Разностные схемы для уравнений эллиптического типа. Итерационные методы решений сеточных уравнений

Курдоякова Марина

Задание: Найти решение задачи:

$$Lu = -f(x,y)$$

где

$$Lu = \frac{\partial}{\partial x}\left(\ln(2+x)\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(\ln(2+y)\frac{\partial u}{\partial y}\right), \quad 0 < x < 1, 0 < y < 1,$$

$$u(x,y)|_\Gamma = \mu(x,y)$$

следующими методами:

1. Методом простой итерации
2. Методом итерации с оптимальным параметром
3. Методом Зейделя
4. Попеременно-треугольным итерационным методом

Отладка решения происходила на функции $u^*(x,y) = \sin(\pi x)\cos(\pi y)$

Алгоритм решения задачи:

Рассматривается задача Дирихле для эллиптического уравнения

$$-Lu = f(x,y), \quad (x,y) \in G$$

$$u = \mu(x,y), \quad (x,y) \in \Gamma$$

Пусть $\bar{G} = G \cup \Gamma = \{0 \leqslant x \leqslant l_x, 0 \leqslant y \leqslant l_y\}$ — прямоугольник, а

$$Lu = \frac{\partial}{\partial x}\left(p(x,y)\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(q(x,y)\frac{\partial u}{\partial y}\right)$$

Здесь $p(x,y), q(x,y)$ — достаточно гладкие функции такие, что $0 < c_1 \leqslant p(x,y) \leqslant c_2, 0 < d_1 \leqslant q(x,y) \leqslant d_2$, где c_1, c_2, d_1, d_2 - постоянные.
Разобьем отрезок $[0, l_x]$ на N равных частей. Обозначим $h_x = l_x/N, x_i = ih_x, 0 \leqslant i \leqslant N$.
Разобьем отрезок $[0, l_y]$ на M равных частей. Обозначим $h_y = l_y/M, y_j = jh_y, 0 \leqslant j \leqslant M$.

Обозначим $u_{ij} = u(x_i, y_j)$. Заменяем оператор L во всех внутренних узлах разностным оператором

$$L_h u_{ij} = p_{i+\frac{1}{2}j}\frac{u_{i+1j} - u_{ij}}{h_x^2} - p_{i-\frac{1}{2}j}\frac{u_{ij} - u_{i-1j}}{h_x^2} + q_{ij+\frac{1}{2}}\frac{u_{ij+1} - u_{ij}}{h_y^2} - q_{ij-\frac{1}{2}}\frac{u_{ij} - u_{ij-1}}{h_y^2}$$

$$1 \leqslant i \leqslant N-1; \quad 1 \leqslant j \leqslant M-1$$

1. Метод простой итерации.
   Расчетная формула метода простой итерации в общем случае имеет вид:

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

   где $1 \leqslant i \leqslant N-1; \quad 1 \leqslant j \leqslant M-1$

2. Метод итерации с оптимальным параметром.
   В общем случае расчетная формула метода итерации с оптимальным параметром имеет вид:

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

   Для простейшего случая $\tau = h^2/4$. В простейшем случае метод итерации с оптимальным параметром совпадает с методом простой итерации.

3. Метод Зейделя (Некрасова).
   В общем случае расчетная формула примет вид:

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

   где $1 \leqslant i \leqslant N-1; \quad 1 \leqslant j \leqslant M-1$

4. Попеременно-треугольный итерационный метод.

$$\delta = \frac{4}{c_1} \frac{4}{h_x^2} \sin^2 \frac{\pi h_x}{2l_x} + d_1 \frac{4}{h_y^2} \sin^2 \frac{\pi h_y}{2l_y}, \quad \Delta = c_2 \frac{4}{h_x^2} + d_2 \frac{4}{h_y^2}$$

   Обозначим

$$\eta = \delta / \Delta.$$

   И получаем

$$\omega = 2/\sqrt{\delta\Delta}, \quad \gamma_1 = \frac{\delta}{(2 + 2\sqrt{\eta})}, \quad \gamma_2 = \frac{\delta}{4\sqrt{\eta}}, \tau = 2/\left(\gamma_1 + \gamma 2\right)$$

   Обозначим $\kappa_1 = \omega/h_x^2, \quad \kappa_2 = \omega/h_y^2$ю. И тогда в общем случае, исходя из аппроксимации, формулы будут выглядеть следующим образом:

$$\bar{U}_{ij} = \frac{\kappa_1 p_{i-\frac{1}{2}j} \bar{U}_{i-1j} + \kappa_2 q_{ij-\frac{1}{2}} \bar{U}_{ij-1} + F_{ij}^{k-1}}{1 + \kappa_1 p_{i-\frac{1}{2}j} + \kappa_2 q_{ij-\frac{1}{2}}}, i = 1, \ldots, N-1, j = 1, \ldots, M-1; \bar{U} \mid \gamma_{h_x h_y} = 0$$

$$U_{ij}^k = \frac{\kappa_1 p_{i+\frac{1}{2}j} U_{i+1j}^k + \kappa_2 q_{ij+\frac{1}{2}} U_{ij+1}^k + \bar{U}_{ij}}{1 + \kappa_1 p_{i+\frac{1}{2}j} + \kappa_2 q_{ij+\frac{1}{2}}}, i = N-1, \ldots, 1, j = M-1, \ldots, 1; U^k \mid \gamma_{h_x h_y} = 0$$

$$\begin{gathered} (E + \omega R)\bar{w} = \Phi^{k-1}, \bar{w} \mid \gamma_{h_x h_y} = 0 \\ \Phi^{k-1} = L_h u^{k-1} + f, \quad u^{k-1} \mid \gamma_{h_x h_y} = \mu \\ \left(E + \omega R^T\right) w^k = \bar{w}, \quad w^k \mid \gamma_{h_x h_y} = 0 \\ U^k = U^{k-1} + \tau w^k \end{gathered}$$

Код программы: .

```
import numpy as np
from math import sqrt, sin, pi, cos, log, exp, acos
from scipy.integrate import dblquad, quad
import matplotlib.pyplot as plt
import pylab
from matplotlib import cm
from mpl_toolkits.mplot3d import Axes3D

ax, lx = [0,1]
```

```
ay, ly = [0,1]
n = 15
m = 20
hx = (lx-ax)/n
hy = (ly-ay)/m

eps = 0.001
p = lambda x,y: log(2 + x)
q = lambda x,y: log(2 + y)
mu = lambda x,y: sin(pi*x)*cos(pi*y)
solution_func = lambda x,y: sin(pi*x)*cos(pi*y)

x = [i*hx for i in range(n+1)]
y = [i*hy for i in range(m+1)]

solution = np.zeros((n+1, m+1))
for i in range(n+1):
    for j in range(m+1):
        solution[i][j] = solution_func(x[i],y[j])

c1, c2, d1, d2 = 1,0,1,0
for i in range (1,n):
    for j in range(1,m):
        if  p(x[i],y[j]) < c1:
            c1 = p(x[i],y[j])
        if p(x[i],y[j]) > c2:
            c2 = p(x[i],y[j])
        if q(x[i],y[j]) < d1:
            d1 = q(x[i],y[j])
        if q(x[i],y[j]) > d2:
            d2 = q(x[i],y[j])

delta = (c1*4*sin(pi*hx/(2*lx))**2)/hx**2 + (d1*4*sin(pi*hy/(2*ly))**2)/hy**2
Delta = c2*4/hx**2 + d2*4/hy**2
nu = delta/Delta
omega = 2/(sqrt(delta*Delta))
gamma1 = delta/(2 + 2*sqrt(nu))
gamma2 = delta/(4*sqrt(nu))
k1 = omega/hx**2
k2 = omega/hy**2
zeta = delta/Delta
# zeta = gamma1/gamma2
ro = (1 - zeta)/(1 + zeta)

def Lu(u):
    L1 = p(x[i] + hx/2, y[j])*(u[i+1][j] - u[i][j])/hx**2
    L2 = p(x[i] - hx/2, y[j])*(u[i][j] - u[i-1][j])/hx**2
    L3 = q(x[i], y[j] + hy/2)*(u[i][j+1] - u[i][j])/hy**2
    L4 = q(x[i], y[j] - hy/2)*(u[i][j] - u[i][j-1])/hy**2
    L = L1 - L2 + L3 - L4
    return L

def dirichle(u):
    for i in range(n+1):
```

```
        u[i][0] = mu(x[i], 0)
        u[i][m] = mu(x[i], ly)
    for j in range(1, m):
        u[0][j] = mu(0,y[j])
        u[n][j] = mu(lx,y[j])
    return u

def f(x,y):
    a = (sin(pi*x)*(sin(pi*y) + pi*(y + 2)*log(y+2)*cos(pi*y)))/(y+2)
    b = (cos(pi*y)*(cos(pi*x) - pi*(x + 2)*log(x+2)*sin(pi*x)))/(x+2)
    fi = pi*(a - b)
    return fi

def simple_itt(u):
    U = np.zeros((n+1, m+1))
    U = dirichle(U)
    for i in range(1,n):
        for j in range(1,m):
            u1 = (p(x[i] - hx/2, y[j]) * u[i-1][j])/hx**2
            u2 = (p(x[i] + hx/2, y[j]) * u[i+1][j])/hx**2
            u3 = (q(x[i], y[j] - hy/2) * u[i][j-1])/hy**2
            u4 = (q(x[i], y[j] + hy/2) * u[i][j+1])/hy**2
            u5 = p(x[i] - hx/2, y[j])/hx**2
            u6 = p(x[i] + hx/2, y[j])/hx**2
            u7 = q(x[i], y[j] - hy/2)/hy**2
            u8 = q(x[i], y[j] + hy/2)/hy**2
            U[i][j] = (u1 + u2 + u3 + u4 + f(x[i],y[j]))/(u5 + u6 + u7 + u8)
    return U

def optimal(u):
    tau = 2/(delta + Delta)
    U = np.zeros((n+1, m+1))
    U = dirichle(U)
    for i in range(1,n):
        for j in range(1,m):
            u1 = p(x[i] + hx/2, y[j])*(u[i+1][j] - u[i][j])/hx**2
            u2 = p(x[i] - hx/2, y[j])*(u[i][j] - u[i-1][j])/hx**2
            u3 = q(x[i], y[j] + hy/2)*(u[i][j+1] - u[i][j])/hy**2
            u4 = q(x[i], y[j] - hy/2)*(u[i][j] - u[i][j-1])/hy**2
            U[i][j] = u[i][j] + tau*(u1 - u2 + u3 - u4 + f(x[i],y[j]))
    return U

def zeidel(u):
    U = np.zeros((n+1, m+1))
    U = dirichle(U)
    for i in range(1,n):
        for j in range(1,m):
            u1 = (p(x[i] - hx/2, y[j]) * U[i-1][j])/hx**2
            u2 = (p(x[i] + hx/2, y[j]) * u[i+1][j])/hx**2
            u3 = (q(x[i], y[j] - hy/2) * U[i][j-1])/hy**2
            u4 = (q(x[i], y[j] + hy/2) * u[i][j+1])/hy**2
            u5 = p(x[i] - hx/2, y[j])/hx**2
            u6 = p(x[i] + hx/2, y[j])/hx**2
            u7 = q(x[i], y[j] - hy/2)/hy**2
```

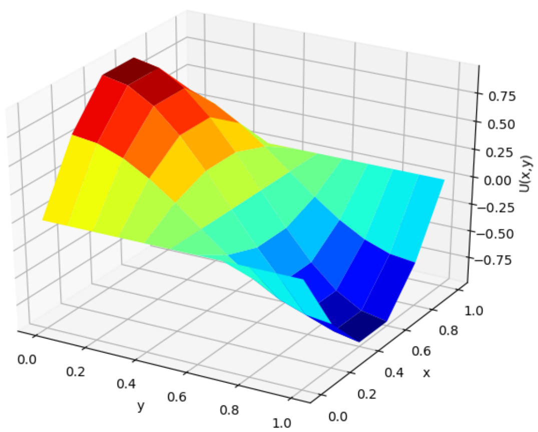
```
            u8 = q(x[i], y[j] + hy/2)/hy**2
            U[i][j] = (u1 + u2 + u3 + u4 + f(x[i],y[j]))/(u5 + u6 + u7 + u8)
    return U

def altern_triang(Uk):
    tau = 2/(gamma1 + gamma2)
    U = np.zeros((n+1, m+1)); W = np.zeros((n+1, m+1))
    WW = np.zeros((n+1, m+1)); F = np.zeros((n+1, m+1))
    u = np.zeros((n+1, m+1))

    u = np.copy(Uk)
    u = dirichle(u)

    for i in range(1,n):
        for j in range(1,m):
            F[i][j] = Lu(u) + f(x[i],y[j])
            uu1 = k1*p(x[i] - hx/2, y[j])*WW[i-1][j]
            uu2 = k2*q(x[i], y[j] - hy/2)*WW[i][j-1]
            uu3 = k1*p(x[i] - hx/2, y[j])
            uu4 = k2*q(x[i], y[j] - hy/2)
            WW[i][j] = (uu1 + uu2 + F[i][j])/(1 + uu3 + uu4)

    for i in range(n-1,0,-1):
        for j in range(m-1,0,-1):
            u1 = k1*p(x[i] + hx/2, y[j])*W[i+1][j]
            u2 = k2*q(x[i], y[j] + hy/2)*W[i][j+1]
            u3 = k1*p(x[i] + hx/2, y[j])
            u4 = k2*q(x[i], y[j] + hy/2)
            W[i][j] = (u1 + u2 + WW[i][j])/(1 + u3 + u4)

    for i in range(n+1):
        for j in range(m+1):
            U[i][j] = Uk[i][j] + tau*W[i][j]
    return U

def plot_surface(x, t, U):
    fig = pylab.figure()
    ax = Axes3D(fig)
    x, t = np.meshgrid(x, t)
    ax.plot_surface(x, t, U,cmap = cm.jet)
    ax.set_xlabel('y')
    ax.set_ylabel('x')
    ax.set_zlabel('U(x,y)')
    pylab.show()

def main(eps):
    U = np.zeros((n+1, m+1)); Y = np.zeros((n+1, m+1))
    Q = np.zeros((n+1, m+1)); O = np.zeros((n+1, m+1))
    J = np.zeros((n+1, m+1)); u = np.zeros((n+1, m+1))

    all_eps = 1; k = 0
    while all_eps > eps:
    # for l in range(k):
        U = simple_itt(U)
```

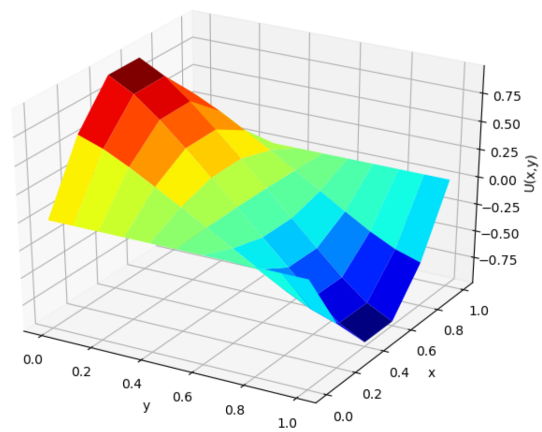
```
        Y = optimal(Y)
        Q = zeidel(Q)
        O = altern_triang(O)

        eps1, eps2, eps3, eps4, eps5 = 1,1,1,1,1
        for i in range (1,n):
            for j in range(1,m):
                if solution[i][j] < eps1:
                    eps1 = solution[i][j]
                if U[i][j] < eps2:
                    eps2 = U[i][j]
                if Y[i][j] < eps3:
                    eps3 = Y[i][j]
                if Q[i][j] < eps4:
                    eps4 = Q[i][j]
                if O[i][j] < eps5:
                    eps5 = O[i][j]

        e1 = abs(eps1-eps2)
        e2 = abs(eps1-eps3)
        e3 = abs(eps1-eps4)
        e4 = abs(eps1-eps5)

        all_eps = max(e1,e2,e3)
        k += 1
        # if k%10 == 0:
        #   plot_surface(y, x, u)

    print("%.6f" % e1, 'simple_itt')
    print("%.6f" % e2, 'optimal')
    print("%.6f" % e3, 'zeidel')
    print("%.6f" % e4, 'altern_triang')
    print('k = ', k)

    # print(y)
    # for i in range(n+1):
    #   print('{0:.1f}'.format(x[i]), U[i])

    plot_surface(y, x, solution)
    plot_surface(y, x, U)
    plot_surface(y, x, Y)
    plot_surface(y, x, Q)
    plot_surface(y, x, O)

main(eps)
```

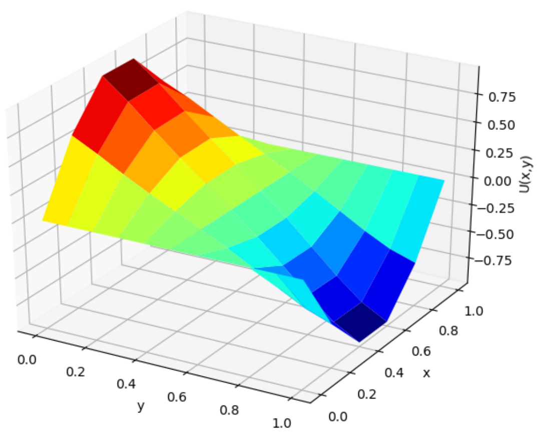
Результаты:

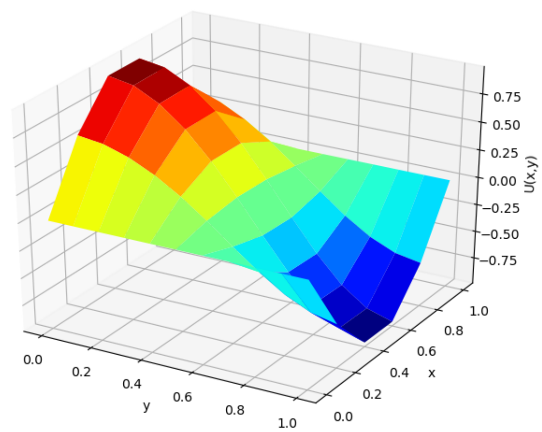
Алгоритм останавливается, когда достагнута заданная точность. Каждый раз прошу выводить значение k, чтобы она не выходила за максимальное значение. Так же вывожу таблицу погрешностей сначала для разницы максимального элемента данного решения и точного, потом для последующих иттераций.

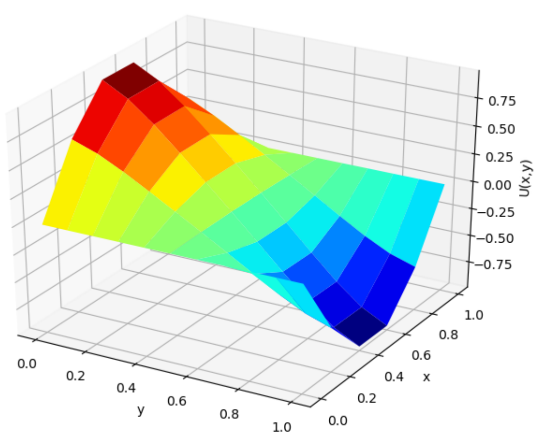
Давайте сначала прогоним на мелкой сетке, для примера возьмем N = 5, M = 10 eps = 0.1





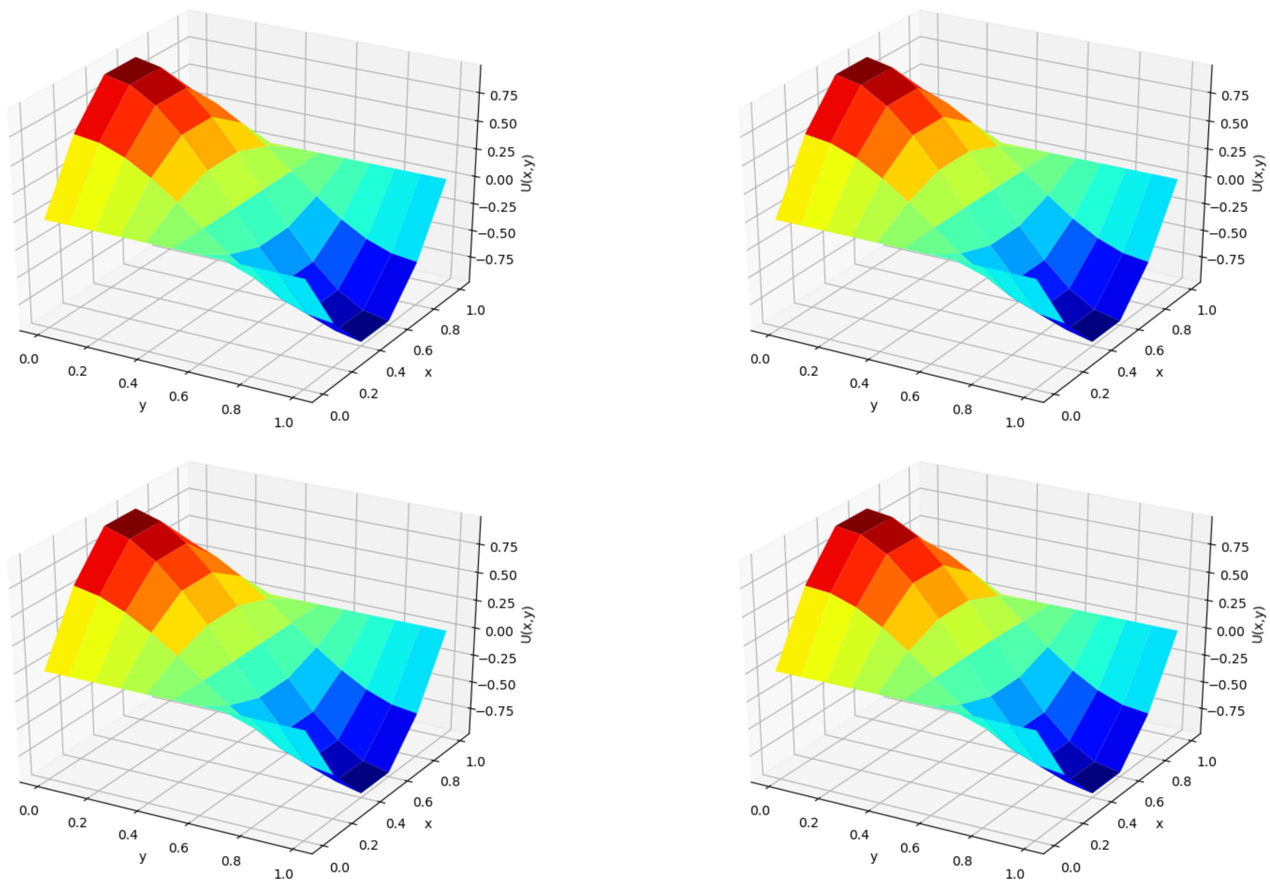


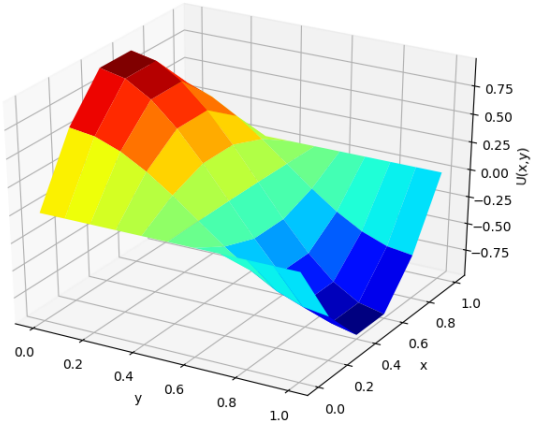




1	0.835176	0.838025	0.793368	0.802677
2	0.406988	0.426618	0.320846	0.396306
3	0.322462	0.332401	0.184708	0.399232
4	0.229911	0.245203	0.120045	0.239766
5	0.184532	0.196169	0.082868	0.155438
6	0.142792	0.155841	0.059373	0.110865
7	0.115288	0.126640	0.043628	0.083692
8	0.091586	0.102911	0.032631	0.062095
9	0.073862	0.084104	0.024699	0.032056

eps $= 0.01$

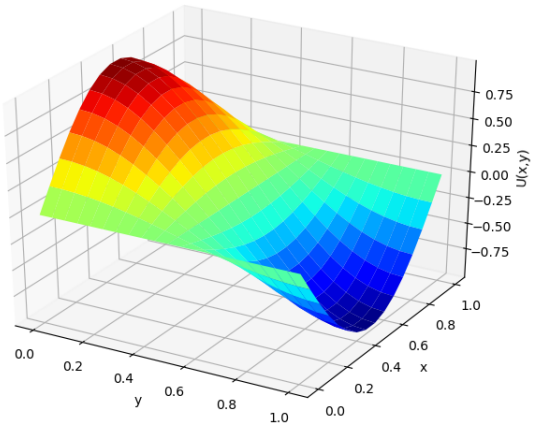


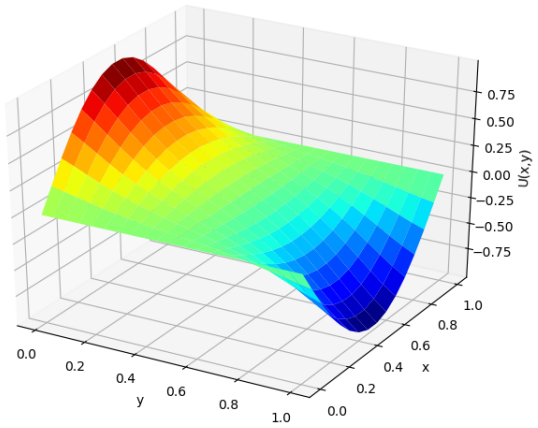


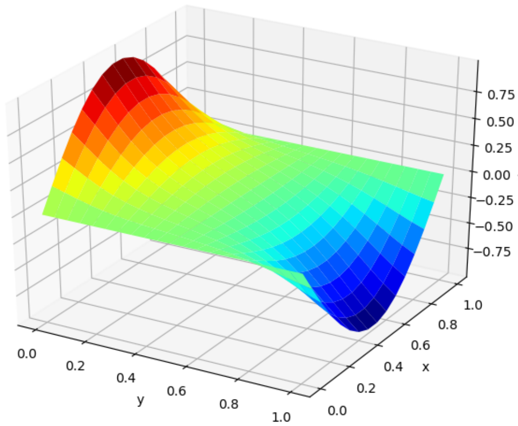
для разных k

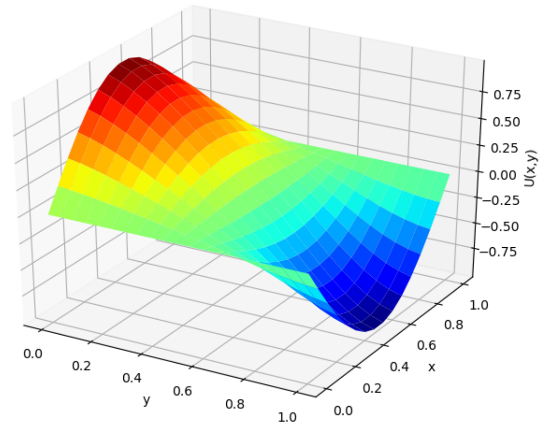
k	simple_itt	optimal	zeidel	ltern_triang
1	0.835176	0.838025	0.793368	0.802677
2	0.406988	0.426618	0.320846	0.396306
3	0.322462	0.332401	0.184708	0.399232
4	0.229911	0.245203	0.120045	0.239766
5	0.184532	0.196169	0.082868	0.155438
6	0.142792	0.155841	0.059373	0.110865
7	0.115288	0.126640	0.043628	0.083692
8	0.091586	0.102911	0.032631	0.062095
9	0.073862	0.084104	0.024699	0.032056
10	0.059093	0.068662	0.018825	0.020943
11	0.047488	0.056014	0.014374	0.016188
12	0.037952	0.045545	0.010936	0.010505
13	0.030309	0.036868	0.008234	0.005178
14	0.024058	0.029657	0.006079	0.003861
15	0.019004	0.023657	0.004337	0.019870
16	0.014871	0.018662	0.002916	0.007056
17	0.011513	0.014503	0.001744	0.003890
18	0.008763	0.011041	0.000773	0.001607
19	0.006521	0.008161	0.000039	0.000103

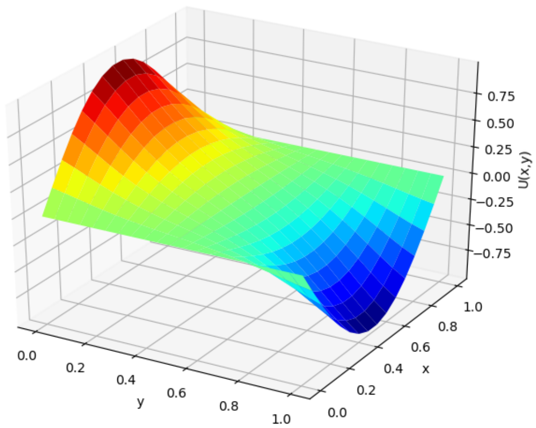
Теперь рассмотрим решения для более крутной сетки: возьмем N = 15, M = 20
eps = 0.1





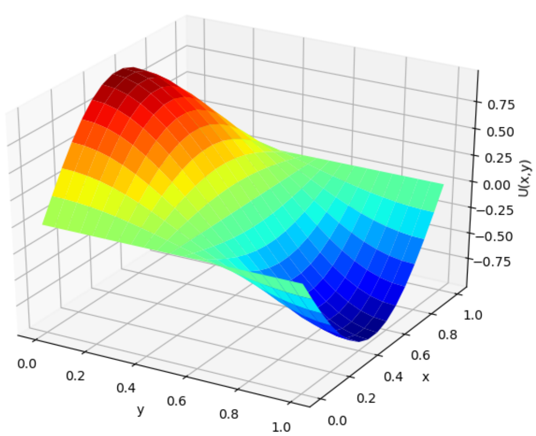


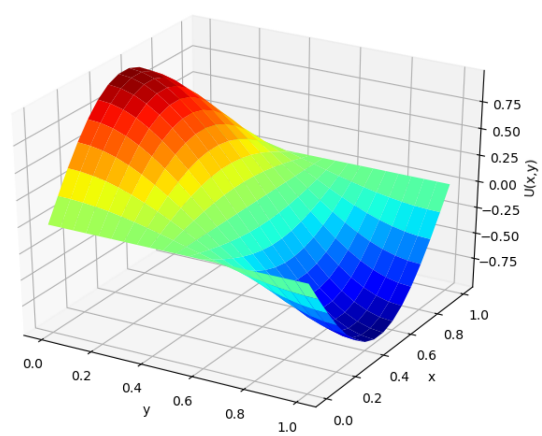


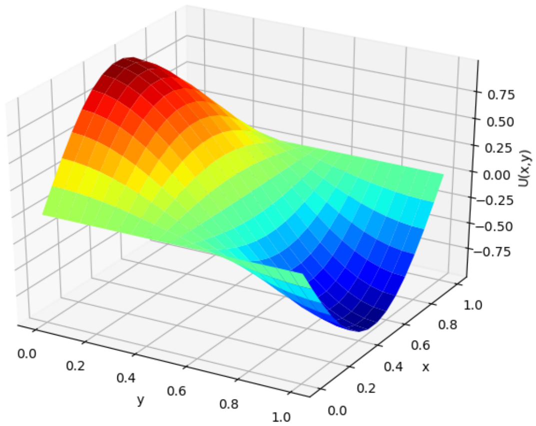


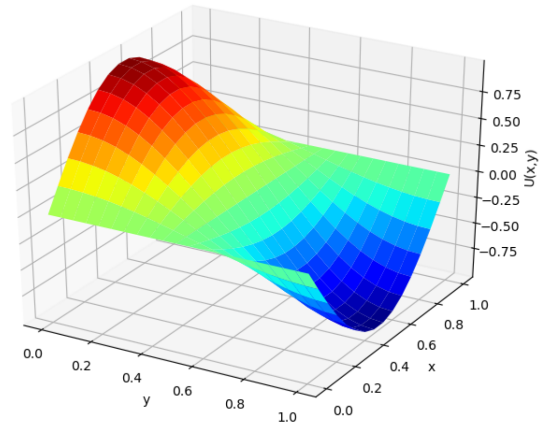
1	0.967117	0.967934	0.953697	0.957737
5	0.370462	0.384643	0.251298	0.350427
10	0.231448	0.243212	0.130608	0.191448
20	0.126520	0.136159	0.053367	0.093650
25	0.099016	0.108043	0.036585	0.069217
26	0.094499	0.103412	0.034036	0.063451
27	0.090243	0.099044	0.031693	0.061920

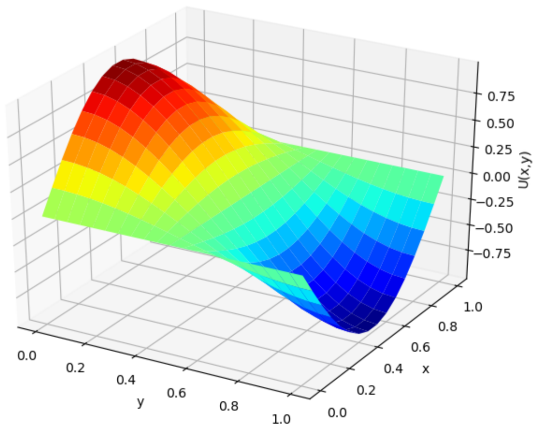
eps $= 0.001$











1	0.967117	0.967934	0.953697	0.957737
10	0.231448	0.243212	0.130608	0.191448
20	0.126520	0.136159	0.053367	0.093650
50	0.034182	0.040366	0.007512	0.009550
70	0.015383	0.019232	0.002683	0.008362
100	0.004651	0.005913	0.000616	0.001104
115	0.002514	0.003012	0.000229	0.000914
125	0.001640	0.001788	0.000064	0.000141
136	0.000995	0.000874	0.000062	0.000081

И в конце приведу сетку значений для всех методов:

Точное решение

x/y	0.0	0.2	0.4	0.6	0.8	1.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.2	5.87785e-01	4.7552e-01	1.8163e-01	-1.8163e-01	-4.7552e-01	-5.8778e-01
0.4	9.51056e-01	7.6942e-01	2.9389e-01	-2.9389e-01	-7.6942e-01	-9.5105e-01
0.6	9.51056e-01	7.6942e-01	2.9389e-01	-2.9389e-01	-7.6942e-01	-9.5105e-01
0.8	5.87785e-01	4.7552e-01	1.8163e-01	-1.8163e-01	-4.7552e-01	-5.8778e-01
1.0	1.22464e-16	9.9076e-17	3.7843e-17	-3.7843e-17	-9.9076e-17	-1.2246e-16

Простой иттерации

0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.2	0.58778525	0.47951815	0.18413436	-0.18246288	-0.47799612	-0.58778525

```
0.4   0.95105652    0.77710582    0.29842008   -0.29584347   -0.77458322   -0.95105652
0.6   0.95105652    0.77800774    0.29877802   -0.29630152   -0.77543623   -0.95105652
0.8   0.58778525    0.48135541    0.18485718   -0.18338151   -0.4797456    -0.58778525
1.0  1.224646e-16  9.907600e-17  3.784366e-17  -3.784366e-17 -9.907600e-17 -1.224646e-16
-------------------------------------------
С оптимальный переметром

0.0   0.0           0.0           0.0           0.0           0.0           0.0
0.2   0.58778525    0.47951671    0.18413221   -0.18246488   -0.47799728   -0.58778525
0.4   0.95105652    0.77710369    0.2984169    -0.29584642   -0.77458494   -0.95105652
0.6   0.95105652    0.77800577    0.29877508   -0.29630425   -0.77543782   -0.95105652
0.8   0.58778525    0.48135427    0.18485547   -0.18338309   -0.47974652   -0.58778525
1.0  1.224646e-16  9.907600e-17  3.784366e-17  -3.784366e-17 -9.907600e-17 -1.224646e-16
-------------------------------------------
Зейделя

0.0   0.0           0.0           0.0           0.0           0.0           0.0
0.2   0.58778525    0.47951804    0.1841342    -0.18246304   -0.47799621   -0.58778525
0.4   0.95105652    0.77710566    0.29841984   -0.29584371   -0.77458337   -0.95105652
0.6   0.95105652    0.77800759    0.29877778   -0.29630174   -0.77543637   -0.95105652
0.8   0.58778525    0.48135532    0.18485704   -0.18338164   -0.47974568   -0.58778525
1.0  1.224646e-16  9.907600e-17  3.784366e-17  -3.784366e-17 -9.907600e-17 -1.224646e-16
-------------------------------------------
Попеременно-треугольный

0.0   0.0           0.0           0.0           0.0           0.0           0.0
0.2   0.58778525    0.47951804    0.1841342    -0.18246304   -0.47799621   -0.58778525
0.4   0.95105652    0.77711346    0.29841245   -0.29584391   -0.77458328   -0.95105693
0.6   0.95105652    0.77800349    0.29877734   -0.29630147   -0.77543647   -0.95105628
0.8   0.58778525    0.48135562    0.18485727   -0.18338184   -0.47974528   -0.58778368
1.0  1.224646e-16  9.907600e-17  3.784366e-17  -3.784366e-17 -9.907600e-17 -1.224646e-16
```