Московский Авиационный Институт

(национальный исследовательский университет)

Институт №8 “Информационные технологии и прикладная математика”

Кафедра 806 “ Вычислительная математика и программирование”

Лабораторная работа № 5 по курсу “Численные методы”

на тему

“Численные методы решения ДУЧП параболического типа”

Студент: Четвергов А. О.

Группа: М8О-409Б-19

Вариант: 3

Преподаватель: Пивоваров Д. Е.

Москва

2022

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 τ, h.

2. **Вариант:**

3.

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

$u(0,t) = \exp(-at)$,

$u(\pi,t) = -\exp(-at)$,

$u(x,0) = \cos x$.

Аналитическое решение: $U(x,t) = \exp(-at)\cos x$.

3. **Теория:**

Требуется решить третью начально-краевую задачу для параболического уравнения:

$$\begin{cases} \frac{\partial u}{\partial t} = a^2\frac{\partial^2 u}{\partial x^2} + b\frac{\partial u}{\partial x} + gu, \quad 0 < x < l, \quad t > 0; & (5.21) \\ \alpha\frac{\partial u(0,t)}{\partial x} + \beta u(0,t) = \varphi_0(t), \quad x = 0, \quad t > 0; & (5.22) \\ \gamma\frac{\partial u(l,t)}{\partial x} + \delta u(l,t) = \varphi_l(t), \quad x = l, \quad t > 0; & (5.23) \\ u(x,0) = \psi(x), \quad 0 \le x \le l, \quad t = 0. & (5.24) \end{cases}$$

Нанесем на пространственно-временную область $0 \le x \le l, 0 \le t \le T$ конечно-разностную сетку $\omega_{h\tau}$

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

С пространственным шагом h=l/N и шагом по времени τ=T/K.

Введем два временных слоя: нижний $t^k = k\tau$, на котором распределение искомой функции $u(x_j, t^k), j = \overline{0,N}$ известно и верхний временной слой $t^{k+1} = (k+1)\tau$, на котором распределение искомой функции $u(x_j, t^{k+1}), j = \overline{0,N}$ подлежит определению.

Сеточной функцией задачи u_j^k назовем однозначное отображение целых аргументов j, k в значения функции $u_j^k = u(x_j, t^k)$.

На введенной сетке введем сеточные функции u_j^k, u_j^{k+1}, первая из которых известна, вторая – подлежит определению. Для ее определения в задаче аппроксимируем дифференциальные операторы отношением конечных разностей, получим

$$\left.\frac{\partial u}{\partial t}\right|_j^k = \frac{u_j^{k+1} - u_j^k}{\tau} + O(\tau), \tag{5.13}$$

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_j^k = \frac{u_{j+1}^k - 2u_j^k + u_{j-1}^k}{h^2} + O(h^2). \tag{5.14}$$

Подставляя (5.13), (5.14) в задачу, получим ***явную конечно-разностную схему*** для этой задачи в форме

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

$$u_0^k = \varphi_0(t^k), \quad u_N^k = \varphi_l(t^k), \quad k = \overline{0, K}; \quad u_j^0 = \psi(x_j), \quad j = \overline{0, N}, \tag{5.15}$$

Где для каждого j-го уравнения все значения сеточной функции известны, за исключением одного - u_j^{k+1}, которое может быть определено явно из соотношений (5.15).

Если в (5.14) дифференциальный оператор по пространственной переменной аппроксимировать отношением конечных разностей на верхнем временном слое

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_j^{k+1} = \frac{u_{j+1}^{k+1} - 2u_j^{k+1} + u_{j-1}^{k+1}}{h^2} + O(h^2), \tag{5.16}$$

То после подстановки (5.13), (5.16) в задачу, получим ***неявную конечно-разностную схему*** для этой задачи

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

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

Рассмотрим неявно-явную схему с весами для простейшего уравнения теплопроводности

$$\frac{u_j^{k+1} - u_j^k}{\tau} = \theta a^2 \frac{u_{j+1}^{k+1} - 2u_j^{k+1} + u_{j-1}^{k+1}}{h^2} + (1-\theta) a^2 \frac{u_{j+1}^k - 2u_j^k + u_{j-1}^k}{h^2}, \tag{5.20}$$

Где θ – вес неявной части конечно-разностной схемы, 1- θ – вес для явной части, причем $0 \le \theta \le 1$. При $\theta = 1$ имеем полностью неявную схему, при $\theta = 0$ – полностью явную схему, и при $\theta = 1/2$ – ***схему Кранка-Николсона***.

Двухточечная аппроксимация с первым порядком:

$$\alpha \frac{u_1^{k+1} - u_0^{k+1}}{h} + \beta u_0^{k+1} = \varphi_0(t^{k+1}) + O(h),$$

$$\gamma \frac{u_N^{k+1} - u_{N-1}^{k+1}}{h} + \delta u_N^{k+1} = \varphi_l(t^{k+1}) + O(h).$$

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:

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

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

4. **Программа:**

Начальные данные:

```
a = 0.1
l = np.pi
N = 10
K = 1000
T = 1
h = l/N
tau = T/K
sigma = (a * tau) / h**2
```

Сетка:

```
ti = np.zeros(K)
xi = np.zeros(N)

for i in range(N):
    xi[i] = h * i
for i in range(K):
    ti[i] = tau * i
```

Дано:

```
def phi0(t):
    return np.exp(-a*t)

def phi1(t):
    return (np.exp(-a*t) * (-1))

def psi(x):
    return np.cos(x)
```

```
def analytical_solution(x,t):
    return (np.exp(-a*t)*np.cos(x))
```

Явная конечно-разностная схема:

```
def explicit():

    u = np.zeros((K,N))

    for k in range(0,K):
        u[k][0] = phi0(tau * k)

    for j in range(0,N):
        u[0][j] = psi(j * h)

    for k in range(1,K):
        for j in range(1,N-1):
            u[k][j] = sigma * u[k-1][j+1] + (1-2*sigma) * u[k-1][j] + sigma *
u[k-1][j-1]

        u[k][0] = phi0(k * tau)
        u[k][-1] = phi1(k * tau)

    return u
```

Неявная конечно-разностная схема:

```
def implicit():

    u = np.zeros((K,N))

    for k in range(0,K):
        u[k][0] = phi0(tau * k)

    for j in range(0,N):
        u[0][j] = psi(j * h)

    a = np.zeros(N)
    b = np.zeros(N)
    c = np.zeros(N)
    d = np.zeros(N)

    for k in range(1,K):

        a[0] = 0
        b[0] = 1
        c[0] = 0
        d[0] = phi0(tau * k)

        for j in range(1,N-1):
            a[j] = sigma
            b[j] = -(1 + 2 * sigma)
            c[j] = sigma
            d[j] = -u[k-1][j]

        a[-1] = 0
        b[-1] = 1
        c[-1] = 0
        d[-1] = phi1(tau * k)

        u[k] = swp(a,b,c,d)

    return u
```

Неявно-явная:

```
def implicit_explicit(omega):

    u = np.zeros((K,N))

    imp = implicit()
    exp = explicit()

    for k in range(0, K):
        for j in range(0, N):
            u[k][j] = imp[k][j] * omega + exp[k][j] * (1 - omega)

    return u
```

Погрешность:

```
def accuracy_error(ti,xi,u):
    res = np.zeros(K)
    for i in range(K):
        res[i] = np.max(np.abs(u[i] - np.array([analytical_solution(x,ti[i])
for x in xi])))

    plt.figure(figsize=(16, 8))
    plt.plot(ti[1:], res[1:])
    plt.xlabel('t')
    plt.ylabel('error')
    plt.grid(True)
    plt.show()
```

Погрешность в различные моменты времени:

```
def result(ti, xi, u1, u2, u12):
    fig,ax = plt.subplots(4)
    fig.set_figheight(10)
    fig.set_figwidth(15)
    t = 0
    for i in range(4):
        ax[i].plot(xi, u1[t, :], label='explicit')
        ax[i].plot(xi, u2[t, :], label='implicit')
        ax[i].plot(xi, u12[t, :], label='implicit_explicit')
        ax[i].plot(xi, [analytical_solution(x, ti[t]) for x in xi],
label='Analytic')
        ax[i].grid(True)
        ax[i].set_xlabel('x')
        ax[i].set_ylabel('u')
        ax[i].set_title(f'Решения при t = {t / K}')
        t = K - (i+1)*100

    plt.legend()
    plt.show()
```

Метод прогонки:

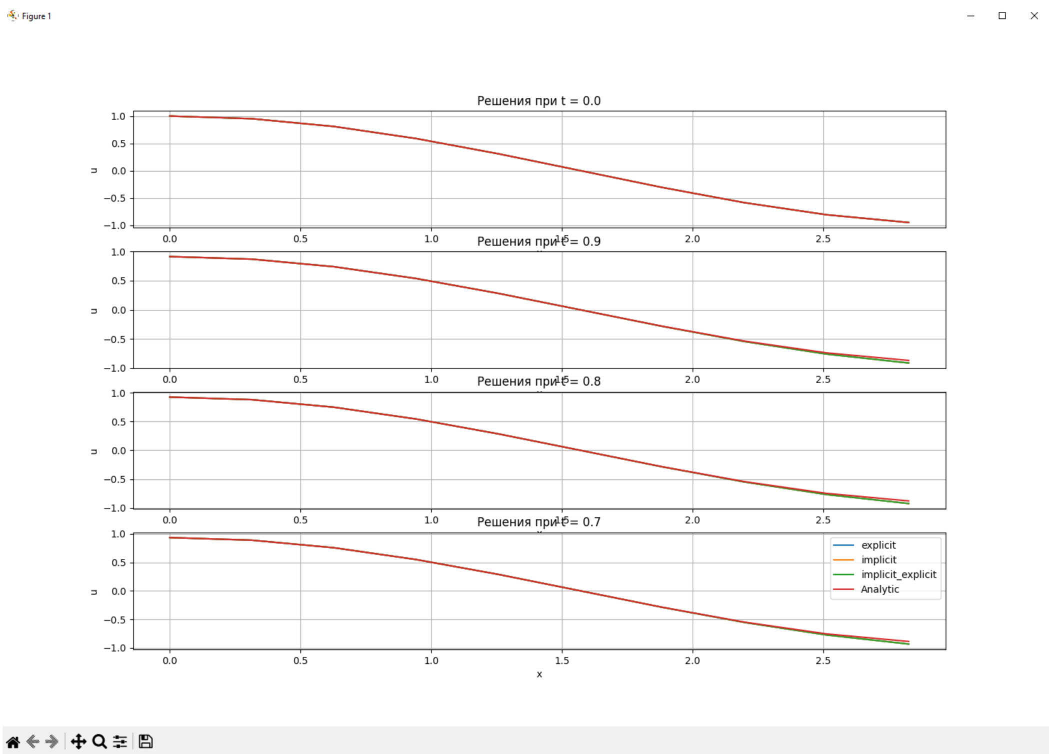
```
def swp(a,b,c,d):

    i = np.shape(d)[0]

    def searchP():
        P = np.zeros(i)
        P[0] = -c[0] / b[0]
        for j in range(1, len(P)):
            P[j] = -c[j] / (b[j] + a[j] * P[j - 1])
        return P

    def searchQ():
```

```
        Q = np.zeros(i)
        Q[0] = d[0] / b[0]
        for j in range(1, len(Q)):
            Q[j] = (d[j] - a[j] * Q[j - 1]) / (b[j] + a[j] * P[j - 1])
        return Q

    def searchX():
        X = np.zeros(i)
        X[i - 1] = Q[i - 1]
        for j in range(len(X) - 2, -1, -1):
            X[j] = P[j] * X[j + 1] + Q[j]
        return X

    P = searchP()
    Q = searchQ()
    X = searchX()

    return X
```

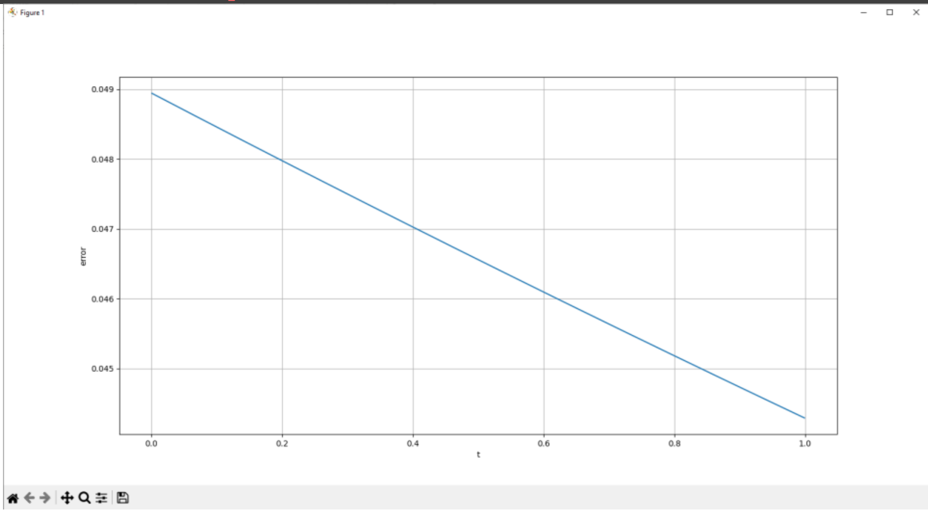
Main:

```
u1 = explicit()
u2 = implicit()
u12 = implicit_explicit(0.5)

result(ti,xi,u1,u2,u12)

accuracy_error(ti,xi,u1)
accuracy_error(ti,xi,u2)
accuracy_error(ti,xi,u12)
```

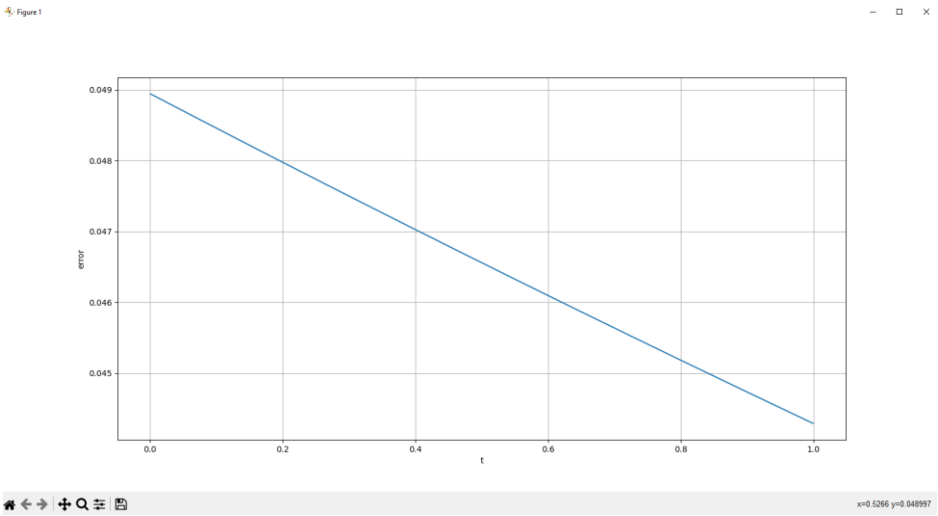
5. Результат:



Явная конечно-разностная схема



Неявная конечно-разностная схема



Неявно-явная схема