Московский авиационный институт

(национальный исследовательский университет)

Институт № 8 «Информационные технологии и прикладная математика»

Лабораторная работа № 5

По курсу «Численные методы»

«Численное решение уравнений с частными производными параболического типа»

Выполнила:

Офицерова Т.И.

Группа:

М8О-409Б-19

Преподаватель:

Пивоваров Д.Е.

Москва, 2022

Задача

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 τ, h.

Описание метода

Классический пример уравнения параболического типа – уравнение теплопроводности, которое в одномерном по пространству случае имеет вид:

$$\frac{\partial u}{\partial t} = a^2 \frac{\partial^2 u}{\partial x^2}, \quad 0 < x < l, \ t > 0 .$$

Если на границах $x=0$ и $x=l$ заданы значения искомой функции $u(x,t)$ *в виде*

$$u(0,t) = \varphi_0(t), \quad x = 0, \quad t > 0;$$

$$u(l,t) = \varphi_l(t), \quad x = l, \quad t > 0,$$

- граничные условия первого рода.

И заданы начальные условия:

$$u(x,0) = \psi(x), \quad 0 \le x \le l, \quad t = 0$$

Также на границах могут быть заданы значения производных искомой функции по пространственной переменной, то есть граничные условия второго рода:

$$\frac{\partial u(0,t)}{\partial x} = \varphi_0(t), \quad x = 0, \quad t > 0;$$

$$\frac{\partial u(l,t)}{\partial x} = \varphi_l(t), \quad x = l, \quad t > 0,$$

Или граничные условия третьего рода, то есть линейные комбинации искомой функции и ее производной по пространственной переменной:

$$\alpha \frac{\partial u(0,t)}{\partial x} + \beta u(0,t) = \varphi_0(t), \quad x = 0, \quad t > 0;$$

$$\gamma \frac{\partial u(l,t)}{\partial x} + \delta u(l,t) = \varphi_l(t), \quad x = l, \quad t > 0,$$

Для решения такой задачи применяют метод конечных разностей.

Для этого вводится понятие разностной сетки с пространственным шагом h и временным τ

$$\omega_{h\tau} = \{x_j = jh, \ j = \overline{0,N}; \ t^k = k\tau, \ k = \overline{0,K}\}$$

Также вводится понятие сеточной функции – однозначное отображение целых аргументов j, k в значения функции $u_j^k = u(x_j, t_k)$.

Вводится два временных слоя: нижний $t^k = k\tau$, на котором распределение искомой функции известно, и верхний $t^{k+1} = (k+1)\tau$, на котором распределение искомой функции подлежит определению.

Далее аппроксимируем дифференциальные операторы отношением конечных разностей.

При аппроксимации второй производной по пространству на нижнем временном слое, получаем явную конечно-разностную схему.

$$\left.\frac{\partial u}{\partial t}\right|_j^k = \frac{u_j^{k+1} - u_j^k}{\tau} + O(\tau),$$

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_j^k = \frac{u_{j+1}^k - 2u_j^k + u_{j-1}^k}{h^2} + O(h^2).$$

Явная конечно-разностная схема:

$$\frac{u_j^{k+1} - u_j^k}{\tau} = a^2 \frac{u_{j+1}^k - 2u_j^k + u_{j-1}^k}{h^2} + O(\tau + h^2), \quad j = \overline{1, N-1}, \quad k = \overline{0, K-1},$$

$$u_0^k = \varphi_0(t^k), \quad u_N^k = \varphi_l(t^k), \quad k = \overline{0, K}; \quad u_j^0 = \psi(x_j), \quad j = \overline{0, N},$$

где для каждого уравнения неизвестна только одна величина u_j^{k+1}, которую можно явно выразить:

$$u_j^{k+1} = \sigma \cdot u_{j+1}^k + (1 - 2\sigma) u_j^k + \sigma \cdot u_{j-1}^k, \quad \sigma = \frac{a^2 \tau}{h^2}, \quad j = \overline{1, N-1}, \quad k = 0, 1, 2 \ldots,$$

Данная схема будет устойчива при условии σ <= ½.

При аппроксимации второй производной по пространству на верхнем временном слое, получаем неявную конечно-разностную схему.

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_j^{k+1} = \frac{u_{j+1}^{k+1} - 2u_j^{k+1} + u_{j-1}^{k+1}}{h^2} + O(h^2),$$

Неявная конечно-разностная схема:

$$\frac{u_j^{k+1} - u_j^k}{\tau} = a^2 \frac{u_{j+1}^{k+1} - 2u_j^{k+1} + u_{j-1}^{k+1}}{h^2} + O(\tau + h^2), \quad j = \overline{1, N-1}, \quad k = \overline{0, K-1},$$

$$u_0^{k+1} = \varphi_0(t^{k+1}), \quad u_N^{k+1} = \varphi_l(t^{k+1}), \quad k = \overline{0, K-1}; \quad u_j^0 = \psi(x_j), \quad j = \overline{0, N}.$$

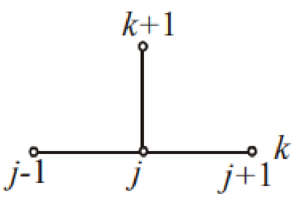
где для нахождения сеточной функции на верхнем временном слое необходимо решить СЛАУ с трехдиагональной матрицей:

$$\begin{array}{l} a_1 = 0; \\ \\ c_{N-1} = 0; \end{array} \begin{cases} b_1 u_1^{k+1} + c_1 u_2^{k+1} = d_1, \quad j = 1 \\ a_j u_{j-1}^{k+1} + b_j u_j^{k+1} + c_j u_{j+1}^{k+1} = d_j, \quad j = \overline{2, N-2} \\ a_{N-1} u_{N-2}^{k+1} + b_{N-1} u_{N-1}^{k+1} = d_{N-1}, \quad j = N-1, \end{cases}$$

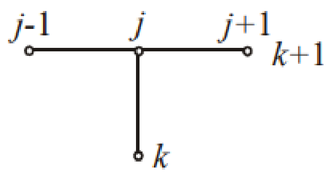
где $a_j = \sigma, j = \overline{2, N-1}; \quad b_j = -(1 + 2\sigma), \quad j = \overline{1, N-1}; \quad c_j = \sigma, \quad j = \overline{1, N-2};$

$$d_j = -u_j^k, \quad j = \overline{2, N-2}; \quad d_1 = -(u_1^k + \sigma\varphi_0(t^{k+1})); \quad d_{N-1} = -(u_{N-1}^k + \sigma\varphi_l(t^{k+1})); \quad \sigma = \frac{a^2\tau}{h^2}.$$

Приведем шаблоны конечно-разностных схем (их геометрическую интерпретацию на конечно-разностной сетке):



шаблон явной схемы



шаблон неявной схемы

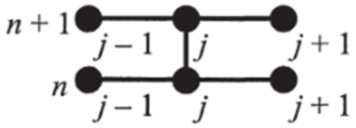
Явно-неявная схема имеет вид:

$$\frac{u_j^{k+1}-u_j^k}{\tau}=\theta a^2\frac{u_{j+1}^{k+1}-2u_j^{k+1}+u_{j-1}^{k+1}}{h^2}+(1-\theta)a^2\frac{u_{j+1}^k-2u_j^k+u_{j-1}^k}{h^2}$$

где ϴ - вес неявной части, причем 0≤ϴ≤1. При ϴ=0 получаем явную схему, при ϴ=1 – неявную схему, при ϴ=1/2 – схему Кранка-Николсона.

Как и в неявной схеме для решения явно-неявной необходимо решать СЛАУ с трехдиагональной матрицей.

Шаблон схемы Кранка-Николсона:



Также рассмотрим три варианта аппроксимации граничных условий, содержащих производные.

Двухточечная аппроксимация с первым порядком:

$$\left.\frac{\partial u}{\partial x}\right|_{j=0}^{k+1}=\frac{u_1^{k+1}-u_0^{k+1}}{h}+O(h); \qquad \left.\frac{\partial u}{\partial x}\right|_{j=N}^{k+1}=\frac{u_N^{k+1}-u_{N-1}^{k+1}}{h}+O(h),$$

Трехточечная аппроксимация со вторым порядком:

$$\frac{\partial u}{\partial x}(0,t^{k+1})=\frac{-3u_0^{k+1}+4u_1^{k+1}-u_2^{k+1}}{2h}+O(h^2),$$

$$\frac{\partial u}{\partial x}(l,t^{k+1})=\frac{u_{N-2}^{k+1}-4u_{N-1}^{k+1}+3u_N^{k+1}}{2h}+O(h^2).$$

Двухточечная аппроксимация со вторым порядком:

Для получения формул разложим u_1^{k+1}в ряд Тейлора в окрестности x = 0 и u_N^{k+1} в окрестности x = l:

$$u_1^{k+1}=u\left(o+h,t^{k+1}\right)=u_0^{k+1}+\left.\frac{\partial u}{\partial x}\right|_0^{k+1}h+\left.\frac{\partial^2 u}{\partial x^2}\right|_0^{k+1}\frac{h^2}{2}+O\left(h^3\right)$$

$$u_{N-1}^{k+1}=u\left(l-h,t^{k+1}\right)=u_N^{k+1}-\left.\frac{\partial u}{\partial x}\right|_N^{k+1}h+\left.\frac{\partial^2 u}{\partial x^2}\right|_N^{k+1}\frac{h^2}{2}+O\left(h^3\right)$$

Далее при помощи информации из исходного уравнения, выражая оттуда вторую производную и подставляя полученные выражения, получим:

$$\left.\frac{\partial u}{\partial x}\right|_0^{k+1}=\frac{2a^2}{h\left(2a^2-bh\right)}\cdot\left(u_1^{k+1}-u_0^{k+1}\right)-\frac{h}{2a^2-bh}\cdot\left.\frac{\partial u}{\partial t}\right|_0^{k+1}+\frac{gh}{2a^2-bh}\cdot u_0^{k+1}+O_1\left(h^2\right),$$

$$\left.\frac{\partial u}{\partial x}\right|_N^{k+1}=\frac{2a^2}{h\left(2a^2+bh\right)}\cdot\left(u_N^{k+1}-u_{N-1}^{k+1}\right)+\frac{h}{2a^2+bh}\cdot\left.\frac{\partial u}{\partial t}\right|_N^{k+1}-\frac{gh}{2a^2+bh}\cdot u_N^{k+1}+O_2\left(h^2\right).$$

Вариант

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \cos x(\cos t + \sin t),$$
$$u(0,t) = \sin t,$$
$$u_x(\frac{\pi}{2},t) = -\sin t,$$
$$u(x,0) = 0,$$

Аналитическое решение: $U(x,t) = \sin t \cos x$.

Решение и код

```
import math
import matplotlib.pyplot as plt
import numpy as np

N = 10
K = 100
T = 1
l = np.pi / 2
h = l/N
tau = T/K
sigma = tau/(h ** 2)
x = np.linspace(0,l,N)
t = np.linspace(0,T,K)
Xp, Tp = np.meshgrid(x, t)

print(sigma)
print(tau)
print(h)
```

```
0.40528473456935116
0.01
0.15707963267948966
```

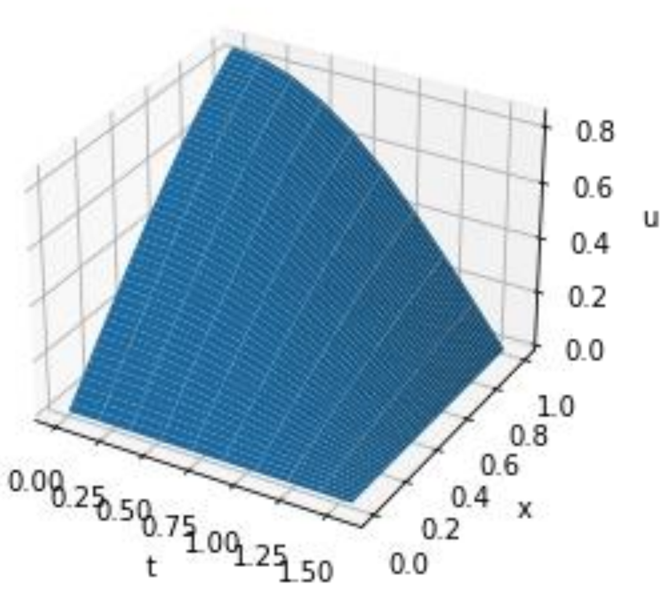
входные функции

```
def psi(x):
    return 0
def f(x, t):
    return np.cos(x)*(np.cos(t)+np.sin(t))
def phi0(t):
    return np.sin(t)
def phi1(t):
    return -np.sin(t)
def solution(x,t):
    return np.sin(t)*np.cos(x)
```

точное решение

```
def analitic(x,t, K):
    u = [0]*K
    for i in range(K):
        u[i] = [0]*N
    for i in range(K):
        for j in range(N):
            u[i][j] = solution(x[j], t[i])
    return u
```

```
u = analitic(x,t, K)
fig = plt.figure()
ax = plt.axes(projection ='3d')
ax.set_xlabel('t')
ax.set_ylabel('x')
ax.set_zlabel('u')
ax.plot_surface(Xp,Tp,np.array(u))
plt.show()
```



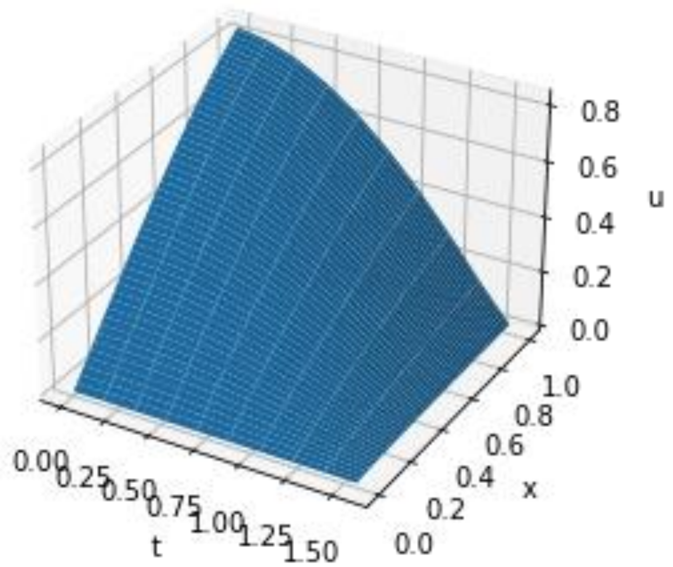
явная схема с различными видами апроксимации

```
def explicit_solve(l, N, K, T, app):
    u = [0]*K
    for i in range(K):
        u[i] = [0]*N
    for j in range(N):
        u[0][j] = psi(j * h)
    for k in range(K):
        u[k][0] = phi0(tau * k)
#         u[k][-1] = phi1(tau * k)

    for k in range(1, K):
        for j in range(1, N-1):
            u[k][j] = u[k-1][j]+tau*(u[k-1][j-1]-2*u[k-1][j]+u[k-1][j+1])/h**
2+tau*f(j*h, tau*k)
    if app == 1:
        for k in range(K):
            u[k][-1] = phi1(tau * k)*h + u[k][-2]
    if app == 2:
        for k in range(K):
            u[k][-1] = (phi1(k * tau) * 2 * h - u[k][-3] + 4 * u[k][-2]) / 3
    if app == 3:
        for k in range(K):
            u[k][-1] = (phi1(k * tau) + u[k][-2] / h + 2 * tau * u[k - 1][-1]
/ h) / \
            (1 / h + 2 * tau / h)
    return u
```

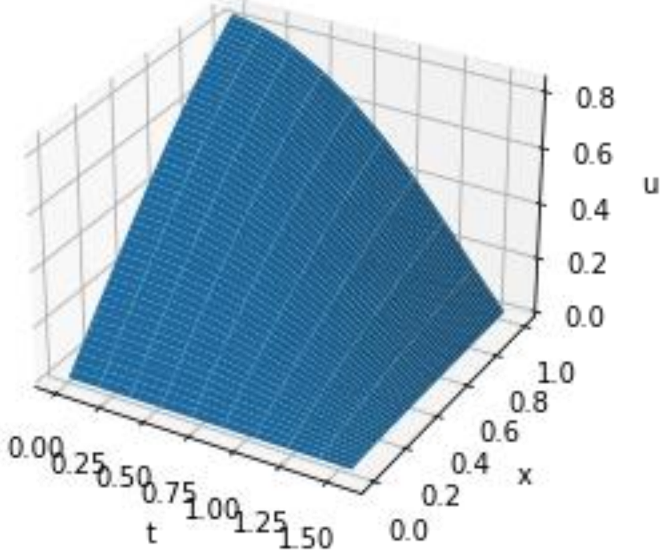
(двухточечная аппроксимация первого порядка)

```
exp1 = explicit_solve(l, N, K, T, 1)
fig = plt.figure()
ax = plt.axes(projection ='3d')
ax.set_xlabel('t')
ax.set_ylabel('x')
ax.set_zlabel('u')
ax.plot_surface(Xp, Tp, np.array(exp1))
plt.show()
```



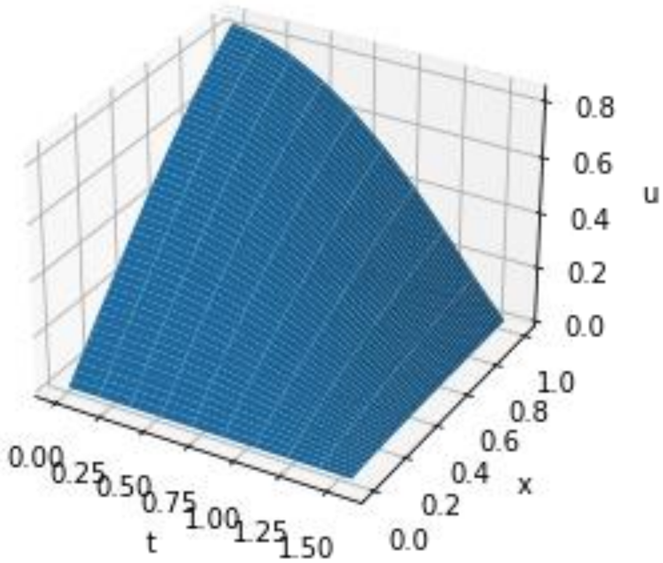
трехточечная аппроксимация со вторым порядком

```
exp2 = explicit_solve(l, N, K, T, 2)
fig = plt.figure()
ax = plt.axes(projection ='3d')
ax.set_xlabel('t')
ax.set_ylabel('x')
ax.set_zlabel('u')
ax.plot_surface(Xp, Tp, np.array(exp2))
plt.show()
```



двухточечная аппроксимация со вторым порядком

```
exp3 = explicit_solve(l, N, K, T, 3)
fig = plt.figure()
ax = plt.axes(projection ='3d')
ax.set_xlabel('t')
ax.set_ylabel('x')
ax.set_zlabel('u')
ax.plot_surface(Xp, Tp, np.array(exp3))
plt.show()
```




```
def tma(a, b, c, d):
    n = len(a)
    p, q = [], []
    p.append(-c[0] / b[0])
    q.append(d[0] / b[0])
    for i in range(1, n):
        p.append(-c[i] / (b[i] + a[i] * p[i - 1]))
```

```
        q.append((d[i] - a[i] * q[i - 1]) / (b[i] + a[i] * p[i - 1]))
    x = [0 for _ in range(n)]
    x[n - 1] = q[n - 1]
    for i in range(n-2, -1, -1):
        x[i] = p[i] * x[i+1] + q[i]
    return x
```

неявная схема

```
def implicit_solve(l, N, K, T):
    sigma = tau / (h ** 2)
    a = np.zeros(N)
    b = np.zeros(N)
    c = np.zeros(N)
    d = np.zeros(N)
    u = [0]*K
    for i in range(K):
        u[i] = [0]*N
    for j in range(N):
        u[0][j] = psi(j * h)
    for k in range(K):
        u[k][0] = phi0(tau*k)
    for k in range(1, K):
        a[0] = 0
        b[0] = -(1 + 2 * sigma)
        c[0] = sigma
        d[0] = -u[k-1][0]-sigma*phi0((k)*tau)
        for j in range(1, N):
            a[j] = sigma
            b[j] = -(1 + 2 * sigma)
            c[j] = sigma
            d[j] = -u[k-1][j] - tau * f(j * h, (k-1) * tau)
        a[-1] = sigma
        b[-1] = -(1 + 2 * sigma)
        c[-1] = 0
        d[-1] = -h*phi1(tau*(k))*h-u[k][-1]-tau*f(N*h,tau*(k+1))
        u[k] = tma(a, b, c, d)
    return u

imp = implicit_solve(l, N, K, T)
fig = plt.figure()
ax = plt.axes(projection ='3d')
ax.set_xlabel('t')
ax.set_ylabel('x')
ax.set_zlabel('u')
ax.plot_surface(Xp, Tp, np.array(imp))
plt.show()
```

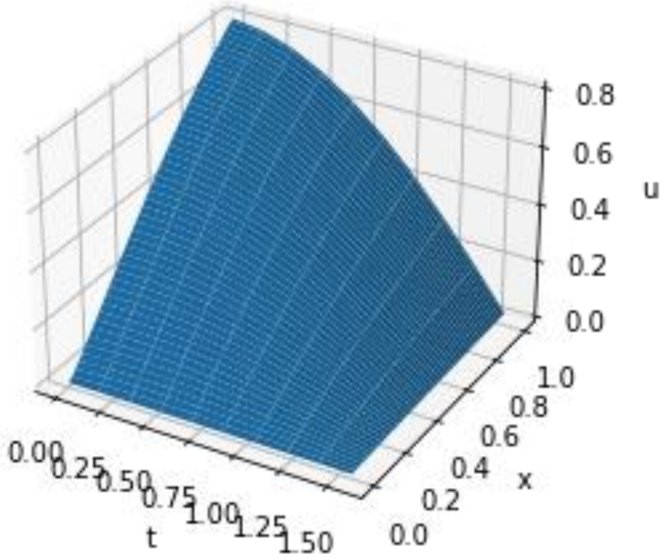
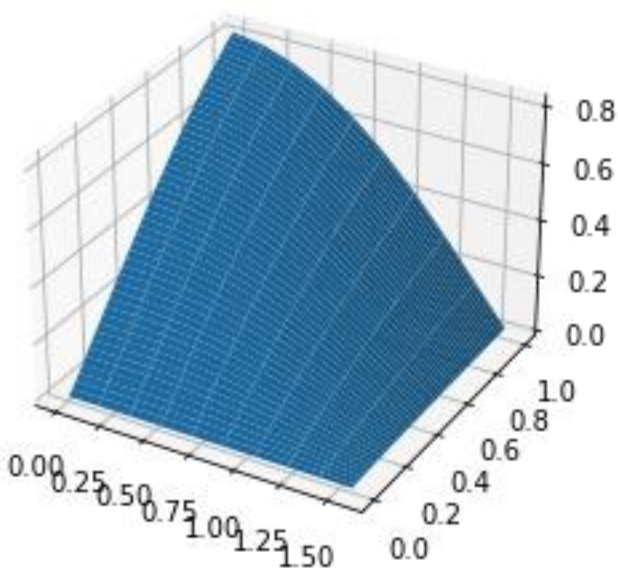


Схема Кранка-Николсона (смешанная)

```
def crank_nicholson_solve(l, N, K, T):
    tetta = 0.5
    u = np.zeros((K, N))
    imp = implicit_solve(l, N, K, T)
    exp = explicit_solve(l, N, K, T, 1)
    for k in range(0, K):
        for i in range(N):
            u[k][i] = imp[k][i] * tetta + exp[k][i] * (1 - tetta)
    return u

kn = crank_nicholson_solve(l, N, K, T)
fig = plt.figure()
ax = plt.axes(projection ='3d')
ax.plot_surface(Xp, Tp, np.array(kn))
plt.show()
```



среднеквадратические ошибки (явная, неявная и смешанная схемы)

```
def pogr(res, u):
    return math.sqrt(sum([sum([(u[i][j]-res[i][j])**2 for j in range(len(x))]
) \
                          for i in range(len(t))]))
```

```
pogr(exp1, u)
```

```
0.24007206456321856
```

```
pogr(imp, u)
```

```
0.5802814484493348
```

```
pogr(kn, u)
```

```
0.39054545119420625
```

Выводы

В ходе выполнения данной работы для решения начально-краевой задачи для дифференциального уравнения параболического типа были реализованы явная и неявная конечно-разностные схемы, схема Кранка-Николсона. Также реализованы три варианта аппроксимации начальных условий, содержащих производные: двухточечная аппроксимация с первым порядком, трехточечная со вторым и двухточечная со вторым. Путем сравнения результатов с приведенным точным решением была вычислена погрешность методов путем нахождения средневкадратической ошибки.