Московский авиационный институт

(Национальный исследовательский университет)

Факультет прикладной математики и физики

Кафедра вычислительной математики и программирования

**Лабораторная работа №7**

«ЧИСЛЕННЫЕ МЕТОДЫ РЕШЕНИЯ ОБЫКНОВЕННЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ»

Вариант 4

Выполнил: Железнов Д.Е.

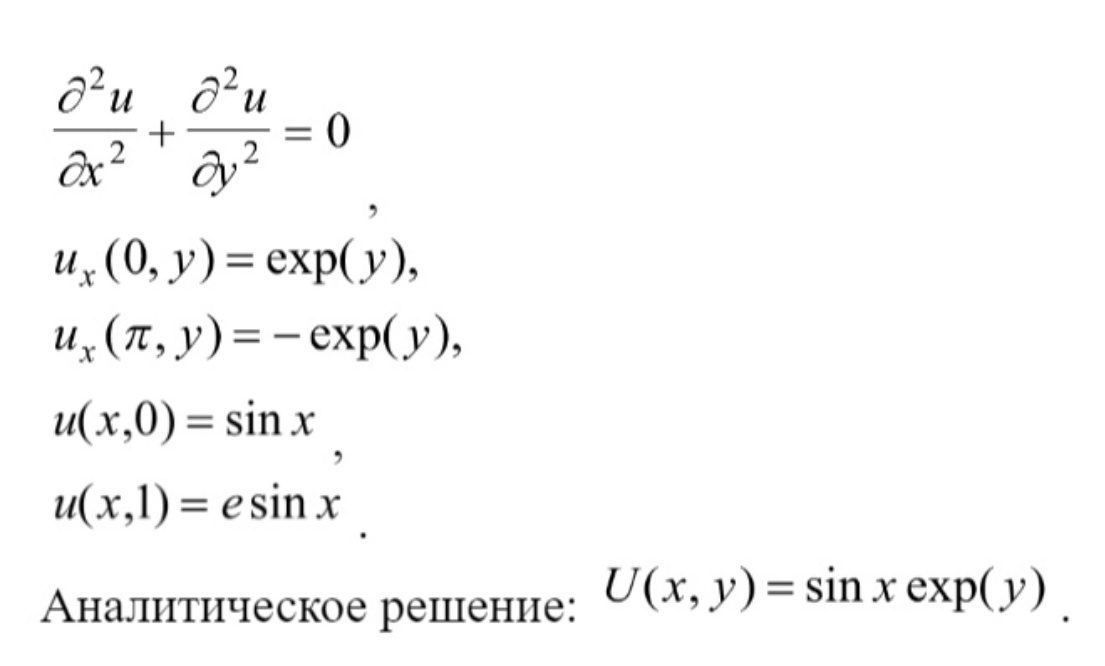
Группа: М8О-409Б-20

Проверил: Пивоваров Д.Е.

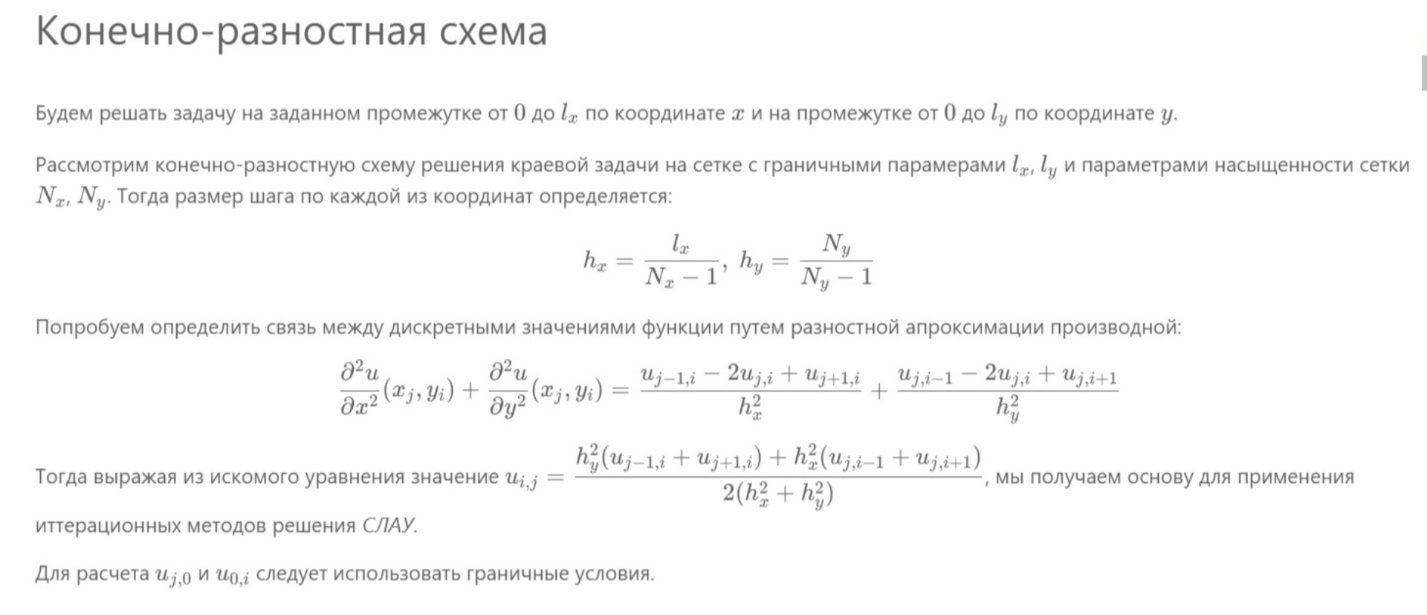
Дата:

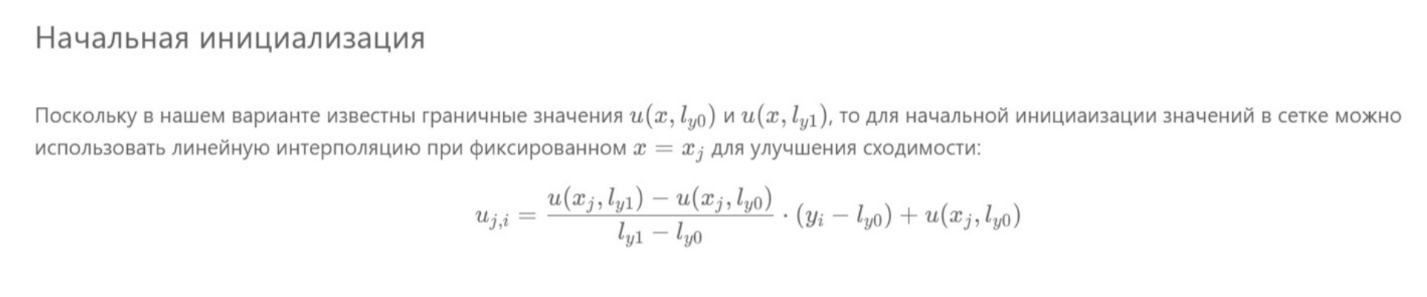
Оценка:

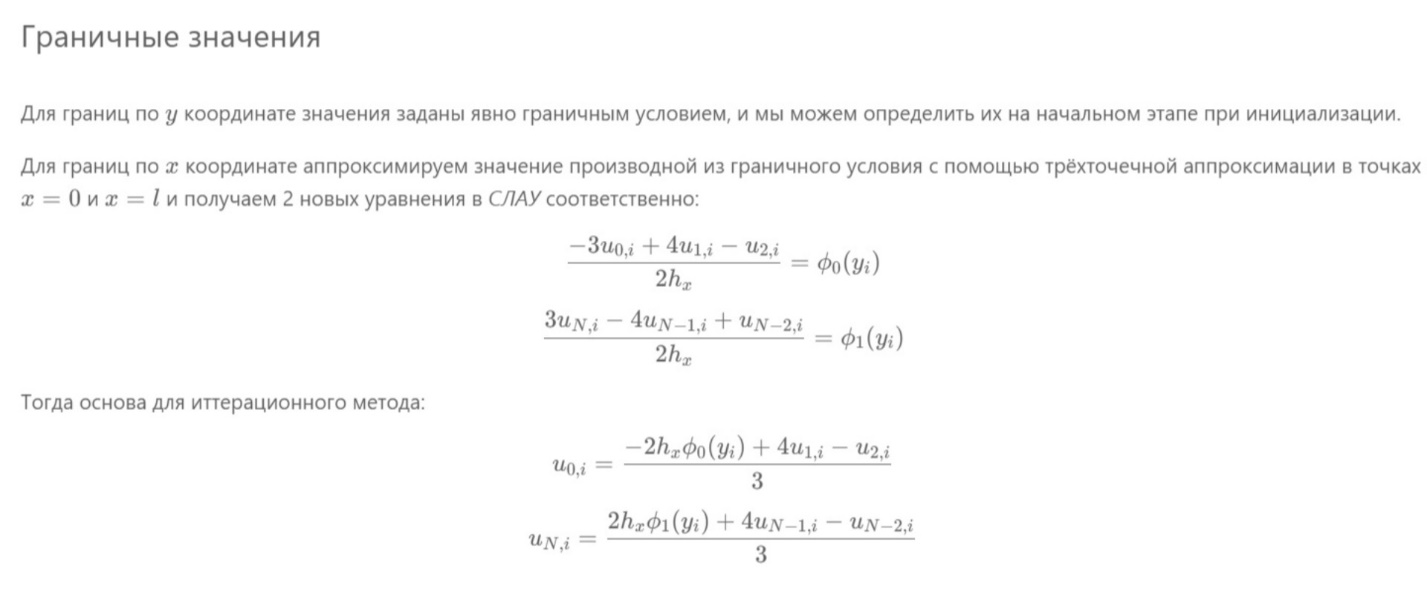
Задание: Решить краевую задачу для дифференциального уравнения эллиптического типа. Аппроксимацию уравнения произвести с использованием центрально-разностной схемы. Для решения дискретного аналога применить следующие методы: *метод простых итераций (метод Либмана), метод Зейделя, метод простых итераций с верхней релаксацией.* Вычислить погрешность численного решения путем сравнения результатов с приведенным в задании аналитическим решением *u (x, t)*. Исследовать зависимость погрешности от сеточных параметров *τ* и *h*.

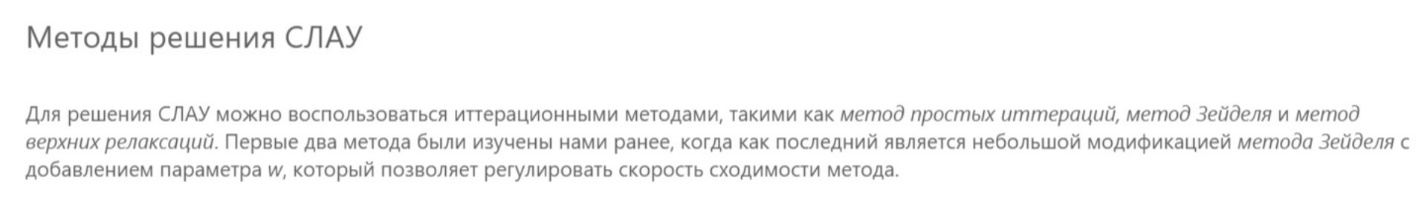


Теоретическая часть:







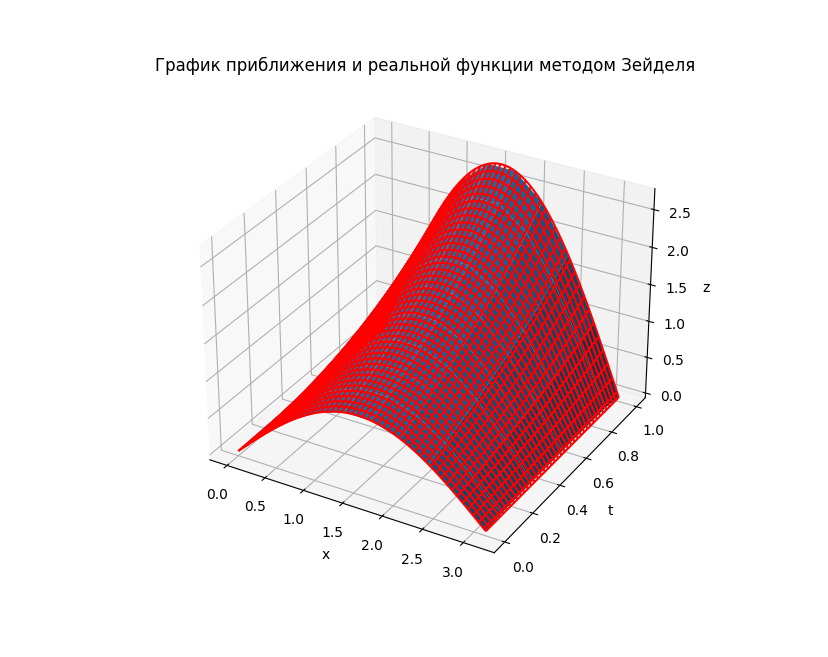


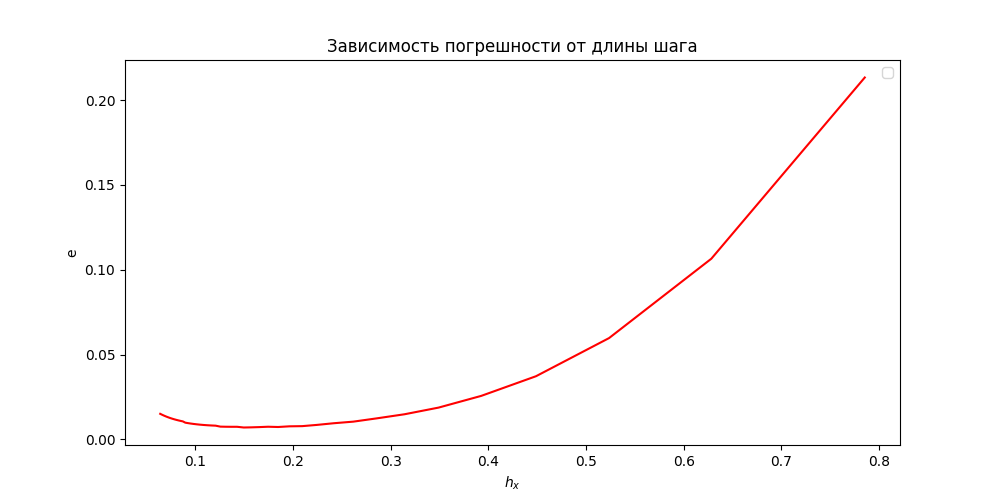
**Код программы:**

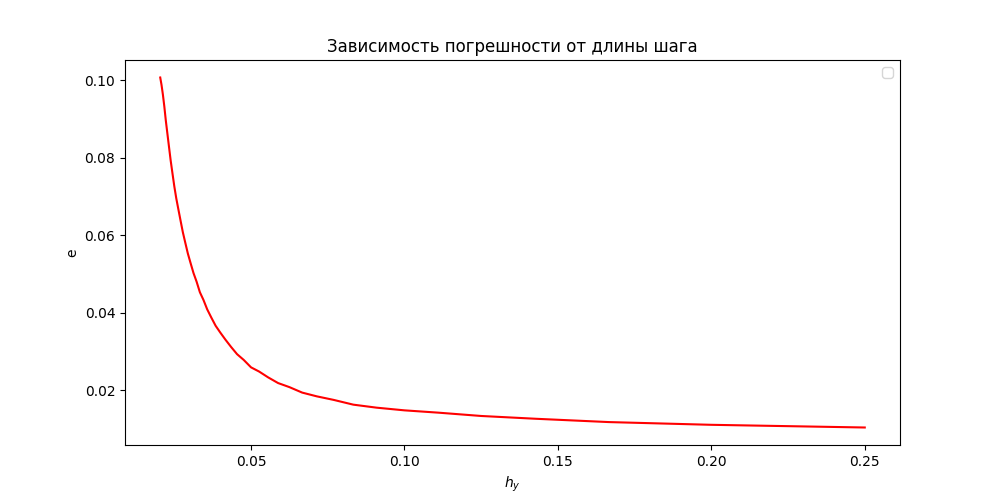
import numpy as np  
import matplotlib.pyplot as plt  
from mpl\_toolkits.mplot3d import Axes3D  
  
def psi\_0(x):  
 return np.sin(x)  
  
def psi\_1(x):  
 return np.sin(x) \* np.e  
  
def phi\_0(y):  
 return np.exp(y)  
  
def phi\_1(y):  
 return -np.exp(y)  
  
*# analytic solve*def u(x, y):  
 return np.exp(y)\*np.sin(x)  
  
  
def func(lx=[0, np.pi], ly=[0, 1], Nx=10, Ny=10, eps=0.001, metod='zeidel', w=1, print\_itters=False):  
 lx = np.array(lx)  
 ly = np.array(ly)  
  
 hx = (lx[1] - lx[0]) / (Nx - 1)  
 hy = (ly[1] - ly[0]) / (Ny - 1)  
  
 def zeidel(X, Y, M, w):  
 return relaxation(X, Y, M, 1)  
  
 def relaxation(X, Y, M, w):  
  
 norm = 0.0  
 hx2 = hx \* hx  
 hy2 = hy \* hy  
  
 for i in range(1, Ny - 1):  
 diff = w \* ((-2 \* hx \* phi\_0(Y[i][0]) + 4 \* M[i][1] - M[i][2]) / 3 - M[i][0])  
 M[i][0] += diff  
 diff = abs(diff)  
 norm = diff if diff > norm else norm  
 for j in range(1, Nx - 1):  
 diff = hy2 \* (M[i][j - 1] + M[i][j + 1])  
 diff += hx2 \* (M[i - 1][j] + M[i + 1][j])  
 diff /= 2 \* (hy2 + hx2)  
 diff -= M[i][j]  
 diff \*= w  
 M[i][j] += diff  
 diff = abs(diff)  
 norm = diff if diff > norm else norm  
 diff = w \* ((2 \* hx \* phi\_1(Y[i][-1]) + 4 \* M[i][-2] - M[i][-3]) / 3 - M[i][-1])  
 M[i][-1] += diff  
 diff = abs(diff)  
 norm = diff if diff > norm else norm  
  
 return norm  
  
 def simple\_eiler(X, Y, M, w):  
 temp = [[0.0 for \_ in range(Nx)] for \_ in range(Ny)]  
 norm = 0.0  
 hx2 = hx \* hx  
 hy2 = hy \* hy  
  
 for i in range(1, Ny - 1):  
 temp[i][0] = (-2 \* hx \* phi\_0(Y[i][0]) + 4 \* M[i][1] - M[i][2]) / 3  
 diff = abs(temp[i][0] - M[i][0])  
 norm = diff if diff > norm else norm  
 for j in range(1, Nx - 1):  
 temp[i][j] = hy2 \* (M[i][j - 1] + M[i][j + 1])  
 temp[i][j] += hx2 \* (M[i - 1][j] + M[i + 1][j])  
 temp[i][j] /= 2 \* (hy2 + hx2)  
 diff = abs(temp[i][j] - M[i][j])  
 norm = diff if diff > norm else norm  
 temp[i][-1] = (2 \* hx \* phi\_1(Y[i][-1]) + 4 \* M[i][-2] - M[i][-3]) / 3  
 diff = abs(temp[i][0] - M[i][0])  
 norm = diff if diff > norm else norm  
  
 for i in range(1, Ny - 1):  
 M[i] = temp[i]  
  
 return norm  
  
 if metod == 'zeidel':  
 method = zeidel  
 elif metod == 'limban':  
 method = simple\_eiler  
 else:  
 method = relaxation  
  
 x = list(np.arange(lx[0], lx[1] + 1 / (10 \* Nx), hx))  
 y = list(np.arange(ly[0], ly[1] + 1 / (10 \* Nx), hy))  
  
 X = [x for \_ in range(Ny)]  
 Y = [[y[i] for \_ in x] for i in range(Ny)]  
  
 ans = [[0 for \_ in range(Nx)] for \_ in range(Ny)]  
 for j in range(Nx):  
 coeff = (psi\_1(X[-1][j]) - psi\_0(X[0][j])) / (ly[1] - ly[0])  
 addition = psi\_0(X[0][j])  
 for i in range(Ny):  
 ans[i][j] = coeff \* (Y[i][j] - ly[0]) + addition  
  
 itters = 0  
  
 while (method(X, Y, ans, w) >= eps):  
 itters += 1  
  
 if print\_itters:  
 print(f"Кол-во итераций: {itters}")  
  
 return np.array(X), np.array(Y), np.array(ans)  
  
  
def real\_z(lx0, lx1, ly0, ly1, f):  
 x = np.arange(lx0, lx1 + 0.005, 0.005)  
 y = np.arange(ly0, ly1 + 0.005, 0.005)  
 X = np.ones((y.shape[0], x.shape[0]))  
 Y = np.ones((x.shape[0], y.shape[0]))  
 Z = np.ones((y.shape[0], x.shape[0]))  
 for i in range(Y.shape[0]):  
 Y[i] = y  
 Y = Y.T  
 for i in range(X.shape[0]):  
 X[i] = x  
 for i in range(Z.shape[0]):  
 for j in range(Z.shape[1]):  
 Z[i, j] = f(X[i, j], Y[i, j])  
 return X, Y, Z  
lx=[0, np.pi]  
ly=[0, 1]  
Nx=10  
Ny=10  
w = 0.5  
  
X, Y, Z = func(lx, ly, Nx, Ny, metod='zeidel', print\_itters=True)  
  
  
fig = plt.figure(figsize=(10, 10))  
ax = fig.add\_subplot(projection='3d')  
ax.plot\_wireframe(\*real\_z(lx[0], lx[1], ly[0], ly[1], u), color="red")  
ax.plot\_surface(X, Y, Z)  
ax.set(xlabel='x', ylabel='t', zlabel='z', title='График приближения и реальной функции методом Зейделя')  
plt.show()  
  
  
  
def epsilon(x, y, z, f):  
 ans = 0.0  
 for i in range(len(z)):  
 for j in range(len(z[i])):  
 ans += (z[i][j] - f(x[i][j], y[i][j]))\*\*2  
 return (ans/(len(z[0])\*len(z)))\*\*0.5  
  
def get\_graphic\_h(solver, real\_f):  
 h = []  
 e = []  
 for N in range(4, 50):  
 x, y, z = solver(Nx=N, metod='zeidel')  
 h.append(np.pi/N)  
 e.append(epsilon(x, y, z, real\_f))  
 return h, e  
  
plt.figure(figsize = (10, 5))  
plt.title("Зависимость погрешности от длины шага")  
h, e = get\_graphic\_h(func, u)  
  
plt.plot(h, e, color = "red")  
plt.xlabel("$h\_x$")  
plt.ylabel("e")  
plt.legend()  
plt.show()  
  
def epsilon(x, y, z, f):  
 ans = 0.0  
 for i in range(len(z)):  
 for j in range(len(z[i])):  
 ans += (z[i][j] - f(x[i][j], y[i][j]))\*\*2  
 return (ans/(len(z[0])\*len(z)))\*\*0.5  
  
def get\_graphic\_h(solver, real\_f):  
 h = []  
 e = []  
 for N in range(4, 50):  
 x, y, z = solver(Ny=N, metod='zeidel')  
 h.append(1/N)  
 e.append(epsilon(x, y, z, real\_f))  
 return h, e  
  
plt.figure(figsize = (10, 5))  
plt.title("Зависимость погрешности от длины шага")  
h, e = get\_graphic\_h(func, u)  
  
plt.plot(h, e, color = "red")  
plt.xlabel("$h\_y$")  
plt.ylabel("e")  
plt.legend()  
plt.show()  
  
  
X, Y, Z = func(lx, ly, Nx, Ny, metod='relax', w=w, print\_itters=True)  
  
fig = plt.figure(figsize=(10, 10))  
ax = fig.add\_subplot(projection='3d')  
ax.plot\_wireframe(\*real\_z(lx[0], lx[1], ly[0], ly[1], u), color="red")  
ax.plot\_surface(X, Y, Z)  
ax.set(xlabel='x', ylabel='t', zlabel='z', title='График приближения и реальной функции явным методом')  
plt.show()  
  
def epsilon(x, y, z, f):  
 ans = 0.0  
 for i in range(len(z)):  
 for j in range(len(z[i])):  
 ans += (z[i][j] - f(x[i][j], y[i][j]))\*\*2  
 return (ans/(len(z[0])\*len(z)))\*\*0.5  
  
def get\_graphic\_h(solver, real\_f):  
 h = []  
 e = []  
 for N in range(4, 50):  
 x, y, z = solver(Nx=N, metod='relax', w=0.5)  
 h.append(np.pi/N)  
 e.append(epsilon(x, y, z, real\_f))  
 return h, e  
  
plt.figure(figsize = (10, 5))  
plt.title("Зависимость погрешности от длины шага")  
h, e = get\_graphic\_h(func, u)  
  
plt.plot(h, e, color = "red")  
plt.xlabel("$h\_x$")  
plt.ylabel("e")  
plt.show()  
  
def epsilon(x, y, z, f):  
 ans = 0.0  
 for i in range(len(z)):  
 for j in range(len(z[i])):  
 ans += (z[i][j] - f(x[i][j], y[i][j]))\*\*2  
 return (ans/(len(z[0])\*len(z)))\*\*0.5  
  
def get\_graphic\_h(solver, real\_f):  
 h = []  
 e = []  
 for N in range(4, 50):  
 x, y, z = solver(Ny=N, metod='relax', w=0.5)  
 h.append(1/N)  
 e.append(epsilon(x, y, z, real\_f))  
 return h, e  
  
plt.figure(figsize = (10, 5))  
plt.title("Зависимость погрешности от длины шага")  
h, e = get\_graphic\_h(func, u)  
  
plt.plot(h, e, color = "red")  
plt.xlabel("$h\_y$")  
plt.ylabel("e")  
plt.show()  
  
  
X, Y, Z = func(lx, ly, Nx, Ny, metod='limban', print\_itters=True)  
  
fig = plt.figure(figsize=(10, 10))  
ax = fig.add\_subplot(projection='3d')  
ax.plot\_wireframe(\*real\_z(lx[0], lx[1], ly[0], ly[1], u), color="red")  
ax.plot\_surface(X, Y, Z)  
ax.set(xlabel='x', ylabel='t', zlabel='z', title='График приближения и реальной функции явным методом')  
plt.show()  
  
def epsilon(x, y, z, f):  
 ans = 0.0  
 for i in range(len(z)):  
 for j in range(len(z[i])):  
 ans += (z[i][j] - f(x[i][j], y[i][j]))\*\*2  
 return (ans/(len(z[0])\*len(z)))\*\*0.5  
  
def get\_graphic\_h(solver, real\_f):  
 h = []  
 e = []  
 for N in range(4, 50):  
 x, y, z = solver(Nx=N, metod='limban')  
 h.append(np.pi/N)  
 e.append(epsilon(x, y, z, real\_f))  
 return h, e  
  
plt.figure(figsize = (10, 5))  
plt.title("Зависимость погрешности от длины шага")  
h, e = get\_graphic\_h(func, u)  
  
plt.plot(h, e, color = "red")  
plt.xlabel("$h\_x$")  
plt.ylabel("e")  
plt.show()  
  
def epsilon(x, y, z, f):  
 ans = 0.0  
 for i in range(len(z)):  
 for j in range(len(z[i])):  
 ans += (z[i][j] - f(x[i][j], y[i][j]))\*\*2  
 return (ans/(len(z[0])\*len(z)))\*\*0.5  
  
def get\_graphic\_h(solver, real\_f):  
 h = []  
 e = []  
 for N in range(4, 50):  
 x, y, z = solver(Ny=N, metod='limban')  
 h.append(1/N)  
 e.append(epsilon(x, y, z, real\_f))  
 return h, e  
  
plt.figure(figsize = (10, 5))  
plt.title("Зависимость погрешности от длины шага")  
h, e = get\_graphic\_h(func, u)  
  
plt.plot(h, e, color = "red")  
plt.xlabel("$h\_y$")  
plt.ylabel("e")  
plt.show()

**Результат:**

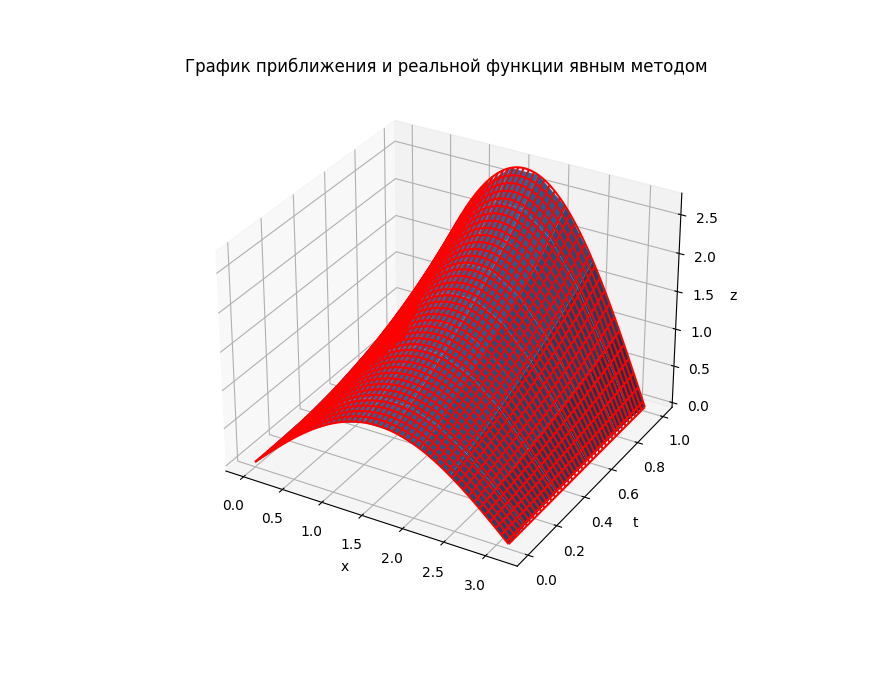
Кол-во итераций: 27

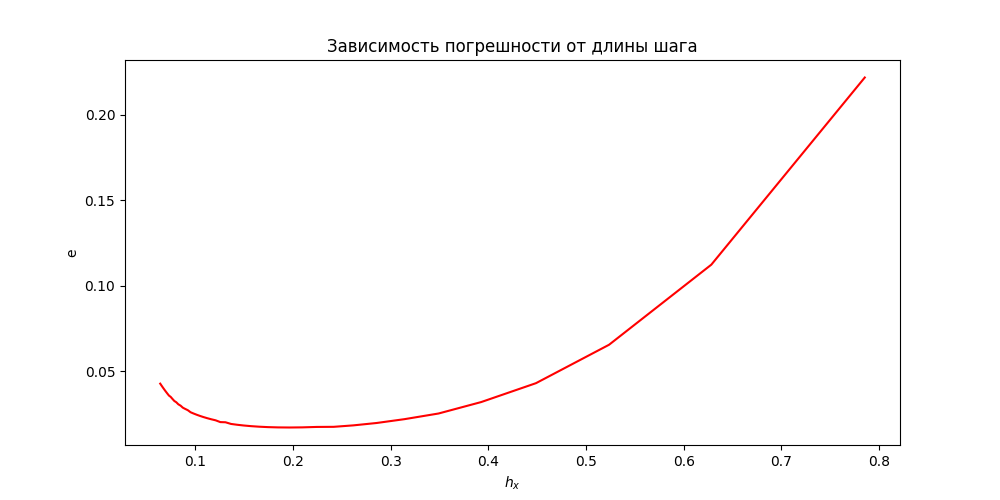


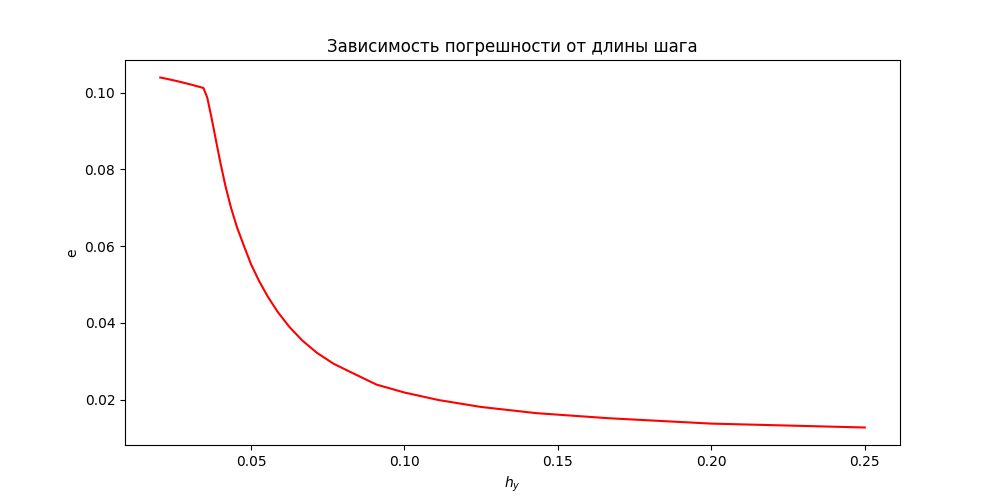




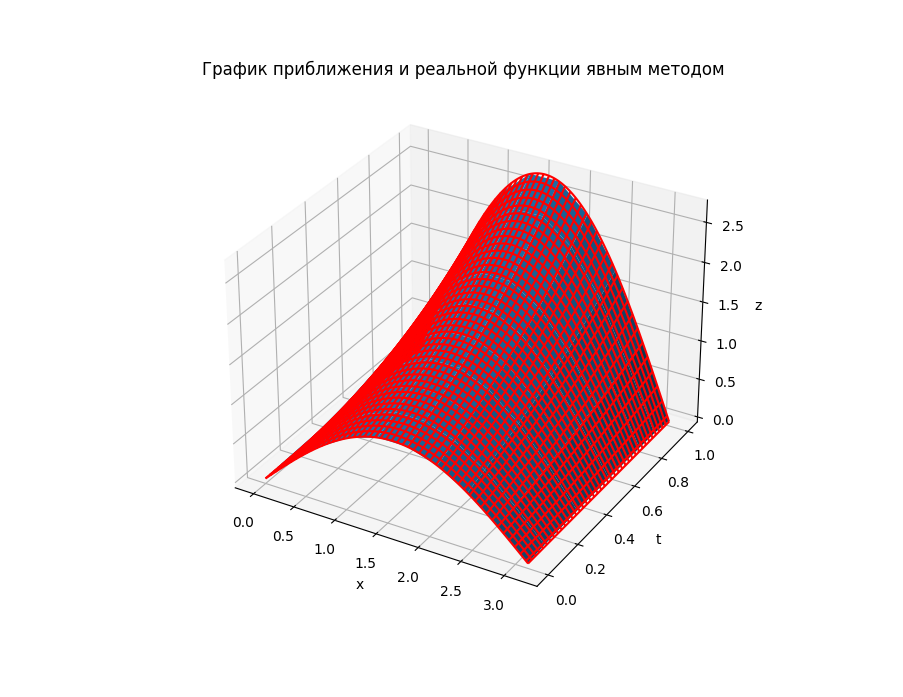
Кол-во итераций: 53

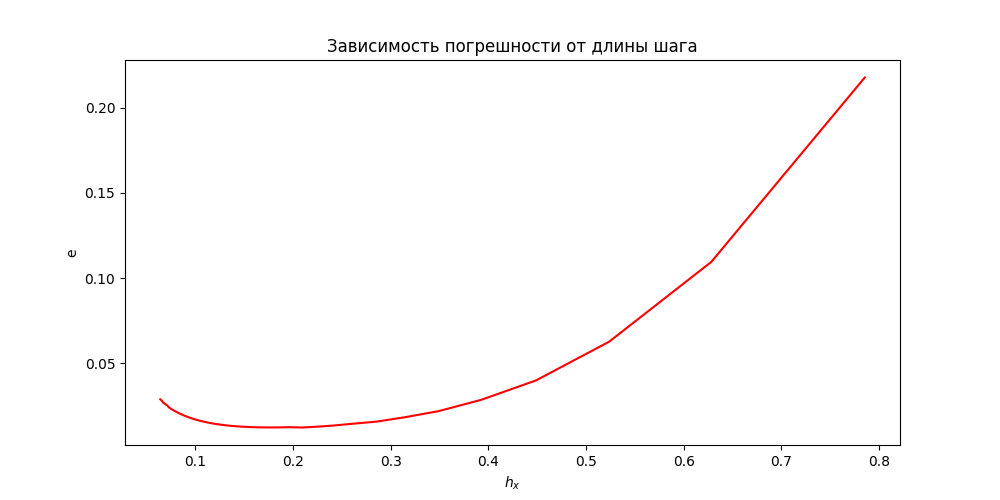


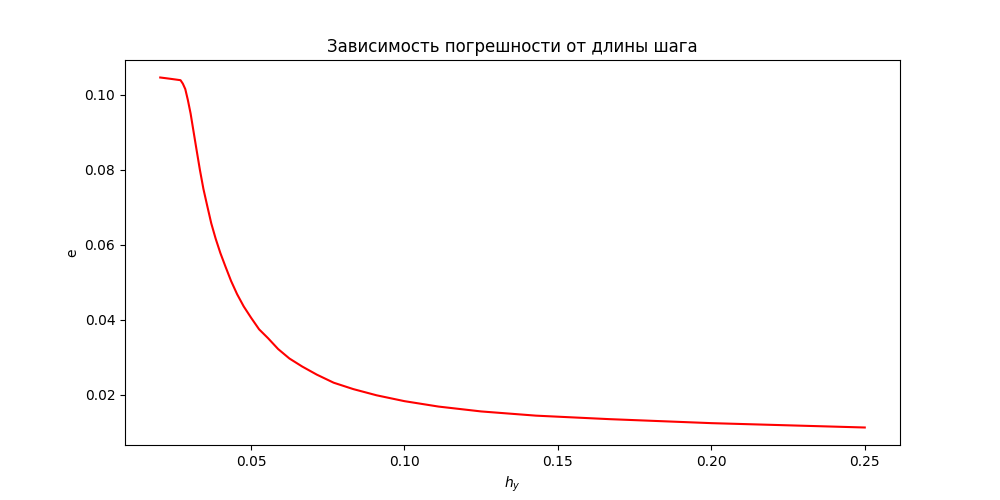




Кол-во итераций: 41







**Вывод:**

В ходе лабораторной работы решена краевая задачу для дифференциального уравнения эллиптического типа. Аппроксимация уравнения произведена с использованием центрально-разностной схемы. Для решения дискретного аналога применены следующие методы: *метод простых итераций (метод Либмана), метод Зейделя, метод простых итераций с верхней релаксацией.* Вычислена погрешность численного решения путем сравнения результатов с приведенным в задании аналитическим решением *u (x, t)*. Исследована зависимость погрешности от сеточных параметров *τ* и *h*.