Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский Государственный Университет имени М.В.Ломоносова»
Физический факультет

Отчёт по практическому заданию №3

Быстрое

Студен группы №437: Белашов Егор Юрьевич (T0=2.67)

Москва - 2024

Оглавление

Теоретическое введение

Постановка задачи

Численно решить уравнение теплопроводности

$$\begin{cases} \dfrac{\partial^2 T}{\partial x^2} - \dfrac{\partial T}{\partial t} = 0 \\ T|_{x=0} = T|_{x=10} = 0 \\ T(x,t)|_{t=0} = T_0(x-x_0)^2 e^{-(x-x_0)^2} \end{cases} \qquad t \in [0;1], 0 \le x \le 10$$

- Явная схема
    - Условие устойчивости
    - Сравнение сеточной диффузии с аналитическим решением
- Схема Кранка-Николсона
    - Условие устойчивости
    - Сравнение сеточной диффузии с аналитическим решением
- Результаты
    - Для комбинаций dxdt: (0.1, 0.01), (0.1, 0.005)
    - Графики для моментов времени t: 0, 0.1, 0.2, 0.3, 0.5, 1

Схемы решения уравнения и их устойчивость

Будем обозначать индексом i – номер узла сетки по оси x, а индексом n – номер узла сетки по времени. Для нахождения условий устойчивости схемы, пропустим гармонический сигнал и поставим условие того, что множитель перехода между узлами по времени меньше единицы. Также, получим сеточную диффузию, подставив в разностные решения решение в виде $e^{i(\omega t - kx)}$ и получим связь ω и k. В аналитическом случае эта связь выглядит так: $\Gamma(k) = -i\omega(k) = k^2$. Тогда разностные схемы для явной схемы и неявной схемы Кранка-Николсона выглядят следующим образом:

Явная схема

$$T_{i,n+1} = T_{i,n} + \frac{\Delta t}{\Delta x^2}\left(T_{i+1,n} - 2T_{i,n} + T_{i-1,n}\right)$$

Устойчивость:

$$\lambda e^{ikx_i} = e^{ikx_i} + \frac{\Delta t}{\Delta x^2}\left(e^{ikx_{i+1}} - 2e^{ikx_i} + e^{ikx_{i-1}}\right)$$

$$\lambda = 1 - 4\frac{\Delta t}{\Delta x^2}\sin^2\left(\frac{k\Delta x}{2}\right) \le 1$$

$$\left(\frac{\Delta t}{\Delta x^2}\right)_{\text{явн}} \le \frac{1}{2}$$

Сеточная диффузия:

$$\Gamma(k) = -\frac{1}{\Delta t}\ln\left(1 - 4\frac{\Delta t}{\Delta x^2}\sin^2\left(\frac{k\Delta x}{2}\right)\right)$$

Неявная схема

$$T_{i,n+1} = T_{i,n} + \frac{\Delta t}{\Delta x^2}\left(T_{i+1,n+1} - 2T_{i,n+1} + T_{i-1,n+1}\right)$$

Устойчивость:

$$\lambda e^{ikx_i} = e^{ikx_i} + \frac{\Delta t}{2\Delta x^2}\left(e^{ikx_{i+1}} - 2e^{ikx_i} + e^{ikx_{i-1}}\right) + \lambda\frac{\Delta t}{2\Delta x^2}\left(e^{ikx_{i+1}} - 2e^{ikx_i} + e^{ikx_{i-1}}\right)$$

$$\lambda = \frac{1 - 4\frac{\Delta t}{\Delta x^2}\sin^2\left(\frac{k\Delta x}{2}\right)}{1 + 4\frac{\Delta t}{\Delta x^2}\sin^2\left(\frac{k\Delta x}{2}\right)} \le 1$$

Выражение выполнено при любом соотношении $\frac{\Delta t}{\Delta x^2}$, поэтому неявная схема всегда устойчива.

Сеточная диффузия:

$$\Gamma(k) = -\frac{1}{\Delta t}\ln\left(\frac{1 - 2\frac{\Delta t}{\Delta x^2}\sin^2\left(\frac{k\Delta x}{2}\right)}{1 + 2\frac{\Delta t}{\Delta x^2}\sin^2\left(\frac{k\Delta x}{2}\right)}\right)$$

Метод прогонки

Для решения по неявной схеме, необходимо на каждом шаге составлять СЛАУ и решать его, матрица этой системы выглядит так:

$$\begin{pmatrix} b_0 & c_0 & & & 0 \\ a_1 & b_1 & c_1 & & \\ & \ddots & \ddots & \ddots & \\ & & a_{I-1} & b_{I-1} & c_{I-1} \\ 0 & & & a_I & b_I \end{pmatrix} \begin{pmatrix} T_{0,n} \\ T_{1,n} \\ \vdots \\ T_{I-1,n} \\ T_{I,n} \end{pmatrix} = \begin{pmatrix} T_{0,n+1} \\ T_{1,n+1} \\ \vdots \\ T_{I-1,n+1} \\ T_{I,n+1} \end{pmatrix}$$

Сделав некоторые вычисления, можно получить две рекурсивные зависимости:

$$\alpha_i = -\frac{c_i}{a_i\alpha_{i-1} + b_i} \quad \beta_i = \frac{d_i - a_i\beta_{i-1}}{a_i\alpha_{i-1} + b_i}$$

$$\alpha_0 = -\frac{c_0}{b_0} \quad \beta_0 = \frac{d_0}{b_0}$$

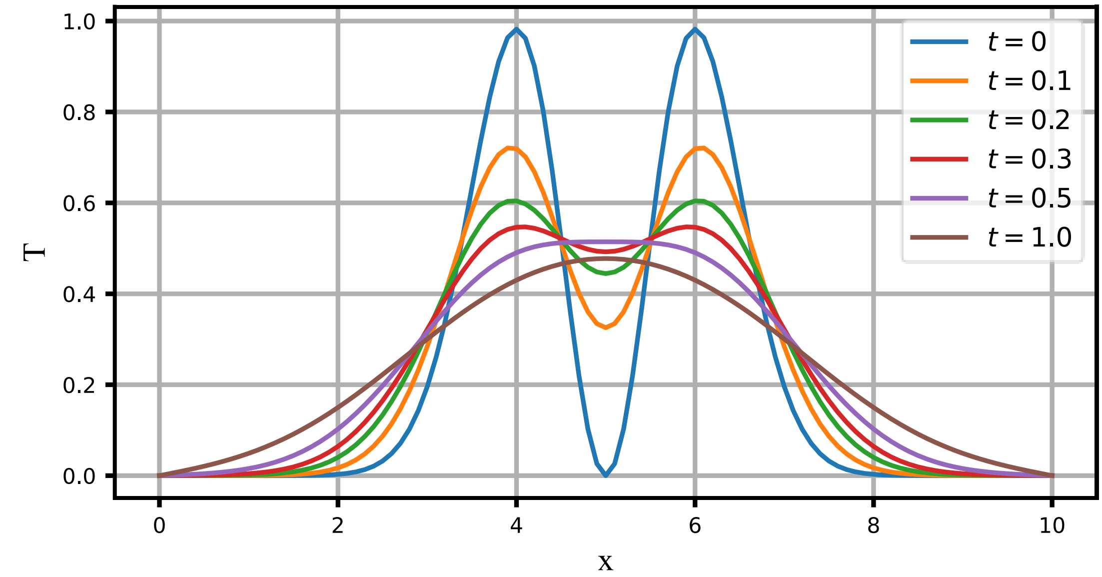
Через которые выражаются значения на новом слое:

$$T_{I,n+1} = \beta_I = \frac{d_I - a_I\beta_{I-1}}{a_I\alpha_{I-1} + b_I} \quad T_{i,n+1} = \alpha_i T_{i+1,n+1} + \beta_i$$

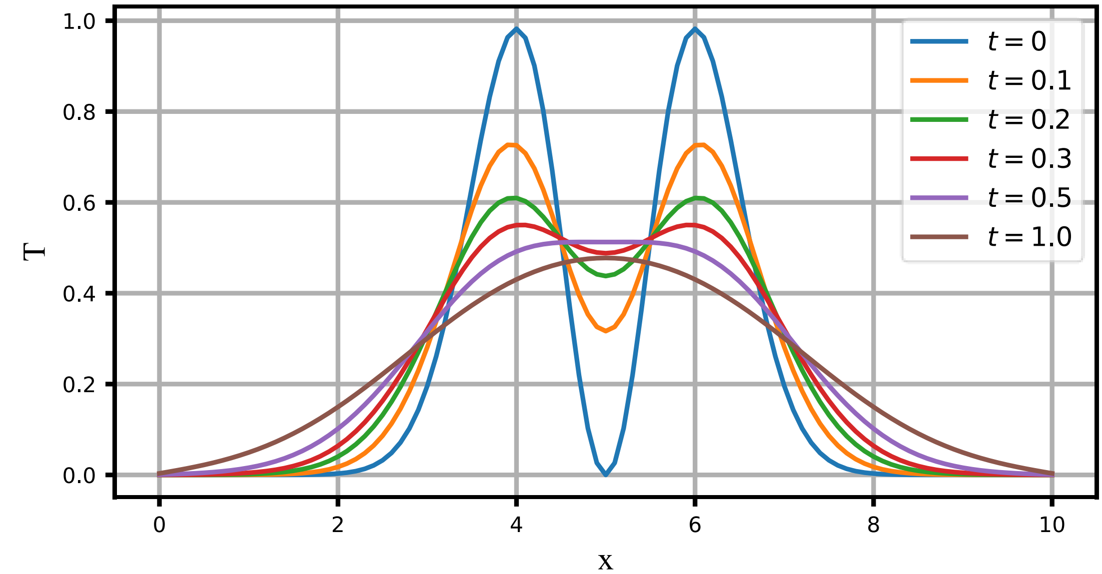
Результаты

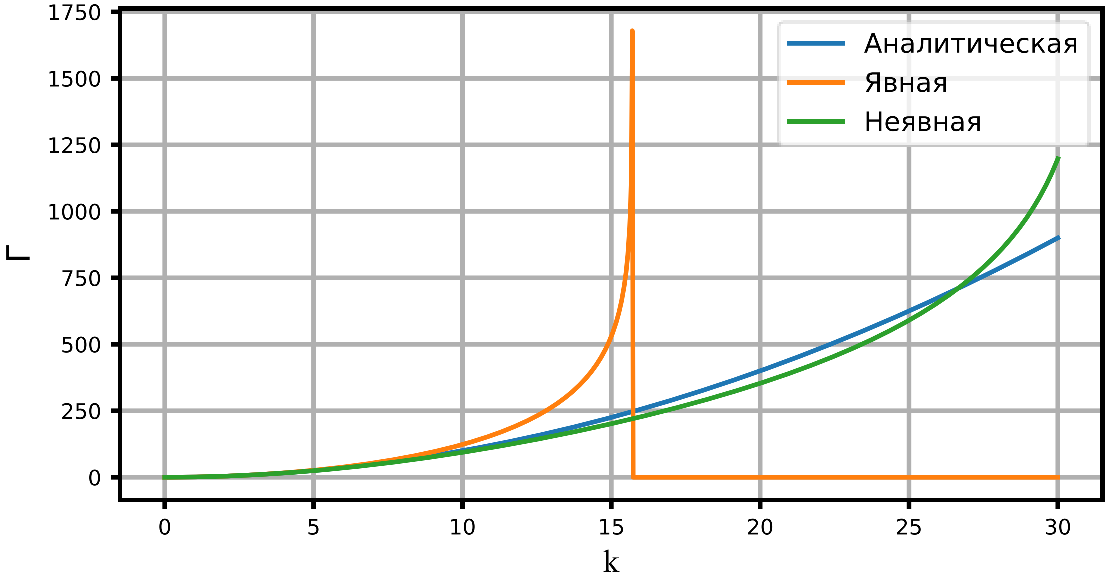
Решение уравнения теплопроводности

Явная схема



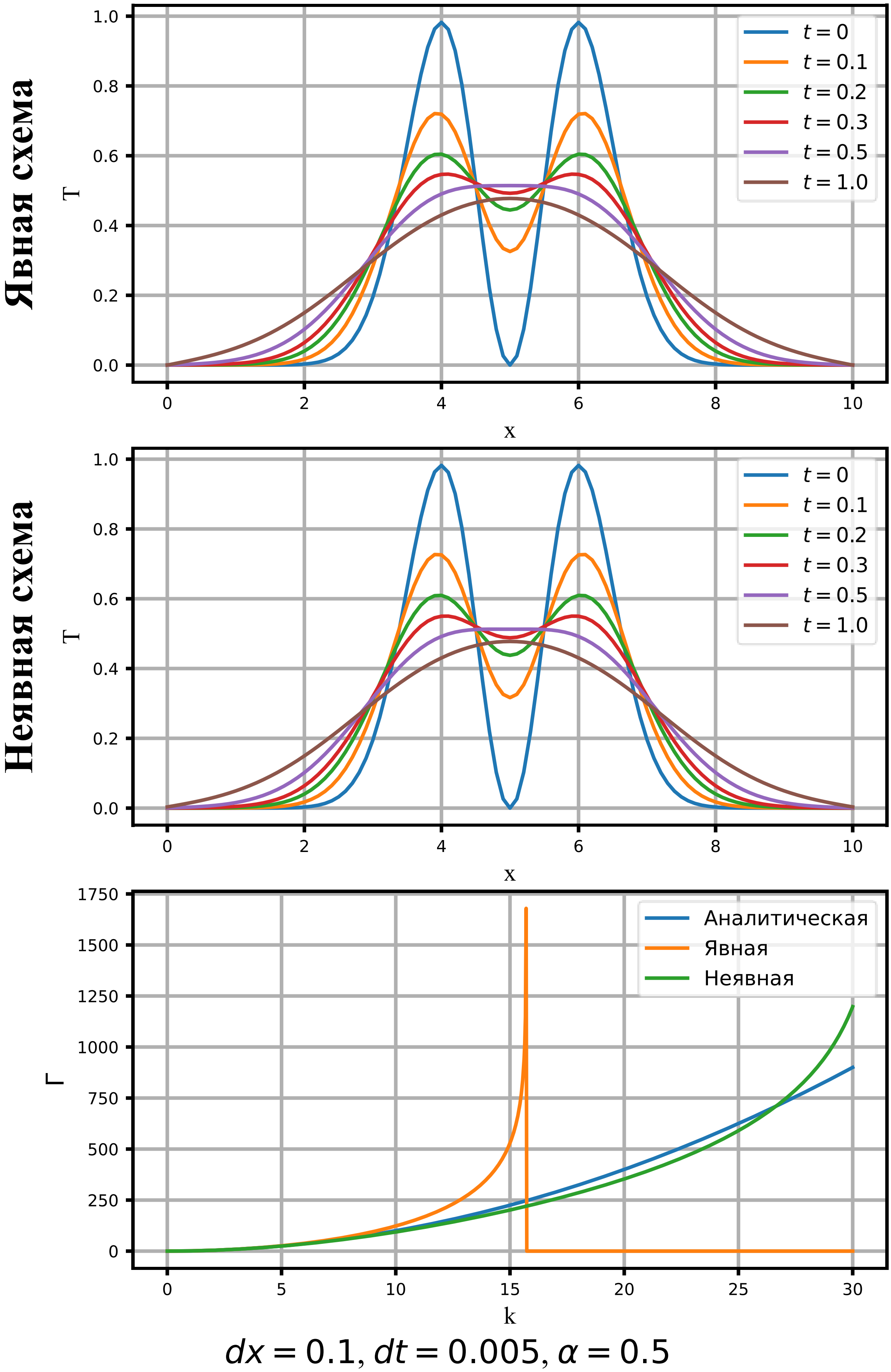
Неявная схема





$dx = 0.1, dt = 0.005, \alpha = 0.5$

Решение уравнения теплопроводности



Листинг программы

[1]:
```python
import numpy
from belashovplot import TiledPlot
from tqdm import tqdm
from copy import deepcopy
```

[2]:
```python
def explicit(initial:numpy.ndarray, alpha:float):
    initial = deepcopy(initial)
    x_dependence = deepcopy(initial[0])
    for i in range(1, initial.shape[0]):
        shifted_dependencies = [x_dependence[0:-2], x_dependence[1:-1], x_dependence[2:]]
        x_dependence[1:-1] += alpha*(shifted_dependencies[0] - 2*shifted_dependencies[1] + shifted_dependencies[2])
        x_dependence[0] = initial[i][0]
        x_dependence[-1] = initial[i][-1]
        initial[i] = deepcopy(x_dependence)
    return initial
```

[3]:
```python
def implicit(initial:numpy.ndarray, alpha:float):
    initial = deepcopy(initial)
    a_array = numpy.ones(initial.shape[1]) * (-alpha)
    b_array = numpy.ones(initial.shape[1]) * (1 + 2*alpha)
    c_array = numpy.ones(initial.shape[1]) * (-alpha)
    a_array[0] = 0
    c_array[-1] = 0

    d_array = deepcopy(initial[0])
    for n in range(1, initial.shape[0]):
        alpha_array = numpy.zeros(initial.shape[1]-1)
        beta_array = numpy.zeros(initial.shape[1]-1)
        alpha_array[0] = -c_array[0] / b_array[0]
        beta_array[0] = d_array[0] / b_array[0]
        for i in range(1, initial.shape[1]-1):
            alpha_array[i] = - c_array[i] / (a_array[i]*alpha_array[i-1] + b_array[i])
            beta_array[i] = (d_array[i] - a_array[i]*beta_array[i-1]) / (a_array[i]*alpha_array[i-1] + b_array[i])
        d_array[-1] = (d_array[-1] - a_array[-1]*beta_array[-1]) / (a_array[-1]*alpha_array[-1] + b_array[-1])
        for i in reversed(range(0, initial.shape[1]-1)):
            d_array[i] = alpha_array[i]*d_array[i+1] + beta_array[i]
        initial[n] = deepcopy(d_array)
    return initial
```

[4]:
```python
def initial_field(nx:int, nt:int, T0:float=2.67, Tx0:float=0.0, Tx1:float=0.0, x0:float=0.0, x1:float=10.0, t:float=1.0):
    x_space = numpy.linspace(x0, x1, nx)
    t_space = numpy.linspace(0, t, nt)
    dx = x_space[1] - x_space[0]
    dt = t_space[1] - t_space[0]
    initial = numpy.zeros((nt, nx))
    initial[:, 0] = Tx0
    initial[:, -1] = Tx1
    initial[0] = T0 * ((x_space - 5)**2) * numpy.exp(-(x_space - 5)**2)
    return initial, dt, dx
```

[5]:
```python
def get_by_time(result:numpy.ndarray, t:float=1.0, *timings):
    dt = t / (result.shape[0] - 1)
    x_dependencies = []
    for timing in timings:
        x_dependencies.append(result[int(timing/dt)])
    return x_dependencies
```

Результат для $(dx, dt) = (0.1, 0.01)$

[6]:
```python
time_points = (0, 0.1, 0.2, 0.3, 0.5, 1.0)
nx = 101
nt = 101
x_space = numpy.linspace(0, 10, nx)
initial, dt, dx = initial_field(nx, nt)
alpha = dt/(dx**2)
print(dx, dt, alpha)
```

```
0.1 0.01 0.9999999999999998
```

[7]:
```python
plot = TiledPlot(7.3, 9.7)
plot.title("Решение уравнения теплопроводности")
plot.description.bottom(f"$dx={dx}$, $dt={dt}$, $\\alpha={round(alpha, 2)}$")
plot.description.row.left("Явная схема", 0)
plot.description.row.left("Неявная схема", 1)
plot.width_to_height(2.0)

result = explicit(initial, alpha)
curves = get_by_time(result, 1.0, *time_points)
axes = plot.axes.add(0,0)
axes.grid(True)
# axes.plot(result[0])
for curve, time in zip(curves, time_points):
    axes.plot(x_space, curve, label=f"$t={round(time, 1)}$")
axes.legend()
plot.graph.label.x("x")
plot.graph.label.y("T")

result = implicit(initial, alpha)
curves = get_by_time(result, 1.0, *time_points)
axes = plot.axes.add(0,1)
axes.grid(True)
# axes.plot(result[0])
for curve, time in zip(curves, time_points):
    axes.plot(x_space, curve, label=f"$t={round(time, 1)}$")
axes.legend()
plot.graph.label.x("x")
plot.graph.label.y("T")

k_space = numpy.linspace(0, 30, 1000)
analitic_G = k_space**2
temp = 1 - 4*dt*(numpy.sin(k_space*dx/2)**2)/(dx**2)
explicit_G = -numpy.log(numpy.where(temp > 0, temp, 1))/dt
temp1 = 1 - 2*dt*(numpy.sin(k_space*dx/2)**2)/(dx**2)
temp2 = 1 + 2*dt*(numpy.sin(k_space*dx/2)**2)/(dx**2)
temp = temp1 / temp2
implicit_G = -numpy.log(numpy.where(temp > 0, temp, 1))/dt
axes = plot.axes.add(0,2)
axes.grid(True)
axes.plot(k_space, analitic_G, label='Аналитическая')
axes.plot(k_space, explicit_G, label='Явная')
axes.plot(k_space, implicit_G, label='Неявная')
axes.legend()
plot.graph.label.x("k")
plot.graph.label.y("$\Gamma$")

plot.save("../figures/HW3G1.svg")
plot.show()
```

Результат для $(dx, dt) = (0.1, 0.001)$

[8]:
```python
time_points = (0, 0.1, 0.2, 0.3, 0.5, 1.0)
nx = 101
nt = 201
x_space = numpy.linspace(0, 10, nx)
initial, dt, dx = initial_field(nx, nt)
alpha = dt/(dx**2)
print(dx, dt, alpha)
```

```
0.1 0.005 0.49999999999999999
```

[10]:
```python
plot = TiledPlot(7.3, 9.7)
plot.title("Решение уравнения теплопроводности")
plot.description.bottom(f"$dx={dx}$, $dt={dt}$, $\\alpha={round(alpha, 2)}$")
plot.description.row.left("Явная схема", 0)
plot.description.row.left("Неявная схема", 1)
plot.width_to_height(2.0)

result = explicit(initial, alpha)
curves = get_by_time(result, 1.0, *time_points)
axes = plot.axes.add(0,0)
axes.grid(True)
# axes.plot(result[0])
for curve, time in zip(curves, time_points):
    axes.plot(x_space, curve, label=f"$t={round(time, 1)}$")
axes.legend()
plot.graph.label.x("x")
plot.graph.label.y("T")

result = implicit(initial, alpha)
curves = get_by_time(result, 1.0, *time_points)
axes = plot.axes.add(0,1)
axes.grid(True)
# axes.plot(result[0])
for curve, time in zip(curves, time_points):
    axes.plot(x_space, curve, label=f"$t={round(time, 1)}$")
axes.legend()
plot.graph.label.x("x")
plot.graph.label.y("T")

k_space = numpy.linspace(0, 30, 1000)
analitic_G = k_space**2
temp = 1 - 4*dt*(numpy.sin(k_space*dx/2)**2)/(dx**2)
explicit_G = -numpy.log(numpy.where(temp > 0, temp, 1))/dt
temp1 = 1 - 2*dt*(numpy.sin(k_space*dx/2)**2)/(dx**2)
temp2 = 1 + 2*dt*(numpy.sin(k_space*dx/2)**2)/(dx**2)
temp = temp1 / temp2
implicit_G = -numpy.log(numpy.where(temp > 0, temp, 1))/dt
axes = plot.axes.add(0,2)
axes.grid(True)
axes.plot(k_space, analitic_G, label='Аналитическая')
axes.plot(k_space, explicit_G, label='Явная')
axes.plot(k_space, implicit_G, label='Неявная')
axes.legend()
plot.graph.label.x("k")
plot.graph.label.y("$\Gamma$")

plot.save("../figures/HW3G2.svg")
plot.show()
```