Московский авиационный институт
(национальный исследовательский университет)

Институт №8 "Информационные технологии и прикладная математика"
Кафедра 806 «Вычислительная математика и программирование»

Лабораторная работа №7

по дисциплине:

Численные методы

Вариант №5

Выполнил: студент группы М8О-409Б-20
Искренкова А.В.
Принял: Пивоваров Е.Д.
Оценка:____

Москва, 2023 г.

1. 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 $U(x, y)$. Исследовать зависимость погрешности от сеточных параметров h_x, h_y.

Уравнение:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = -u$$

$$u_x(0, y) = \cos(y)$$

$$u_x(1, y) - u(1, y) = 0$$

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

$$u\left(x, \frac{\pi}{2}\right) = 0$$

Аналитическое решение:

$$U(x, y) = x\cos(y)$$

2. Решение

$$\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h_x^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h_y^2} = -u_{i,j}$$

/Пусть $h_x = h_y$/

- Метод простых итераций:

$$u_{i,j}^{k+1} = \frac{u_{i+1,j}^k + u_{i-1,j}^k + u_{i,j+1}^k + u_{i,j-1}^k}{4 - h^2}$$

- Метод Зейделя:

$$u_{i,j}^{k+1} = \frac{u_{i+1,j}^k + u_{i-1,j}^{k+1} + u_{i,j+1}^k + u_{i,j-1}^{k+1}}{4 - h^2}$$

- Метод простых итераций с релаксацией:

$$u_{i,j}^{k+1} = \left(\frac{u_{i+1,j}^k + u_{i-1,j}^k + u_{i,j+1}^k + u_{i,j-1}^k}{4 - h^2}\right)\text{C} + u_{i,j}^k(1 - \text{C})$$

- Метод Зейделя с релаксацией:

$$u_{i,j}^{k+1} = \left(\frac{u_{i+1,j}^k + u_{i-1,j}^{k+1} + u_{i,j+1}^k + u_{i,j-1}^{k+1}}{4 - h^2}\right)\text{C} + u_{i,j}^k(1 - \text{C})$$

Аппроксимация граничных условий:

$$u_{0,j}^{k} = u_{1,j}^{k} - h_x \cos(y)$$

$$u_{N,j}^{k} = \frac{u_{N-1,j}^{k}}{(1 - h_x)}$$

Заполнение начального уровня (инициализацию) проведем с помощью линейной интерполяции. Учитывая, что $u(x, 0) = 0$ и $u\left(x, \frac{\pi}{2}\right) = 0$, имеем:

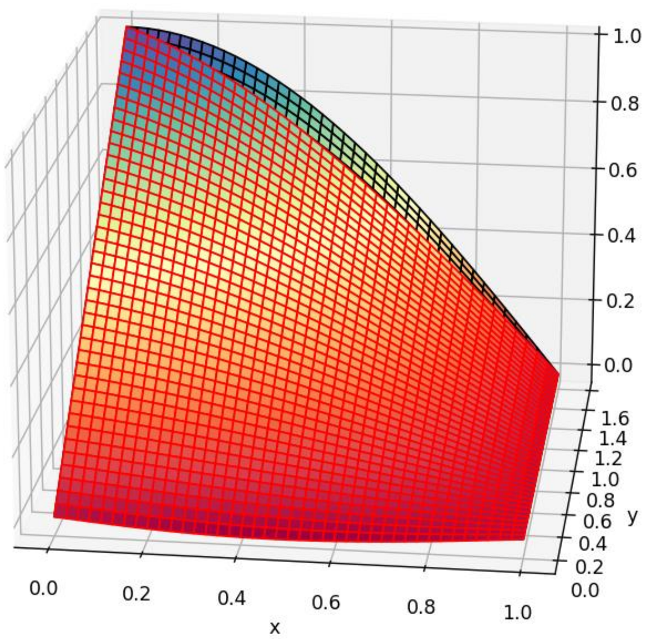
$$u_{i,j}^{0} = x_i * \frac{1 - y_j}{\frac{\pi}{2}}$$

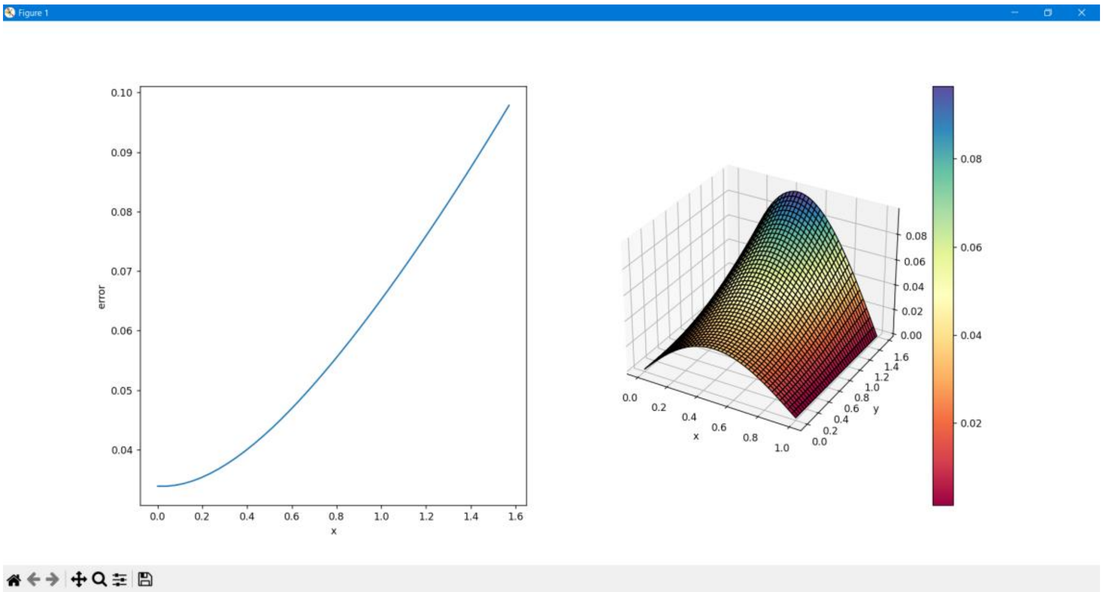
Погрешность между численным и аналитическим решением рассчитывается как абсолютная.

Параметры задачи: разбиение по x и по y = 40, eps = 0.000001

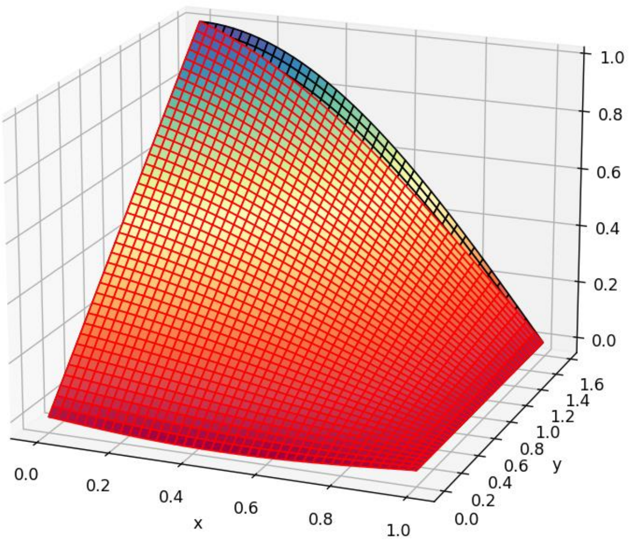
3. Вывод программы

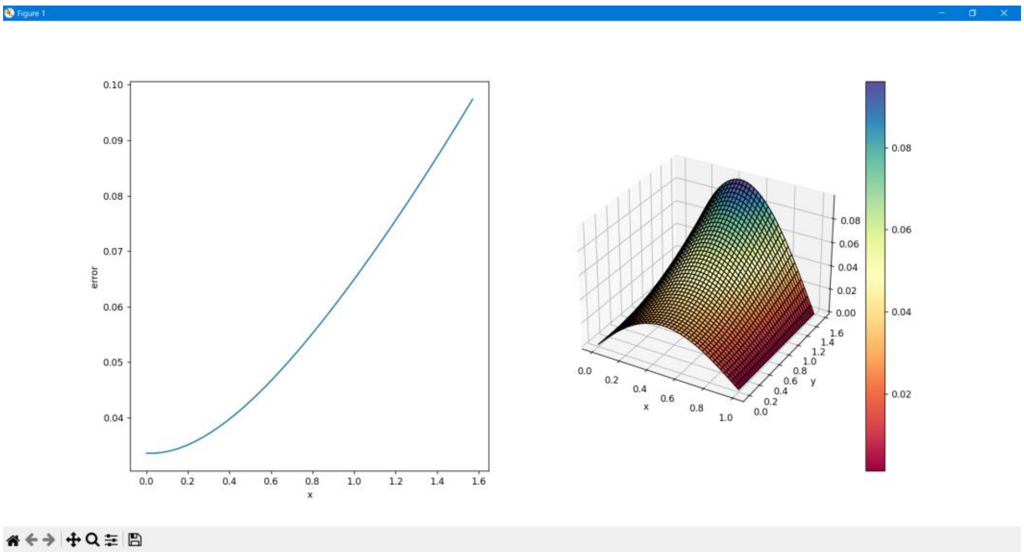
- Метод простых итераций (MPI k = 4214)



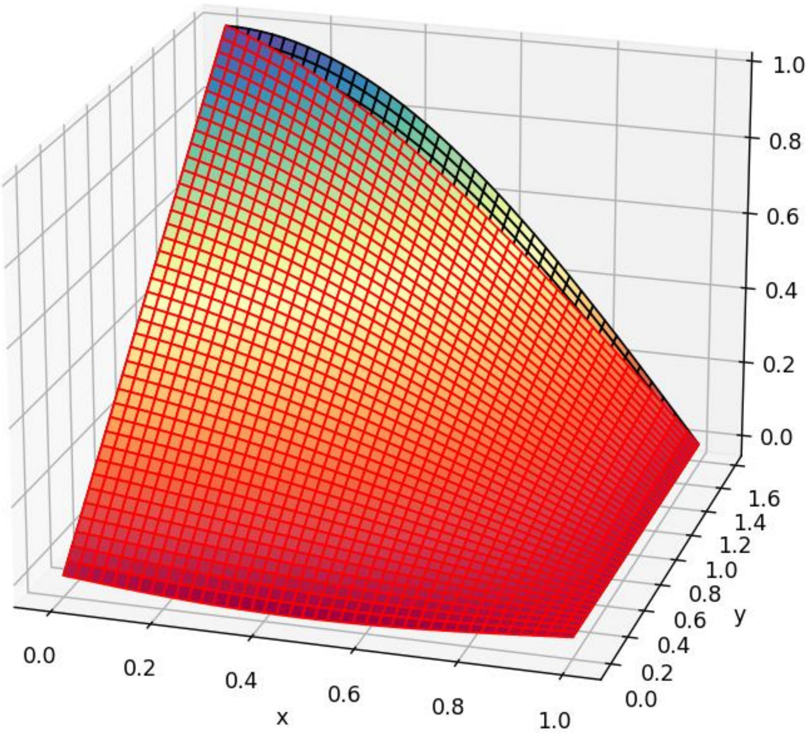


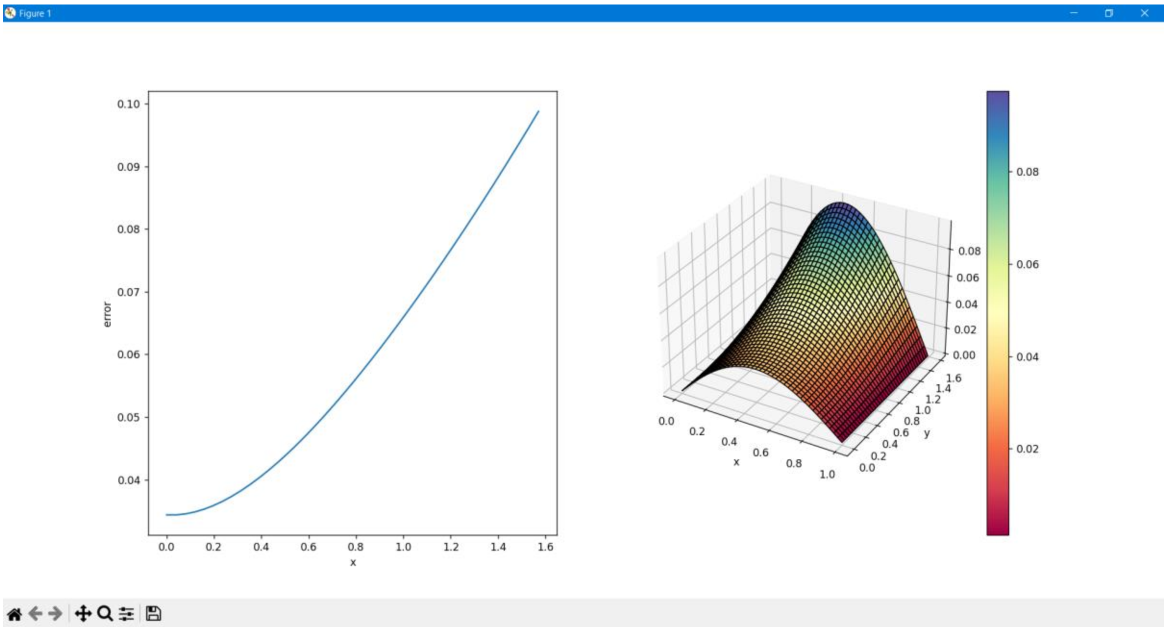
- Метод Зейделя (ZEI k = 2510)



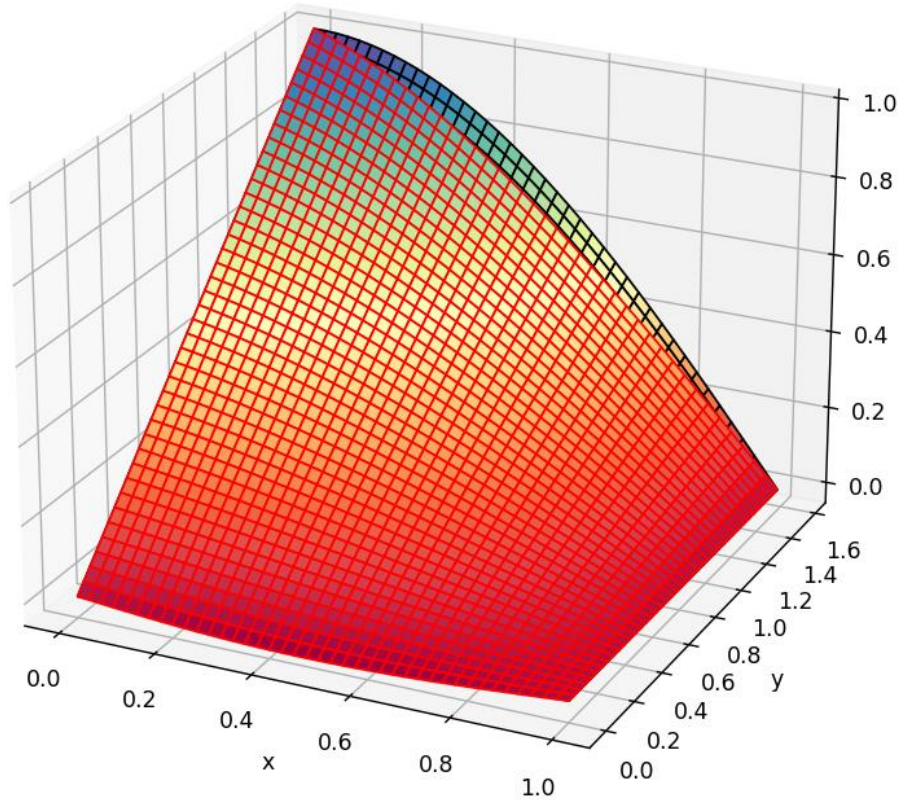


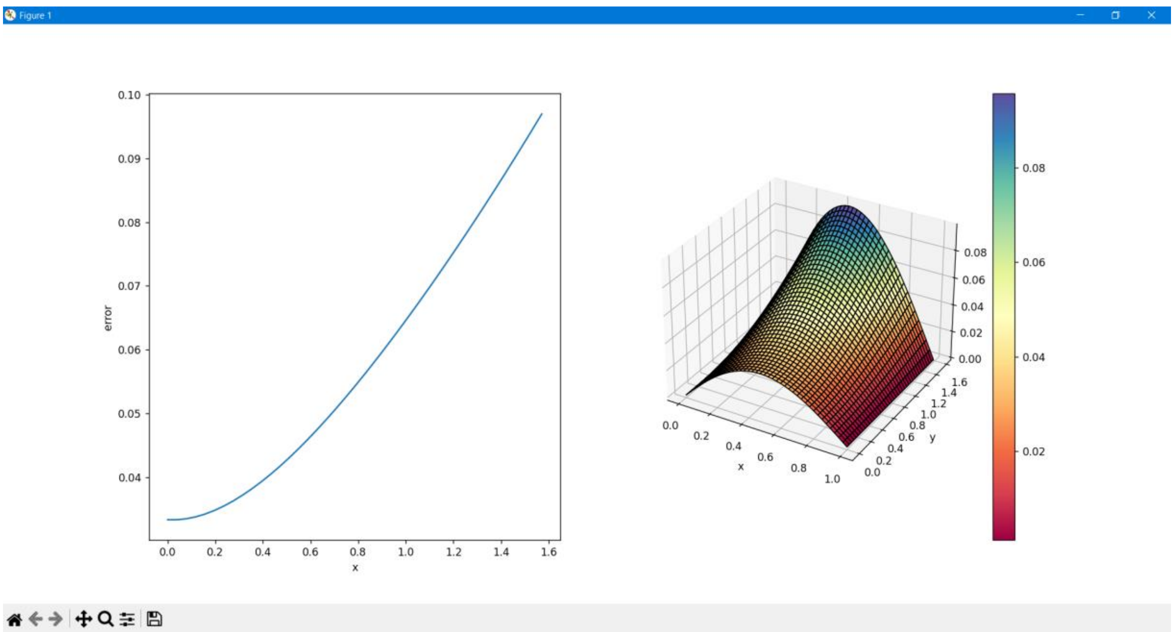
- Метод простых итераций с релаксацией (C = 0.5, MPI_relax k = 7059)





- Метод Зейделя с релаксацией (C = 1.91 – найдено подбором; ZEI_relax k = 373)





4. Листинг

```
import numpy as np
import matplotlib.pyplot as plt
from math import sqrt
import matplotlib.cm

def stop(L,U,x,y):
    maxx = 0
    for i in range(len(x)):
        for j in range(len(y)):
            if abs(U[i,j] - L[i,j]) > maxx:
                maxx = abs(U[i,j] - L[i,j])
    return maxx

def true_fval(x, y):
    return x*np.cos(y)

lbx = 0
ubx = 1
nx = 40
hx = (ubx - lbx)/nx

lby = 0
uby = np.pi/2
ny = 40
hy = (uby - lby)/ny

x = np.arange(lbx, ubx + hx, hx)
y = np.arange(lby, uby + hy, hy)

eps = 0.000001

def MPI(hx, hy, eps, lbx, lby, ubx, uby):
    x = np.arange(lbx, ubx + hx, hx)
    y = np.arange(lby, uby + hy, hy)
    U = np.zeros((len(x),len(y)))
    for i in range(len(x)):
        U[i,0] = x[i]
        U[i,-1] = 0
    k = 0
    for j in range(1,len(y)-1):
        for i in range(len(x)):
            U[i,j] = U[i,0] * (y[-1]-y[j])/y[-1]
    while True:
        k = k+1
        L = np.copy(U)
        U = np.zeros((len(x),len(y)))
        for i in range(len(x)):
```

```
            U[i,0] = x[i]
            U[i,-1] = 0
        for j in range(1, len(y)-1):
            U[0,j] = L[1,j]-hx*np.cos(y[j])
            U[-1,j] = L[-2,j]/(1-hx)

        for j in range(1, len(y)- 1):
            for i in range(1, len(x)- 1):
                U[i,j] = (L[i+1,j] + L[i-1,j] + L[i,j+1] + L[i,j-1])/(4-hx*hy)
        if stop(L,U,x,y) <= eps:
            print('MPI eps = ',stop(L,U,x,y))
            print('MPI k = ', k)
            break

    return U

def ZEI(hx, hy, eps, lbx, lby, ubx, uby):
    x = np.arange(lbx, ubx + hx, hx)
    y = np.arange(lby, uby + hy, hy)
    U = np.zeros((len(x),len(y)))
    for i in range(len(x)):
        U[i,0] = x[i]
        U[i,-1] = 0
    k = 0
    for j in range(1,len(y)-1):
        for i in range(len(x)):
            U[i,j] = U[i,0] * (y[-1]-y[j])/y[-1]
    while True:
        k = k+1
        L = np.copy(U)
        U = np.zeros((len(x),len(y)))
        for i in range(len(x)):
            U[i,0] = x[i]
            U[i,-1] = 0
        for j in range(1, len(y)-1):
            U[0,j] = L[1,j]-hx*np.cos(y[j])
            U[-1,j] = L[-2,j]/(1-hx)

        for j in range(1, len(y)- 1):
            for i in range(1, len(x)- 1):
                U[i,j] = (L[i+1,j] + U[i-1,j] + L[i,j+1] + U[i,j-1])/(4-hx*hy)
        if stop(L,U,x,y) <= eps:
            print('ZEI eps = ',stop(L,U,x,y))
            print('ZEI k = ', k)
            break

    return U

def MPI_relax(hx, hy, eps, lbx, lby, ubx, uby, C):
```

```
    x = np.arange(lbx, ubx + hx, hx)
    y = np.arange(lby, uby + hy, hy)
    U = np.zeros((len(x),len(y)))
    for i in range(len(x)):
        U[i,0] = x[i]
        U[i,-1] = 0
    k = 0
    for j in range(1,len(y)-1):
        for i in range(len(x)):
            U[i,j] = U[i,0] * (y[-1]-y[j])/y[-1]
    while True:
        k = k+1
        L = np.copy(U)
        U = np.zeros((len(x),len(y)))
        for i in range(len(x)):
            U[i,0] = x[i]
            U[i,-1] = 0
        for j in range(1, len(y)-1):
            U[0,j] = L[1,j]-hx*np.cos(y[j])
            U[-1,j] = L[-2,j]/(1-hx)

        for j in range(1, len(y)- 1):
            for i in range(1, len(x)- 1):
                U[i,j] = ((L[i+1,j] + L[i-1,j] + L[i,j+1] + L[i,j-1])/(4-hx*hy))*C + L[i,j]*(1-C)
        if stop(L,U,x,y) <= eps:
            print('MPI_relax eps = ',stop(L,U,x,y))
            print('MPI_relax k = ', k)
            break

    return U

def ZEI_relax(hx, hy, eps, lbx, lby, ubx, uby,C):
    x = np.arange(lbx, ubx + hx, hx)
    y = np.arange(lby, uby + hy, hy)
    U = np.zeros((len(x),len(y)))
    for i in range(len(x)):
        U[i,0] = x[i]
        U[i,-1] = 0
    k = 0
    for j in range(1,len(y)-1):
        for i in range(len(x)):
            U[i,j] = U[i,0] * (y[-1]-y[j])/y[-1]
    while True:
        k = k+1
        L = np.copy(U)
        U = np.zeros((len(x),len(y)))
        for i in range(len(x)):
            U[i,0] = x[i]
            U[i,-1] = 0
        for j in range(1, len(y)-1):
```

```python
            U[0,j] = L[1,j]-hx*np.cos(y[j])
            U[-1,j] = L[-2,j]/(1-hx)

        for j in range(1, len(y)- 1):
            for i in range(1, len(x)- 1):
                U[i,j] = ((L[i+1,j] + U[i-1,j] + L[i,j+1] + U[i,j-1])/(4-hx*hy))*C + L[i,j]*(1-C)
        if stop(L,U,x,y) <= eps:
            print('ZEI_relax eps = ',stop(L,U,x,y))
            print('ZEI_relax k = ', k)
            break

    return U

def plot_U(z, u):
    x = np.arange(lbx, ubx + hx, hx)
    y = np.arange(lby, uby + hy, hy)
    xx, yy = np.meshgrid(x, y)
    fig = plt.figure(figsize = (20, 7))
    d = abs(z - u)


    ax1 = fig.add_subplot(1, 2, 1)

    plt.xlabel("x")
    plt.ylabel("error")

    ax2 = fig.add_subplot(1, 2, 2, projection = "3d")

    plt.xlabel("x")
    plt.ylabel("y")

    surf = ax2.plot_surface(xx, yy, d,
                             edgecolors = ["black"], linewidth = 1,
                             cmap = matplotlib.cm.Spectral, shade = True, antialiased = True)
    fig.colorbar(surf)

    ppp = []

    for i in range(len(y)):
        mmm = d[i][0]
        for j in range(len(x)):
            mmm = max(d[i][j], mmm)
        ppp.append(mmm)

    ax1.plot(y, ppp)

    fig1, ax = plt.subplots(1, 1, figsize = (20, 20), subplot_kw = {"projection": "3d"})
    ax.view_init(30, 30)
```

```
    surf = ax.plot_surface(xx, yy, z,
                         edgecolors = ["black"], linewidth = 1,
                         cmap = matplotlib.cm.Spectral, shade = True, antialiased = True)
    surf1 = ax.plot_surface(xx, yy, u,
                         edgecolors = ["red"], linewidth = 1,
                         cmap = matplotlib.cm.Spectral, shade = False, antialiased = True)
    plt.xlabel("x")
    plt.ylabel("y")

    #fig1.colorbar(surf)
    #fig1.colorbar(surf1)

    plt.show()
    return

z = np.zeros((len(x),len(y)))
for i in range(len(x)):
        for j in range(len(y)):
            z[i,j]= true_fval(x[i],y[j])

#u1 = MPI(hx, hy, eps, lbx, lby, ubx, uby)
#u2 = ZEI(hx, hy, eps, lbx, lby, ubx, uby)
#u3 = MPI_relax(hx, hy, eps, lbx, lby, ubx, uby,0.5)
u4 = ZEI_relax(hx, hy, eps, lbx, lby, ubx, uby,1.91)

#plot_U(z,u1)
#plot_U(z,u2)
#plot_U(z,u3)
plot_U(z,u4)

```

5. Вывод

В ходе выполнения лабораторной работы были изучены метод простых итераций и метод Зейделя решений начально-краевой задачи для дифференциального уравнения эллиптического типа. Была применена двухточечная аппроксимация первого порядка граничных условий и линейная интерполяция для инициализации итерационных методов. Были получены результаты в графическом представлении и подсчитаны погрешности для каждого варианта решения.