**Московский авиационный институт**

**(Национальный исследовательский университет)**

Институт №8 «Компьютерные науки и прикладная математика»

Кафедра вычислительной математики и программирования

# Лабораторная работа №5

по курсу «Численные методы»

Студент: Мариечев К.Д.

Группа: М8О-409Б-20

Проверил: Пивоваров Д.Е.

Дата:

Оценка:

Москва, 2023

# 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 $\tau, h$.

Вариант 9:

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

$u_x(0,t) - u(0,t) = -\exp(-at)(\cos(bt) + \sin(bt))$,

$u_x(\pi,t) - u(\pi,t) = \exp(-at)(\cos(bt) + \sin(bt))$,

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

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

# Теория

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

Пусть $N$ разбиений по $x$, $k$ разбиений по $t$. Тогда $h=l/N$, $\tau=T/k$, $x_i=l+ih$, $t_k=k\tau$.

Во внутренних узлах конечно-разностная сетки **явная схема** имеет вид:

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

Тогда

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

**Аппроксимация граничных условий**

1 способ (двухточечная аппроксимация с первым порядком)

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

Тогда граничные условия примут вид:

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

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

Выразим $u_0^{k+1}, u_N^{k+1}$

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

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

2 способ (трехточечная аппроксимация со вторым порядком)

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

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

Подставим в граничные условия и выразим $u_0^{k+1}, u_N^{k+1}$

$$u_0^{k+1} = \frac{\varphi_0}{\beta - 3\alpha/2h} + \frac{\alpha(4u_1^{k+1} - u_2^{k+1})}{2h(\beta - 3\alpha/2h)}$$

$$u_N^{k+1} = \frac{\varphi_l}{\delta - 3\gamma/2h} - \frac{\gamma(u_{N-2}^{k+1} - 4u_{N-1}^{k+1})}{2h(\delta - 3\gamma/2h)}$$

3 способ (двухточечная аппроксимация со вторым порядком)

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

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

Подставим значения второй производной в граничных узлах, полученные из дифференциального уравнения

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{j=0,N}^{k+1} = \left(\frac{1}{a^2}\cdot\frac{\partial u}{\partial t} - \frac{b}{a^2}\cdot\frac{\partial u}{\partial x}\right)_{j=0,N}^{k+1},$$

и найдем из полученных выражений значения первой производной $\left.\frac{\partial u}{\partial x}\right|_{j=0,N}^{k+1}$

в граничных узлах с порядком $O(\tau + h^2)$

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

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

Подставим $\left.\frac{\partial u}{\partial x}\right|_0^{k+1}$ и $\left.\frac{\partial u}{\partial x}\right|_N^{k+1}$ и аппроксимируя полученные

соотношения в соответствующих граничных узлах (при этом $\left.\frac{\partial u}{\partial t}\right|_0^{k+1} = \left(u_0^{k+1} - u_0^k\right)/\tau + O(\tau)$,

$\left.\frac{\partial u}{\partial t}\right|_N^{k+1} = \left(u_N^{k+1} - u_N^k\right)/\tau + O(\tau)$)

Во внутренних узлах конечно-разностная сетки **гибридная схема** имеет вид:

$$\frac{u_i^{k+1} - u_i^k}{\tau} = \theta a\frac{u_{i+1}^{k+1} - 2u_i^{k+1} + u_{i-1}^{k+1}}{h^2} + (1-\theta)a\frac{u_{i+1}^k - 2u_i^k + u_{i-1}^k}{h^2}$$
$$+ \theta b\frac{u_{i+1}^{k+1} - u_{i-1}^{k+1}}{2h} + (1-\theta)b\frac{u_{i+1}^k - u_{i-1}^k}{2h}$$

При $\theta = 1$ – неявная схема, при $\theta = 1/2$ – схема Кранка - Николсона

Составим систему линейных уравнений, где коэффициенты

При $u_{i-1}^{k+1}$: $\quad -\frac{a\theta\tau}{h^2} - \frac{b\theta\tau}{2h}$

При $u_i^{k+1}$: $1 + \frac{2a\theta\tau}{h^2}$

При $u_{i+1}^{k+1}$: $-\frac{a\theta\tau}{h^2} - \frac{b\theta\tau}{2h}$

Правая часть: $u_i^k + \tau(1-\theta)a\frac{u_{i+1}^k - 2u_i^k + u_{i-1}^k}{h^2} + \tau(1-\theta)b\frac{u_{i+1}^k - u_{i-1}^k}{2h}$

**Аппроксимация граничных условий**

1 способ (двухточечная аппроксимация с первым порядком)

Коэффициенты СЛАУ:

При $u_0^{k+1}$: $\beta - \frac{\alpha}{h}$

При $u_1^{k+1}$: $\frac{\alpha}{h}$

Правая часть: $\varphi_0(t^{k+1})$

При $u_{N-1}^{k+1}$: $\delta - \frac{\gamma}{h}$

При $u_N^{k+1}$: $\frac{\gamma}{h}$

Правая часть: $\varphi_l(t^{k+1})$

Матрица коэффициентов имеет трехдиагональный вид. Найдем $u^{k+1}$, решив СЛАУ методом прогонки.

2 способ (трехточечная аппроксимация со вторым порядком)

Коэффициенты СЛАУ:

При $u_0^{k+1}$: $-3\alpha + 2h\beta$

При $u_1^{k+1}$: $4\alpha$

При $u_2^{k+1}$: $-\alpha$

Правая часть: $2h * \varphi_0(t^{k+1})$

При $u_{N-2}^{k+1}$: $\gamma$

При $u_{N-1}^{k+1}$: $-4\gamma$

При $u_N^{k+1}$: $3\gamma + 2h\delta$

Правая часть: $2h * \varphi_l(t^{k+1})$

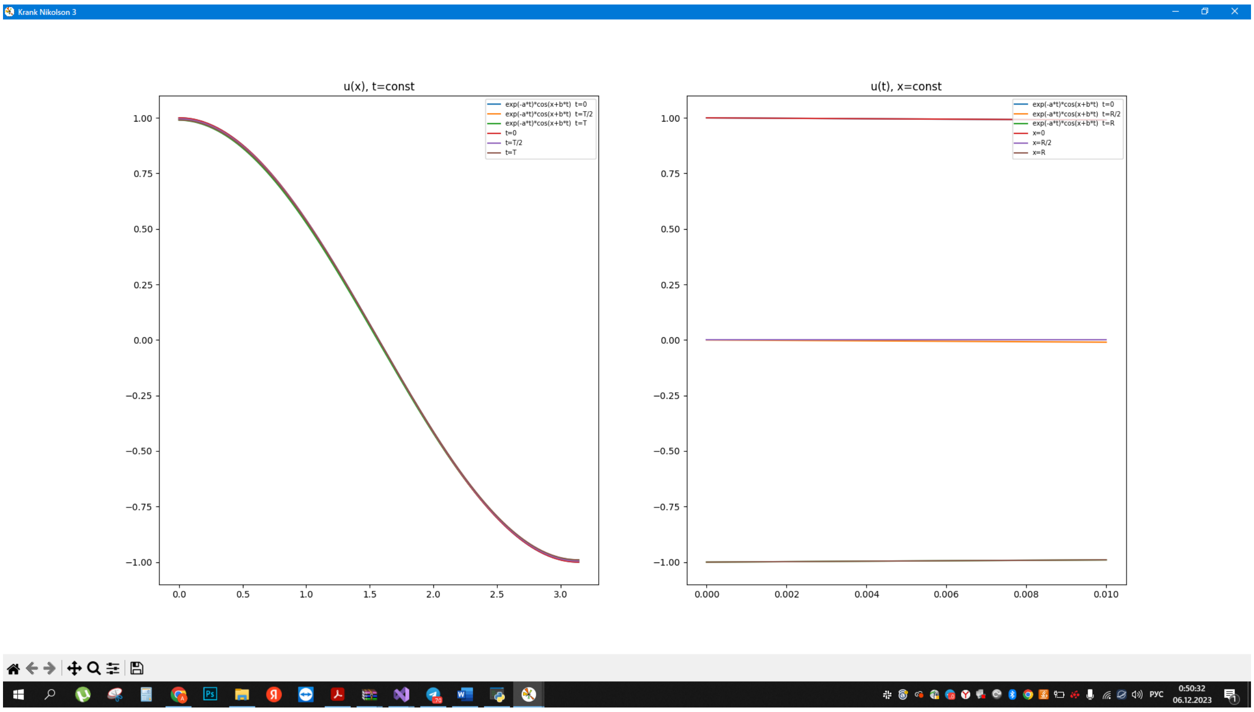
Приведем матрицу коэффициентов к трехдиагональному виду. Найдем $u^{k+1}$, решив СЛАУ методом прогонки.

3 способ (двухточечная аппроксимация со вторым порядком)

Аналогично 1 и 2 способу приводим подобные слагаемые при $u_{i-1}^{k+1}, u_i^{k+1}, u_{i+1}^{k+1}$ и получаем трехдиагональную матрицу. Найдем $u^{k+1}$, решив СЛАУ методом прогонки.

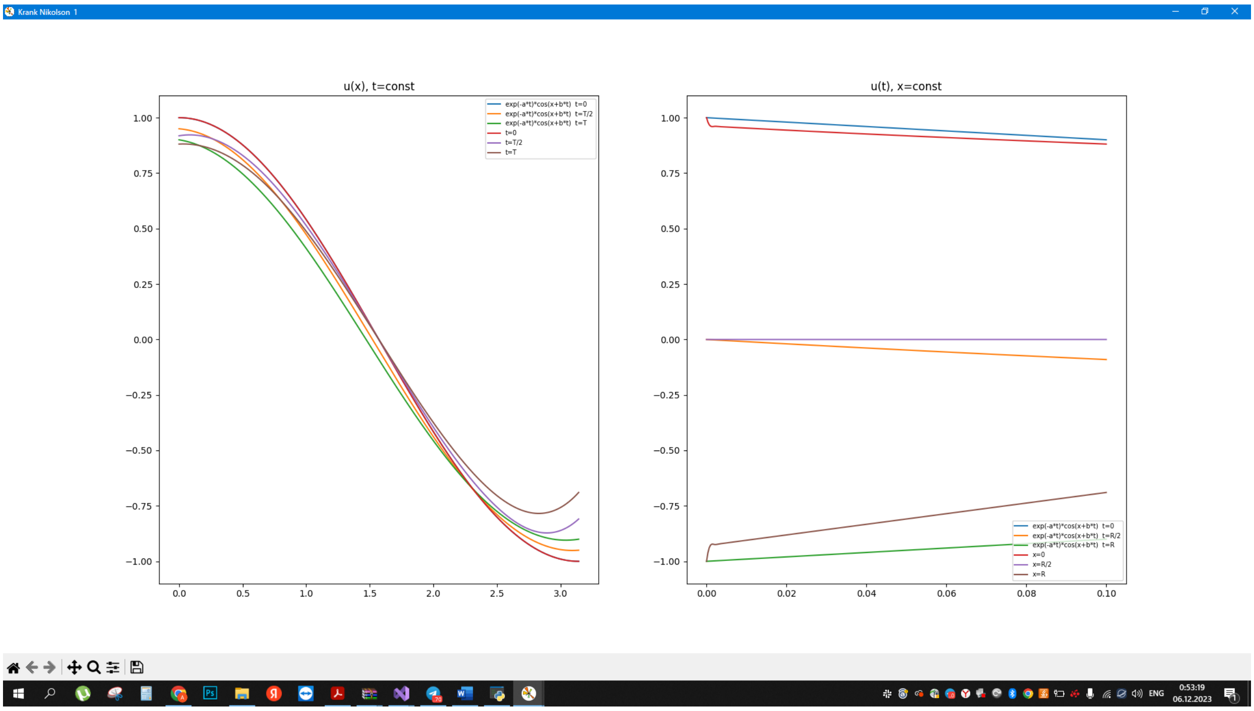
## Решение

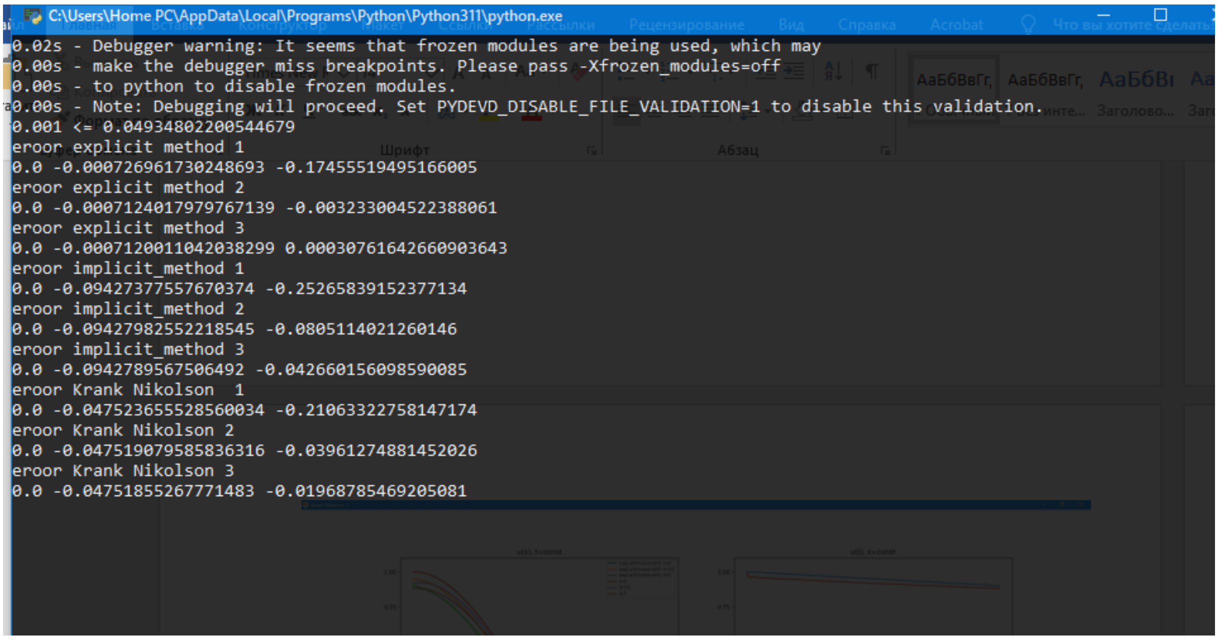
При k=100, N=10, T=0.01



```
C:\Users\Home PC\AppData\Local\Programs\Python\Python311\python.exe
.02s - Debugger warning: It seems that frozen modules are being used, which may
.00s - make the debugger miss breakpoints. Please pass -Xfrozen_modules=off
.00s - to python to disable frozen modules.
.00s - Note: Debugging will proceed. Set PYDEVD_DISABLE_FILE_VALIDATION=1 to disable this validation.
.0001 <= 0.04934802200544679
roor explicit method 1
.0 -8.074688590099018e-05 -0.08253965680800446
roor explicit method 2
.0 -8.074663971175278e-05 -0.0022373387249982057
roor explicit method 3
.0 -8.074663277301154e-05 7.426230161167169e-05
roor implicit_method 1
.0 -0.009868365868610241 -0.08851916643589564
roor implicit_method 2
.0 -0.009868366006183509 -0.007832705692035868
roor implicit_method 3
.0 -0.009868366009370686 -0.0005183044703199746
roor Krank Nikolson 1
.0 -0.004974799056805218 -0.08549345689051147
roor Krank Nikolson 2
.0 -0.004974798983354617 -0.005007865250414745
roor Krank Nikolson 3
.0 -0.004974798981011452 -0.0002191499607533398
```

k=100, N=10, T=0.1





```
C:\Users\Home PC\AppData\Local\Programs\Python\Python311\python.exe
0.02s - Debugger warning: It seems that frozen modules are being used, which may
0.00s - make the debugger miss breakpoints. Please pass -Xfrozen_modules=off
0.00s - to python to disable frozen modules.
0.00s - Note: Debugging will proceed. Set PYDEVD_DISABLE_FILE_VALIDATION=1 to disable this validation.
0.001 <= 0.04934802200544679
eroor explicit method 1
0.0 -0.0007269617302486693 -0.17455519495166005
eroor explicit method 2
0.0 -0.0007124017979767139 -0.003233004522388061
eroor explicit method 3
0.0 -0.0007120011042038299 0.0003076164266090364З
eroor implicit_method 1
0.0 -0.09427377557670374 -0.25265839152377134
eroor implicit_method 2
0.0 -0.09427982552218545 -0.0805114021260146
eroor implicit_method 3
0.0 -0.0942789567506492 -0.04266015609859008 5
eroor Krank Nikolson  1
0.0 -0.047523655528560034 -0.21063322758147174
eroor Krank Nikolson 2
0.0 -0.047519079585836316 -0.03961274881452026
eroor Krank Nikolson 3
0.0 -0.04751855267771483 -0.01968785469205081
```

```python
# явный метод
# 1 способ апроксимации


# t=0
K=0
u=np.zeros((k+1,n+1))

for i in range(n+1):
    x=l+i*h
    u[0][i]=math.cos(x)

for K in range(0,k):
    for i in range(1,n):
        u[K+1][i]=tau*(a*(u[K][i+1]-2*u[K][i]+u[K][i-1])/h**2+b*(u[K][i+1]-u[K][i-
1])/(2*h))+u[K][i]
    t=tau*(K+1)
    u[K+1][0]=(-alpha/h)*u[K+1][1]/(betta-alpha/h)-math.exp(-
a*t)*(math.cos(b*t)+math.sin(b*t))/(betta-alpha/h)
    u[K+1][n]=(gamma/h)*u[K+1][n-1]/(delta+gamma/h)+math.exp(-
a*t)*(math.cos(b*t)+math.sin(b*t))/(delta+gamma/h)



# 2 способ апроксимации

K=0
u=np.zeros((k+1,n+1))

for i in range(n+1):
    x=l+i*h
    u[0][i]=math.cos(x)

for K in range(0,k):
    for i in range(1,n):
        u[K+1][i]=tau*(a*(u[K][i+1]-2*u[K][i]+u[K][i-1])/h**2+b*(u[K][i+1]-u[K][i-
1])/(2*h))+u[K][i]
    t=tau*(K+1)
```

```
    u[K+1][0]=(-alpha/(2*h))*(4*u[K+1][1]-u[K+1][2])/(betta-3*alpha/(2*h))-
math.exp(-a*t)*(math.cos(b*t)+math.sin(b*t))/(betta-3*alpha/(2*h))
    u[K+1][n]=(-gamma/(2*h))*(u[K+1][n-2]-4*u[K+1][n-
1])/(delta+3*gamma/(2*h))+math.exp(-
a*t)*(math.cos(b*t)+math.sin(b*t))/(delta+3*gamma/(2*h))



# 3 способ апроксимации

K=0
u=np.zeros((k+1,n+1))

for i in range(n+1):
    x=l+i*h
    u[0][i]=math.cos(x)

D=2*a*tau*h*betta-tau*b*betta*h**2-2*a*alpha*tau-b*h**2
C=2*a*gamma*tau+gamma*h**2+2*a*h*tau*delta+b*tau*delta*h**2

for K in range(0,k):
    for i in range(1,n):
        u[K+1][i]=tau*(a*(u[K][i+1]-2*u[K][i]+u[K][i-1])/h**2+b*(u[K][i+1]-u[K][i-
1])/(2*h))+u[K][i]
    t=tau*(K+1)

    u[K+1][0]=(-2*alpha*a*tau*u[K+1][1])/D-(alpha*u[K][0]*h**2)/D+(tau*h*(2*a-
b*h)*(-math.exp(-a*t)*(math.cos(b*t)+math.sin(b*t))))/D
    u[K+1][n]=(2*gamma*a*tau*u[K+1][n-
1])/C+(gamma*u[K][n]*h**2)/C+(tau*h*(2*a+b*h)*(math.exp(-
a*t)*(math.cos(b*t)+math.sin(b*t))))/C



#гибридная схема
def hybrid_method(tetta, name1,name2,name3):
    # 1 способ апроксимации

    K=0
    u=np.zeros((k+1,n+1))

    for i in range(n+1):
        x=l+i*h
        u[0][i]=math.cos(x)

    A=np.zeros((n+1,n+1))

    for K in range(0,k):
        A=np.zeros((n+1,n+1))
        B=np.zeros((n+1,1))
        t=tau*(K+1)
        A[0][0]=betta-alpha/h
        A[0][1]=alpha/h
        B[0][0]=-math.exp(-a*t)*(math.cos(b*t)+math.sin(b*t))

        A[n][n-1]=-gamma/h
        A[n][n]=delta+gamma/h
        B[n][0]=math.exp(-a*t)*(math.cos(b*t)+math.sin(b*t))

        for i in range(1,n):
            A[i][i-1]=-a*tetta*tau/h**2-b*tetta*tau/(2*h)
            A[i][i]=1+2*a*tetta*tau/h**2
            A[i][i+1]=-a*tetta*tau/h**2+b*tetta*tau/(2*h)
            B[i][0]=tau*(1-tetta)*(a*(u[K][i+1]-2*u[K][i]+u[K][i-
1])/h**2+b*(u[K][i+1]-u[K][i-1])/(2*h))+u[K][i]
```

```
        x=running_method(A,B,n+1)

        for i in range(0,n+1):
            u[K+1][i]=x[i]



    # 2 способ апроксимации

    K=0
    u=np.zeros((k+1,n+1))

    for i in range(n+1):
        x=l+i*h
        u[0][i]=math.cos(x)

    A=np.zeros((n+1,n+1))

    for K in range(0,k):
        A=np.zeros((n+1,n+1))
        B=np.zeros((n+1,1))

        for i in range(1,n):
            A[i][i-1]=-a*tetta*tau/h**2-b*tetta*tau/(2*h)
            A[i][i]=1+2*a*tetta*tau/h**2
            A[i][i+1]=-a*tetta*tau/h**2+b*tetta*tau/(2*h)
            B[i][0]=tau*(1-tetta)*(a*(u[K][i+1]-2*u[K][i]+u[K][i-
1])/h**2+b*(u[K][i+1]-u[K][i-1])/(2*h))+u[K][i]

        t=tau*(K+1)

        A[0][0]= (-3*alpha+2*h*betta)-A[1][0]*(-alpha)/A[1][2]
        A[0][1]=4*alpha-A[1][1]*(-alpha)/A[1][2]
        B[0][0]=-2*h*math.exp(-a*t)*(math.cos(b*t)+math.sin(b*t))-B[1][0]*(-
alpha)/A[1][2]

        A[n][n-1]=-4*gamma-A[n-1][n-1]*gamma/A[n-1][n-2]
        A[n][n]=3*gamma+2*h*delta-A[n-1][n]*gamma/A[n-1][n-2]
        B[n][0]=2*h*math.exp(-a*t)*(math.cos(b*t)+math.sin(b*t))-B[n-
1][0]*gamma/A[n-1][n-2]
        x=running_method(A,B,n+1)

        for i in range(0,n+1):
            u[K+1][i]=x[i]


    # 3 способ апроксимации

    K=0
    u=np.zeros((k+1,n+1))

    for i in range(n+1):
        x=l+i*h
        u[0][i]=math.cos(x)

    A=np.zeros((n+1,n+1))

    for K in range(0,k):
        A=np.zeros((n+1,n+1))
        B=np.zeros((n+1,1))
        t=tau*(K+1)
        A[0][0]=-2*a*alpha/(h*(2*a-b*h))-alpha*h/(tau*(2*a-b*h))+betta
        A[0][1]=2*a*alpha/(h*(2*a-b*h))
        B[0][0]=-alpha*h/(tau*(2*a-b*h))*u[K][0]-math.exp(-
a*t)*(math.cos(b*t)+math.sin(b*t))
```

```
        A[n][n-1]=-2*a*gamma/(h*(2*a+b*h))
        A[n][n]=2*a*gamma/(h*(2*a+b*h))+gamma*h/(tau*(2*a+b*h))+delta
        B[n][0]=gamma*h/(tau*(2*a+b*h))*u[K][n]+math.exp(-
a*t)*(math.cos(b*t)+math.sin(b*t))

        for i in range(1,n):
            A[i][i-1]=-a*tetta*tau/h**2-b*tetta*tau/(2*h)
            A[i][i]=1+2*a*tetta*tau/h**2
            A[i][i+1]=-a*tetta*tau/h**2+b*tetta*tau/(2*h)
            B[i][0]=tau*(1-tetta)*(a*(u[K][i+1]-2*u[K][i]+u[K][i-
1])/h**2+b*(u[K][i+1]-u[K][i-1])/(2*h))+u[K][i]
        x=running_method(A,B,n+1)

        for i in range(0,n+1):
            u[K+1][i]=x[i]
```

# Вывод

В данной работе была решена начально-краевая задача для ДУ параболического типа тремя способами:

- с помощью явной конечно-разностной схемы
- с помощью неявной конечно-разностной схемы
- с помощью схемы Кранка-Николсона

С помощью каждого метода получилось решить заданное ДУ с хорошей точностью.

Явная конечно-разностная схема легко считается, но она не всегда устойчива и, соответственно, не всегда гарантирует хороший результат.

Неявная схема абсолютно устойчива, нотребует больших вычислительных затрат на решение СЛАУ

Схема Кранка-Николсона "комбинирует" предыдущие схемы, поэтому имеет наименьшую погрешность. Но при этом она по-прежнему использует сложные вычисления.