**Московский авиационный институт**

(национальный исследовательский институт)

**Лабораторные работы**

**По курсу**

**Численные методы**

Выполнил: Смирнов Д.А.

Группа: М8О-403Б-18

Москва 2021

**Лабораторная №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. Исследовать зависимость погрешности от сеточных параметров.

Вариант 8

Коэффициенты a,b,c:

4,0,2

, .



,

Аналитическое решение: .

Теория

Классическим примером уравнения параболического типа является уравнение теплопроводности (диффузии). В одномерном по пространству случае однородное (без источников энергии) уравнение теплопроводности имеет вид

Граничные условия первого рода:

Начальные условия:

Задача (1) - (4) называется первой начально-краевой задачей для уравнения теплопроводности.

Если же на границах заданы значения производных искомых функций по пространственной переменной

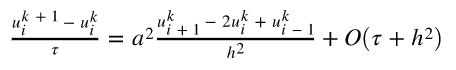
То есть граничные условия второго рода, то задачу (1), (5), (6), (4) называют второй начально-краевой задачей для уравнения теплопроводности (1).

Если на границах заданы линейные комбинации искомой функции и ее производной по пространственной переменной.

То есть граничные условия третьего рода, то задачу (1), (7), (8), (4) называют третьей начально-краевой задачей для уравнения теплопроводности (1).

Методы решения:

**Явная конечно-разностная схема** для решения уравнения параболического типа:



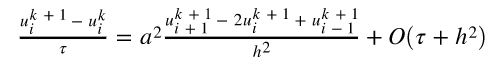
Сеточную функцию можно выразить :



Схема является условно устойчивой с условием, накладываемым на сеточные характеристики :



**Неявная конечно-разностная схема** для решения уравнения параболического типа:

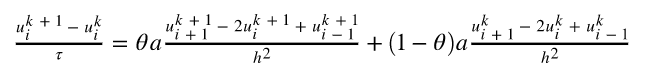


Сеточную функцию на верхнем временном слое можно получить из решения СЛАУ с трехдиагональной матрицей.

Эта СЛАУ в форме, пригодной для использования метода прогонки, имеет вид:



**Неявно-явная схема** с весами:



При  имеем полностью неявную схему, при   - полностью явную схему, и  - схему Кранка-Николсона.

Для схемы Кранка-Николсона порядок аппроксимации составляет при ,

Схема Кранка-Николсона при   абсолютно устойчива и имеет второй порядок аппроксимации по времени и пространственной переменной x.

Код:

Определим функции для аппроксимаций граничных условий:

def implicitApprox1(stepH,stepTau,k,m,param\_a,param\_c,a,b,c,d):

    # двухточечная со вторым порядком точности

    a[0] =  - (1 / (2\*stepH))

    c[0] = 1 / (2\*stepH)

    d[0] = LEFT((k+1) \* stepTau,param\_a,param\_c)

    c[1] = c[1] - a[1]\*c[0]/a[0]

    d[1] = d[1] - a[1]\*d[0]/a[0]

    b[m - 1] = 1

    d[m - 1] = RIGHT((k+1) \* stepTau,param\_a,param\_c)

    d[m - 2] = d[m - 2] - d[m - 1]\*c[m - 2]

def implicitApprox2(SOLVE,stepH,stepTau,k,m,param\_a,param\_c,a,b,c,d):

    # трехточечная со вторым порядком точности

    alpha = 1

    beta = 0

    delta = 1

    gamma = 0

    param\_b = 0

    b[0] = 2\*param\_a/stepH + stepH/stepTau - param\_c\*stepH - (beta/alpha)\*(2\*param\_a - param\_b\*stepH)

    c[0] = - 2\*param\_a/stepH

    d[0] = (stepH/stepTau)\*SOLVE[k][0] - LEFT((k+1) \* stepTau,param\_a,param\_c) \* (2\*param\_a - param\_b\*stepH)/alpha

    a[m - 1] = 0

    b[m - 1] = 1

    d[m - 1] = RIGHT((k+1) \* stepTau,param\_a,param\_c)

**Неявный метод**

def implicit(param\_a, param\_c, stepH, stepTau, m, n, approx\_type = 0):

    x\_end = 0

    x\_start = np.pi/2

    SOLVE = FIRST(x\_end, stepH, m, n)

    sigma = param\_a / stepH\*\*2

    for k in range(0,n-1):

        a = np.zeros(m)

        b = np.zeros(m)

        c = np.zeros(m)

        d = np.zeros(m)

        for j in range(1, m-1):

            a[j] = sigma

            b[j] = - (1/stepTau + 2 \* sigma - param\_c)

            c[j] = sigma

            d[j] = - SOLVE[k][j]/stepTau

        if approx\_type == 0:

            implicitApprox0(stepH,k,m,param\_a,param\_c,b,c,d)

        if approx\_type == 1:

            implicitApprox1(stepH,stepTau,k,m,param\_a,param\_c,a,b,c,d)

        elif approx\_type == 2:

            implicitApprox2(SOLVE,stepH,stepTau,k,m,param\_a,param\_c,a,b,c,d)

        if approx\_type != 1:

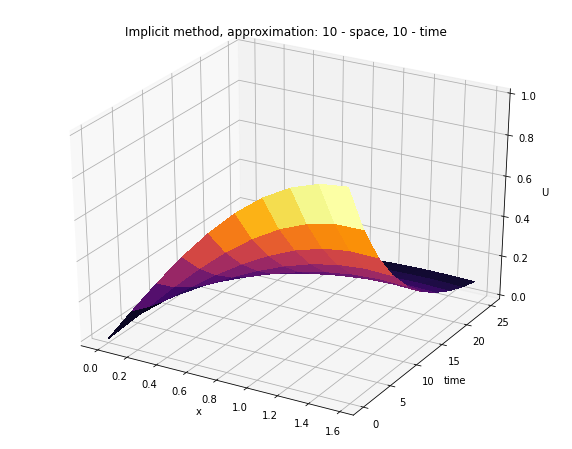
            Y = runMethod(a, b, c, d, m)

        else:

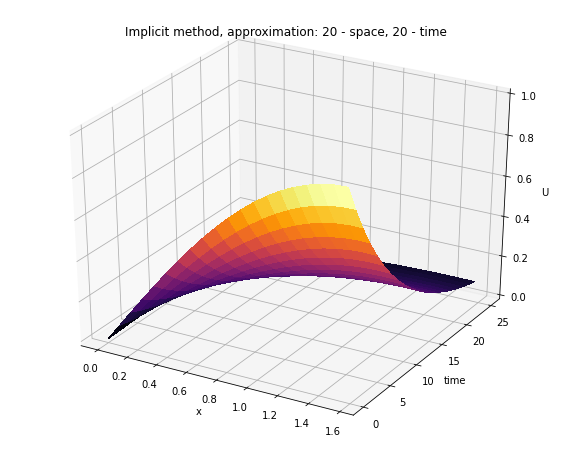
            Y = preRun(a, b, c, d, m)

        SOLVE[k + 1] = Y

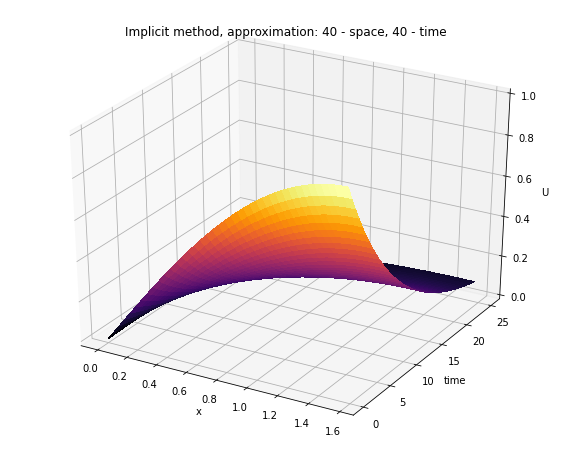
    return SOLVE



Ошибка 0.04987056081427133

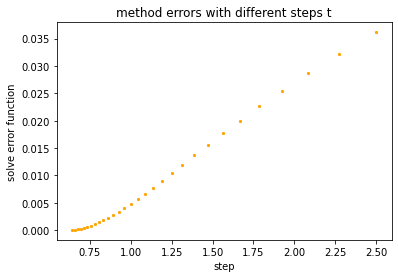


Ошибка 0.03639311412484216

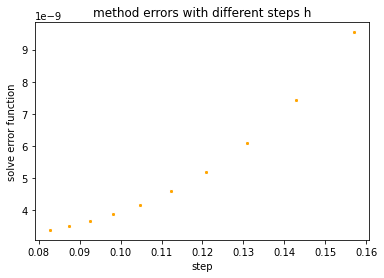


Ошибка 0.02609729391352843

Ошибка с различным шагом по времени



Ошибка с различным шагом по пространству



**Явный метод**

def explicit(param\_a, param\_c, stepH, stepTau, m, n, approx\_type = 0):

    x\_end = 0

    x\_start = np.pi/2

    SOLVE = FIRST(x\_end, stepH, m, n)

    sigma = param\_a / stepH\*\*2

    for k in range(0,n-1):

        for j in range(1, m-1):

            Rpart = sigma\*(SOLVE[k][j+1] - 2\*SOLVE[k][j] + SOLVE[k][j-1]) + param\_c\*(SOLVE[k][j])

            SOLVE[k+1][j] = SOLVE[k][j] + Rpart\*stepTau

        if approx\_type == 0:

            # двухточечная с первым порядком точности

            b = - (1 / stepH)

            c = SOLVE[k][1] / stepH

            d = LEFT(k \* stepTau,param\_a,param\_c)

            SOLVE[k+1][0] = (d - c)/b

            d = RIGHT(k \* stepTau,param\_a,param\_c)

            SOLVE[k+1][m-1] = d

        if approx\_type == 1:

            # двухточечная со вторым порядком точности

            b =  - (1 / (2\*stepH))

            c = SOLVE[k][2] \* (1 / (2\*stepH))

            d = LEFT(k \* stepTau,param\_a,param\_c)

            SOLVE[k+1][0] = (d - c)/b

            d = RIGHT(k \* stepTau,param\_a,param\_c)

            SOLVE[k+1][m-1] = d

        elif approx\_type == 2:

            # трехточечная со вторым порядком точности

            a = (4\*SOLVE[k][1] - SOLVE[k][2])/(2\*stepH)

            b = (-3/(2\*stepH) - 1)

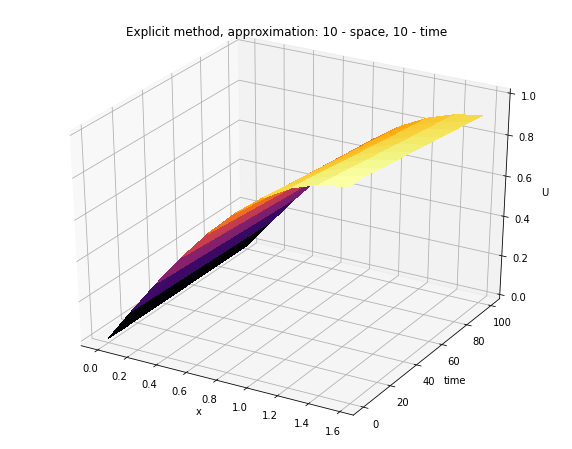
            c = LEFT(k \* stepTau,param\_a,param\_c)

            SOLVE[k+1][0] = (c - a)/b

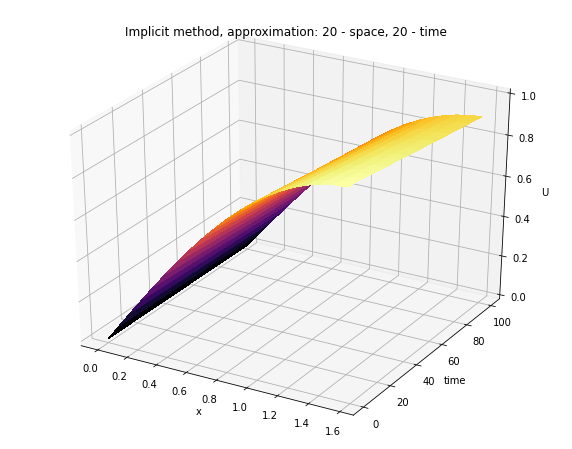
            c = RIGHT(k \* stepTau,param\_a,param\_c)

            SOLVE[k+1][m-1] = c

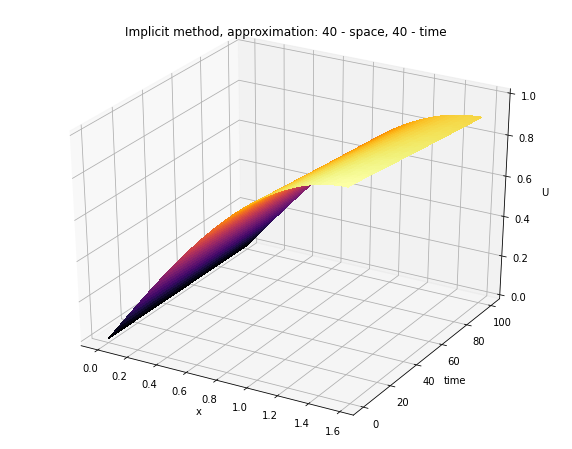
    return SOLVE



Ошибка 1.2345342223122427e-05

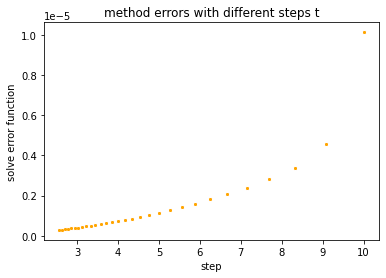


Ошибка 1.6485390830811178e-06

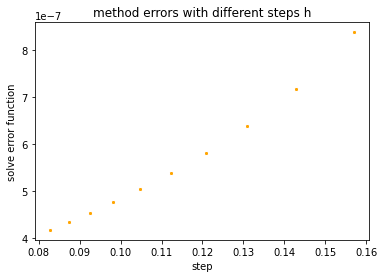


Ошибка 2.791480720467311e-07

Ошибка с разным шагом по времени



Ошибка с разным шагом по пространству



**Метод KN**

def KN(param\_a, param\_c, stepH, stepTau, m, n, x\_end, approx\_type = 0):

    SOLVE = FIRST(x\_end, stepH, m, n)

    sigma = param\_a / stepH\*\*2

    for k in range(0,n-1):

        a = np.zeros(m)

        b = np.zeros(m)

        c = np.zeros(m)

        d = np.zeros(m)

        for j in range(1, m-1):

            Rpart = param\_a\*(SOLVE[k][j+1] - 2\*SOLVE[k][j] + SOLVE[k][j-1])/stepH\*\*2 +param\_c\*SOLVE[k][j]

            a[j] = sigma/2

            b[j] = - (1/stepTau + sigma - param\_c/2)

            c[j] = sigma/2

            d[j] = - SOLVE[k][j]/stepTau - Rpart/2

        if approx\_type == 0:

            implicitApprox0(stepH,k,m,param\_a,param\_c,b,c,d)

        if approx\_type == 1:

            implicitApprox1(stepH,stepTau,k,m,param\_a,param\_c,a,b,c,d)

        elif approx\_type == 2:

            implicitApprox2(SOLVE,stepH,stepTau,k,m,param\_a,param\_c,a,b,c,d)

        if approx\_type != 1:

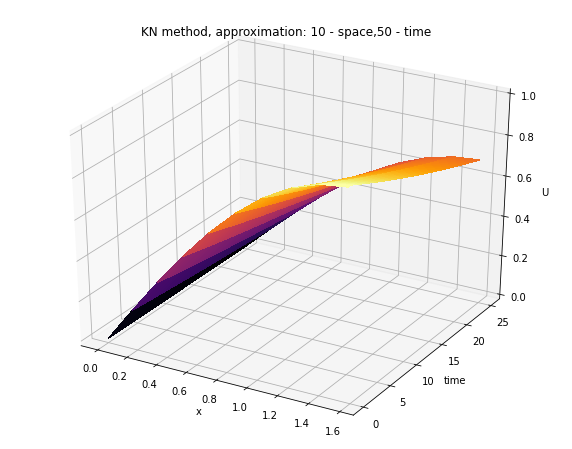
            Y = runMethod(a, b, c, d, m)

        else:

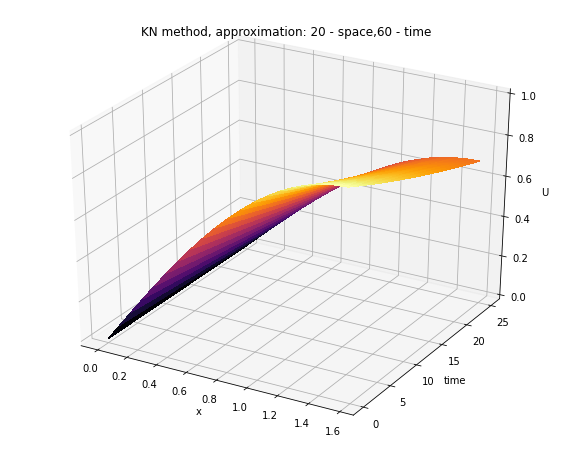
            Y = preRun(a, b, c, d, m)

        SOLVE[k + 1] = Y

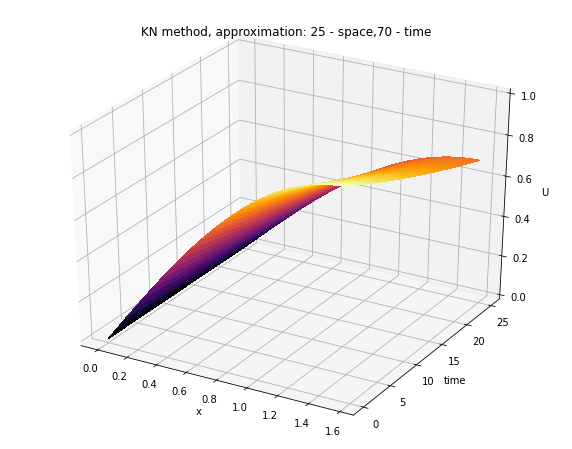
    return SOLVE



Ошибка 3.0277003702938885e-07

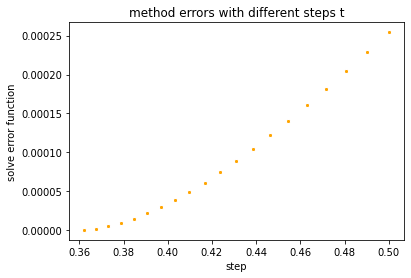


Ошибка 8.864703263892001e-09

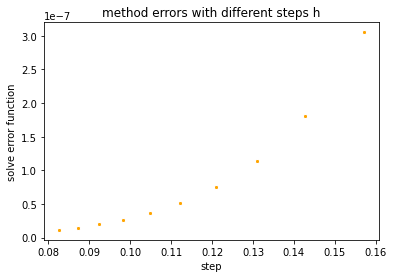


Ошибка 3.136201986373934e-09

Ошибка с разным шагом по времени:



Ошибка с разным шагом по пространству:



**Вывод**  
В результате выполнения лабораторной работы были освоены 3 конечно-разностные схемы для решения начально-краевой задачи для дифференциального уравнения параболического типа : явная конечно-разностная схема, неявная конечно-разностная схема и схема Кранка - Николсона.

Были построены графики ошибок, на которых показано падение ошибки при росте числа разбиений по пространству и времени.