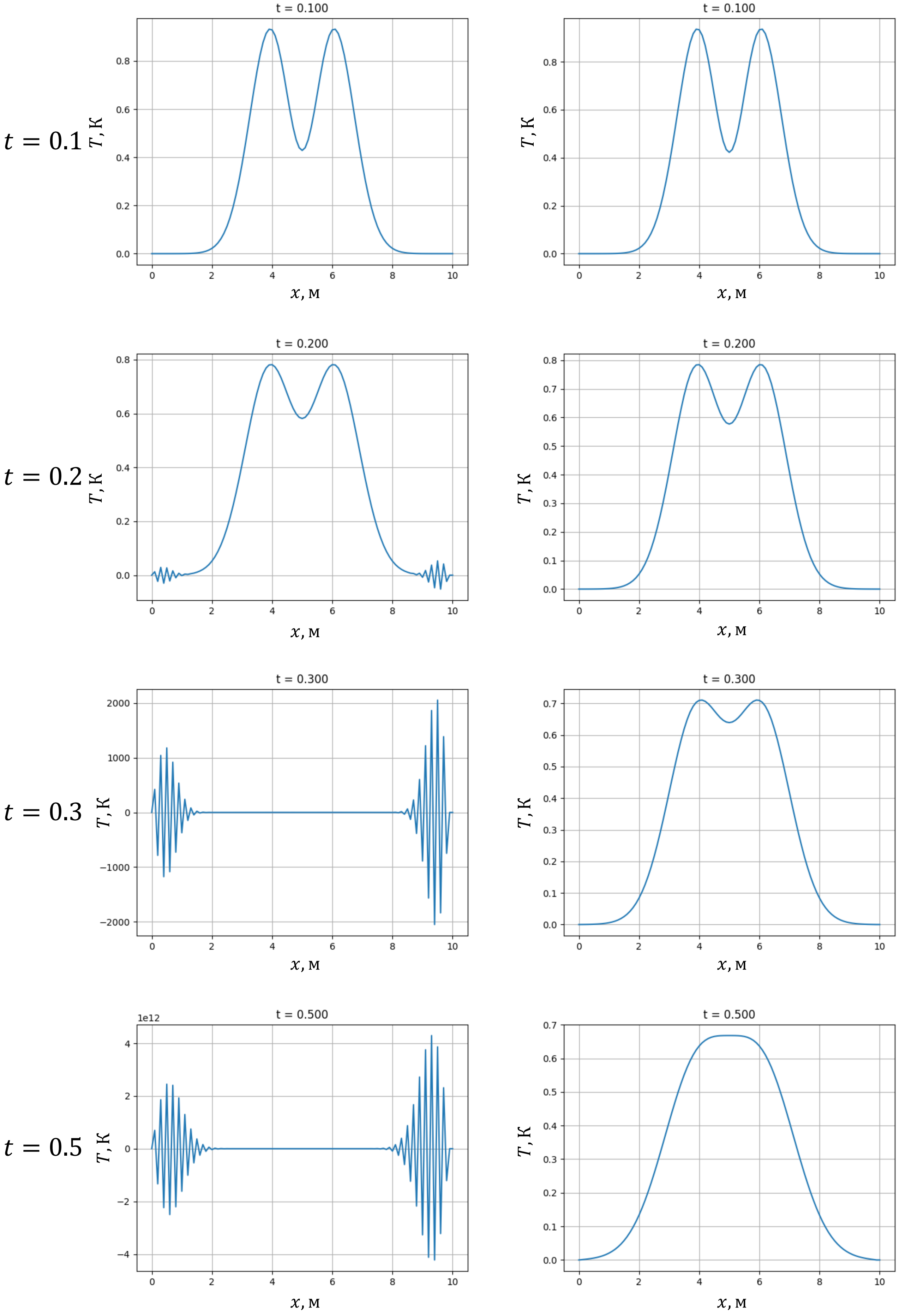
$$T_j^0 = T_0(x_j - x_0)^2 e^{-(x_j - x_0)^2}. \tag{7}$$

Шаблон явной схемы:

$$T_j^{n+1} = T_j^n + \frac{\Delta t}{(\Delta x)^2}\left(T_{j+1}^n - 2T_j^n + T_{j-1}^n\right) \tag{8}$$

Сравним результаты, полученные с разным шагом по времени:

	$dx = 0.1$ $dt = 0.01$	$dx = 0.1$ $dt = 0.005$
$t = 0$		



t = 0.1
t = 0.2
t = 0.3
t = 0.5
t = 0.100
t = 0.200
t = 0.300
t = 0.500
T, К
x, м
1e12

$t = 1$

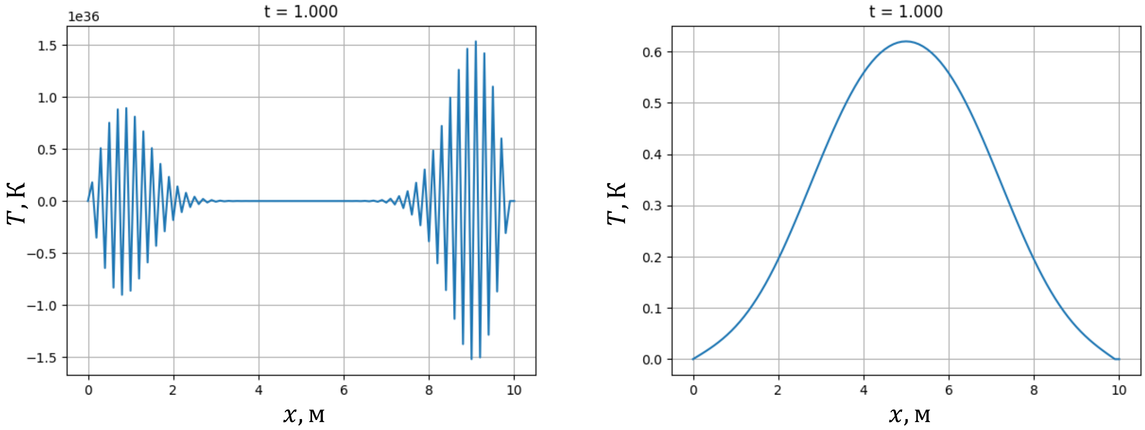


Рис. 1. Результаты численного решения уравнения теплопроводности явной схемой в разные моменты времени и с разным шагом по времени $dt = 0.005$

Решение во все моменты времени представлено на рисунке 2.

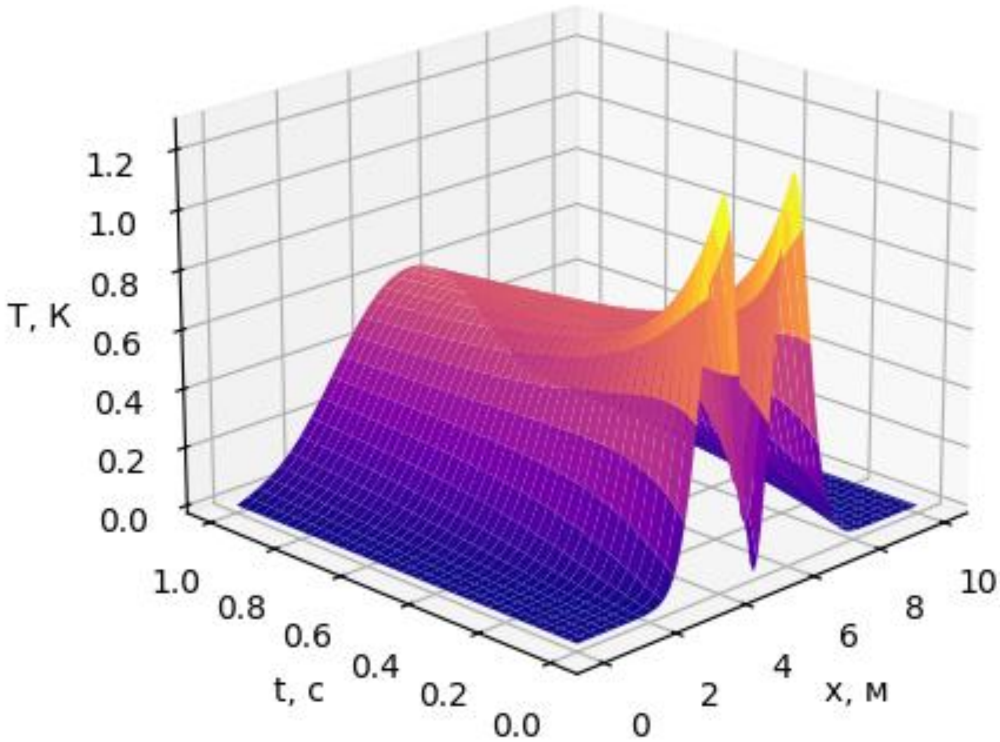


Рис. 2. Численное решение уравнения теплопроводности явной схемой с шагом по времени $dt = 0.005$

3. Схема Кранка-Николсона

Шаблон схемы Кранка-Николсона:

$$T_j^{n+1} = T_j^n + \frac{\Delta t}{2(\Delta x)^2}\left(T_{j+1}^n - 2T_j^n + T_{j-1}^n\right) + \frac{\Delta t}{2(\Delta x)^2}\left(T_{j+1}^{n+1} - 2T_j^{n+1} + T_{j-1}^{n+1}\right) \quad (9)$$

Так как схема неявная, применяем метод прогонки:

$$a_j u_{j+1} + b_j u_j + c_j u_{j-1} = f_j \quad (10)$$

$$u_j \to T_j^{n+1}, \qquad f_j \to T_j^n \quad (11)$$

Для поставленной задачи имеем:

$$T_j^{n+1}\left(1 + \frac{\Delta t}{(\Delta x)^2}\right) - \frac{\Delta t}{2(\Delta x)^2} T_{j+1}^{n+1} - \frac{\Delta t}{2(\Delta x)^2} T_{j-1}^{n+1} = \frac{\Delta t}{2(\Delta x)^2} T_{j+1}^n + \left(1 - \frac{\Delta t}{(\Delta x)^2}\right) T_j^n + \frac{\Delta t}{2(\Delta x)^2} T_{j-1}^n \quad (12)$$

$$2T_j^{n+1}\left(\frac{(\Delta x)^2}{\Delta t} + 1\right) - T_{j+1}^{n+1} - T_{j-1}^{n+1} = T_{j+1}^n + 2\left(\frac{(\Delta x)^2}{\Delta t} - 1\right) T_j^n + T_{j-1}^n \quad (13)$$

Отсюда имеем:

$$\begin{cases} a_j = -1, \ \ c_j = -1 \\ b_j = 2\left(\dfrac{(\Delta x)^2}{\Delta t} + 1\right) \\ f_j = T_{j+1}^n + 2\left(\dfrac{(\Delta x)^2}{\Delta t} - 1\right) T_j^n + T_{j-1}^n \end{cases} \quad (14)$$

$$\begin{cases} a_{j-1} = \dfrac{1}{2\left(\dfrac{(\Delta x)^2}{\Delta t} + 1\right) - a_j} \\ b_{j-1} = \dfrac{f_j + b_j}{2\left(\dfrac{(\Delta x)^2}{\Delta t} - 1\right) - a_j} \\ f_j = T_{j+1}^n + 2\left(\dfrac{(\Delta x)^2}{\Delta t} - 1\right) T_j^n + T_{j-1}^n \end{cases} \quad (15)$$

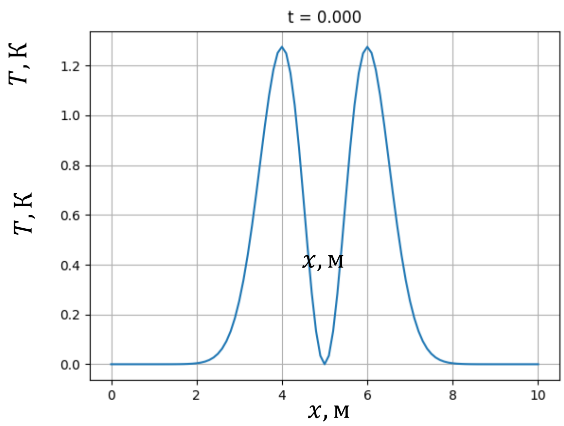
Тогда решение представимо в виде:

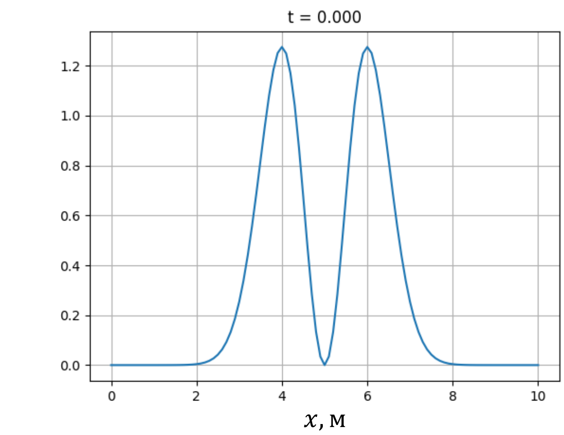
$$T_{j+1}^{n+1} = a_j T_j^{n+1} + b_j \quad (16)$$

Сравним результаты, полученные с разным шагом по времени:

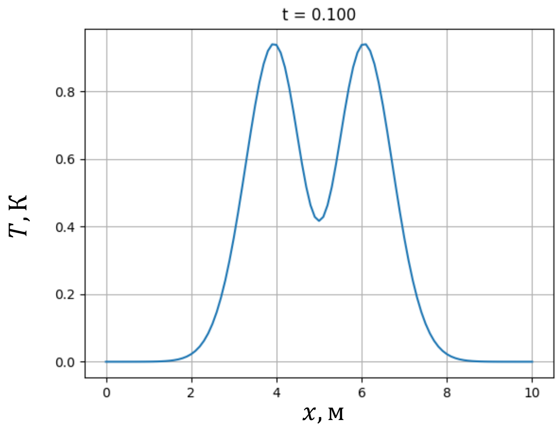
$dx = 0.1$, $dt = 0.01$	$dx = 0.1$, $dt = 0.005$

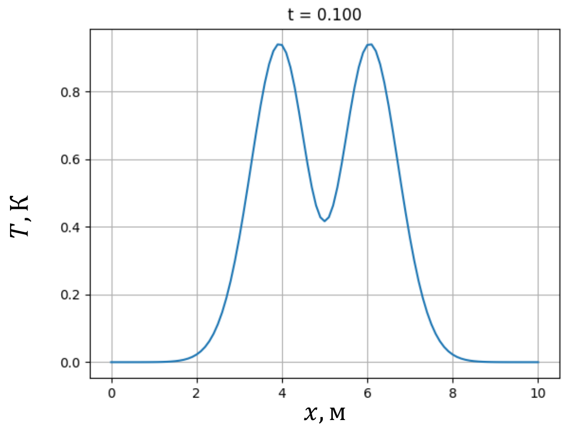
$t = 0$



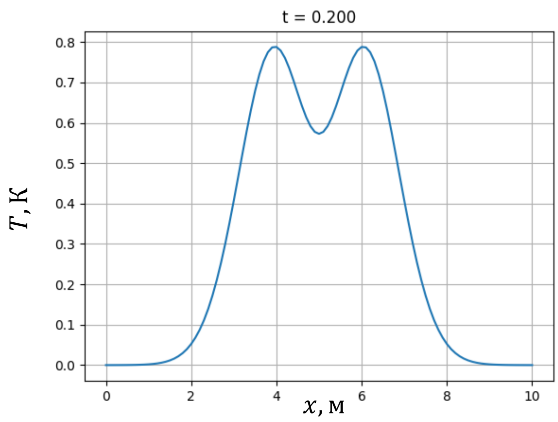


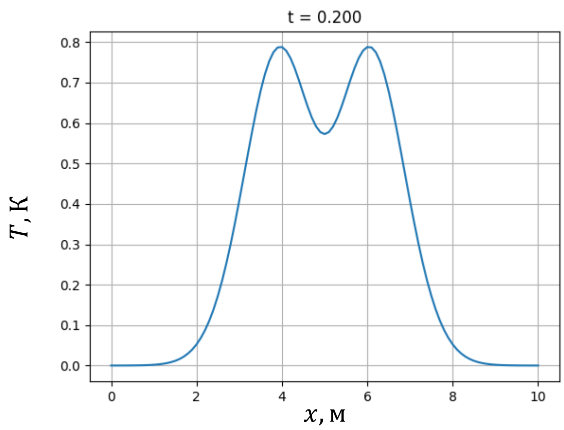
$t = 0.1$





$t = 0.2$





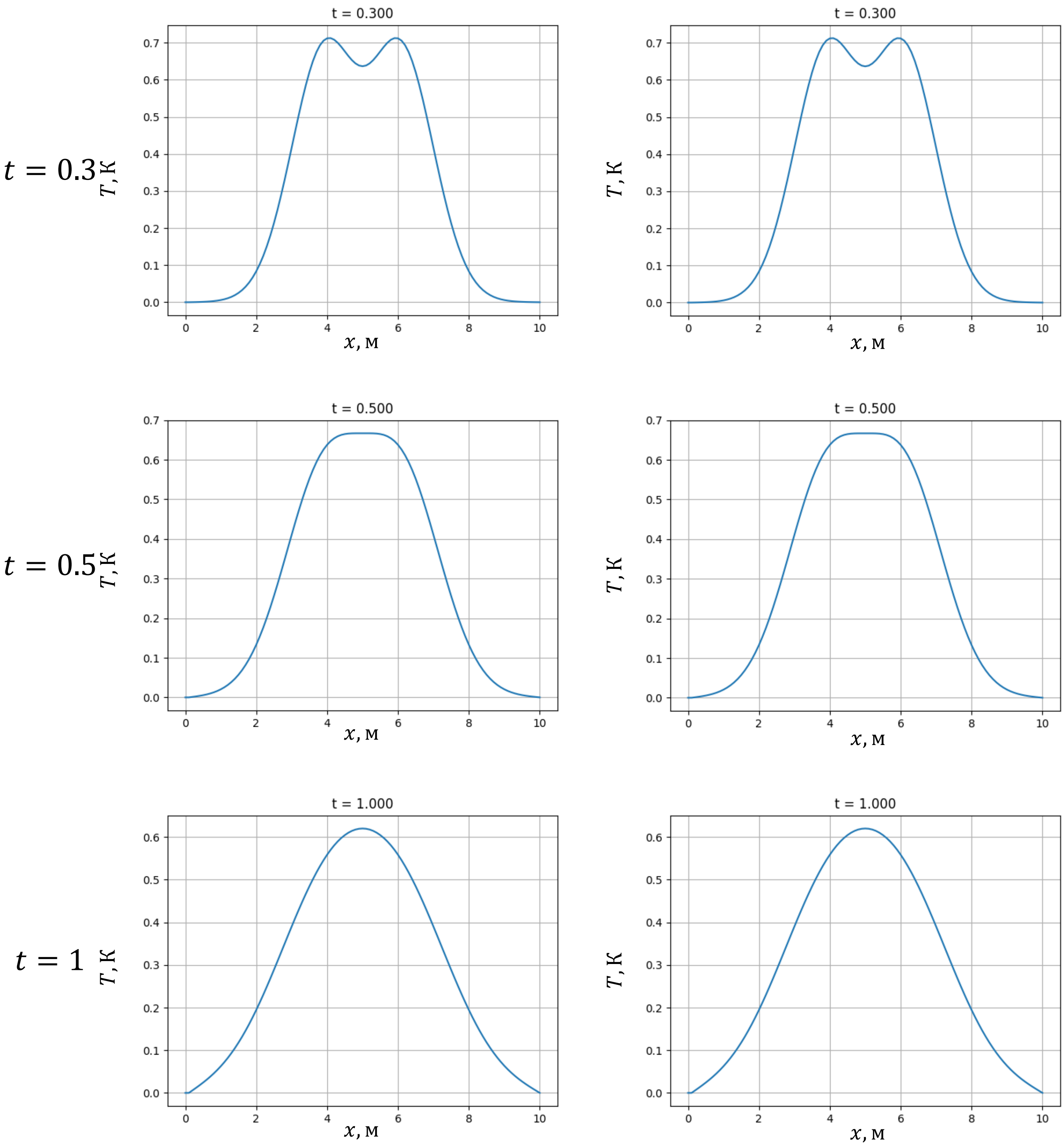


Рис. 3. Результаты численного решения уравнения теплопроводности методом Кранка-Николсона в разные моменты времени и с разным шагом по времени

Решение во все моменты времени:

$$dx = 0.1 \quad dt = 0.01 \qquad\qquad dx = 0.1 \quad dt = 0.005$$

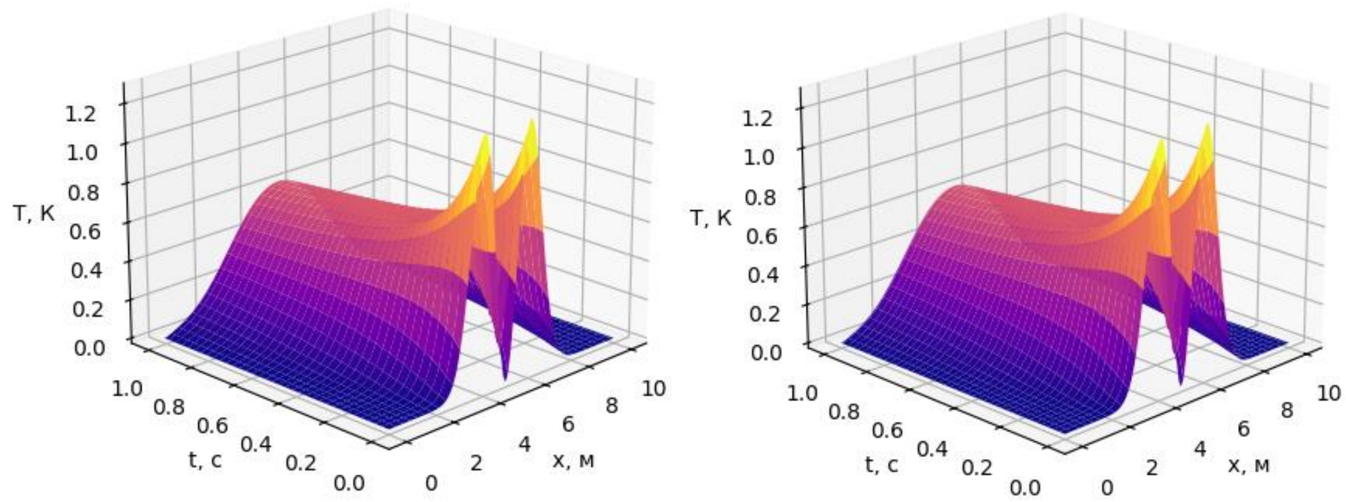


Рис. 4. Решение для схемы Кранка-Николсона с разными шагами по времени

4. Условия устойчивости

1. Явная схема

Для пространственной Фурье-моды $\hat{T}_j^n e^{ikx_j}$ с учётом $\hat{T}_j^{n+1} = \lambda \hat{T}_j^n$ имеем:

$$\hat{T}_k^{n+1} e^{ikx_j} = \hat{T}_k^n e^{ikx_j} + \alpha\left(\hat{T}_k^n e^{ik(x_j+\Delta x)} - 2\hat{T}_k^n e^{ikx_j} + \hat{T}_k^n e^{ik(x_j-\Delta x)}\right)$$

$$= \hat{T}_k^n e^{ikx_j}\left(1 + \alpha\left(e^{ik\Delta x} + e^{-ik\Delta x} - 2\right)\right) \quad (17)$$

$$\Rightarrow \quad \lambda = 1 - 2\alpha(1 - \cos k\Delta x) \quad (18)$$

$$\lambda = 1 - 4\alpha \sin^2 \frac{k\Delta x}{2}, \quad (19)$$

где $\alpha = \frac{\Delta t}{(\Delta x)^2}$

Схема устойчива при $|\lambda| \leq 1$, то есть при

$$4\alpha \sin^2 \frac{k\Delta x}{2} \leq 2 \quad (20)$$

$$\Rightarrow \quad \alpha \leq \frac{1}{2} \quad (21)$$

Следовательно, условие устойчивости явной схемы:

$$\frac{\Delta t}{(\Delta x)^2} \leq \frac{1}{2} \quad (22)$$

Таким образом, становится понятно, почему при $dt = 0.01$ при $dx = 0.1$ решение получилось ошибочным.

2. Схема Кранка-Николсона

Подставим пространственную Фурье-моду $\hat{T}_k^n e^{ikx_j}$ в шаблон схемы Кранка-Николсона с учётом $\hat{T}_k^{n+1} = \lambda \hat{T}_k^n$:

$$\hat{T}_k^{n+1} e^{ikx_j} = \hat{T}_k^n e^{ikx_j} + \frac{\alpha}{2}\left(\hat{T}_k^n e^{ik(x_j+\Delta x)} - 2\hat{T}_k^n e^{ikx_j} + \hat{T}_k^n e^{ik(x_j+\Delta x)}\right)$$

$$+ \frac{\alpha}{2}\left(\hat{T}_k^{n+1} e^{ik(x_j+\Delta x)} - 2\hat{T}_k^{n+1} e^{ikx_j} + \hat{T}_k^{n+1} e^{ik(x_j+\Delta x)}\right) \quad (23)$$

$$\hat{T}_k^{n+1}\left(1-\frac{\alpha}{2}\left(e^{ik\Delta x}+e^{-ik\Delta x}-2\right)\right)=\hat{T}_k^n\left(1+\frac{\alpha}{2}\left(e^{ik\Delta x}+e^{-ik\Delta x}-2\right)\right) \quad (24)$$

$$\lambda=\frac{\hat{T}_k^{n+1}}{\hat{T}_k^n}=\frac{1-\alpha(1-\cos k\Delta x)}{1+\alpha(1-\cos k\Delta x)}=\frac{1-2\alpha\sin^2\frac{k\Delta x}{2}}{1+2\alpha\sin^2\frac{k\Delta x}{2}} \quad (25)$$

Следовательно,

$$|\lambda|\leq 1\ \forall\ \Delta x,\Delta t \quad (26)$$

Таким образом, схема абсолютно устойчива.

5. Сеточная диффузия

Из формулы для сеточной диффузии $\Gamma\Delta t = -\ln|\lambda|$ имеем:

$$\Gamma = \begin{cases} -\ln\left|1 - 4\dfrac{\Delta t}{(\Delta x)^2}\sin^2\dfrac{k\Delta x}{2}\right|\dfrac{1}{\Delta t} & \text{для явной схемы} \\ -\ln\left|\dfrac{1 - 2\dfrac{\Delta t}{(\Delta x)^2}\sin^2\dfrac{k\Delta x}{2}}{1 + 2\dfrac{\Delta t}{(\Delta x)^2}\sin^2\dfrac{k\Delta x}{2}}\right|\dfrac{1}{\Delta t} & \text{для схемы Кранка} - \text{Николсона} \\ k^2 & \text{для исходного уравнения} \end{cases} \quad (27)$$

где $k_N = \frac{\pi}{\Delta x}$. С учётом равенства $\frac{k\Delta x}{2} = \frac{k}{k_N}\frac{\pi}{2}$ построены графики сеточной диффузии для шагов интегрирования по времени $\Delta t = 0.01$ и $\Delta t = 0.005$:

$dx = 0.1$, $dt = 0.01$ $\qquad$ $dx = 0.1$, $dt = 0.005$

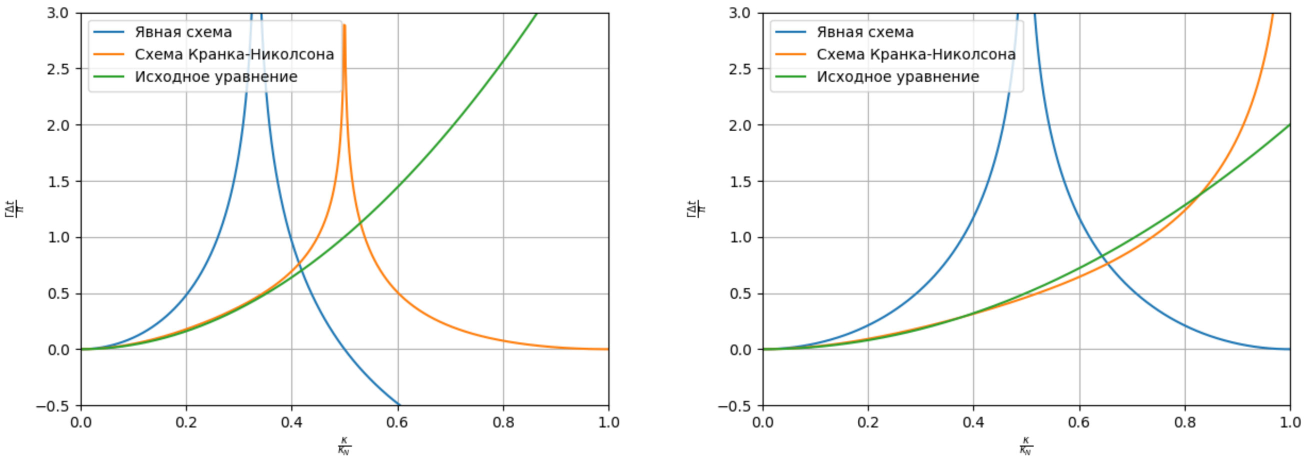


Рис. 5. Графики сеточной диффузии для явной схемы и схемы Кранка-Николсона в сравнении с точным решением

6. Листинг программы

```
from PIL import Image
from glob import glob
import numpy as np
import matplotlib.pyplot as plt
from tqdm import tqdm
import os

T_0 = (5 + 15 + 32) / 15
x_0 = 5

x_max = 10
t_max = 1


def plot_graphs(T: np.array, x: np.array, t: float,
                save: bool = False, folder: str = None):
    """
    Строит обычные графики в каждый момент времени
    :param T: решение уравнения в заданный момент времени
    :param x: массив с координатами
    :param t: массив со временем
    :param save: Если True, сохраняет графики на диск. Если False, только
показывает
    :param folder: Если save=True, должна быть указана папка. Задаётся с
функциях с решением уравнений
    :return: None
    """
    plt.figure('Plot Graph')

    plt.title(f't = {t:.3f}')

    plt.plot(x, T)
    plt.grid()

    if save:
        if folder is not None:
            if not os.path.isdir(folder):
                os.mkdir(folder)
            plt.savefig(f'{folder}/t = {t:.3f}.png')
        else:
            print('Укажи папку для сохранения графиков!')
            exit()
    plt.show(block=False)
    plt.pause(.01)
    plt.clf()


def save_gif(folder: str):
    """
    Собирает все сохранённые картинки, делает гифку и удаляет все графики,
кроме нужных моментов времени
    :param folder: Если save=True, должна быть указана папка. Задаётся с
функциях с решением уравнений
    :return: None
    """
    files = glob(f'{folder}/*.png')

    frames = []

    for file in files:
```

```
        if 't = ' in file.split("\\")[-1]:
            frame = Image.open(file)
            frames.append(frame)

    frames[0].save(f'{folder}/Solution.gif',
                   save_all=True,
                   append_images=frames[1:],
                   optimize=True,
                   duration=50,
                   loop=0)

    times_to_save = [0, 0.1, 0.2, 0.3, 0.5, 1]
    times_to_save = [f't = {times_to_save[i]:.3f}' for i in
range(len(times_to_save))]

    pbar = tqdm(total=len(files))
    pbar.set_description('Удаляю лишние графики')

    for file in files:
        if file.split("\\")[-1].replace('.png', '') not in times_to_save:
            if 't = ' in file.split("\\")[-1]:
                os.remove(file)

        pbar.update(1)


def surface_graph(T: np.array, x: np.array, t: np.array,
                  save: bool = False, folder: str = None):
    """
    Строит объёмный график поверхности решения
    :param T: решение уравнения
    :param x: массив с координатами
    :param t: массив со временем
    :param save: Если True, сохраняет графики на диск. Если False, только
показывает
    :param folder: Если save=True, должна быть указана папка. Задаётся с
функциях с решением уравнений
    :return: None
    """
    plt.figure('Surface Graph')

    X, Y = np.meshgrid(x, t)

    ax = plt.axes(projection='3d')
    ax.plot_surface(X, Y, T, cmap='plasma')
    ax.set_title('Решение уравнения теплопроводности')

    azim = -135
    ax.view_init(elev=20, azim=azim)

    ax.set_xlabel('x, m')
    ax.set_ylabel('t, s')
    ax.set_zlabel('T')

    if save:
        if folder is not None:
            if not os.path.isdir(folder):
                os.mkdir(folder)

            plt.savefig(f'{folder}/Surface.png')
        else:
            print('Укажи папку для сохранения графиков!')
            exit()
```

```
    plt.show()


def initial_conditions(x: np.array, t: np.array) -> np.array:
    """
    Применяет все начальные и граничные условия
    :param x: массив с координатами
    :param t: массив со временем
    :return: массив T
    """
    T = np.zeros((t.size, x.size))

    # Добавляем начальные условия
    T[0, :] = T_0 * (x - x_0) ** 2 * np.exp(-(x - x_0) ** 2)

    # Добавляем граничные условия
    T[:, 0] = 0
    T[:, -1] = 0

    return T


def explicit_scheme(dx: float, dt: float, save: bool = False):
    """
    Решает уравнение явной схемой
    :param dx: шаг по x
    :param dt: шаг по t
    :param save: Если True, сохраняет графики на диск. Если False, только
показывает
    :return: None
    """

    if save:
        folder = 'Explicit Scheme'
        if folder is not None:
            if not os.path.isdir(folder):
                os.mkdir(folder)
        folder = f'{folder}/dx = {dx:.3f}, dt = {dt:.3f}'
    else:
        folder = ''

    x = np.arange(0, x_max + dx, step=dx)
    t = np.arange(0, t_max + dt, step=dt)
    T = initial_conditions(x=x, t=t)

    plot_graphs(T[0], x, t[0], save=save, folder=folder)

    pbar = tqdm(total=t[:-1].size)

    for n in np.arange(t.size-1):
        for j in np.arange(1, x.size-2):
            T[n + 1, j] = T[n, j] + dt / dx ** 2 * (T[n, j + 1] - 2 * T[n, j]
+ T[n, j - 1])

        # Добавляем граничные условия
        T[:, 0] = 0
        T[:, -1] = 0

        pbar.set_description(f'Явная схема. Посчитано для t = {t[n +
1]:.3f}')
        pbar.update(1)

        plot_graphs(T[n + 1], x, t[n + 1], save=save, folder=folder)
```

```
    surface_graph(T, x, t, save=True, folder=folder)
    if save:
        save_gif(folder=folder)


def crank_nicolson_method(dx: float, dt: float, save: bool = False):
    """
    Решает уравнение методом Кранка-Николсона
    :param dx: шаг по x
    :param dt: шаг по t
    :param save: Если True, сохраняет графики на диск. Если False, только
показывает
    :return: None
    """

    if save:
        folder = 'Crank Nicolson Method'
        if folder is not None:
            if not os.path.isdir(folder):
                os.mkdir(folder)

        folder = f'{folder}/dx = {dx:.3f}, dt = {dt:.3f}'
    else:
        folder = ''

    x = np.arange(0, x_max + dx, step=dx)
    t = np.arange(0, t_max + 2 * dt, step=dt)

    alpha = dt / dx ** 2

    a = 0 * x
    a[-1] = -1

    b = 0 * x
    b[-1] = 2 * (1 / alpha + 1)

    f = 0 * x

    T = initial_conditions(x, t)

    plot_graphs(T[0], x, t[0], save=save, folder=folder)

    pbar = tqdm(total=t.size-1)

    for n in np.arange(t.size-1):
        for j in np.arange(x.size - 2, 1, -1):
            a[j - 1] = 1 / (2 * (1 + 1 / alpha) - a[j])
            f[j] = T[n, j + 1] + 2 * (1 / alpha - 1) * T[n, j] + T[n, j - 1]
            b[j - 1] = (f[j] + b[j]) / (2 * (1 / alpha + 1) - a[j])

        for j in np.arange(1, x.size-1):
            T[n + 1, j + 1] = a[j] * T[n + 1, j] + b[j]

        pbar.set_description(f'Схема Кранка-Николсона. Посчитано для t =
{t[n]:.3f}')
        pbar.update(1)

        plot_graphs(T[n], x, t[n], save=save, folder=folder)

    surface_graph(T, x, t, save=save, folder=folder)
    if save:
        save_gif(folder=folder)
```

```python
def diffusion(dx: float, dt: float, save: bool = False):
    """
    Строит графики сеточной диффузии
    :param save: Если True, сохраняет график в корневую папку
    :param dx: шаг по x
    :param dt: шаг по t
    :return: None
    """
    f = np.arange(0, t_max + dt, .001) * np.pi / 2
    G_1 = -np.log(np.abs(1 - 4 * dt / dx**2 * np.sin(f)**2)) / np.pi
    G_2 = -np.log(np.abs(1 - 2 * dt / dx**2 * np.sin(f)**2)) / ((1 + 2 * dt /
dx**2 * np.sin(f)**2) / np.pi)
    G_3 = dt / np.pi * (2 / dx * f)**2

    f = f * 2 / np.pi
    plt.plot(f, G_1, label='Явная схема')
    plt.plot(f, G_2, label='Схема Кранка-Николсона')
    plt.plot(f, G_3, label='Исходное уравнение')

    plt.grid()
    plt.legend()

    if save:
        plt.savefig(f'Diffusion dx = {dx:.1f} dt = {dt:.3f}.png')

    plt.show()


dx = .1
dt = .005

explicit_scheme(dx=dx, dt=dt, save=True)
crank_nicolson_method(dx=dx, dt=dt, save=True)
diffusion(dx, dt, save=True)
```