

**Московский авиационный институт**  
(национальный исследовательский университет)

Институт №8 "Информационные технологии и прикладная математика"  
Кафедра 806 «Вычислительная математика и программирование»

**Лабораторная работа №5**  
по дисциплине:  
**Численные методы**  
Вариант №5

Выполнил: студент группы М8О-409Б-20  
Искренкова А.В.  
Принял: Пивоваров Е.Д.  
Оценка: \_\_\_\_\_

Москва, 2023 г.

# 1. Задание

Используя явную и неявную конечно-разностные схемы, а также схему Кранка - Николсона, решить начально-краевую задачу для дифференциального уравнения параболического типа. Осуществить реализацию трех вариантов аппроксимации граничных условий, содержащих производные: двухточечная аппроксимация с первым порядком, трехточечная аппроксимация со вторым порядком, двухточечная аппроксимация со вторым порядком. В различные моменты времени вычислить погрешность численного решения путем сравнения результатов с приведенным в задании аналитическим решением  $U(x, t)$ . Исследовать зависимость погрешности от сеточных параметров  $\tau, h$ .

Уравнение:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \sin(\pi x)$$

$$u(0, t) = 0$$

$$u(1, t) = 0$$

$$u(x, 0) = 0$$

Аналитическое решение:

$$U(x, t) = \frac{1}{\pi^2} (1 - \exp(-\pi^2 t)) \sin(\pi x)$$

# 2. Решение

- Явная схема:

$$\frac{u_j^{k+1} - u_j^k}{\tau} = \frac{u_{j-1}^k - 2u_j^k + u_{j+1}^k}{h^2} + f(x_j, t^k)$$

- Неявная схема:

$$\frac{u_j^{k+1} - u_j^k}{\tau} = \frac{u_{j-1}^{k+1} - 2u_j^{k+1} + u_{j+1}^{k+1}}{h^2} + f(x_j, t^{k+1})$$

- Гибридная схема (при  $\theta = 0.5$ , схема Кранка-Николсона)

$$\begin{aligned} \frac{u_j^{k+1} - u_j^k}{\tau} = & (\theta) \left( \frac{u_{j-1}^{k+1} - 2u_j^{k+1} + u_{j+1}^{k+1}}{h^2} + f(x_j, t^k) \right) \\ & + (1 - \theta) \left( \frac{u_{j-1}^k - 2u_j^k + u_{j+1}^k}{h^2} + f(x_j, t^k) \right) \end{aligned}$$

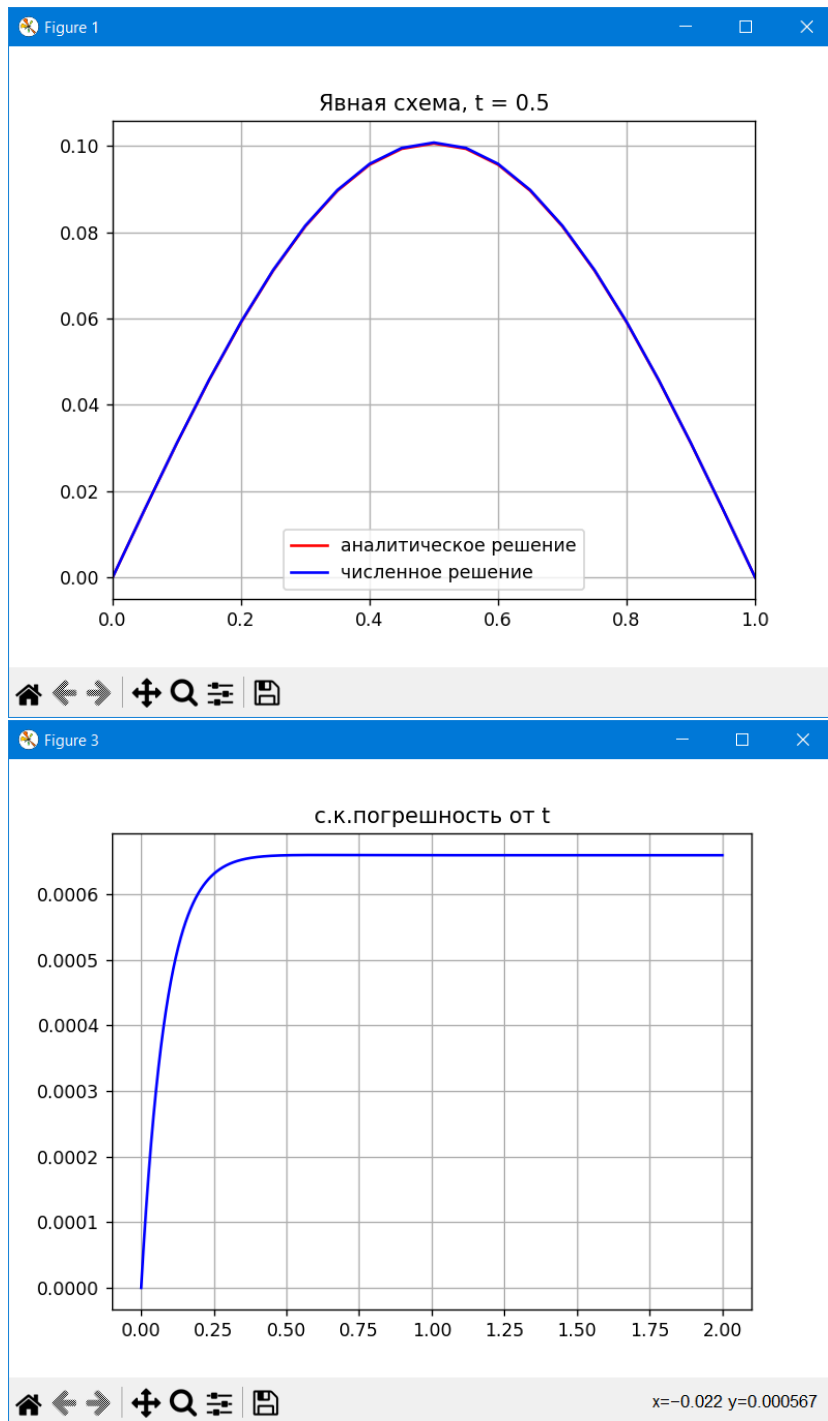
Для решения задачи не используется аппроксимация граничных условий, так как они нулевые.

Погрешность между численным и аналитическим решением рассчитывается как среднеквадратическая.

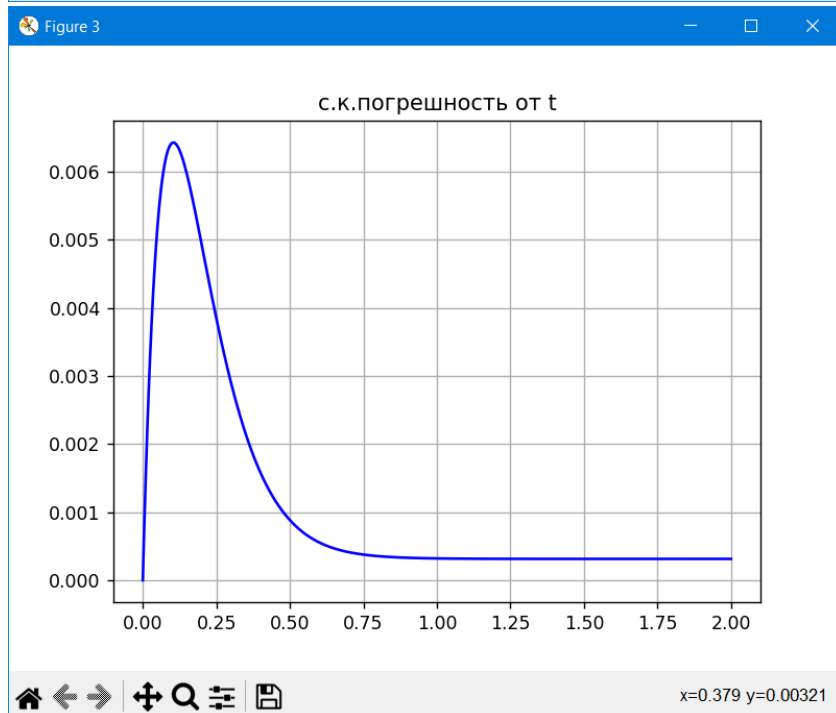
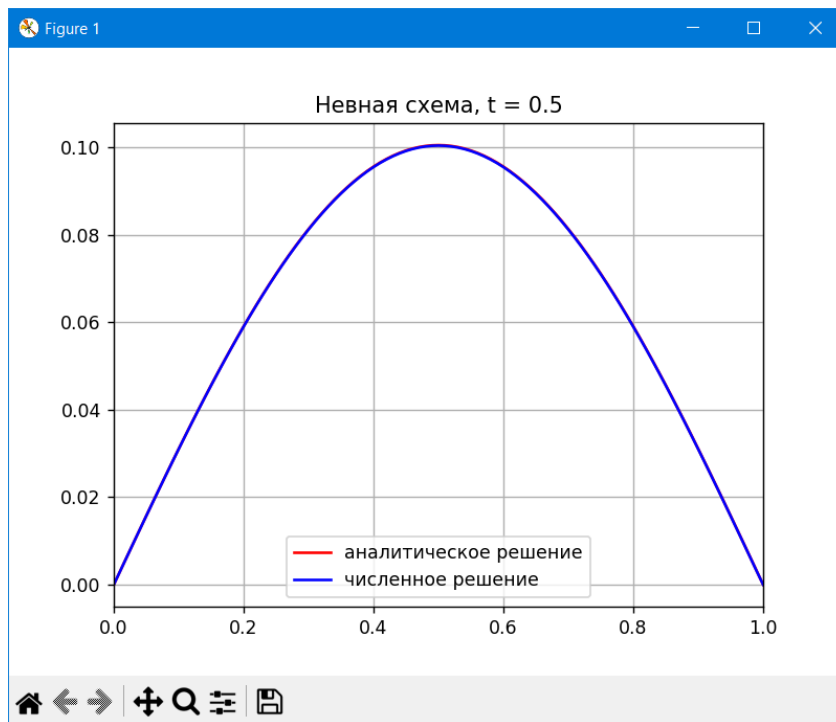
Параметры задачи:  $T = 2, h = 0.05, \tau = 0.0005$ .

# 3. Вывод программы

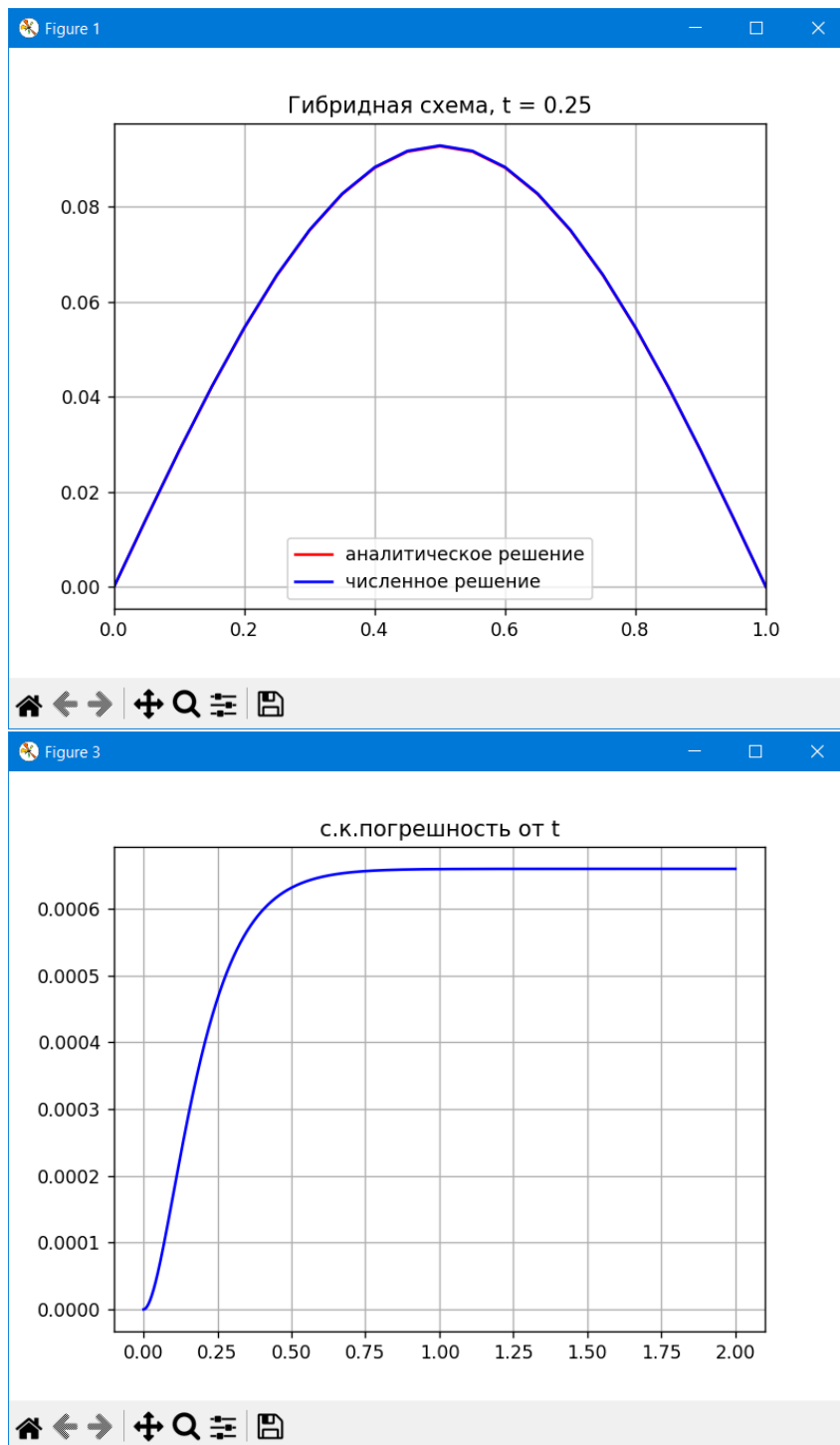
- Явная схема



- Неявная схема



- Схема Кранка-Николсона



#### 4. Листинг

```

5.     import matplotlib
6.     import numpy as np
7.     import math
8.     import matplotlib.pyplot as plt
9.     from math import sqrt
10.
11.     def psi(x):
12.         return 0
13.
14.     def phi0(t):
15.         return 0
16.
17.     def phi1(t):
18.         return 0
19.
20.     def true_fval(x, t):
21.         return (1 / (math.pi ** 2)) * (1 - np.exp((-math.pi ** 2) * t)) *
22.         np.sin(math.pi * x)
23.
24.     def f(x):
25.         return np.sin(math.pi * x)
26.
27.     def norma(a):
28.         norm = 0
29.         for i in range(len(a)):
30.             norm += a[i] ** 2
31.         return sqrt(norm)
32.
33.     # метод прогонки
34.     def tridiagonal(a, b, c, d):
35.         n = len(d)
36.         x = np.zeros(n)
37.         p = [-c[0] / b[0]]
38.         q = [d[0] / b[0]]
39.         for i in range(1, n):
40.             p.append(-c[i] / (b[i] + a[i] * p[i - 1]))
41.             q.append((d[i] - a[i] * q[i - 1]) / (b[i] + a[i] * p[i - 1]))
42.         x[-1] = q[-1]
43.         for i in reversed(range(n - 1)):
44.             x[i] = p[i] * x[i + 1] + q[i]
45.         return x
46.
47.     def ExScheme(a, lb, ub, h, tau, T):
48.         # разбиение осей
49.         x = np.arange(lb, ub + h, h)
50.         t = np.arange(0, T + tau, tau)
51.         # строим конечно-разностную сетку
52.         U = np.zeros((len(t), len(x)))
53.         # заполним первый уровень
54.         for j in range(len(x)):
55.             U[0, j] = psi(x[j])

```

```

55.     # прямая схема
56.     for i in range(1, len(t)):
57.         for j in range(1, len(x)-1):
58.             U[i, j] = a*tau/(h**2)*U[i-1,j-1]+(1-2*a*tau/(h**2))*U[i-
23.             1,j]+a*tau/(h**2)*U[i-1,j+1]+tau*f(x[j])
59.             U[i, 0] = phi0(t[i])
60.             U[i, -1] = phi1(t[i])
61.
62.     return U
63.
64. def ImScheme(a, lb, ub, h, tau, T):
65.     x = np.arange(lb, ub + h, h)
66.     t = np.arange(0, T + tau, tau)
67.     U = np.zeros((len(t), len(x)))
68.     for j in range(len(x)):
69.         U[0, j] = psi(x[j])
70.
71.     for i in range(1, len(t)):
72.         aa = np.zeros(len(x)-2)
73.         bb = np.zeros(len(x)-2)
74.         cc = np.zeros(len(x)-2)
75.         dd = np.zeros(len(x)-2)
76.         dd[0] = -(U[i - 1, 1] + a * tau / (h ** 2) * phi0(t[i])) - tau
23.         * f(x[1])
77.         dd[-1] = -(U[i - 1, len(x) - 1] + a * tau / (h ** 2) *
23.         phi1(t[i])) - tau * f(x[len(x) - 2])
78.         bb[0] = -(1 + 2 * a * tau / (h ** 2))
79.         bb[-1] = -(1 + 2 * a * tau / (h ** 2))
80.         cc[0] = a * tau / (h ** 2)
81.         aa[-1] = a * tau / (h ** 2)
82.         for j in range(1, len(x) - 3):
83.             aa[j] = a * tau / (h ** 2)
84.             bb[j] = -(1 + 2 * a * tau / (h ** 2))
85.             cc[j] = a * tau / (h ** 2)
86.             dd[j] = -U[i - 1, j+1] - tau * f(x[j+1])
87.         xx = tridiagonal(aa, bb, cc, dd)
88.         for j in range(1, len(x)-1):
89.             U[i, j] = xx[j-1]
90.
91.     return U
92.
93. def Hybrid(a, lb, ub, h, tau, T, teta):
94.     x = np.arange(lb, ub + h, h)
95.     t = np.arange(0, T + tau, tau)
96.     U = np.zeros((len(t), len(x)))
97.     for j in range(len(x)):

```

```

98.         U[0, j] = psi(x[j])
99.
100.    for i in range(1, len(t)):
101.        aa = np.zeros(len(x) - 2)
102.        bb = np.zeros(len(x) - 2)
103.        cc = np.zeros(len(x) - 2)
104.        dd = np.zeros(len(x) - 2)
105.        dd[0] = -(1-teta)*a*tau/(h**2)*U[i-1,0]-(1-(1-
teta)*2*a*tau/(h**2))*U[i-1,1]-(1-teta)*a*tau/(h**2)*U[i-1,2]\
106.            -a*tau/(h**2)*teta*phi0(t[i])-tau*f(x[1])
107.        dd[-1] = -(1-teta)*a*tau/(h**2)*U[i-1,len(x)-3]-(1-(1-
teta)*2*a*tau/(h**2))*U[i-1,len(x)-2]-(1-teta)*a*tau/(h**2)*U[i-
1,len(x)-1]\
108.            -a*tau/(h**2)*teta*phi1(t[i])-tau*f(x[len(x)-2])
109.        bb[0] = -(1 + 2 * a * tau / (h ** 2)*teta)
110.        bb[-1] = -(1 + 2 * a * tau / (h ** 2)*teta)
111.        cc[0] = a * tau / (h ** 2)*teta
112.        aa[-1] = a * tau / (h ** 2)*teta
113.        for j in range(1, len(x) - 3):
114.            aa[j] = a * tau / (h ** 2)*teta
115.            bb[j] = -(1 + 2 * a * tau / (h ** 2)*teta)
116.            cc[j] = a * tau / (h ** 2)*teta
117.            dd[j] = -(1-teta)*a*tau/(h**2)*U[i-1,j]-(1-(1-
teta)*2*a*tau/(h**2))*U[i-1,j+1]-(1-teta)*a*tau/(h**2)*U[i-1,j+2]-
tau*f(x[j+1])
118.        xx = tridiagonal(aa, bb, cc, dd)
119.        for j in range(1, len(x) - 1):
120.            U[i, j] = xx[j - 1]
121.
122.    return U
123.
124. def plot_ex(a, lb, ub, h, tau, T, k):
125.     x = np.arange(lb, ub + h, h)
126.     t = np.arange(0, T + tau, tau)
127.     plt.figure(1)
128.     plt.title('Явная схема, t = ' + str(t[k]))
129.     plt.grid()
130.     plt.plot(x, true_fval(x, t[k]), color='red', label='аналитическое
решение')
131.     U = ExScheme(a, lb, ub, h, tau, T)
132.     plt.plot(x, U[k, :], color='blue', label='численное решение')
133.     plt.legend()
134.     plt.xlim((0, ub))
135.     plt.figure(2)
136.     plt.title('Погрешность на срезе t =' + str(t[k])+' от x')
137.     plt.grid()
138.     eps = []
139.     for i in range(len(x)):
140.         a = np.abs(true_fval(x[i], t[k]) - U[i, :])
141.         eps = np.append(eps, a)
142.     plt.plot(x, a, color='green')

```



```

143.     plt.figure(3)
144.     plt.title('с.к.погрешность от t')
145.     plt.grid()
146.     eps = []
147.     for i in range(len(t)):
148.         a = true_fval(x,t[i]) - U[i, :]
149.         eps = np.append(eps, norma(a))
150.     plt.plot(t, eps, color='blue')
151.
152.     plt.show()
153.     return
154.
155. def plot_im(a, lb, ub, h, tau, T, k):
156.     x = np.arange(lb, ub + h, h)
157.     t = np.arange(0, T + tau, tau)
158.     plt.figure(1)
159.     plt.title('Невная схема, t = ' + str(t[k]))
160.     plt.grid()
161.     plt.plot(x, true_fval(x, t[k]), color='red', label='аналитическое
решение')
162.     U = ImScheme(a, lb, ub, h, tau, T)
163.     plt.plot(x, U[k, :], color='blue', label='численное решение')
164.     plt.legend()
165.     plt.xlim((0, ub))
166.     plt.figure(2)
167.     plt.title('Погрешность на срезе t =' + str(t[k])+' от x')
168.     plt.grid()
169.     eps = []
170.     for i in range(len(x)):
171.         a = np.abs(true_fval(x[i], t[k]) - U[i, :])
172.         eps = np.append(eps, a)
173.     plt.plot(x, a, color='green')
174.     plt.figure(3)
175.     plt.title('с.к.погрешность от t')
176.     plt.grid()
177.     eps = []
178.     for i in range(len(t)):
179.         a = true_fval(x, t[i]) - U[i, :]
180.         eps = np.append(eps, norma(a))
181.     plt.plot(t, eps, color='blue')
182.
183.     plt.show()
184.     return
185.
186. def plot_Hybrid(a, lb, ub, h, tau, T, k,teta):
187.     x = np.arange(lb, ub + h, h)
188.     t = np.arange(0, T + tau, tau)
189.     plt.figure(1)
190.     plt.title('Гибридная схема, t = ' + str(t[k]))
191.     plt.grid()

```

```

192.     plt.plot(x, true_fval(x, t[k]), color='red', label='аналитическое
        решение')
193.     U = Hybrid(a, lb, ub, h, tau, T, teta)
194.     plt.plot(x, U[k, :], color='blue', label='численное решение')
195.     plt.legend()
196.     plt.xlim((0, ub))
197.     plt.figure(2)
198.     plt.title('Погрешность на срезе t =' + str(t[k]) + ' от x')
199.     plt.grid()
200.     eps = []
201.     for i in range(len(x)):
202.         a = np.abs(true_fval(x[i], t[k]) - U[i, :])
203.         eps = np.append(eps, a)
204.     plt.plot(x, a, color='green')
205.     plt.figure(3)
206.     plt.title('с.к.погрешность от t')
207.     plt.grid()
208.     eps = []
209.     for i in range(len(t)):
210.         a = true_fval(x, t[i]) - U[i, :]
211.         eps = np.append(eps, norma(a))
212.     plt.plot(t, eps, color='blue')
213.
214.     plt.show()
215.     return
216.
217. plot_ex(1,0,1,0.05,0.0005,2,1000)
218. plot_im(1, 0, 1, 0.01, 0.005, 2, 100)
219. plot_Hybrid(1,0,1,0.05,0.0005,2,500,0.5)
220.

```

## 5. Вывод

В ходе выполнения лабораторной работы были изучены явная, неявная и гибридная схемы решений начально-краевой задачи для дифференциального уравнения параболического типа. Также были изучены три варианта аппроксимации граничных условий. Были получены результаты в графическом представлении и подсчитаны погрешности для каждого варианта решения.