Московский авиационный институт

(национальный исследовательский институт)

Лабораторная работа

По курсу

Численные методы

Выполнил: Смирнов Д.А.

Группа : М8О-403Б-18

Москва 2021

Лабораторная №3

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. Исследовать зависимость погрешности от сеточных параметров.

Вариант 8

$$\begin{cases} \dfrac{\partial^2 u}{\partial x^2}+\dfrac{\partial^2 u}{\partial y^2}=-2\dfrac{\partial u}{\partial x}-3u \\ u(0,y)=\cos y \\ u\left(\dfrac{\pi}{2},y\right)=0 \\ u(x,0)=e^{-x}cosx \\ u\left(x,\dfrac{\pi}{2}\right)=0 \end{cases}$$

Аналитическое решение:

$u(x,y)=e^{-x}\cos(x)\cos(y)$.

Теоретический материал

Рассмотрим уравнение Пуассона, которое является классическим примером эллиптического уравнения

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}=f(x,y),(1)$$

Или уравнение Лапласа при $f(x,y)=0$

Первая краевая задача для уравнения Лапласа или Пуассона называется задачей Дирихле

$$\begin{cases} \dfrac{\partial^2 u}{\partial x^2}+\dfrac{\partial^2 u}{\partial y^2}=f(x,y),(x,y)\epsilon\Omega; & (2)\\ u(x,y)|_1=\varphi(x,y),(x,y)\epsilon\Gamma; & (3) \end{cases}$$

Если на границе Г задается нормальная производная искомой функции, то соответствующая вторая краевая задача называется задачей Неймана для уравнения Лапласа или Пуассона

$$\begin{cases} \dfrac{\partial^2 u}{\partial x^2}+\dfrac{\partial^2 u}{\partial y^2}=f(x,y),(x,y)\epsilon\Omega; & (4)\\ \dfrac{\partial u(x,y)}{\partial n}|_\Gamma=\varphi(x,y),(x,y)\epsilon\Gamma; & (5) \end{cases}$$

При этом n – направление внешней к границе Г нормали.

Третья краевая задача для уравнения Пуассона (Лапласа) имеет вид:

$$\begin{cases} \dfrac{\partial^2 u}{\partial x^2} + \dfrac{\partial^2 u}{\partial y^2} = f(x,y), (x,y)\epsilon\Omega; & (4) \\ \dfrac{\partial u(x,y)}{\partial n}|_{\Gamma} = \varphi(x,y), (x,y)\epsilon\Gamma; & (5) \end{cases}$$

Разностные схемы для аппроксимации

Для решения – строим сетку по x и y. И на ней аппроксимируем задачу во внутренних узлах с помощью отношения конечных разностей по следующей схеме:

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

$i=1,\ldots,N_1-1, j=1,\ldots,N_2-1$

В результате получаем СЛАУ, которую можно решить разными итерационными методами.

Метод Либмана:

$$\frac{u_{i-1j}^{k} - 2u_{ij}^{k+1} + u_{i+1j}^{k}}{h_x^2} + \frac{u_{ij-1}^{k} - 2u_{ij}^{k+1} + u_{ij+1}^{k}}{h_y^2} = -2\frac{u_{i+1j}^{k} - u_{i-1j}^{k}}{2h_x} - 3u_{ij}^{k}$$

$$u_{ij}^{k+1} = \frac{\left(u_{i-1j}^{k} + u_{i+1j}^{k}\right)h_y^2 + \left(u_{ij-1}^{k} + u_{ij+1}^{k}\right)h_x^2 + \left(u_{i+1j}^{k} - u_{i-1j}^{k}\right)h_x h_y^2 + 3u_{ij}^{k}}{2\left(h_y^2 + h_x^2\right)}$$

Метод Зейделя:

$$\frac{u_{i-1j}^{k+1} - 2u_{ij}^{k+1} + u_{i+1j}^{k}}{h_x^2} + \frac{u_{ij-1}^{k+1} - 2u_{ij}^{k+1} + u_{ij+1}^{k}}{h_y^2} = -2\frac{u_{i+1j}^{k+1} - u_{i-1j}^{k+1}}{2h_x} - 3u_{ij}^{(k)}$$

$$u_{ij}^{k+1} = \frac{\left(u_{i-1j}^{k+1} + u_{i+1j}^{k}\right)h_y^2 + \left(u_{ij-1}^{k+1} + u_{ij+1}^{k}\right)h_x^2 + \left(u_{i+1j}^{k+1} - u_{i-1j}^{k+1}\right)h_y^2 h_x + 3u_{ij}^{k}}{2\left(h_x^2 + h_y^2\right)}$$

Программа

Определяем начально-краевые условия

```
# левая граница по y
def Uleft(y):
    return np.cos(y)

# правая граница по y
def Uright(y):
    return 0

# левая граница по x
def Ubottom(x):
    return np.exp(-x)*np.cos(x)

# правая граница по x
def Uupper(x):
    return 0

# абслотное решение
def Solution(x, y):
    return np.exp(-x)*np.cos(x)*np.cos(y)
```

Метод Либмана:

```
def Libman(x, y, hx, hy, epsilon):
    len_x = len(x)
    len_y = len(y)
    prevU = np.zeros([len_y,len_x])
    curU = np.zeros([len_y,len_x])
         # интерполяция
    for i in range(0,len_x):
        coeff = (Uupper(x[i]) - Ubottom(x[i]))/(len_y-1)
        addition = Ubottom(x[i])
        for j in range(0,len_y):
            curU[j][i] = coeff*j + addition

    while(True):
        prevU = copy.deepcopy(curU)
        for j in range(1,len_y-1):
            curU[j][0] = Uleft(y[j])
            for i in range(1,len_x-1):
                coeff_left = (2/hx**2 + 2/hy**2 - 3 + 2/hx)
                Rpart = (prevU[j][i-1]+ prevU[j][i+1])/hx**2 + (prevU[j+1][i]+prevU[j-1][i])/hy**2 +
2*prevU[j][i-1]/hx
                curU[j][i] = Rpart/coeff_left


            curU[j][-1] = Uright(y[j])
        norm = maxNorm(curU, prevU, len_x, len_y)


        if (norm<=epsilon):
            break
    return curU
```

Метод Зейделя:

```
def Seidel(x, y, hx, hy, epsilon):
    len_x = len(x)
    len_y = len(y)
    prevU = np.zeros([len_y,len_x])
    curU = np.zeros([len_y,len_x])
         # интерполяция
    for i in range(0,len_x):
```

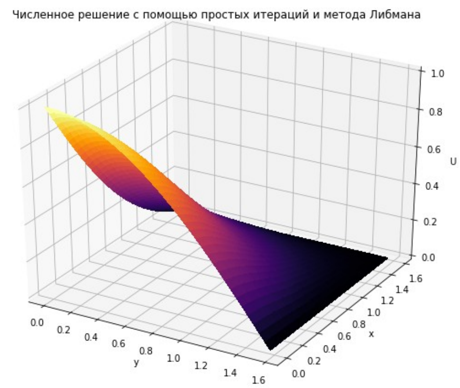
```
        coeff = (Uupper(x[i]) - Ubottom(x[i]))/(len_y-1)
        addition = Ubottom(x[i])
        for j in range(0,len_y):
            curU[j][i] = coeff*j + addition

    while(True):
        prevU = copy.deepcopy(curU)
        for j in range(1,len_y-1):
            curU[j][0] = Uleft(y[j])
            for i in range(1,len_x-1):
                coeff_left = (2/hx**2 + 2/hy**2 - 3 + 2/hx)
                Rpart = (curU[j][i-1]+ prevU[j][i+1])/hx**2 + (prevU[j+1][i]+curU[j-1][i])/hy**2 +
2*curU[j][i-1]/hx
                curU[j][i] = Rpart/coeff_left


            curU[j][-1] = Uright(y[j])
        norm = maxNorm(curU, prevU, len_x, len_y)

        if (norm<=epsilon):
            break
    return curU
```

Метод с верхней релаксацией:

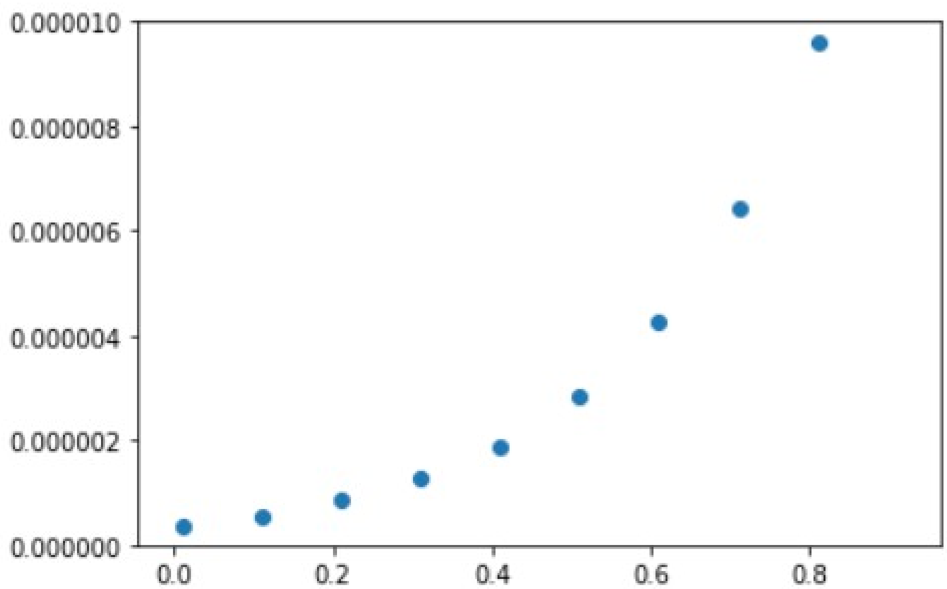
```
def Relaxation(x, y, hx, hy, epsilon):
    len_x = len(x)
    len_y = len(y)
    prevU = np.zeros([len_y,len_x])
    curU = np.zeros([len_y,len_x])
         # интерполяция
    for i in range(0,len_x):
        coeff = (Uupper(x[i]) - Ubottom(x[i]))/(len_y-1)
        addition = Ubottom(x[i])
        for j in range(0,len_y):
            curU[j][i] = coeff*j + addition

    relax_param = 0.6

    while(True):
        prevU = copy.deepcopy(curU)
        for j in range(1,len_y-1):
            curU[j][0] = Uleft(y[j])
            for i in range(1,len_x-1):
                coeff_left = (2/hx**2 + 2/hy**2 - 3 + 2/hx)
                Rpart = (prevU[j][i-1]+ prevU[j][i+1])/hx**2 + (prevU[j+1][i]+prevU[j-1]
[i])/hy**2 + 2*prevU[j][i-1]/hx
                M = Rpart/coeff_left

                curU[j][i] = (1 - relax_param) * prevU[j][i] + relax_param * M
            curU[j][-1] = Uright(y[j])
        norm = maxNorm(curU, prevU, len_x, len_y)

        if (norm<=epsilon):
            break
    return curU
```

Результат работы:

Метод Либмана:



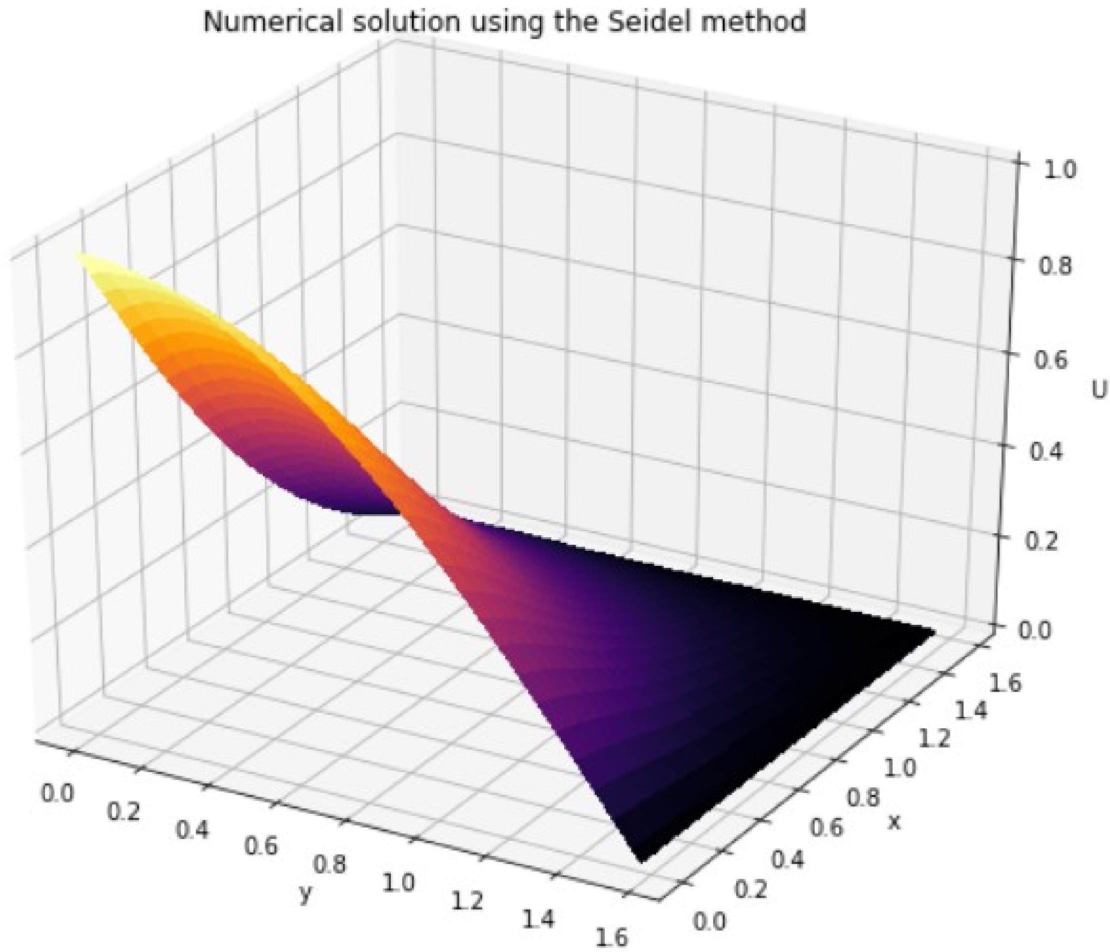
Покажем ошибку в зависимости от длины шага по пространству

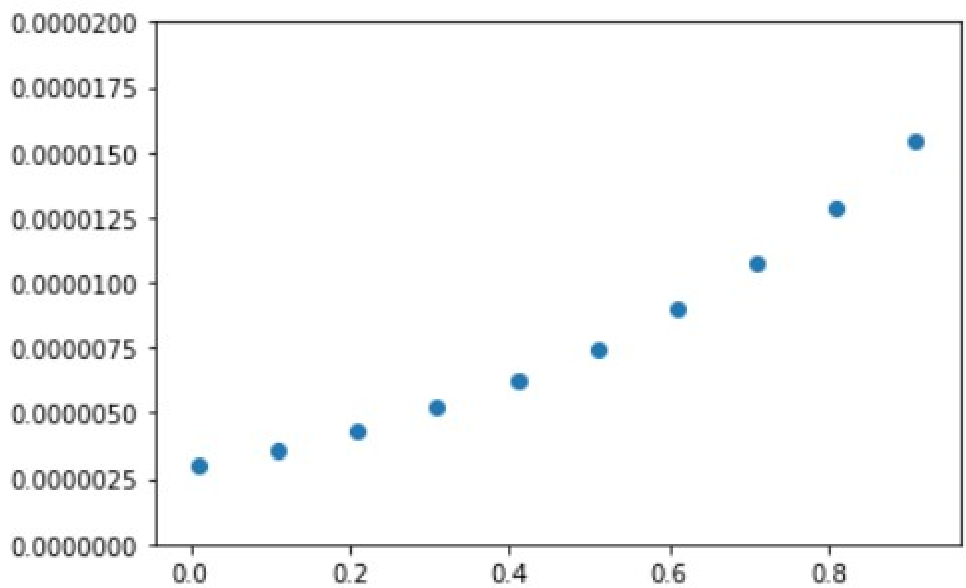


In [2]:
```python
import time

start_dt = time.time()

U_libman = Libman(x,y,x_step, y_step, epsilon)

end_dt = time.time()

print("execution time: {0}".format(end_dt - start_dt))
```

```
execution time: 63.00409150123596
```

Метод Зейделя:





In [9]:
```
import time

start_dt = time.time()

U_seidel = Seidel(x,y,x_step,y_step,epsilon)

end_dt = time.time()

print("execution time: {0}".format(end_dt - start_dt))
```

```
execution time: 108.00409150123596
```

Вывод: в лабораторной работе №3 изучил методы Либмана и Зейделя, построил график зависимости ошибки от размера шага по пространству и сравнил процессорное время выполнения алгоритма: алгоритм Зейделя работает в 1.71 раз медленнее.