|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

**ФАКУЛЬТЕТ** ***ИУК «Информатика и управление»***

**КАФЕДРА** \_\_***ИУК4 «Программное обеспечение ЭВМ, информационные технологии»***

**ЛАБОРАТОРНАЯ РАБОТА №4**

**«Метод сеток для решения уравнения параболического типа»**

**ДИСЦИПЛИНА: «Моделирование»**

|  |  |  |
| --- | --- | --- |
| Выполнил: студент гр. ИУК4-62Б | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ( Карельский М.К. )  (Подпись) |
| Проверил: | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ( Никитенко У.В. )  (Подпись) |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | | |
|  | - Балльная оценка:  - Оценка: | |

Калуга, 2023

**Цель:** сформировать практические навыки анализа возможностей построения и выделения наиболее важных свойств объектов моделей для моделирования и использования специализированных программных пакетов и библиотек для стандартных вычислений и визуализации результатов численного или приближенно-аналитического решения ДУЧП2 параболического типа на основе сравнения результатов.

**Задачи:** решить уравнение, указанное в варианте методом аппроксимации дифференциального оператора, выбрать среду для проведения расчетов и вычислительного эксперимента. Написать программу, реализующую решение задачи. Оценить результаты расчетов. Визуализировать результаты.

**Задание:**

Найти решение задачи

используя различные разностные схемы

* явную схему порядка с аппроксимацией производных в граничных условиях с порядком ;
* схему с весами порядка при σ = 0, σ = 1, σ = 1/2 с аппроксимацией производных в граничных условиях с порядком .

По решению задачи должен быть представлен отчет, содержащий

1. Алгоритм решения задачи.
2. Тестирование алгоритма на решениях, для которых разностная схема точно аппроксимирует дифференциальную задачу.
3. Тестирование алгоритма, например, на решениях , , , , на которых разностная схема неточно аппроксимирует дифференциальную задачу.
4. Таблицы решения на «крупной» сетке независимо от шагов по t и x, с которыми строится решение (N = 5, 10, 20)
5. Таблицы, характеризующие точность решения и внутреннюю сходимость

**Вариант 9**

**Решение:**

***Явная разностная схема***

Аппроксимируем данное уравнение в узле :

имеет вид:

В случае данного уравнения:

После подстановки получаем:

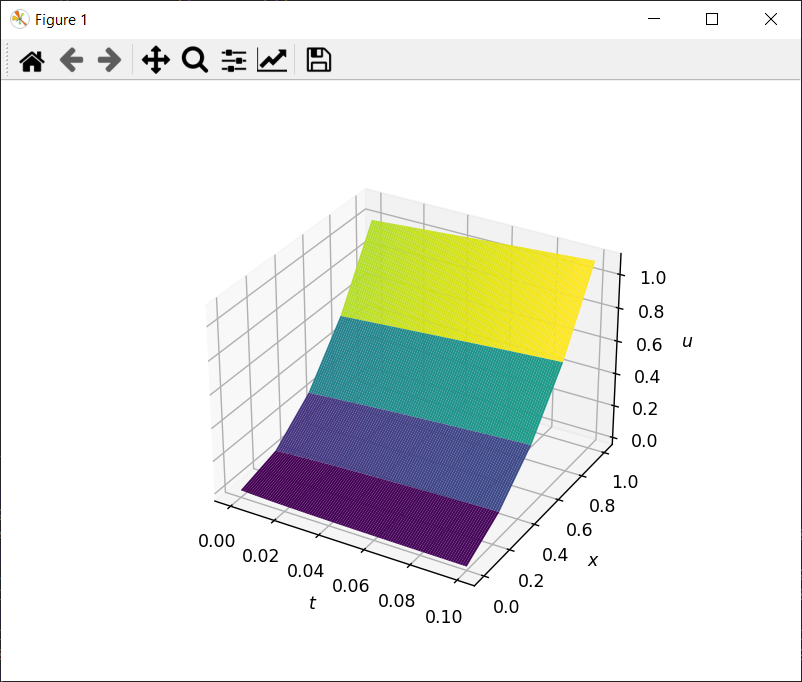
Найдем начальные условия:

Найдем граничные условия:

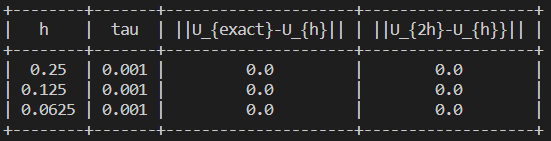
Найдем нужные функции из условия:

После подстановки и выражения получаем:

Проведем тестирование на

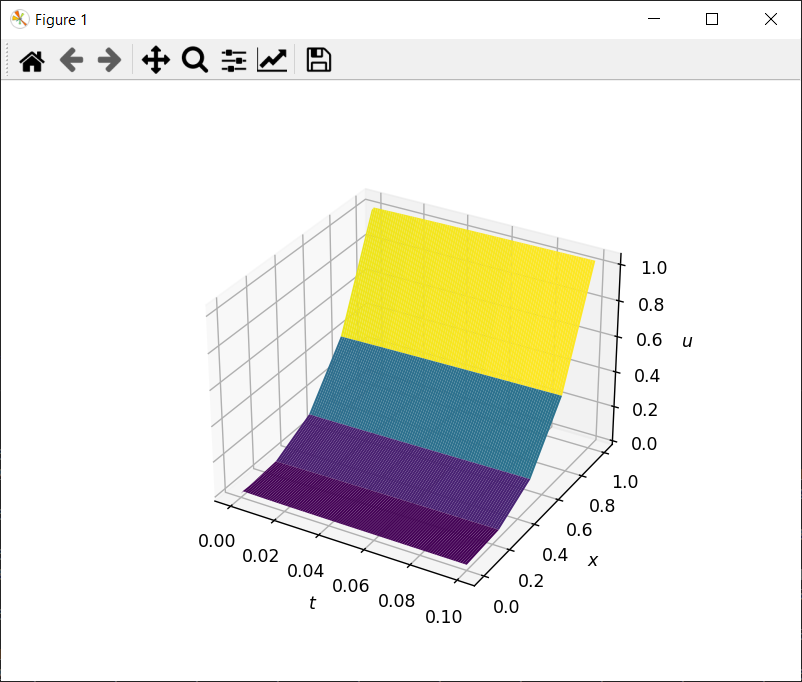


**Рис. 1.** График функции

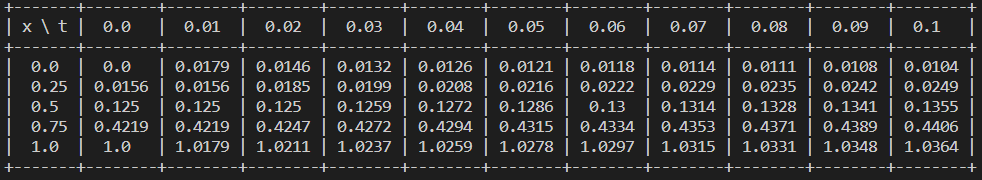


**Рис. 2.** Точность решения

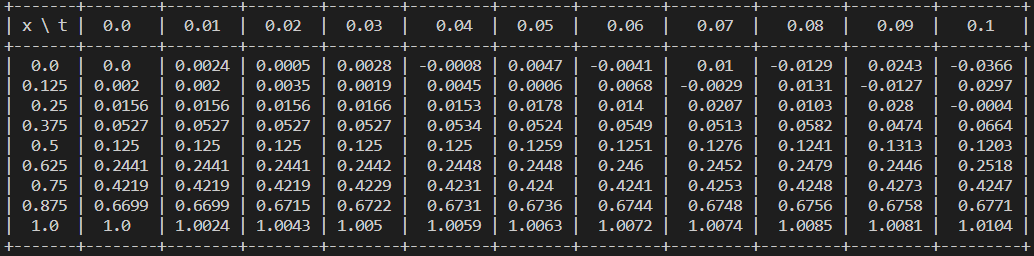
Проведем тестирование на



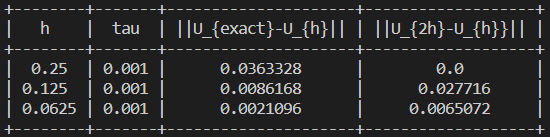
**Рис. 3.** График функции



**Рис. 4.** Аппроксимация при h = 0.25, t = 0.01



**Рис. 5.** Аппроксимация при h = 0.125, t = 0.01



**Рис. 6.** Точность решения

***Схема с весами***

Найдем начальные условия:

Найдем граничные условия:

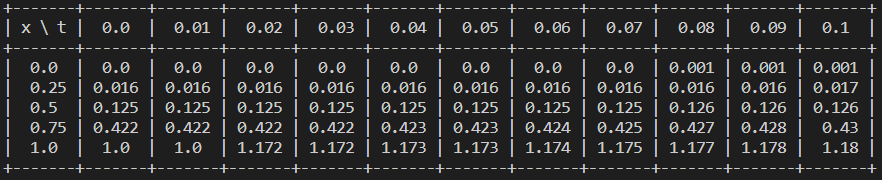
Найдем коэффициенты системы, решив которую можно получить решения на последующих слоях:

Имея:

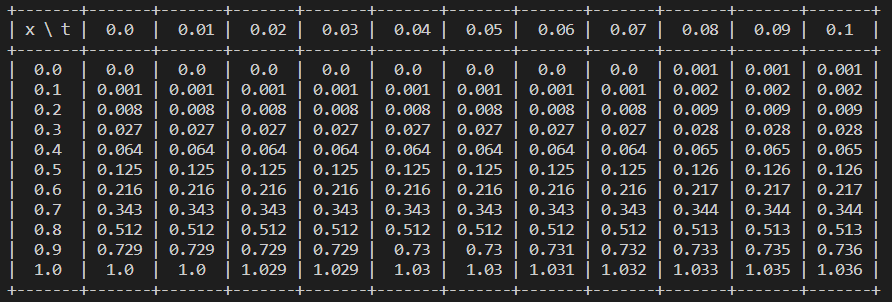
Составим систему:

Проведем тестирование на

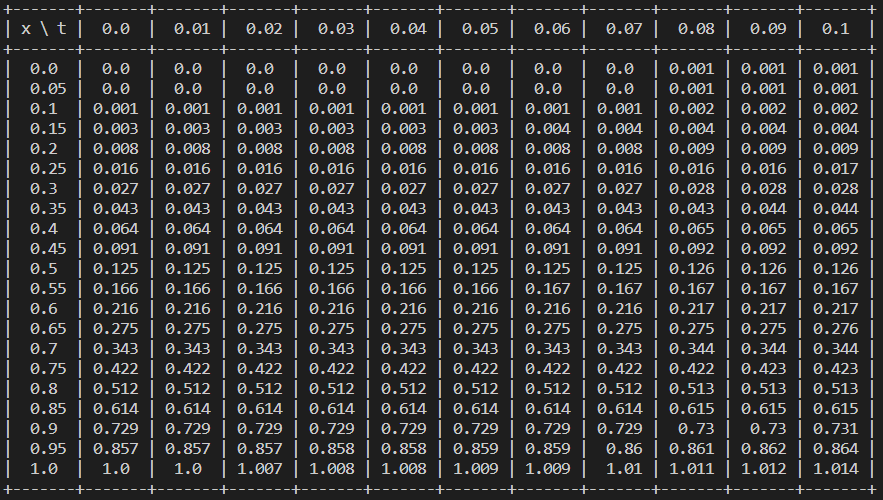
Найдем результат для :



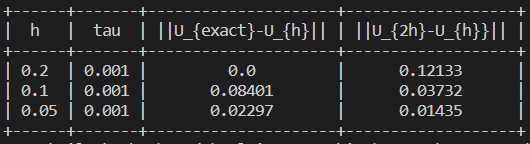
**Рис. 7.** Аппроксимация при h = 0.25, t = 0.01



**Рис. 8.** Аппроксимация при h = 0.1, t = 0.01

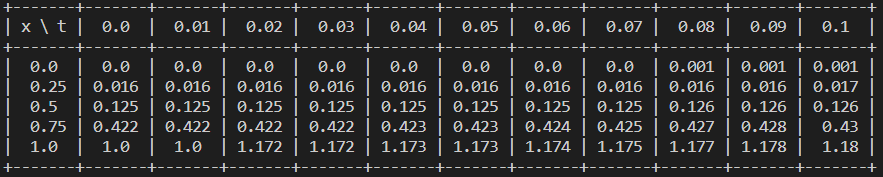
****

**Рис. 9.** Аппроксимация при h = 0.05, t = 0.01

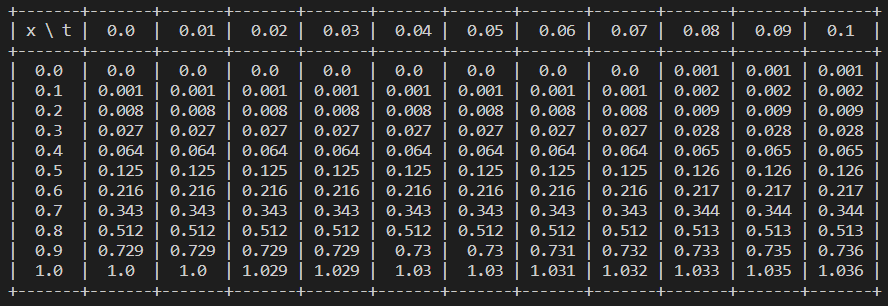
****

**Рис. 10.** Точность решения

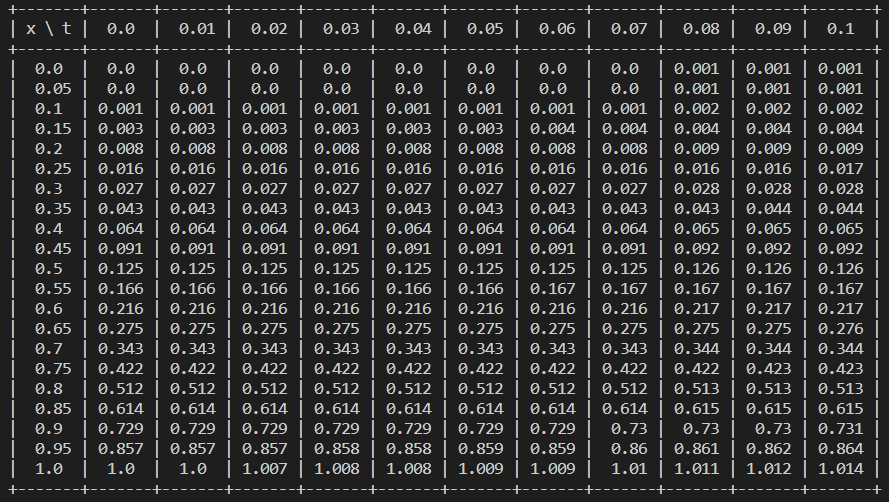
Найдем результат для :

****

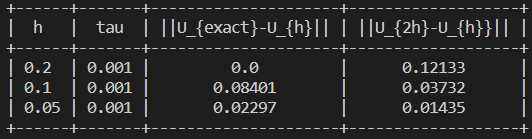
**Рис. 11.** Аппроксимация при h = 0.25, t = 0.01

****

**Рис. 12.** Аппроксимация при h = 0.1, t = 0.01

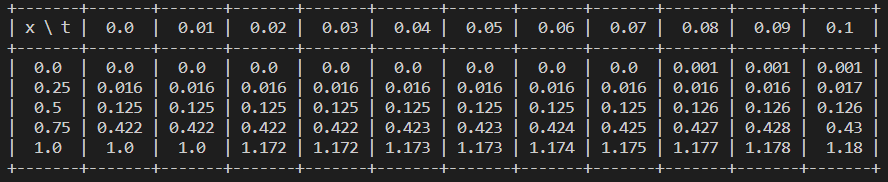
****

**Рис. 13.** Аппроксимация при h = 0.05, t = 0.01

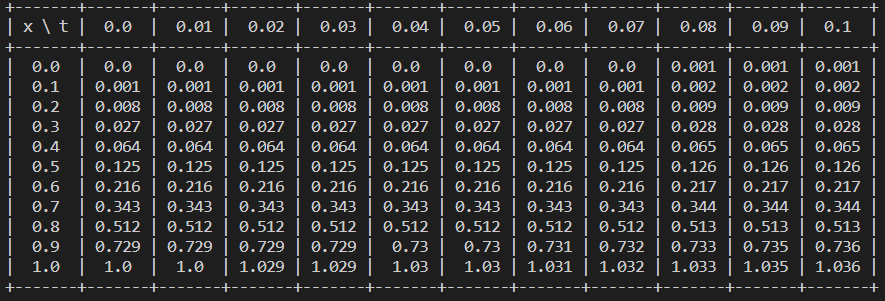
****

**Рис. 14.** Точность решения

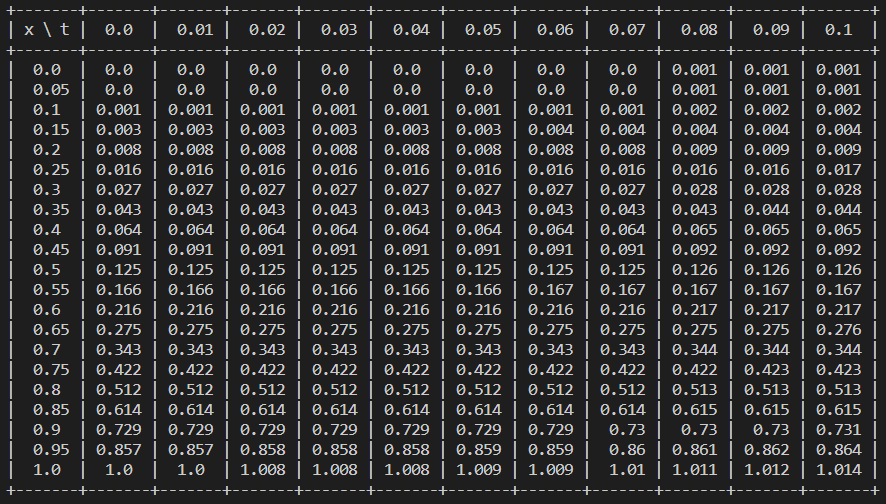
Найдем результат для :

****

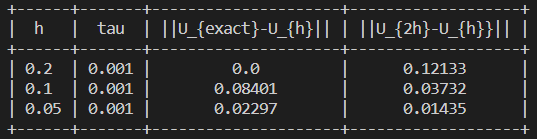
**Рис. 15.** Аппроксимация при h = 0.25, t = 0.01

****

**Рис. 16.** Аппроксимация при h = 0.1, t = 0.01



**Рис. 17.** Аппроксимация при h = 0.05, t = 0.01



**Рис. 18.** Точность решения

**Вывод:** в ходе выполнения лабораторной работы были получены практические навыки анализа возможностей построения и выделения наиболее важных свойств объектов моделей для моделирования и использования специализированных программных пакетов и библиотек для стандартных вычислений и визуализации результатов численного или приближенно-аналитического решения ДУЧП2 параболического типа на основе сравнения результатов.

**ПРИЛОЖЕНИЯ**

**Листинг:**

***LW4\_1.py***

import numpy as np

import matplotlib.pyplot as plt

from prettytable import PrettyTable

def f(x, t):

    return 3\*t\*\*2 - 6\*x + (x\*\*3 + t\*\*3)\*np.sin(x)

    #return (x\*\*2 + t)\*np.sin(x) - 1

def lu(u: np.array, x: np.linspace, t: np.linspace, i, k, h):

     return (u[i + 1, k] - 2\*u[i, k] + u[i - 1, k]) / pow(h, 2) - u[i, k]\*np.sin(x[i])

def solve(h, tau):

    x\_min = 0

    x\_max = 1

    xs = np.arange(x\_min, x\_max + h, h)

    n\_x = len(xs)

    t\_min = 0

    t\_max = 0.1

    ts = np.arange(t\_min, t\_max + tau, tau)

    n\_t = len(ts)

    #phi = lambda x, t: x\*\*2

    phi = lambda x, t: pow(x, 3)

    #alpha = lambda t: t

    alpha = lambda t: pow(t, 3)

    #beta = lambda t: t + 3

    beta = lambda t: 4 + pow(t, 3)

    U = np.zeros((n\_x, n\_t))

    U[:, 0] = [phi(x, t\_min) for x in xs]

    for k in range(1, n\_t):

        for i in range(1, n\_x - 1):

            U[i, k] = U[i, k - 1] + tau \* (lu(U, xs, ts, i, k - 1, h) +

                                           f(xs[i], ts[k - 1]))

        U[0, k] = (2\*h\*alpha(ts[k]) - 4\*U[1, k] + U[2, k]) / (2\*h + 3)

        U[-1, k] = (2\*h\*beta(ts[k]) + 4\*U[-2, k] - U[-3, k]) / (2\*h + 3)

    return [xs, ts, U]

def makeTableFromResult(xs, ts, U):

    table = PrettyTable()

    ts = ts.round(4)

    xs = xs.round(4)

    U = U.round(4)

    table.add\_column("x \ t", xs)

    for k in range(len(ts)):

        table.add\_column(f"{ts[k]}", U[:, k])

    return table

h = 0.125

tau = 0.01

[xs, ts, U] = solve(h, tau)

print("Результат:")

print(makeTableFromResult(xs, ts, U))

fig = plt.figure()

ax = plt.axes(projection='3d')

X, Y = np.meshgrid(ts, xs)

ax.plot\_surface(X, Y, U, rstride=1, cstride=1,

    cmap='viridis', edgecolor='none')

ax.set\_xlabel('$t$')

ax.set\_ylabel('$x$')

ax.set\_zlabel('$u$')

plt.show()

def makeTableFromStep(hs, taus, exact\_diff, diff):

    table = PrettyTable()

    table.add\_column("h", hs)

    table.add\_column("tau", taus)

    table.add\_column("||U\_{exact}-U\_{h}||", diff)

    table.add\_column("||U\_{2h}-U\_{h}}||", exact\_diff)

    return table

hs = []

taus = []

exact\_diff = []

last\_diff = []

[\_, \_, last\_u] = solve(h, tau)

last\_u = last\_u[0::2]

for i in range(3):

    [xs, ts, U] = solve(h, tau)

    u = lambda t, x: x\*\*3 + t\*\*3

    #u = lambda t, x: x\*\*2 + t

    U\_exact = np.array([[u(t, x) for t in ts] for x in xs])

    hs.append(h)

    taus.append(tau)

    exact\_diff.append(np.amax(np.abs(U - U\_exact)))

    last\_diff.append(np.amax(np.abs(last\_u - U[0::2])))

    h /= 2

    last\_u = U

hs = np.array(hs).round(7)

taus = np.array(taus).round(7)

exact\_diff = np.array(exact\_diff).round(7)

last\_diff = np.array(last\_diff).round(7)

print(makeTableFromStep(hs, taus, last\_diff, exact\_diff))

***LW4\_2.py***

import numpy as np

import matplotlib.pyplot as plt

from mpl\_toolkits import mplot3d

import scipy.linalg as la

from prettytable import PrettyTable

def f(x, t):

    return 3\*t\*\*2 - 20\*x\*\*3 + (x\*\*5 + t\*\*3)\*np.sin(x)

def lu(u: np.array, x: np.linspace, t: np.linspace,

       i, k, h):

     return (u[i + 1, k] - 2 \* u[i, k] + u[i - 1, k]) / pow(h, 2) - u[i, k]\*np.sin(x[i])

def solve(h, tau, sigma):

    x\_min = 0

    x\_max = 1

    xs = np.arange(x\_min, x\_max + h, h)

    n\_x = len(xs)

    t\_min = 0

    t\_max = 0.1

    ts = np.arange(t\_min, t\_max + tau, tau)

    n\_t = len(ts)

    phi = lambda x, t: pow(x, 5)

    alpha = lambda t: pow(t, 3)

    beta = lambda t: pow(t, 3) + 6

    U = np.zeros((n\_x, n\_t))

    G = np.zeros((n\_x, n\_t))

    U[:, 0] = [phi(x, t\_min) for x in xs]

    A = np.zeros((n\_x - 1))

    B = np.zeros((n\_x))

    C = np.zeros((n\_x - 1))

    G[0, 0] = h\*alpha(ts[0])

    G[-1, 0] = h\*beta(ts[0])

    for k in range(0, n\_t - 1):

        for i in range(1, n\_x - 1):

            G[i, k+1] = (-1 \* U[i, k]) / tau \

                - (1 - sigma) \* lu(U, xs, ts, i, k, h) \

                - f(xs[i], ts[k])

            A[i - 1] = sigma / h\*\*2

            B[i] = 2\*sigma / h\*\*2 + np.sin(xs[i]) + 1 / tau

            C[i] = sigma / h\*\*2

        B[0] = -(h + 1)

        C[0] =  -1

        A[-1] = -1

        B[-1] = -(h + 1)

        G[0, k+1] = h\*alpha(ts[k+1])

        G[-1, k+1] = h\*beta(ts[k+1])

        matrix = np.array([[0, \*C], B, [\*A, 0]])

        U[:, k+1] = la.solve\_banded((1,1), matrix, G[:, k+1])

    return [xs, ts, U]

def makeTableFromResult(xs, ts, U):

    table = PrettyTable()

    ts = ts.round(4)

    xs = xs.round(4)

    U = U.round(5)

    table.add\_column("x \ t", xs)

    for k in range(len(ts)):

        table.add\_column(f"{ts[k]}", U[:, k])

    return table

h = 0.25

tau = 0.01

sigma = 0.5

[xs, ts, U] = solve(h, tau, sigma)

fig = plt.figure()

ax = plt.axes(projection='3d')

X, Y = np.meshgrid(ts, xs)

ax.plot\_surface(X, Y, U, rstride=1, cstride=1,

    cmap='magma', edgecolor='none')

ax.set\_xlabel('$t$')

ax.set\_ylabel('$x$')

ax.set\_zlabel('$u$')

plt.show()

def makeTableFromStep(hs, taus, diff, exact\_diff):

    table = PrettyTable()

    table.add\_column("h", hs)

    table.add\_column("tau", taus)

    table.add\_column("||U\_{exact}-U\_{h}||", diff)

    table.add\_column("||U\_{2h}-U\_{h}}||", exact\_diff)

    return table

def makeTableFromResult(xs, ts, U):

    table = PrettyTable()

    ts = ts.round(3)

    xs = xs.round(3)

    U = U.round(3)

    table.add\_column("x \ t", xs)

    for k in range(len(ts)):

        table.add\_column(f"{ts[k]}", U[:, k])

    return table

[xs, ts, U] = solve(h, tau, sigma)

print("Результат:")

print(makeTableFromResult(xs, ts, U))

hs = []

taus = []

exact\_diff = []

last\_diff = []

[\_, \_, last\_u] = solve(h, tau, sigma)

last\_u = last\_u[0::2]

for i in range(3):

    [xs, ts, U] = solve(h, tau, sigma)

    u = lambda t, x: x\*\*5 + t\*\*3

    U\_exact = np.array([[u(t, x) for t in ts] for x in xs])

    hs.append(h)

    taus.append(tau)

    exact\_diff.append(np.amax(np.abs(U - U\_exact)))

    last\_diff.append(np.amax(np.abs(last\_u - U[0::2])))

    h /= 2

    last\_u = U

hs = np.array(hs).round(5)

taus = np.array(taus).round(5)

exact\_diff = np.array(exact\_diff).round(5)

last\_diff = np.array(last\_diff).round(5)

print(makeTableFromStep(hs, taus, exact\_diff, last\_diff))