НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Механико-математический факультет

Кафедра: Математика и компьютерные науки

Тлепбергенова Дарья Дулатовна

Отчет по вычислительному практикуму

Решение двумерного эллиптического уравнения методом конечных разностей. Вариант 14.

3 курс, группа 16121

Преподаватель:
Махоткин Олег Александрович

Новосибирск, 2018 г.

1. Постановка задачи.

С помощью 5-точечной схемы свести уравнение задачи Дирихле

$$-a\frac{\partial^2 u}{\partial x^2} - b\frac{\partial^2 u}{\partial y^2} = f(x,y), (x,y) \in D$$

$$u(x,y) = \varphi(x,y), (x,y) \in \partial D$$

в области D к решению системы линейных алгебраических уравнений $Az = F$. Использовать разностную схему с шагами:

$$h_x = h_y = h = \frac{1}{m}$$

$$-a\frac{U_{i-1,j} - 2U_{i,j} + U_{i+1,j}}{h^2} - b\frac{U_{i,j-1} - 2U_{i,j} + U_{i,j+1}}{h^2} = f_{i,j}$$

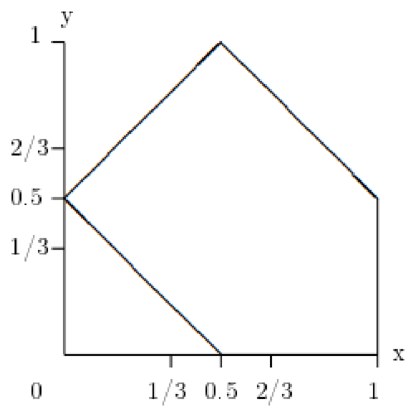
Для

$$a = 1; b = 1$$

$$f(x,y) = 2\pi^2 sin(\pi x) sin(\pi x),$$

$$\varphi(x,y) = sin(\pi x) sin(\pi x)$$

Область D выглядит следующим образом:



1) Найти погрешность аппроксимации разностной схемы на решении краевой задачи.
2) Проверить, что функция $\varphi(x,y)$ является точным решением краевой задачи.

3) Для небольшого числа узлов сетки и двух вариантов нумерации внутренних узлов выписать в симметрическом виде матрицы A. Убедиться, что они являются симметричными и разреженными.

4) Записать полученные матрицы в упакованном виде (по строкам).

5) Найти максимальное и минимальное собственное значения матрицы A для нескольких значений m степенным методом.

6) Используя собственные значения оператора $(Lv)_i = v_{i-1} - 2v_i + v_{i+1}, i = 1,...,m-1, v_0 = v_m = 0$, найти собственные значения двумерного разностного оператора рассматриваемой задачи для $D = Q^2 = [0,1]^2$. Сравнивать полученное значение $\lambda_{min}(h)$ с λ_{min} для дифференциальной задачи. Получить асимптотическое разложение $\lambda_{min}(h)$ по h.

7) Найти решение системы уравнений $Az = F$ методом установления и методом верхней релаксации.

8) Для нескольких значений числа интервалов m найти относительные погрешности разностного решения $\Delta =< |U(x,y) - \varphi(x,y)| > / < |\varphi(x,y)| >$.

   Здесь $< |g(x,y)| >= \sum_{i,j} [(i,j) =\in D] |g(x_i, y_j)|$.

9) Вывести на экран разностное решение $U(x,y)$ и погрешность $U(x,y) - \varphi(x,y)$.

2. Исследование данной схемы на точность и устойчивость.

2.1. Погрешность аппроксимации.

Оценим погрешность аппроксимации данной задачи, для этого, Приблизим оператор Лапласа разностным оператором:

$$\Lambda u = \Lambda_1 u + \Lambda_2 u = u_{x,x} + u_{y,y}$$

Тогда с помощью разложения в ряд Тейлора получаем:

$$\Lambda_1 u = \frac{\partial^2 u}{\partial x^2} + \frac{h^2}{12}\frac{\partial^4 u}{\partial x^4} + O(h^4)$$

$$\Lambda_2 u = \frac{\partial^2 u}{\partial y^2} + \frac{h^2}{12}\frac{\partial^4 u}{\partial y^4} + O(h^4)$$

Тогда

$$\Lambda u - \Delta v = \frac{h^2}{12}\frac{\partial^4 u}{\partial x^4} + \frac{h^2}{12}\frac{\partial^4 u}{\partial y^4} + O(h^4)$$

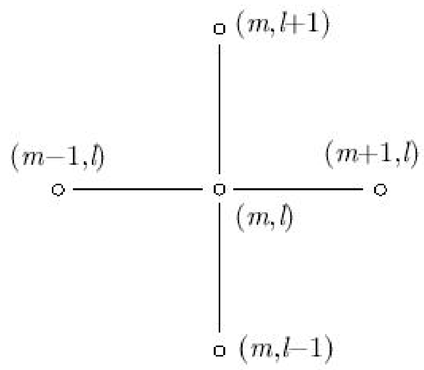
Отсюда следует, что

$$\Lambda u - \Delta v = O(h^2)$$

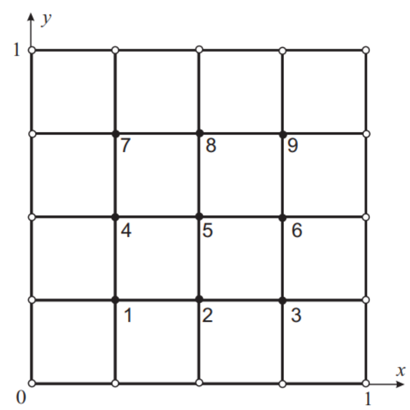
Таким образом, данный разностный оператор аппроксимирует оператор Лапласа со вторым порядком аппроксимации на регулярном шаблоне "крест".

3. Описание вычислительного метода. Алгоритм решения программы.

- Изобразим шаблон нашей схемы:



- Наложим на нашу область равномерную сетку с шагом h и пронумеруем узлы сетки следующим образом:



- С помощью отдельной функции, отделим точки, принадлежащие границе области D, точки, лежащие строго внутри области D и точки не принадлежащие области.
- Для граничных точек сразу вычисляем значения искомой функции
- Для всех внутренних точек выполняем разностную схему и ищем их значения, как решение СЛАУ.На основе этого получаем матрицу.
- Записываем эту матрицу в упакованном виде
- Ищем у полученной матрицы max и min числа
- Решаем матрицу (СЛАУ) с помощью метода верхней релаксации и методом установления
- Заносим оставшиеся значения в матрицу решений и выводим полученный график

4. Код программы

[1]:
```
#%matplotlib inline
%matplotlib notebook
```

[2]:
```
import math
import numpy as np
import scipy.linalg as sla
import matplotlib.pyplot as plt
from scipy.sparse import csc_matrix
from scipy.sparse import csr_matrix
from scipy import sparse
import scipy
import scipy.sparse.linalg as linalg

from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FormatStrFormatter
```

[3]:
```
a = 1
b = 1
m = 4
h = 1 / m
eps = 1e-9
number = np.zeros((m + 1, m + 1), dtype = int)
```

[4]:
```
def check_in(x, y): #function of checking
        return (((-x + 0.5) < y) and ((x + 0.5) > y) and
                ((-x + 1.5) > y) and (x < 1) and (y > 0))
```

[5]:
```
def check_border(x,y):
    return (x >= 0) and (y >= 0) and (x <= 1) and (y <= 1) and (((-x +␣
 ↪0.5) == y) or ((x + 0.5) == y) or ((-x + 1.5) == y) or ((x == 1) and␣
 ↪(y <= 0.5)) or ((y == 0) and (x >= 0.5)));
```

[6]:
```
def f(x, y):
    return 2.0 * math.pi * math.pi * math.sin(math.pi * x) * math.
 ↪sin(math.pi * y)

def phi(x, y):
    return math.sin(math.pi * x) * math.sin(math.pi * y)
```

[7]:
```
cnt = 0
for i in range(1, m):
    for j in range(1, m):
        if check_in(i / m, j / m):
            number[i][j] = cnt
```

```
            cnt += 1
```

```
[8]: x = np.zeros((m + 1, m + 1))
y = np.zeros((m + 1, m + 1))
indicate = np.zeros((m + 1,m + 1),dtype = int)
x1 = 0
y1 = 0
k = 0

for i in range(1, m + 1):
    for j in range(1, m + 1):
        x[i][j] = i / m
        y[i][j] = j / m

for i in range(0, m + 1):
    x1=0
    for j in range(0, m + 1):
        if (check_in(x1,y1)):
            indicate[i][j] = k
            k=k+1
        else :
            if (check_border(x1, y1)):
                indicate[i][j] = -1
            else :
                indicate[i][j] = -2
        x1 = x1 + h
    y1 = y1 + h
```

```
[9]: A = np.zeros((cnt, cnt), dtype = float)
z = np.zeros(cnt)
```

```
[10]: k = 0
y1 = h
for i in range(1, m):
    x1 = h
    for j in range(1, m):
        if (indicate[i][j] >= 0) :
            A[k][k] = 2 * (a + b)/(h*h)
            z[k] =  f(x1, y1)
            if (indicate[i - 1][j] >= 0) :
                A[k][indicate[i - 1][j]] = -b/(h*h)
            else :
                if (indicate[i - 1][j] == -1) :
                    z[k] += b * phi(x[i - 1][j],y[i-1][j])/(h*h)
                else :
                    print("error_1 ")
```

```
            if (indicate[i][j - 1] >= 0) :
                A[k][k - 1] = -a/(h*h)
            else :
                if (indicate[i][j - 1] == -1) :
                    z[k] += a * phi(x[i][j - 1],y[i][j-1])/(h*h)
                else :
                    print("error_2 ")

            if (indicate[i + 1][j] >= 0) :
                A[k][indicate[i + 1][j]] = -b/(h*h)
            else :
                if (indicate[i + 1][j] == -1) :
                    z[k] += b * phi(x[i + 1][j],y[i+1][j])/(h*h)
                else :
                    print("error_3 ")

            if (indicate[i][j + 1] >= 0) :
                A[k][k + 1] = -a/(h*h)
            else :
                if (indicate[i][j + 1] == -1) :
                    z[k] += phi(x[i][j + 1],y[i][j+1])/(h*h)
                else :
                    print("error_4 ")

            k = k + 1
        x1 = x1 + h
    y1 = y1 + h
```

```
[11]: def find_e(A): # Function to find eigs
    size = A.shape[0]
    y = np.array([1 for i in range(size)])
    x = y / np.linalg.norm(y)
    eps = 10e-6
    lam1 = 4
    lam2 = 1

    while abs(lam2 - lam1) > eps:
        y = A.dot(x)
        lam1 = lam2
        lam2 = y.T.dot(x)
        x = y / np.linalg.norm(y)

    eigs = np.linalg.eig(A)[0]

    eigs = np.sort(eigs)
    print("Eigs: ", eigs[0], " ", eigs[size - 1])
```

[12]:
```
print(A)
find_e(A)
A = csr_matrix(A)
#print(A)
#max = scipy.sparse.linalg.eigsh(A,k=5, sigma=0.5)[0]
#print(max)
print(indicate)
```

```
[[ 64. -16.   0. -16.   0.   0.]
 [-16.  64.   0.   0. -16.   0.]
 [  0.   0.  64. -16.   0.   0.]
 [-16.   0. -16.  64. -16. -16.]
 [  0. -16.   0. -16.  64.   0.]
 [  0.   0.   0. -16.   0.  64.]]
Eigs:  27.38807021966818   100.61192978033175
[[-2 -2 -1 -1 -1]
 [-2 -1  0  1 -1]
 [-1  2  3  4 -1]
 [-2 -1  5 -1 -2]
 [-2 -2 -1 -2 -2]]
```

[13]:
```
u_my = linalg.spsolve(A, z)
```

[14]:
```
u = np.zeros((m + 1, m + 1), dtype='float64')
for i in range(m + 1):
    for j in range(m + 1):
        if (indicate[i][j] >= 0):
            u[i][j] = u_my[indicate[i][j]]
        else :
            if (indicate[i][j] == -1):
                u[i][j] = phi(x[i][j], y[i][j])
            else :
                u[i][j] = 0
```

[15]:
```
u_real = np.zeros((m + 1, m + 1), dtype='float64')
for i in range(m + 1):
    for j in range(m + 1):
        if (indicate[i][j] >= -1):
            u_real[i][j] = phi(x[i][j], y[i][j])
        else :
            u_real[i][j] = 0
```

[16]:
```
print(sla.norm(u - u_real))
```

```
0.06423848246925326
```

[17]:
```
X, Y = np.meshgrid(np.linspace(0, 1, m  + 1), np.linspace(0, 1, m  + 1))
```

```
fig = plt.figure()
ax = fig.gca(projection='3d')
#surf = ax.plot_surface(X, Y, u, cmap=cm.coolwarm,
 #                       linewidth=0, antialiased=False)

surf_ = ax.plot_surface(X, Y, u_real, cmap=cm.plasma,
                        linewidth=0, antialiased=False)

ax.set_xlabel('t')
ax.set_ylabel('x')
ax.set_zlabel('u')

#fig.colorbar(surf, shrink=0.5, aspect=5)
fig.colorbar(surf_, shrink=0.5, aspect=5)

plt.show()
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

[18]:

```
X, Y = np.meshgrid(np.linspace(0, 1, m  + 1), np.linspace(0, 1, m  + 1))

print(X.shape, Y.shape)

fig = plt.figure()
ax = fig.gca(projection='3d')
surf = ax.plot_surface(X, Y, u - u_real, cmap=cm.coolwarm,
                       linewidth=0, antialiased=False)


ax.set_xlabel('t')
ax.set_ylabel('x')
ax.set_zlabel('u')

fig.colorbar(surf, shrink=0.5, aspect=5)

plt.show()
```

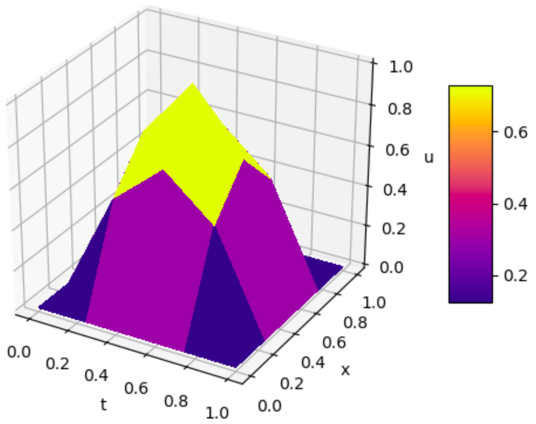
```
(5, 5) (5, 5)
```

```
<IPython.core.display.Javascript object>
```

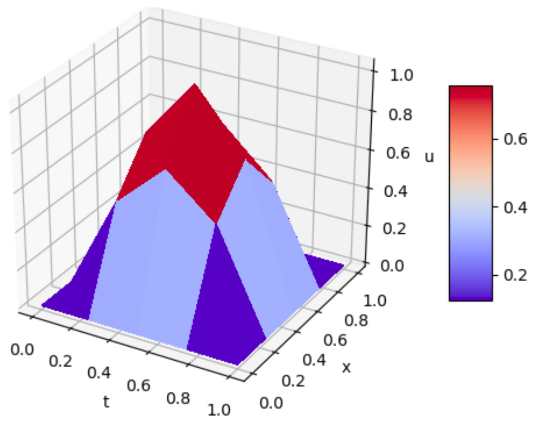
```
<IPython.core.display.HTML object>
```

5. Графический вывод (Тесты)

При $h = 0.25$
Функция решения и действительная функция:



а)

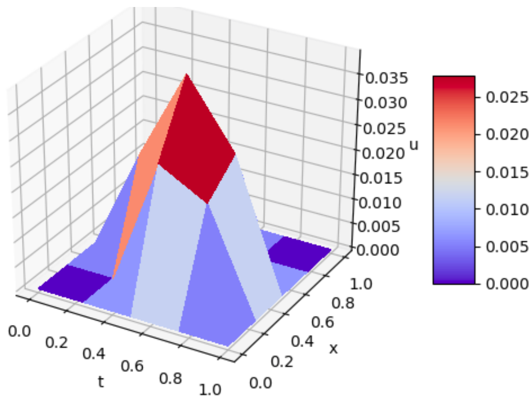


б)

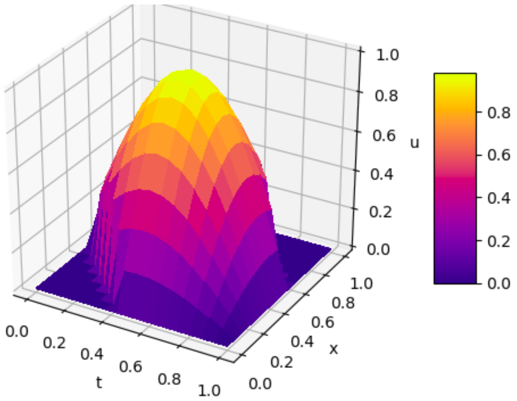
Значения минимального и максимального собственных чисел:

$$Eigs : 27.38807021966818 \;\; 100.61192978033175$$

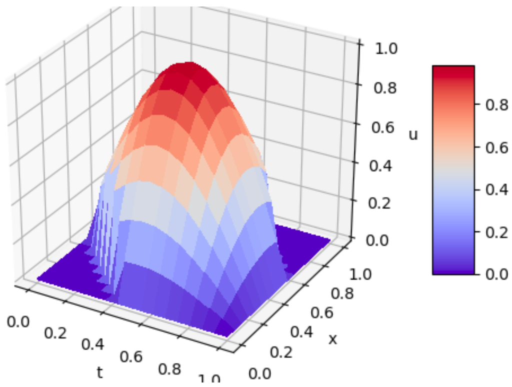
Ошибка выглядит следующим образом:



При $h = 0.625$:
Функция решения и действительная функция:



а)

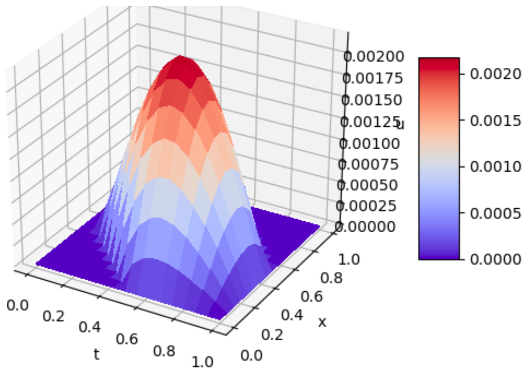


б)

Значения минимального и максимального собственных чисел:

$$Eigs : 31.841084906917786 \;\; 2016.1589150930836$$

Ошибка выглядит следующим образом:



Заметим, что во втором тесте мы увеличили h в 4 раза и наша ошибка уменьшилась в 17,5 раз, даже лучше чем должна была (теоретически должна была уменьшиться в 16 раз).

6. Выводы.

Таким образом мы исследовали задачу Дирихле в ограниченной области и изучили ход ее решения. Убедились в том, что при увеличении шага h, погрешность решения уменьшается примерно в h^2 раз.

Этот метод не очень прост в реализации: в нем участвуют сразу несколько дополнительных методов для поиска собственных чисел и решения системы линейных уравнений. Мы научились работать с упаковочными массивами, графически изображать решение и итерационно решать дифференциальные уравнения.

Таким образом,данный метод можно разными способами модифицировать, что также является плюсом метода.