Санкт-Петербургский политехнический университет Петра Великого
Институт прикладной математики и механики
Высшая школа теоретической механики

Лабораторная работа №3

Уравнение Лапласа. Вариант 6.

Студент: А.А. Дурнев
Преподаватель: Е.Ю.Витохин

Санкт-Петербург
2020

Содержание

1 Постановка задачи

Необходимо, используя метод конечных разностей, составить приближённое решение задачи Дирихле для уравнения Лапласа в квадрате $ABCD$ с вершинами $A(0,0)$, $B(0,1)$, $C(1,1)$, $D(1,0)$. Шаг: $h = 0.2$, точность: $\epsilon = 0.01$.

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \tag{1}$$

где x, y - пространственные координаты.

Граничные условия:

$$U|_{AB} = 30sin(\pi y),\ U|_{BC} = 20x,\ U|_{CD} = 20y,\ U|_{AD} = 30x(1-x) \tag{2}$$

Для численного решения уравнения будем использовать итерационный метод последовательных релаксаций.

2 Описание метода

Задаём сетки по осям x и y:

$$y_j = jh,\ j = 0, \ldots, N \tag{3}$$

$$x_i = ih,\ i = 0, \ldots, N \tag{4}$$

h - шаг сетки по осям x и y, N - количество узлов сетки по осям x и y.

Производные приближаем конечными разностями:

$$\frac{\partial^2 U}{\partial x^2} = \frac{U_{i-1,\,j} - 2U_{i,\,j} + U_{i+1,\,j}}{h^2} \tag{5}$$

$$\frac{\partial^2 U}{\partial y^2} = \frac{U_{i-1,\,j} - 2U_{i,\,j} + U_{i+1,\,j}}{h^2} \tag{6}$$

Подставляя (5) и (6) в уравнение (1) получаем:

$$U_{i-1,j} - 4U_{i,j} + U_{i+1,j} + U_{i,j-1} + U_{i,j+1}$$

Откуда:

$$U_{i,j} = \frac{U_{i-1,j} + U_{i+1,j} + U_{i,j-1} + U_{i,j+1}}{4}$$

Обозначим:

$$\tilde{U}_{i,j} = \frac{U_{i-1,j} + U_{i+1,j} + U_{i,j-1} + U_{i,j+1}}{4}$$

Вычисления будем производить по формуле:

$$U_{i,j}^{k+1} = U_{i,j}^{k} + \omega(\tilde{U}_{i,j}^{k} - U_{i,j}^{k}) \tag{7}$$

Параметр ω в формуле (7) может изменяться в пределах $[0.5, 1.9)$.

Производим вычисления пока не выполнено условие:

$$max|U_{i,j}^{k+1} - U_{i,j}^{k}| \leq \epsilon$$

3 Описание результатов

Определим наилучший параметр по зависимости $k(\omega)$, где k - количество шагов:

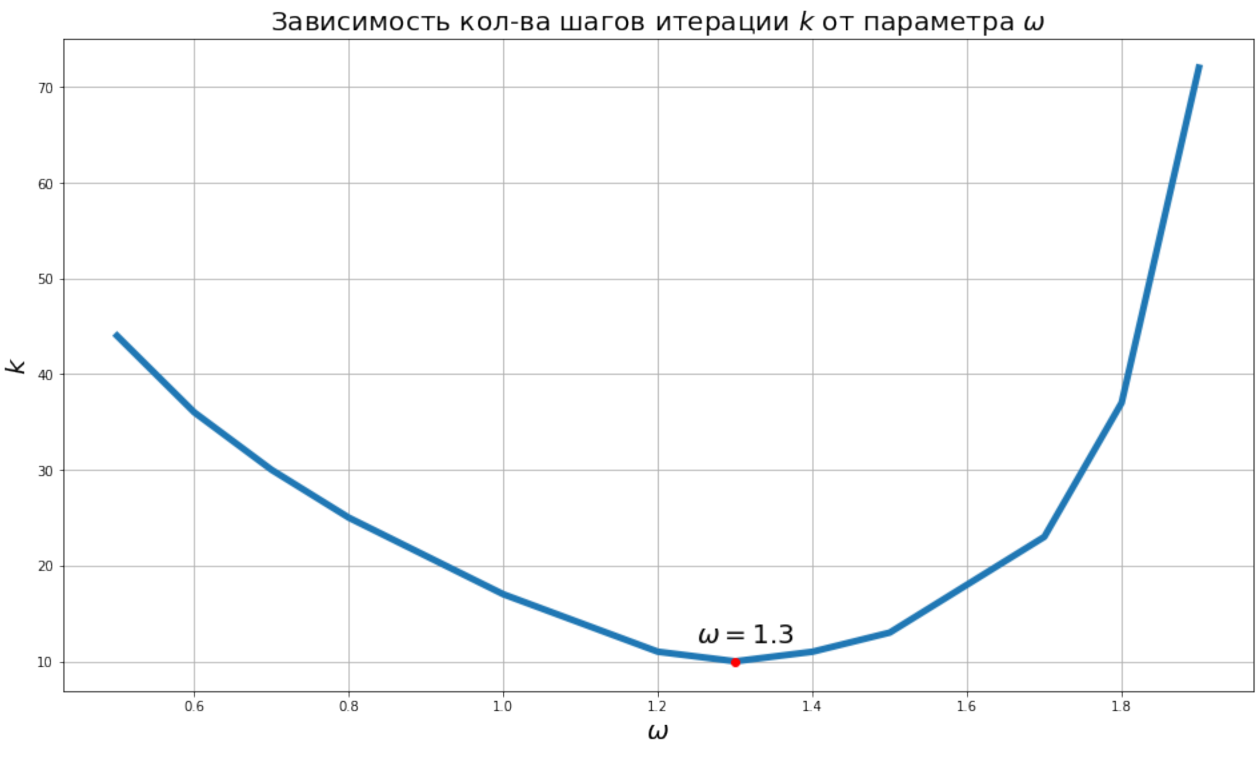

Рис. 1:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	
0	0.5	0.6	0.7	0.8	0.9	1.0	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9	Omega
1	44.0	36.0	30.0	25.0	21.0	17.0	14.0	11.0	10.0	11.0	13.0	18.0	23.0	37.0	72.0	K

Рис. 2: Таблица с результатами ω и k

Решение при оптимальном параметре $\omega = 1.3$:

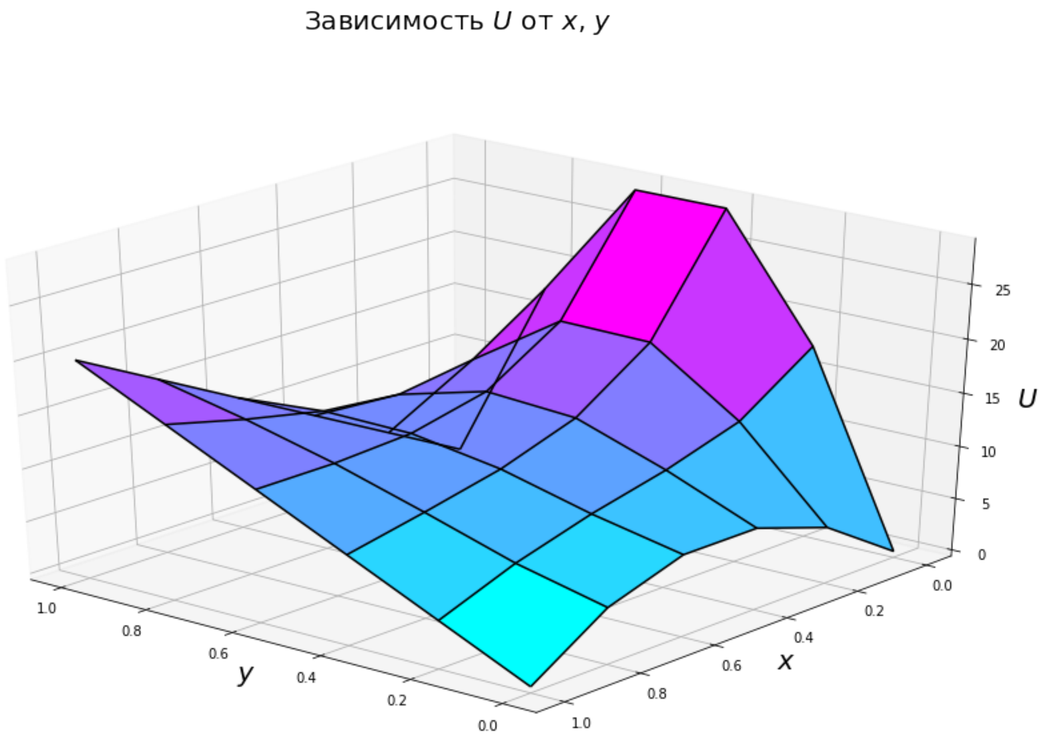

Рис. 3:

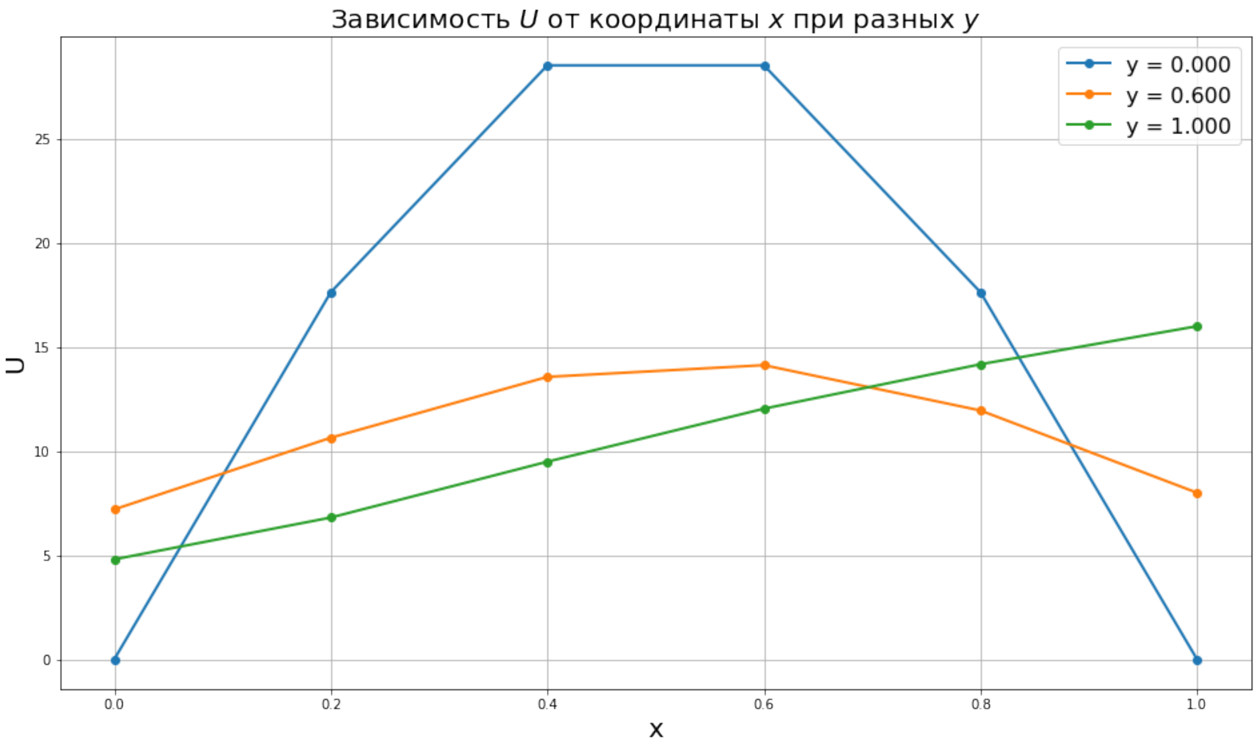
Зависимость U от координаты x при разных y
y = 0.000
y = 0.600
y = 1.000
U
x
0
5
10
15
20
25
0.0
0.2
0.4
0.6
0.8
1.0

Рис. 4:

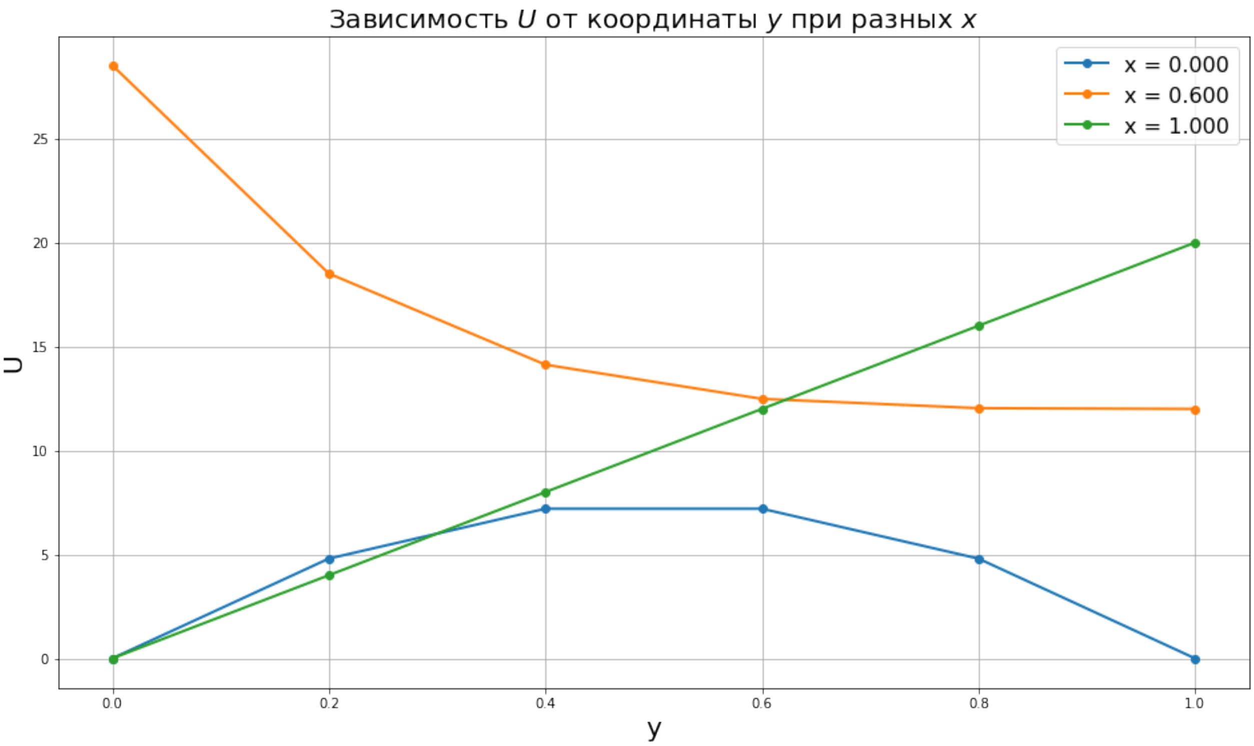

Рис. 5:

Y	0	1	2	3	4	5 X
0	0.0	17.633558	28.531695	28.531695	17.633558	0.0
1	4.8	12.861877	18.368565	18.513517	13.024606	4.0
2	7.2	10.643138	13.565357	14.128627	11.951261	8.0
3	7.2	8.943411	11.121215	12.484788	12.652115	12.0
4	4.8	6.808824	9.491758	12.037243	14.172283	16.0
5	0.0	4.000000	8.000000	12.000000	16.000000	20.0

Рис. 6: Таблица с решением

4 Приложение

Далее представлен код программы на Python:

Листинг 1: Insert code directly in your document

```
import math
import numpy
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FormatStrFormatter
import numpy as np
import pandas as pd
import warnings
warnings.filterwarnings("ignore")
from copy import deepcopy


def diff(A, B):
    max = 0
    for i in range(1, N-1):
        for j in range(1, N-1):
            if abs(A[i][j] - B[i][j]) > max:
                max = abs(A[i][j] - B[i][j])

    return max


def start():

    U = [[0 for i in range(N)] for k in range(N)]
    for i in range(N):
        U[0][i] = round(20 * y_j(i), 3)
    for i in U:
        i[-1] = 20
    for i in range(N):
        U[-1][i] = round(20 * y_j(i) ** 2, 3)
    for i in range(N):
        U[i][0] = round(50 * x_i(i) * (1 - x_i(i)), 3)

    return U

h = 0.2
l = 1
N = round(l / h) + 1
w = 1
e = 0.01

x_i = lambda i: i * h
y_j = lambda j: j * h

W = []
K = []
for w in np.arange(0.5, 2, 0.1):
    W.append(w)
    U = start()
    U_k = deepcopy(U)
```

```
    for i in range(1, N - 1):
        for j in range(1, N - 1):
            U[i][j] = U[i][j] + w * ( 0.25 * (U[i-1][j] +
             U[i+1][j] + U[i][j-1] + U[i][j+1]) - U[i][j] )
    k = 1
    while diff(U, U_k) > e:
        k += 1
        U_k = deepcopy(U)
        for i in range(1, N - 1):
            for j in range(1, N - 1):
                U[i][j] = U[i][j] + w * ( 0.25 * (U[i-1][j] +
                 U[i+1][j] + U[i][j-1] + U[i][j+1]) - U[i][j] )

    K.append(k)

U = start()
U_k = deepcopy(U)
w = 1.3
for i in range(1, N - 1):
    for j in range(1, N - 1):
        U[i][j] = U[i][j] + w * ( 0.25 * (U[i-1][j] +
         U[i+1][j] + U[i][j-1] + U[i][j+1]) - U[i][j] )
k = 1
while diff(U, U_k) > e:
    k += 1
    U_k = deepcopy(U)
    for i in range(1, N - 1):
        for j in range(1, N - 1):
            U[i][j] = U[i][j] + w * ( 0.25 * (U[i-1][j] +
             U[i+1][j] + U[i][j-1] + U[i][j+1]) - U[i][j] )

plt.figure(figsize=(16, 9))
plt.plot(W, K, linewidth=5)
plt.plot([1.3], [10], 'ro')
plt.text(1.25, 12, '$\omega=1.3$', fontsize=20)
plt.title('Title', fontsize=20)
plt.xlabel('$\omega$', fontsize=20)
plt.ylabel('$k$', fontsize=20)
plt.grid()

fig = plt.figure(figsize=(16, 9))
ax = fig.gca(projection='3d')

# Make data.
X = np.arange(0, l + h/10, h)
X, Y = np.meshgrid(X, X)
U = np.array(U)

# Plot the surface.
ax.plot_surface(X, Y, np.array((U)), cmap='cool')
ax.plot_wireframe(X, Y, np.array((U)), color='black')

ax.set_xlabel("y", fontsize=20)
ax.set_ylabel("x", fontsize=20)
ax.set_zlabel("U", fontsize=20)
ax.view_init(30, 230)

fig.suptitle('title', fontsize=20)
```