

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:

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

- Метод Зейделя с релаксацией:

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

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:

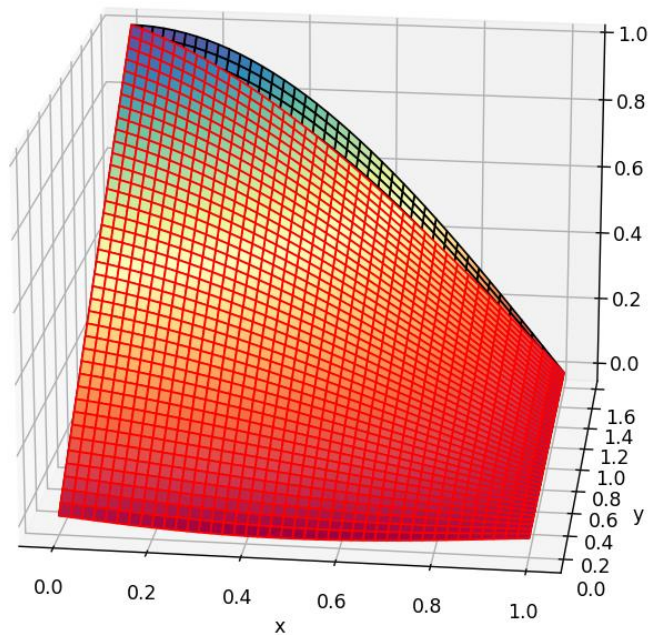
$$u_{i,j}^0 = x_i * \frac{1 - y_j}{\frac{\pi}{2}}$$

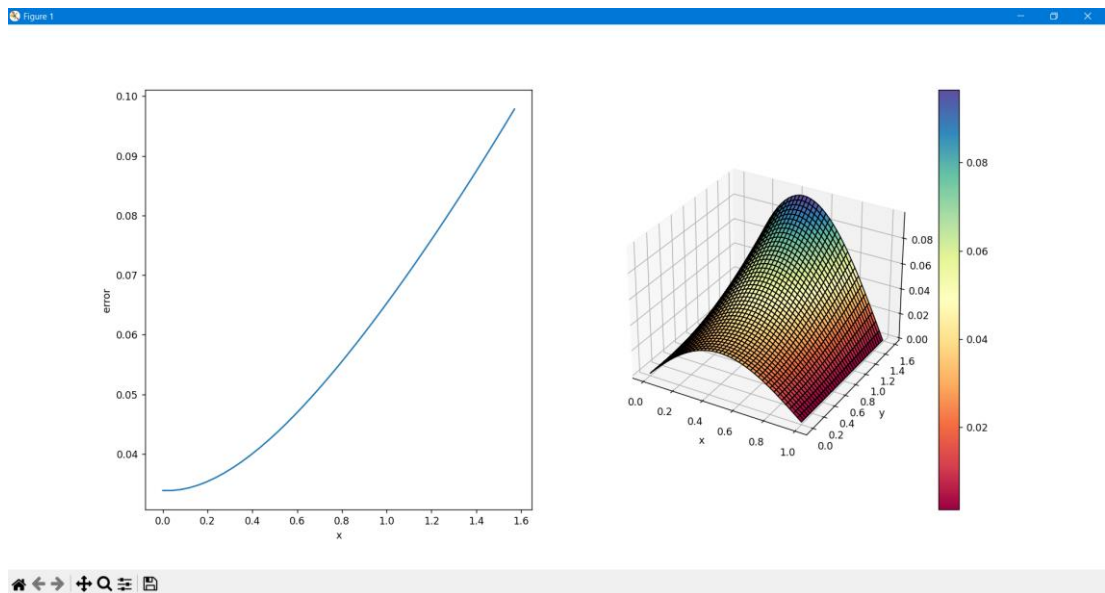
Погрешность между численным и аналитическим решением рассчитывается как абсолютная.

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

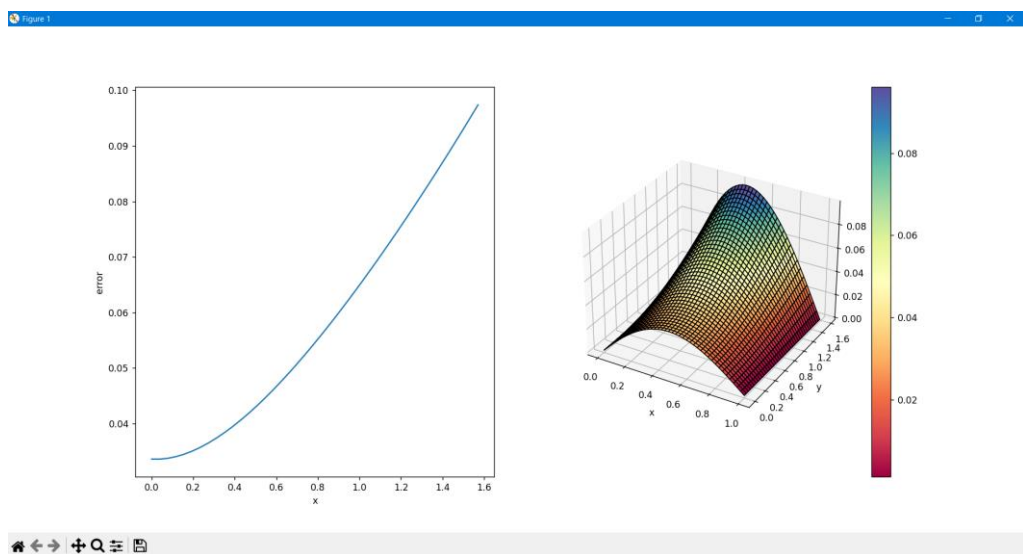
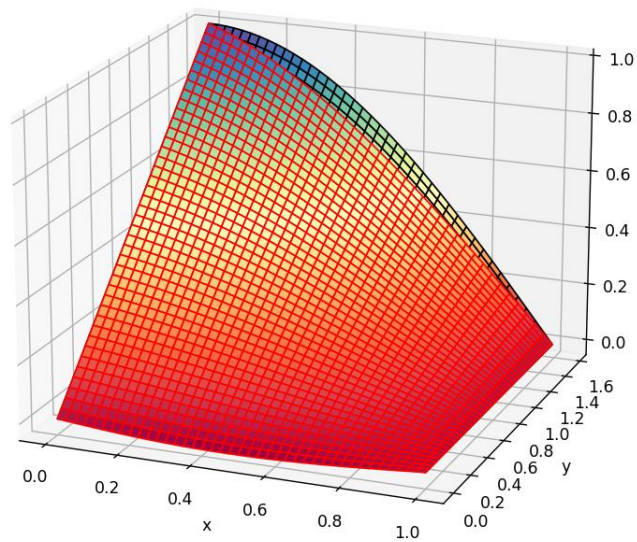
3. Вывод программы

- Метод простых итераций (MPI k = 4214)

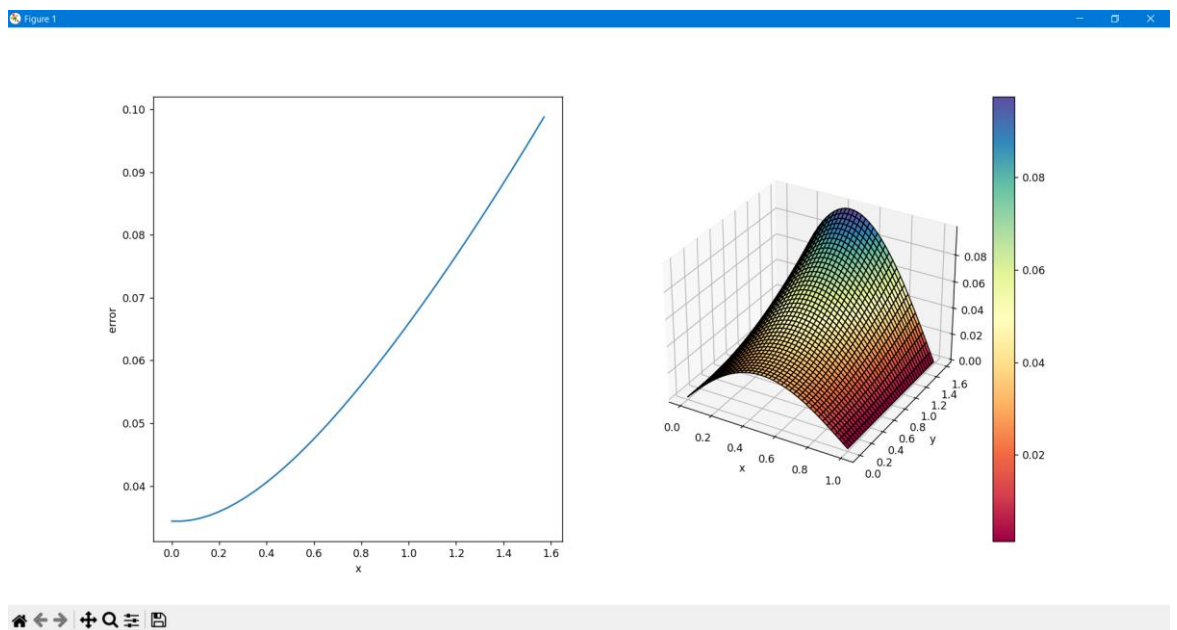
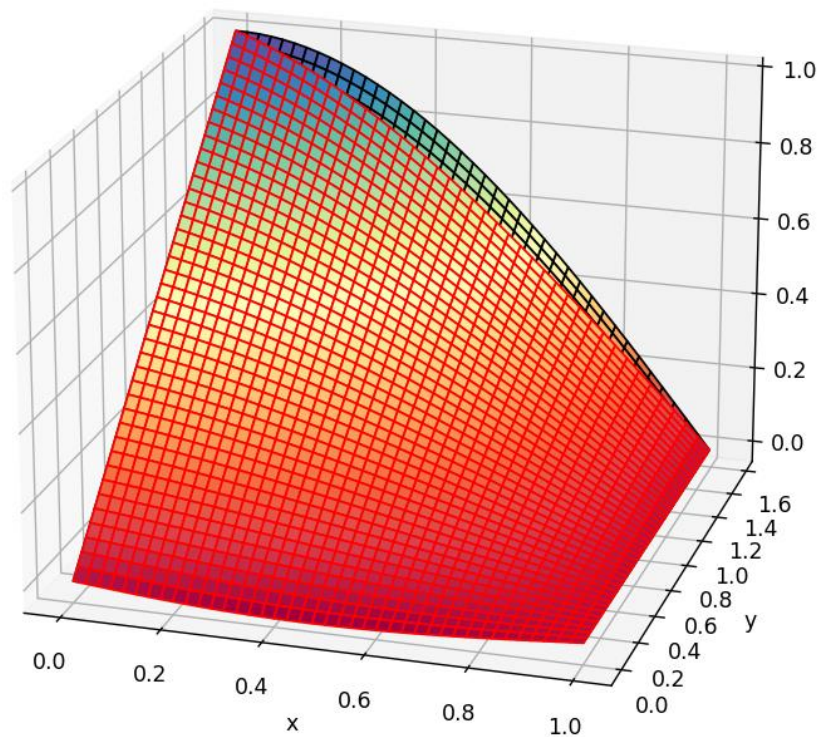




- Метод Зейделя (ZEI $k = 2510$)



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



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

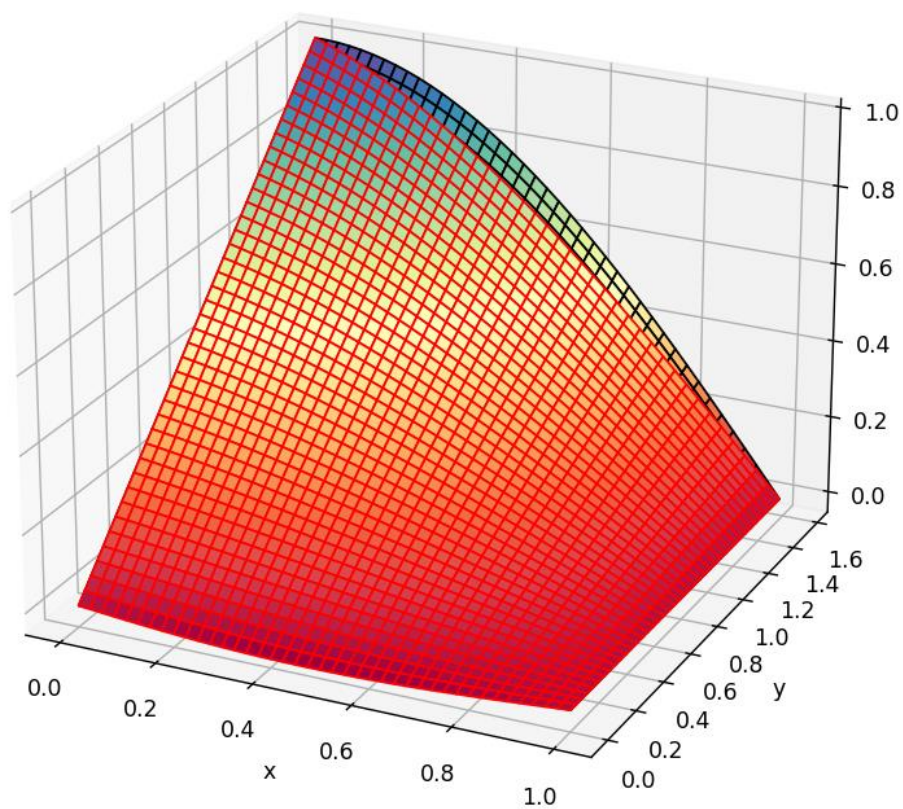
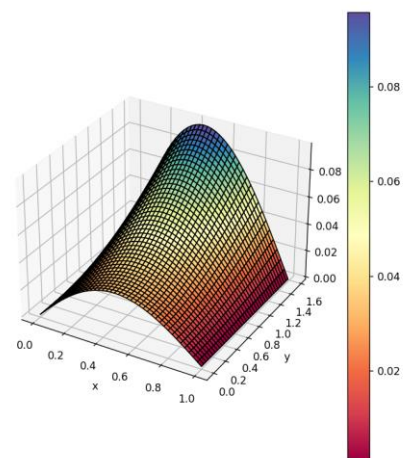
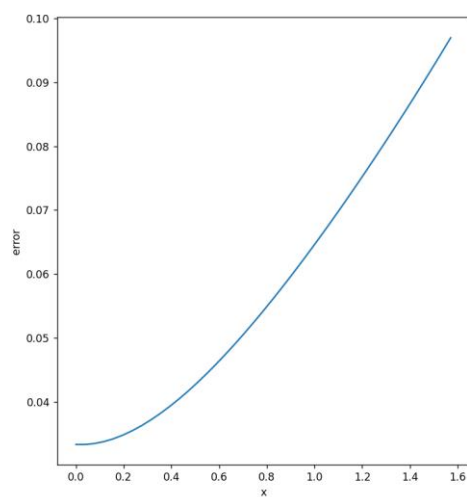


Figure 1



4. Листинг

```
5. import numpy as np
6. import matplotlib.pyplot as plt
7. from math import sqrt
8. import matplotlib.cm
9.
10. def stop(L,U,x,y):
11.     maxx = 0
12.     for i in range(len(x)):
13.         for j in range(len(y)):
14.             if abs(U[i,j] - L[i,j]) > maxx:
15.                 maxx = abs(U[i,j] - L[i,j])
16.     return maxx
17.
18. def true_fval(x, y):
19.     return x*np.cos(y)
20.
21. lbx = 0
22. ubx = 1
23. nx = 40
24. hx = (ubx - lbx)/nx
25.
26. lby = 0
27. uby = np.pi/2
28. ny = 40
29. hy = (uby - lby)/ny
30.
31. x = np.arange(lbx, ubx + hx, hx)
32. y = np.arange(lby, uby + hy, hy)
33.
34. eps = 0.000001
35.
36. def MPI(hx, hy, eps, lbx, lby, ubx, uby):
37.     x = np.arange(lbx, ubx + hx, hx)
38.     y = np.arange(lby, uby + hy, hy)
39.     U = np.zeros((len(x),len(y)))
40.     for i in range(len(x)):
41.         U[i,0] = x[i]
42.         U[i,-1] = 0
43.     k = 0
44.     for j in range(1,len(y)-1):
45.         for i in range(len(x)):
46.             U[i,j] = U[i,0] * (y[-1]-y[j])/y[-1]
47.     while True:
48.         k = k+1
49.         L = np.copy(U)
50.         U = np.zeros((len(x),len(y)))
51.         for i in range(len(x)):
```



```

52.         U[i,0] = x[i]
53.         U[i,-1] = 0
54.     for j in range(1, len(y)-1):
55.         U[0,j] = L[1,j]-hx*np.cos(y[j])
56.         U[-1,j] = L[-2,j]/(1-hx)
57.
58.     for j in range(1, len(y)- 1):
59.         for i in range(1, len(x)- 1):
60.             U[i,j] = (L[i+1,j] + L[i-1,j] + L[i,j+1] + L[i,j-
61. 1])/(4-hx*hy)
62.     if stop(L,U,x,y) <= eps:
63.         print('MPI eps = ',stop(L,U,x,y))
64.         print('MPI k = ', k)
65.         break
66.
67.     return U
68.
69. def ZEI(hx, hy, eps, lbx, lby, ubx, uby):
70.     x = np.arange(lbx, ubx + hx, hx)
71.     y = np.arange(lby, uby + hy, hy)
72.     U = np.zeros((len(x),len(y)))
73.     for i in range(len(x)):
74.         U[i,0] = x[i]
75.         U[i,-1] = 0
76.     k = 0
77.     for j in range(1,len(y)-1):
78.         for i in range(len(x)):
79.             U[i,j] = U[i,0] * (y[-1]-y[j])/y[-1]
80.     while True:
81.         k = k+1
82.         L = np.copy(U)
83.         U = np.zeros((len(x),len(y)))
84.         for i in range(len(x)):
85.             U[i,0] = x[i]
86.             U[i,-1] = 0
87.         for j in range(1, len(y)-1):
88.             U[0,j] = L[1,j]-hx*np.cos(y[j])
89.             U[-1,j] = L[-2,j]/(1-hx)
90.
91.         for j in range(1, len(y)- 1):
92.             for i in range(1, len(x)- 1):
93.                 U[i,j] = (L[i+1,j] + U[i-1,j] + L[i,j+1] + U[i,j-
94. 1])/(4-hx*hy)
95.         if stop(L,U,x,y) <= eps:
96.             print('ZEI eps = ',stop(L,U,x,y))
97.             print('ZEI k = ', k)
98.             break
99.
100.     return U
101.
102. def MPI_relax(hx, hy, eps, lbx, lby, ubx, uby, C):

```



```

101.     x = np.arange(lbx, ubx + hx, hx)
102.     y = np.arange(lby, uby + hy, hy)
103.     U = np.zeros((len(x),len(y)))
104.     for i in range(len(x)):
105.         U[i,0] = x[i]
106.         U[i,-1] = 0
107.     k = 0
108.     for j in range(1,len(y)-1):
109.         for i in range(len(x)):
110.             U[i,j] = U[i,0] * (y[-1]-y[j])/y[-1]
111.     while True:
112.         k = k+1
113.         L = np.copy(U)
114.         U = np.zeros((len(x),len(y)))
115.         for i in range(len(x)):
116.             U[i,0] = x[i]
117.             U[i,-1] = 0
118.         for j in range(1, len(y)-1):
119.             U[0,j] = L[1,j]-hx*np.cos(y[j])
120.             U[-1,j] = L[-2,j]/(1-hx)
121.
122.             for j in range(1, len(y)- 1):
123.                 for i in range(1, len(x)- 1):
124.                     U[i,j] = ((L[i+1,j] + L[i-1,j] + L[i,j+1] + L[i,j-
125.                     1]))/(4-hx*hy))*C + L[i,j]*(1-C)
126.                     if stop(L,U,x,y) <= eps:
127.                         print('MPI_relax eps = ',stop(L,U,x,y))
128.                         print('MPI_relax k = ', k)
129.                         break
130.     return U
131.
132. def ZEI_relax(hx, hy, eps, lbx, lby, ubx, uby,C):
133.     x = np.arange(lbx, ubx + hx, hx)
134.     y = np.arange(lby, uby + hy, hy)
135.     U = np.zeros((len(x),len(y)))
136.     for i in range(len(x)):
137.         U[i,0] = x[i]
138.         U[i,-1] = 0
139.     k = 0
140.     for j in range(1,len(y)-1):
141.         for i in range(len(x)):
142.             U[i,j] = U[i,0] * (y[-1]-y[j])/y[-1]
143.     while True:
144.         k = k+1
145.         L = np.copy(U)
146.         U = np.zeros((len(x),len(y)))
147.         for i in range(len(x)):
148.             U[i,0] = x[i]
149.             U[i,-1] = 0
150.         for j in range(1, len(y)-1):

```

```

151.         U[0,j] = L[1,j]-hx*np.cos(y[j])
152.         U[-1,j] = L[-2,j]/(1-hx)
153.
154.         for j in range(1, len(y)- 1):
155.             for i in range(1, len(x)- 1):
156.                 U[i,j] = ((L[i+1,j] + U[i-1,j] + L[i,j+1] + U[i,j-
157. 1]))/(4-hx*hy))*C + L[i,j]*(1-C)
158.                 if stop(L,U,x,y) <= eps:
159.                     print('ZEI_relax eps = ',stop(L,U,x,y))
160.                     print('ZEI_relax k = ', k)
161.                     break
162.
163.     return U
164.
165. def plot_U(z, u):
166.     x = np.arange(lbx, ubx + hx, hx)
167.     y = np.arange(lby, uby + hy, hy)
168.     xx, yy = np.meshgrid(x, y)
169.     fig = plt.figure(figsize = (20, 7))
170.     d = abs(z - u)
171.
172.     ax1 = fig.add_subplot(1, 2, 1)
173.
174.     plt.xlabel("x")
175.     plt.ylabel("error")
176.
177.     ax2 = fig.add_subplot(1, 2, 2, projection = "3d")
178.
179.     plt.xlabel("x")
180.     plt.ylabel("y")
181.
182.     surf = ax2.plot_surface(xx, yy, d,
183.                             edgecolors = ["black"], linewidth = 1,
184.                             cmap = matplotlib.cm.Spectral, shade =
185. True, antialiased = True)
186.     fig.colorbar(surf)
187.
188.     ppp = []
189.
190.     for i in range(len(y)):
191.         mmm = d[i][0]
192.         for j in range(len(x)):
193.             mmm = max(d[i][j], mmm)
194.         ppp.append(mmm)
195.
196.     ax1.plot(y, ppp)
197.
198.     fig1, ax = plt.subplots(1, 1, figsize = (20, 20), subplot_kw =
{"projection": "3d"})
    ax.view_init(30, 30)

```

```

199.     surf = ax.plot_surface(xx, yy, z,
200.                             edgecolors = ["black"], linewidth = 1,
201.                             cmap = matplotlib.cm.Spectral, shade = True,
    antialiased = True)
202.     surf1 = ax.plot_surface(xx, yy, u,
203.                             edgecolors = ["red"], linewidth = 1,
204.                             cmap = matplotlib.cm.Spectral, shade = False,
    antialiased = True)
205.     plt.xlabel("x")
206.     plt.ylabel("y")
207.
208.     #fig1.colorbar(surf)
209.     #fig1.colorbar(surf1)
210.
211.     plt.show()
212.     return
213.
214.     z = np.zeros((len(x),len(y)))
215.     for i in range(len(x)):
216.         for j in range(len(y)):
217.             z[i,j]= true_fval(x[i],y[j])
218.
219.     #u1 = MPI(hx, hy, eps, Lbx, Lby, ubx, uby)
220.     #u2 = ZEI(hx, hy, eps, Lbx, Lby, ubx, uby)
221.     #u3 = MPI_relax(hx, hy, eps, Lbx, Lby, ubx, uby,0.5)
222.     u4 = ZEI_relax(hx, hy, eps, lbx, lby, ubx, uby,1.91)
223.
224.     #plot_U(z,u1)
225.     #plot_U(z,u2)
226.     #plot_U(z,u3)
227.     plot_U(z,u4)
228.

```

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.