Московский авиационный институт
(национальный исследовательский университет)
Институт № 8 «Информационные технологии и прикладная математика»

# Лабораторная работа № 7

## По курсу «Численные методы»

## «Численное решение уравнений с частными производными эллиптического типа»

**Выполнил:**
Гришин П. Ф.
**Группа:**
М8О-401Б-21
**Преподаватель:**
Ревизников Д. Л.

Москва, 2025

**Задача**

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).

**Описание метода**

Классический пример уравнения эллиптического типа – это уравнение Пуассона

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}=f(x,y);$$

Или уравнение Лапласа при нулевой правой части.

Функция u в этом уравнении может иметь различный физический смысл: стационарное и независящее от времени распределение температуры, скорость потенциального течения идеальной жидкости, распределение напряженностей электрического и магнитного поля, потенциала в силовом поле тяготения и т.д.

Если на границе расчетной области задана искомая функция, то соответствующая первая краевая задача – задача Дирихле:

$$\begin{cases}\dfrac{\partial^2 u}{\partial x^2}+\dfrac{\partial^2 u}{\partial y^2}=f(x,y), & (x,y)\in\Omega;\\ u(x,y)\big|_{\Gamma}=\varphi(x,y), & (x,y)\in\Gamma.\end{cases}$$

Если же на границе задана производная искомой функции, то соответствующая вторая краевая задача называется задачей Неймана:

$$\begin{cases}\dfrac{\partial^2 u}{\partial x^2}+\dfrac{\partial^2 u}{\partial y^2}=f(x,y), & (x,y)\in\Omega;\\ \left.\dfrac{\partial u(x,y)}{\partial n}\right|_{\Gamma}=\varphi(x,y), & (x,y)\in\Gamma.\end{cases}$$

Где n – направление внешней к границе нормали.

Третья же краевая задача для уравнения Пуассона или Лапласа имеет вид

$$\begin{cases}\dfrac{\partial^2 u}{\partial x^2}+\dfrac{\partial^2 u}{\partial y^2}=f(x,y), & (x,y)\in\Omega;\\ \left.\dfrac{\partial u(x,y)}{\partial n}\right|_{\Gamma}+\alpha\, u\big|_{\Gamma}=\varphi(x,y), & (x,y)\in\Gamma.\end{cases}$$

Рассмотрим краевую задачу для уравнения Лапласа или Пуассона в прямоугольнике $x\in[0,l_1],y\in[0,l_2]$ с наложенной сеткой

$$\omega_{h_1,h_2}=\{x_i=ih_1, i=\overline{0,N_1};\quad y_j=jh_2, j=\overline{0,N_2}\}.$$

На сетке аппроксимируем дифференциальную задачу во внутренних узлах с помощью отношения конечных разностей по схеме, имеющей второй порядок по переменным x и y:

$$\frac{u_{i+1,j}-2u_{i,j}+u_{i-1,j}}{h_1^2}+\frac{u_{i,j+1}-2u_{i,j}+u_{i,j-1}}{h_2^2}+O\left(h_1^2+h_2^2\right)=f(x_i,y_j),$$

$$i=\overline{1,N_1-1},\quad j=\overline{1,N_2-1}$$

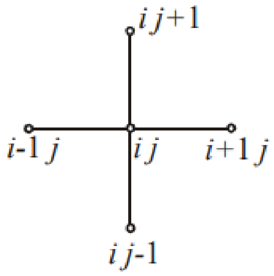
Получаемая СЛАУ имеет пяти-диагональный вид и решается с помощью методов линейной алгебры.

Рассмотрим разностный метод Либмана численного решения задачи Дирихле. Положим $h_1 = h_1 = h$, тогда на k-ой итерации получим:

$$u_{i,j}^{(k+1)} = \frac{1}{4}[u_{i+1,j}^{(k)} + u_{i-1,j}^{(k)} + u_{i,j-1}^{(k)} + u_{i,j+1}^{(k)} - h^2 \cdot f_{i,j}], \quad f_{i,j} = f(x_i, y_j),$$

$$i = \overline{1, N_1 - 1}, \quad j = \overline{1, N_2 - 1}.$$

Укажем схему:




Видно, что для использования полученной формулы необходимо найти значения $u_{i,j}^{(0)}$ на нулевой итерации. Для этого на каждой координатной линии используем линейные интерполяции граничных значений. Полученные значения подставляем в формулу и получаем распределение $u_{i,j}^{(1)}$. Это распределение опять же подставляется в формулу, после чего получаем распределение $u_{i,j}^{(2)}$ и тд.

Процесс Либмана прекращается при условии:

$$\left\| u^{(k+1)} - u^{(k)} \right\| \le \varepsilon, \quad \left\| u^{(k)} \right\| = \max_{i,j} \left| u_{i,j}^{(k)} \right|$$

Далее рассмотрим метод Зейделя, который является улучшением метода Либмана. Различие методов в том, что в методе Зейделя в формулу подставляются не только значения с предыдущей итерации, но и уже полученные значения с текущей итерации. Таким образом мы получаем формулу:

$$u_{i,j}^{(k+1)} = \frac{1}{4}[u_{i+1,j}^{(k)} + u_{i-1,j}^{(k+1)} + u_{i,j+1}^{(k)} + u_{i,j-1}^{(k+1)} - h^2 f_{i,j}]$$

Стоит отметить, что скорость сходимости метода Зейделя выше, чем у метода Либмана. Метод верхней релаксации отличается наличием свободного параметра c, который определяет величину смещения каждой компоненты в зависимости от ее положения на предыдущем шаге:

$$u_{i,j}^{(k+1)} = (1 - \mathrm{c})u_{i,j}^{(k)} + \mathrm{c}\frac{1}{4}[u_{i+1,j}^{(k)} + u_{i-1,j}^{(k+1)} + u_{i,j+1}^{(k)} + u_{i,j-1}^{(k+1)} - h^2 f_{i,j}]$$

Скорость сходимости метода верхней релаксации зависит от выбора параметра c, и при верном выборе она может быть значительно быстрее скоростей сходимости остальных методов.

**Вариант**

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

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

$$u(\frac{\pi}{2}, y) = 0,$$

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

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

Аналитическое решение: $U(x, y) = \cos x \cos y$.

## Решение и код

```
import numpy as np
import matplotlib.pyplot as plt
def phi1(y):
        return np.cos(y)
       def phi2(y):
            return 0
       def phi3(x):
        return np.cos(x)
       def phi4(x):
            return 0
def solution(x, y):
        return np.cos(x)*np.cos(y)
def norm(u):
        return np.abs(u).max()

#//////////////////////////////////////////////////////


l = np.pi / 2
N = 10
```

#аналитическое решение задачи

```
def analitic_solve(l, N, solution):  h = l / (N - 1)
        u = np.zeros((N, N))
        for x in range(N):
                for y in range(N):
                        u[x][y] = solution(x * h, y * h)
        return u
```

#функция заполнения u0

```
def get_u0(N, l): size =
        N - 1 h = l / size
        u = np.zeros((N, N))
        for j in range(N): u[0][j] = phi1(j
                * h)
                u[-1][j] = phi2(j * h)
                u[j][0] = phi3(j * h)
                u[j][-1] = phi4(j * h)
        for i in range(1, N - 1):
                for j in range(1, N - 1):
                        u[i][j] = h * i * u[i][0] + h * j * u[0][j] + (l - h * i) * u[i][
-1] + (l - h * j) * u[-1][j]
        return u
```

#метод Лимбмана

```
def limbman(N, l, eps): size = N -
        1
        h = l / size
        u = get_u0(N, l) k = 0
        while k == 0 or norm(u - u_prev) > eps:
```

```
        u_prev = u.copy()
        for i in range(1, N - 1):
            for j in range(1, N - 1):
                u[i][j] = (u_prev[i+1][j] + u_prev[i-1][j] + u_prev[i][j+1] + u_prev[i][j-1]) / (4 - 2 *
h ** 2)
        k += 1
    return u, k
```

#метод Зейделя

```
def zeidel(N, l, eps): size = N - 1
    h = l / size
    u = get_u0(N, l) k = 0
    while k == 0 or norm(u - u_prev) > eps:
        u_prev = u.copy()
        for i in range(1, N - 1):
            for j in range(1, N - 1):
                u[i][j] = (u_prev[i + 1][j] + u[i - 1][j] + u_prev[i][j + 1]
+ u[i][j - 1]) / (4 - 2 * h ** 2)
        k += 1
    return u, k
```

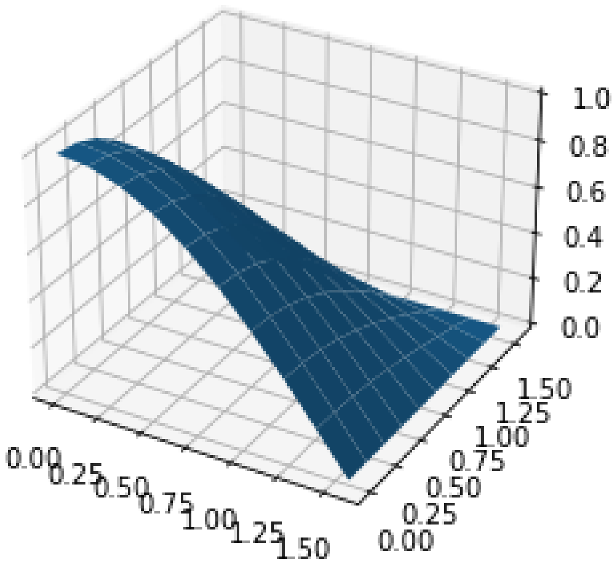
#метод простых итераций с верхней релаксацией

```
def with_relax(N, l, eps, c): size = N - 1
    h = l / size
    u = get_u0(N, l) k = 0
    while k == 0 or norm(u - u_prev) > eps:
        u_prev = u.copy()
        for i in range(1, N - 1):
            for j in range(1, N - 1):
                u[i][j] = (1 - c) * u_prev[i][j] + c * (( + u_prev[i + 1][j]
+ u[i - 1][j] + u_prev[i][j + 1] + u[i][j - 1]) / (4 - 2 * h ** 2)) k += 1
    return u, k
```

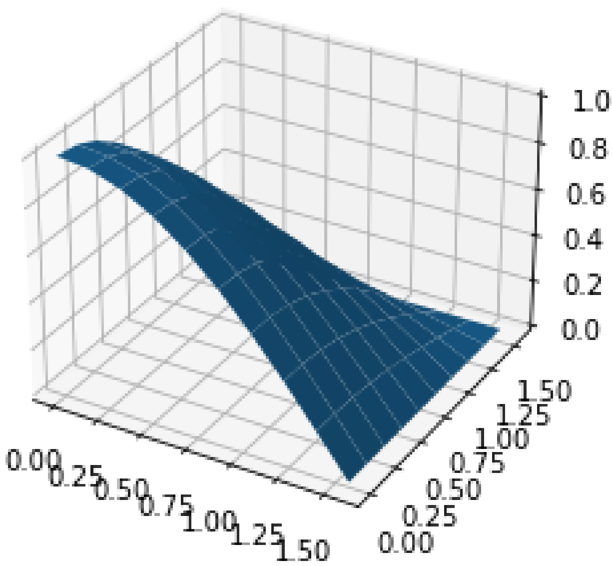
#количество шагов для сходимости каждого метода

```
analitic = analitic_solve(l, N, solution)

limbman_s = limbman(N, l, 0.001)

print(limbman_s[1])
zeidel_s = zeidel(N, l, 0.001)
print(zeidel_s[1])
with_relax_s = with_relax(N, l, 0.001, 1.5)
print(with_relax_s[1])

#////////////////////////////////////////////////////

76
49
20
#////////////////////////////////////////////////////

x = np.linspace(0,l,N) y =
np.linspace(0,l,N)
```

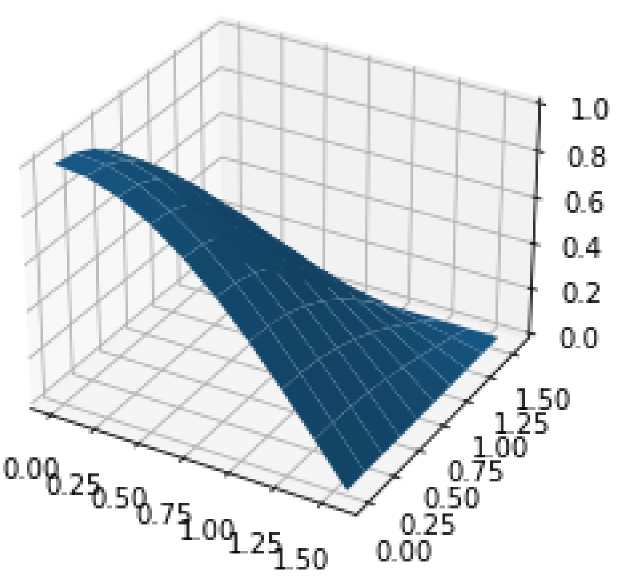
```
x_plt, t_plt = np.meshgrid(x, y) fig =
plt.figure()
ax = plt.axes(projection ='3d')
ax.plot_surface(x_plt,t_plt,np.array(analitic))   plt.show()
#///////////////////////////////////////////////////
```



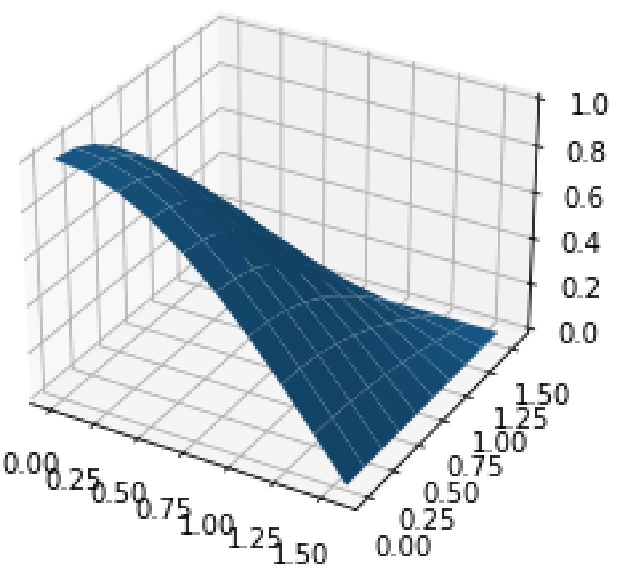

```
#///////////////////////////////////////////////////

fig = plt.figure()
ax = plt.axes(projection ='3d')
ax.plot_surface(x_plt,t_plt,np.array(limbman_s[0]))
plt.show()
#///////////////////////////////////////////////////
```




```
#///////////////////////////////////////////////////
fig = plt.figure()
ax = plt.axes(projection ='3d')
ax.plot_surface(x_plt,t_plt,np.array(zeidel_s[0]))
plt.show()

#///////////////////////////////////////////////////
```




#////////////////////////////////////////////////////////
fig = plt.figure()
ax = plt.axes(projection ='3d')
ax.plot_surface(x_plt,t_plt,np.array(with_relax_s[0]))
plt.show()




#////////////////////////////////////////////////////////
#погрешности каждого метода

def pogr(u, res):
        return np.sqrt(sum([sum([(u[i][j]-res[i][j])**2 for j in range(len(x))])
for i in range(len(y))]))

print(pogr(limbman_s[0], analitic))

print(pogr(zeidel_s[0], analitic))

print(pogr(with_relax_s[0], analitic))

#////////////////////////////////////////////////////////

0.0943016816911137

0.04460112096437301
0.009295647251482082

**Выводы**

В ходе выполнения данной работы для решения краевой задачи для дифференциального уравнения эллиптического типа была произведена аппроксимация с использованием центрально-разностной схемы. Дискретный аналог был решен тремя разными методами: методом простых итераций (Либмана), методом Зейделя и методом простых итераций с верхней релаксацией. Также были вычислены погрешности каждого решения путем сравнения результатов с приведенным аналитическим решением. По результатам можно сделать вывод, что самым точным и быстро сходящимся методом является метод простых итераций с верхней релаксацией, а самым неточным и медленно сходящимся – метод Либмана.