Решение эллиптического уравнения методом простой итерации, Зейделя и верхней релаксации

Постановка задачи

Решить данную краевую задачу для эллиптического уравнения методом простой итерации, методом Зейделя и методом верхней релаксации. Сравнить полученные результаты.

Вывод формул

$$a_{11}(x,y)\frac{\partial^2 u}{\partial x^2} + a_{22}(x,y)\frac{\partial^2 u}{\partial y^2} + a_1(x,y)\frac{\partial u}{\partial x} + a_2(x,y)\frac{\partial u}{\partial y} + a(x,y)u = f(x,y) \tag{1}$$

в данной задаче:

$$\begin{aligned} &a_{11} = (x^2+1) \quad a_1 = 2x \qquad a = 0 \\ &a_{22} = \ln(4+y) \quad a_2 = \frac{1}{4+y} \quad f = 0 \end{aligned} \tag{2}$$

граничные условие:

$$\frac{\partial u}{\partial x}(0,y) = \frac{\partial u}{\partial x}(1,y) = 0; \quad u(x,0) = u(x,1) = 1 + 4x(1-x); \tag{3}$$

область:

$$x \in [0,1]; y \in [0,1] \tag{4}$$

Вводим равномерную сетку: $x_i = ih$, $y_k = kh$, h - шаг сетки.

Решение ищем в виде: $u_{ik} \approx u(x_i, y_k)$

Полученное уравнение:

$$\begin{aligned} a_{11}(x_i,y_k)\frac{u_{i+1,k} - 2u_{ik} + u_{i-1,k}}{h^2} + a_{22}(x_i,y_k)\frac{u_{i,k+1} - 2u_{ik} + u_{i,k+1}}{h^2} + \\ + a_1(x_i,y_k)\frac{u_{i+1,k} - u_{i-1,k}}{2h} + a_2(x_i,y_k)\frac{u_{i,k+1} - u_{i,k-1}}{2h} + a(x_i,y_k)u_{ik} = f(x_i,y_k) \end{aligned} \tag{5}$$

Аппроксимация граничных условий:

$$\frac{\partial u}{\partial x}(0,y) = \frac{u(h,y) - u(0,y)}{h} - \frac{h}{2}\frac{\partial^2 u}{\partial x^2}(0,y) + O(h^2) \tag{6}$$

1. **Метод простых итераций:**
   Подготовка: разрешить уравнение относительно диагонального члена u_{ik}:

$$u_{ik}^{(n+1)} = \frac{1}{E_{ik}}\left(A_{ik}u_{i-1,k}^{(n)} + B_{ik}u_{i,k-1}^{(n)} + C_{ik}u_{i,k+1}^{(n)} + D_{ik}u_{i+1,k}^{(n)}\right) - \frac{F_{ik}}{E_{ik}} \tag{7}$$

здесь:

$$\begin{aligned} &A_{ik} = \frac{a_{11}}{h^2} - \frac{a_1}{2h} \quad D_{ik} = \frac{a_{11}}{h^2} + \frac{a_1}{2h} \quad E_{ik} = \frac{2}{h^2}(a_{11} + a_{22}) - a \\ &B_{ik} = \frac{a_{22}}{h^2} - \frac{a_2}{2h} \quad C_{ik} = \frac{a_{22}}{h^2} - \frac{a_2}{2h} \quad F_{ik} = f(x_i,y_k) \end{aligned} \tag{8}$$

условие прекращения итераций : $\|u^{n+1} - u^n\| < \varepsilon$

2. **Метод Зейделя:**
   Для ускорения сходимости используют метод Зейделя:

$$u_{ik}^{(n+1)} = \frac{1}{E_{ik}} \left(A_{ik} u_{i-1,k}^{(n+1)} + B_{ik} u_{i,k-1}^{(n+1)} + C_{ik} u_{i,k+1}^{(n)} + D_{ik} u_{i+1,k}^{(n)} \right) - \frac{F_{ik}}{E_{ik}} \tag{9}$$

3. **Метод верхней релаксации:**
   При правильном подборе параметра ω метод является наиболее быстро сходящимся. Сначала находят $\tilde{u}_{ik}^{(n+1)}$ по формуле метода Зейделя, но его не считают окончательным, а подвергают релаксации:

$$u_{ik}^{(n+1)} = u_{ik}^{(n)} + \omega(\tilde{u}_{ik}^{(n+1)} - u_{ik}^{(n)}) \tag{10}$$

   Условие сходимости: $\omega \in (1, 2)$. ω_{opt} находят эмпирически.

Текст программы

```
import math
import numpy as np
n = int(raw_input('n= '))
eps = 0.000001; h = 1.0 / n
u = np.zeros((n+1,n+1)); u_next = np.zeros((n+1,n+1))
A = np.zeros((n+1,n+1)); B = np.zeros((n+1,n+1))
C = np.zeros((n+1,n+1)); D = np.zeros((n+1,n+1))
E = np.zeros((n+1,n+1))
for i in range(n+1):
      for k in range(n+1):
            x = i*h; y = k*h
            A[i][k] = (1.0+x**2)/h**2 - x/h; D[i][k] = (1.0+x**2)/h**2 + x/h
            B[i][k] = math.log(4.0+y) / h**2 - 1.0/((4.0+y)*2.0*h)
            C[i][k] = math.log(4.0+y) / h**2 + 1.0/((4.0+y)*2.0*h)
            E[i][k] = 2.0/h**2*(1.0+x**2+math.log(4.0+y))
for i in range(n+1):
      A[0][i] = 0; D[0][i] = 0; E[0][i] = 2.0/h**2*math.log(4.0+(i*h))
      B[0][i] = math.log(4.0+(i*h)) / h**2 - 1.0/((4.0+(i*h))*2.0*h)
      C[0][i] = math.log(4.0+(i*h)) / h**2 + 1.0/((4.0+(i*h))*2.0*h)
A[n] = A[0]; B[n] = B[0]; C[n] = C[0]; D[n] = D[0]; E[n] = E[0]
for i in range(n+1):
      u_next[i][0] = 1.0+4.0*(i*h)*(1-i*h)
      u_next[i][n] = 1.0+4.0*(i*h)*(1-i*h)
def meth_iter(curr,next):
      n_iter = 0
      u = np.array(curr); u_next = np.array(next)
      while (np.max(abs(u_next - u)) > eps):
            u = [[u_next[i][k] for k in range(n+1)] for i in range(n+1)]
            for i in range(n+1):
                  u_next[i][0] = 1.0+4.0*(i*h)*(1.0-i*h)
            for i in range(1,n):
                  for k in range(1,n):
                        u_next[i][k] = 1.0/E[i][k] * (A[i][k]*u[i-1][k]+B[i][k]*u[i
    ][k-1]+C[i][k]*u[i][k+1]+D[i][k]*u[i+1][k])
            for i in range(n+1):
                  u_next[i][n] = 1.0+4.0*(i*h)*(1.0-i*h)
            n_iter = n_iter + 1
      print 'number of iterations = ',n_iter
      return u_next
```

```
def meth_zeidel(curr,next):
    n_iter = 0
    u = np.array(curr); u_next = np.array(next)
    while (np.max(abs(u_next - u)) > eps):
        u = [[u_next[i][k] for k in range(n+1)] for i in range(n+1)]
        for i in range(n+1):
            u_next[i][0] = 1.0+4.0*(i*h)*(1-i*h)
        for i in range(1,n):
            for k in range(1,n):
                u_next[i][k] = 1.0/E[i][k] * (A[i][k]*u_next[i-1][k]+B[i][k]*u_next[i][k-1]+C[i][k]*u[i][k+1]+D[i][k]*u[i+1][k])
        for i in range(n+1):
            u_next[i][n] = 1.0+4.0*(i*h)*(1-i*h)
        n_iter = n_iter + 1
    print 'number of iterations = ',n_iter
    return u_next
def meth_relax(curr,next,omega,return_iter=False):
    n_iter = 0
    u = np.array(curr); u_next = np.array(next)
    while (np.max(abs(u_next - u)) > eps):
        u = [[u_next[i][k] for k in range(n+1)] for i in range(n+1)]
        for i in range(n+1):
            u_next[i][0] = 1.0+4.0*(i*h)*(1-i*h)
        for i in range(1,n):
            for k in range(1,n):
                u_rel = 1.0/E[i][k] * (A[i][k]*u_next[i-1][k]+B[i][k]*u_next[i][k-1]+C[i][k]*u[i][k+1]+D[i][k]*u[i+1][k])
                u_next[i][k] = u[i][k] + omega * (u_rel - u[i][k])
        for i in range(n+1):
            u_next[i][n] = 1.0+4.0*(i*h)*(1-i*h)
        n_iter = n_iter + 1
    if return_iter:
        return n_iter
    else:
        print 'number of iterations = ',n_iter
        return u_next
def find_omega():
  omega0 = 0.95; omega = 1.0
  while (omega <= 2):
    omega0 = omega; omega = omega +0.01
    if (meth_relax(u, u_next, omega, True) - meth_relax(u, u_next, omega0, True) > 0):
      return omega0
  return omega
```

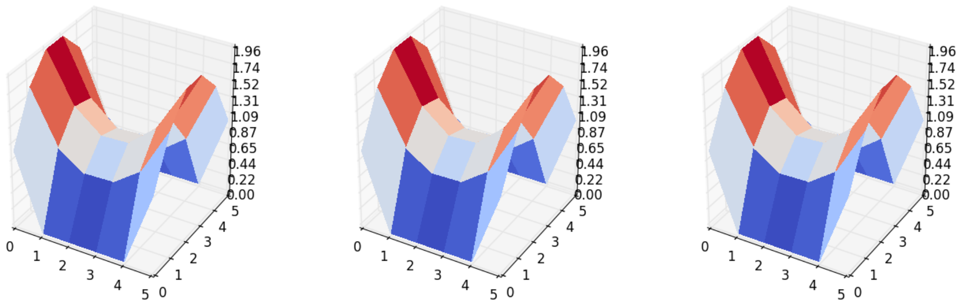
Результаты

Количество требуемых итераций в зависимости от метода и числа узлов сетки:

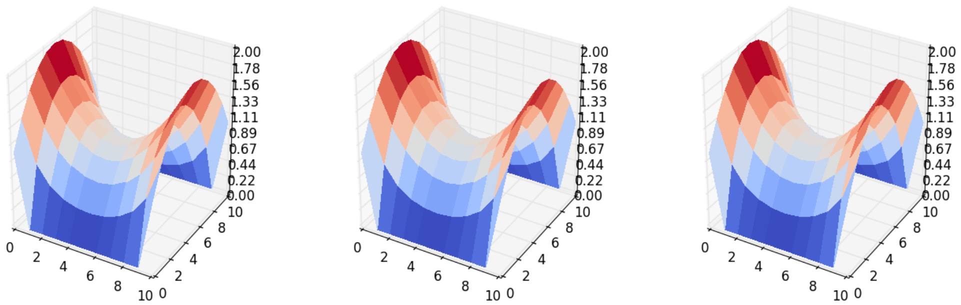
Метод	n = 5	n = 10	n = 15
Простые итерации	59	220	462
Зейделя	33	118	248
Верхняя релаксация	14	28	43

Графики решений

3D график решения для $n = 5$:



3D график решения для $n = 10$:



3D график решения для $n = 15$: