Московский государственный университет имени М.В.Ломоносова

Казахстанский филиал

Факультет вычислительной математики и кибернетики

Отчёт по практикуму по специализации

Численное решение задачи Дирихле для уравнения Пуассона

Составил: студент Калдаров Б.М.

Проверил: преподаватель Нетесов В.В.

Нур-Султан, 2020

Содержание

1 Постановка задачи

Рассматривается задача Дирихле для эллиптического уравнения

$$-Lu = f(x,y), \quad (x,y) \in G, \tag{1}$$

$$u = \mu(x,y), \quad (x,y) \in H. \tag{2}$$

Пусть $\overline{G} = G + H = 0 \leq x \leq l_x, 0 \leq y \leq l_y$ - прямоугольник, а

$$Lu = \frac{\partial}{\partial x}\left(p(x,y)\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(q(x,y)\frac{\partial u}{\partial y}\right) \tag{3}$$

Здесь $p(x,y), q(x,y)$ - достаточно гладкие функции такие, что $0 < c_1 \leq p(x,y) \leq c_2, 0 < d_1 \leq q(x,y) \leq d_2$, где c_1, c_2, d_1, d_2 - постоянные. Обозначим $A = max(c_2, d_2)$. Разобьём отрезок $[0, l_x]$ на N_x равных частей. Обозначим $h_x = \frac{l_x}{N_x}, x_i = ih_x, 0 \leq i \leq N_x$.
Разобьём отрезок $[0, l_y]$ на N_y равных частей. Обозначим $h_y = \frac{l_y}{N_y}, y_j = jh_y, 0 \leq j \leq N_y$.

2 Двухслойная схема с весами

Построим сетку узлов:

$$\overline{\omega_{h_x h_y}} = (x_i, y_j), \quad 0 \leq i \leq N_x, 0 \leq j \leq N_y.$$

Узлы $(x_i, y_j), 1 \leq i \leq N_x - 1, 1 \leq j \leq N_y - 1$ - внутренние, остальные, лежащие на границе прямоугольника, - граничные.
Пусть

$$L_1 u = \frac{\partial}{\partial x}\left(p(x,y)\frac{\partial u}{\partial x}\right), \quad L_2 u = \frac{\partial}{\partial y}\left(q(x,y)\frac{\partial u}{\partial y}\right), \tag{4}$$

так что

$$Lu = L_1 u + L_2 u. \tag{5}$$

Операторы L_1 и L_2 заменим разностными операторами Λ_1 и Λ_2

$$\Lambda_1 = p_{i+\frac{1}{2}j}\frac{u_{i+1j} - u_{ij}}{h_x^2} - p_{i-\frac{1}{2}j}\frac{u_{ij} - u_{i-1j}}{h_x^2}, \tag{6}$$

$$\Lambda_2 = q_{ij+\frac{1}{2}}\frac{u_{ij+1} - u_{ij}}{h_y^2} - q_{ij-\frac{1}{2}}\frac{u_{ij} - u_{ij-1}}{h_y^2}. \tag{7}$$

Здесь

$$p_{i+\frac{1}{2}j} = p(x_i + \frac{h_x}{2}, y_j), \quad p_{i-\frac{1}{2}j} = p(x_i - \frac{h_x}{2}, y_j),$$

$$q_{ij+\frac{1}{2}} = p(x_i, y_j + \frac{h_y}{2}), \quad p_{ij-\frac{1}{2}} = p(x_i, y_j - \frac{h_y}{2}).$$

Обозначим

$$\Lambda u_{ij} = \Lambda_1 u_{ij} + \Lambda_2 u_{ij}, \quad 1 \le i \le N_x - 1, \quad 1 \le j \le N_y - 1.$$

Если $u(x, y)$ имеет не менее четырех непрерывных ограниченных в рассматриваемой области G производных по x и по y, а $p(x, y)$ и $q(x, y)$ - не менее трех, то разностный оператор Λ аппроксимирует дифференциальный L со вторым порядком, т. е.

$$Lu - \Lambda u = O(|h|^2), |h|^2 = h_x^2 + h_y^2.$$

Итак, решение задачи $(1) - (2)$ свелось к решению разностной задачи Дирихле

$$-(\Lambda_1 u_{ij} + \Lambda_2 u_{ij}) = f_{ij}, \quad 1 \le i \le N_x - 1, \quad 1 \le j \le N_y - 1.$$

Аппроксимируем задачу

$$\begin{cases} \frac{\partial u}{\partial t} = L_1 u + L_2 u + f(x, y), \\ u|_H = \mu(x, y), \quad u(x, y, 0) = u_0(x, y). \end{cases} \tag{8}$$

разностной схемой

$$\frac{u_{ij}^{k+1} - u_{ij}^k}{\tau} = \Lambda(\sigma u_{ij}^{k+1} + (1-\sigma)u_{ij}^k + f(x_i, y_j)), \tag{9}$$

$$i = \overline{1, N_x - 1}, \quad j = \overline{1, N_y - 1}, \quad k = 0, 1, 2, ...$$

$$\begin{cases} u_{i0}^{k+1} = \mu(x_i, 0), \quad 1 \leq i \leq N_x - 1, \\ u_{iN_y}^{k+1} = \mu(x_i, l_y), \quad 1 \leq i \leq N_x - 1, \\ u_{0j}^{k+1} = \mu(0, y_j), \quad 1 \leq j \leq N_y - 1, \\ u_{N_xj}^{k+1} = \mu(N_x, y_j), \quad 1 \leq j \leq N_y - 1. \end{cases} \tag{10}$$

Решение при $k = 0$ находится из начального условия в (9)

$$u_{ij}^0 = u_0(x_i, y_j), \quad , i = \overline{1, N_x}, \quad j = \overline{1, N_y}.$$

В данной работе рассматривается чисто неявная схема с значением параметра $\sigma = 1$.

Эта схема устойчива при любых значениях h и τ. Для определения u_{ij}^{k+1} на каждом слое получаем линейную систему

$$u_{ij}^{k+1} - \tau(\Lambda_1 u_{ij}^{k+1} + \Lambda_2 u_{ij}^{k+1}) = u_{ij}^k + \tau f(x_i, y_j), \tag{11}$$

$$i = \overline{1, N_x - 1}, \quad j = \overline{1, N_y - 1}, \quad k = 0, 1, 2, ...$$

Матрица этой системы пятидиагональная и система решается методом матричной прогонки[1].

3 Схема переменных направлений

Эта схема сочетает лучшие качества явной схемы - экономичность и неявной - устойчивость. Наряду с основными значениями u_{ij}^k u_{ij}^{k+1} вводится промежуточное значение $u_{ij}^{k+\frac{1}{2}}$, которое формально можно рассматривать как значение при $t = t_{k+\frac{1}{2}} = t_k + \frac{\tau}{2}$. Решение задачи в этом случае сводится к решению двух систем с трехдиагональными матрицами:

$$\frac{u_{ij}^{k+\frac{1}{2}} - u_{ij}^k}{\frac{\tau}{2}} = \Lambda_1 u_{ij}^{k+\frac{1}{2}} + \Lambda_2 u_{ij}^k + f(x_i, y_j)) \tag{12}$$

$$1 \le i \le N_x - 1, \quad 1 \le j \le N_y - 1.$$

$$\frac{u_{ij}^{k+1} - u_{ij}^{k+\frac{1}{2}}}{\frac{\tau}{2}} = \Lambda_1 u_{ij}^{k+\frac{1}{2}} + \Lambda_2 u_{ij}^{k+1} + f(x_i, y_j)) \tag{13}$$

$$1 \le i \le N_x - 1, \quad 1 \le j \le N_y - 1.$$

$$k = 0, 1, 2, ...$$

В граничных узлах решение должно принимать заданные в (10) значения.

Схема (12) неявна по направлению x и явна по направлению y, а схема (13) ясна по направлению x и неявна по направлению y, что позволяет использовать для нахождения решения одномерные прогонки.

Система (12) с учётом граничных условий (10) может быть записана в следующем виде:

$$\begin{cases} u_{0j}^{k+\frac{1}{2}} = \mu(0, y_j), \\ \overline{A_{ij}} u_{i-1j}^{k+\frac{1}{2}} - \overline{B_{ij}} u_{ij}^{k+\frac{1}{2}} + \overline{C_{ij}} u_{i+1j}^{k+\frac{1}{2}} = \overline{G}_{ij}^{k+\frac{1}{2}ij}, 1 \le i \le N_x - 1, \\ u_{N_x j}^{k+\frac{1}{2}} = \mu(l_x, y_j). \end{cases} \tag{14}$$

Где

$$\overline{G}_{ij}^{k+\frac{1}{2}} = -u_{ij}^{k} - \frac{\tau}{2}(\Lambda_2 u_{ij}^{k} + f(x,y)), \tag{15}$$

$$1 \leq j \leq N_y - 1.$$

В итоге при каждом $1 \leq j \leq N_y - 1$. получили линейную замкнутую систему $N_x + 1$-ого порядка относительно $u_{0j}^{k+\frac{1}{2}}, u_{1j}^{k+\frac{1}{2}}, ..., u_{N_x j}^{k+\frac{1}{2}}$. Матрица системы трёхдиагональная и решается методом прогонки [2].

Прогонки осуществляются вдоль строк. При $j = 0, j = N_y$ решения находятся из (10):

$$\begin{cases} u_{i0}^{k+1} = \mu(x_i, 0), \quad 1 \leq i \leq N_x - 1, \\ u_{iN_y}^{k+1} = \mu(x_i, l_y), \quad 1 \leq i \leq N_x - 1. \end{cases} \tag{16}$$

Система (13) с учётом граничных условий (10) может быть записана в следующем виде:

$$\begin{cases} u_{i0}^{k+\frac{1}{2}} = \mu(x_i, 0), \\ \overline{\overline{A_{ij}}} u_{i-1j}^{k+\frac{1}{2}} - \overline{\overline{B_{ij}}} u_{ij}^{k+\frac{1}{2}} + \overline{\overline{C_{ij}}} u_{i+1j}^{k+\frac{1}{2}} = \overline{\overline{G}}_{ij}^{k+\frac{1}{2}ij}, 1 \leq j \leq N_y - 1, \\ u_{iN_y}^{k+\frac{1}{2}} = \mu(x_i, l_y). \end{cases} \tag{17}$$

Где

$$\overline{\overline{G}}_{ij}^{k+\frac{1}{2}} = -u_{ij}^{k} - \frac{\tau}{2}(\Lambda_1 u_{ij}^{k} + f(x,y)), \tag{18}$$

$$1 \leq i \leq N_x - 1.$$

В итоге при каждом $1 \leq i \leq N_x - 1$. получили линейную замкнутую систему $N_y + 1$-ого порядка относительно $u_{i0}^{k+\frac{1}{2}}, u_{i1}^{k+\frac{1}{2}}, ..., u_{yN_y}^{k+\frac{1}{2}}$. Матрица системы трёхдиагональная и решается методом прогонки [2].

Прогонки осуществляются вдоль столбцов. При $i = 0, i = N_x$ решения находятся из (10):

$$\begin{cases} u_{0j}^{k+1} = \mu(0, y_j), \quad 1 \leq j \leq N_y, \\ u_{N_x j}^{k+1} = \mu(N_x, y_j), \quad 1 \leq j \leq N_y. \end{cases} \tag{19}$$

В нашем случае $(p(x,y) \equiv 1, q(x,y) \equiv 1, h_x = h_y = h, N_x = N_y = N)$

$$\overline{A_{ij}} = \overline{\overline{A_{ij}}} = \frac{\tau}{2h^2}, \overline{B_{ij}} = \overline{\overline{B_{ij}}} = \frac{\tau}{h^2} + 1, \overline{C_{ij}} = \overline{\overline{C_{ij}}} = \frac{\tau}{2h^2}. \tag{20}$$

Итак, рассмотрим алгоритм метода переменных направлений.

1. Из начального условия получаем решение при $k = 0$ во всех точках сетки
$$u_{ij}^0 = u_0(x_i, y_j), 0 \le i \le N_x, 0 \le j \le N_y.$$

2. Полагая $k = 0$, решаем методом прогонки при каждом $1 \le j \le N_y - 1$ систему (14).
Решение при $j = 0$ и $j = N_y$ находится из (16). Тем самым, найдено решение $u_{ij}^{\frac{1}{2}}$ на промежуточном слое $\frac{1}{2}$ во всех точках сетки.

3. Полагая $k = 0$, решаем методом прогонки при каждом $1 \le i \le N_x - 1$ систему (17).
Решение при $i = 0$ и $i = N_x$ находится из (19). Таким образом, найдено решение u_{ij}^1 на слое $k = 1$ во всех точках сетки.

4. Вычислив характеристики полученного решения, увеличиваем номер слоя на единицу $(k = k + 1)$ и повторяем пункты 2 и 3 пока не будет выполнен критерий окончания счёта.

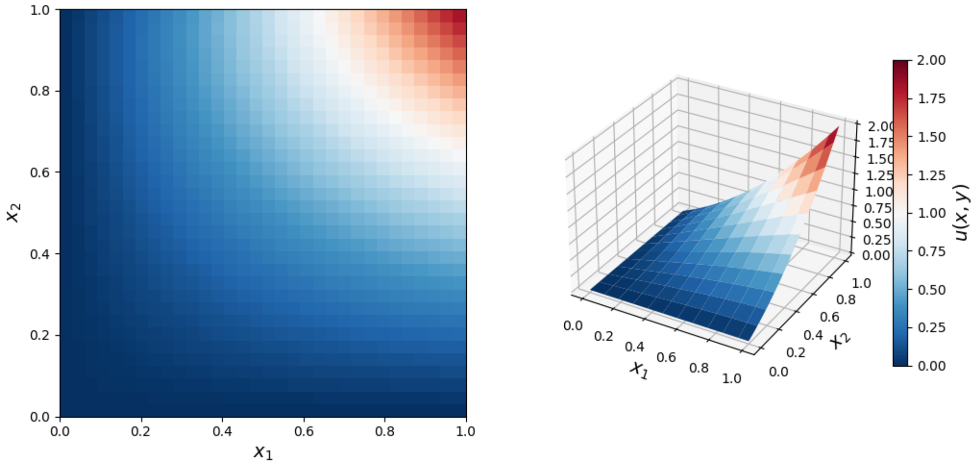
4 Основные результаты

В качестве пробной функции взята функция

$$\mu(x, y) = x^2 y + y^2 x$$

Комментарии: если брать фукнцию содержащую sin, cos погрешность заметно большая, а график аналитического решения "по форме"совпадает. Была взята в качестве пробной функции еще "декартов лист "лемниската"для разнообразия, но в отчете они не будут присутствовать. По программам можно лишь отметить, что схема переменных направлений работает для $n = 32$, и при больших $m \sim 1000$ примерно 7 сек, а матричная прогонка меньше секунды для того же n.

Схема переменных направлений



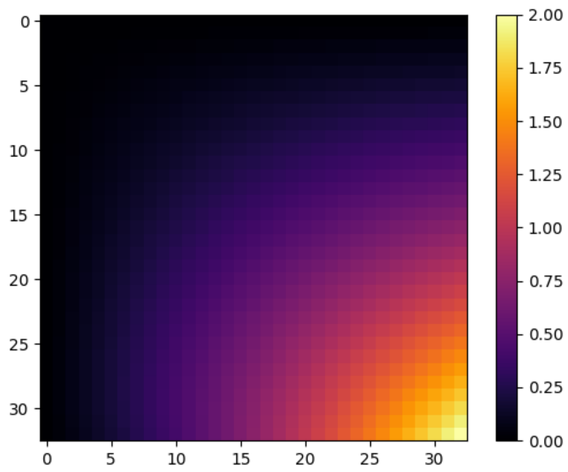
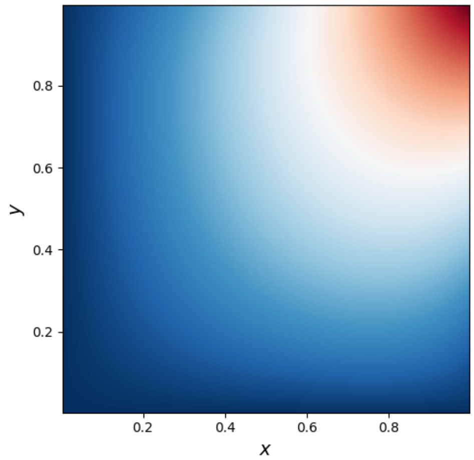
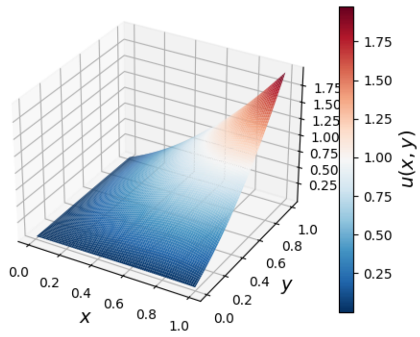


График для $n = 32, ||Ua - U|| = 0.1588.$

Неявная схема. Метод матричной прогонки





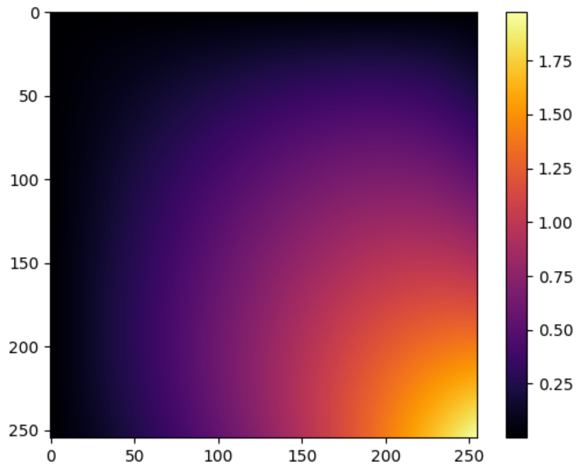


График для $n = 256, ||Ua - U|| = 0.4028.$

Комментарии: при увеличении n норма погрешности уменьшается. Для $n < 256$ норма чуть выше.

5 Листинг программы

5.1 Схема переменных направлений

```
import numpy as np
import sys
import time as t

import matplotlib as mpl
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d.axes3d import Axes3D

def u0(x, y):
        return 0.0

def phi1(x):
        return 0.0

def phi2(x):
        return x + x*x

def phi3(y):
        return 0.0

def phi4(y):
        return y + y*y

def ua(x, y):
        return x*x*y + y*y*x

T = 1.0
L = 1.0
N = 32
N1 = N+1
h = L/N
h2 = h*h
x = np.linspace(0.0, L, N1)
y = np.linspace(0.0, L, N1)

m = 1024
tau = T / m

U0 = np.zeros((N1, N1))
U_old = np.zeros((N1, N1))
U_new = np.zeros((N1, N1))

U0 = np.array([[ua(xi, yi) for yi in y] for xi in x])

U_old = U0

A = np.zeros((N1, N1))
B = np.zeros((N1, N1))
C = np.zeros((N1, N1))
```

```
D = np.zeros((N1, N1))
for i in range(N1):
for j in range(N1):
A[i, j] = -1/(2*h2)
B[i, j] = 1/tau + 1/h2
C[i, j] = -1/(2*h2)

alpha = np.zeros(N1)
beta = np.zeros(N1)

start = t.time()

for k in range(m):
        for i in range(N1):
                U_old[0, i] = phi3(y[i])
                U_new[0, i] = phi3(y[i])
                U_old[N, i] = phi4(y[i])
                U_new[N, i] = phi4(y[i])

        for i in range(1, N):
                for j in range(1, N):
                        D[i, j] = U_old[i, j]/tau + (U_old[i, j-1] - 2*U_old[i, j] + U_old[i, j+1]) / (2*h2)

                for j in range(1, N):
                        alpha[1] = 0
                        beta[1] = phi3(x[j])
                        for i in range(1, N):
                        alpha[i+1] = -C[i, j] / (B[i, j] + A[i, j] * alpha[i])
                        beta[i+1] = (D[i, j] - A[i, j]*beta[i])/(B[i, j] + A[i, j]*alpha[i])
                        U_old[N, j] = phi4(x[j])
                        for i in range(N-1, 0, -1):
                                U_old[i, j] = alpha[i+1] * U_old[i+1, j] + beta[i+1]

        for i in range(N1):
                U_old[0, i] = phi1(x[i])
                U_new[0, i] = phi1(x[i])
                U_old[N, i] = phi2(x[i])
                U_new[N, i] = phi2(x[i])


        for i in range(1, N):
                for j in range(1, N):
                        D[i, j] = U_old[i, j]/tau + (U_old[i, j+1] - 2*U_old[i, j] + U_old[i, j-1]) / (2*h2)

                for i in range(1, N):
                        alpha[1] = 0
                        beta[1] = 0
                        for j in range(1, N):
                        alpha[j+1] = -C[i, j] / (B[i, j] + A[i, j] * alpha[i])
                        beta[j+1] = (D[i, j] - A[i, j]*beta[j])/(B[i, j] + A[i, j]*alpha[j])
                        U_new[i, N] = phi2(y[i])
                        for j in range(N-1, 0, -1):
                                U_new[i, j] = alpha[j+1] * U_new[i, j+1] + beta[j+1]

        U_old[:, :] = U_new[:, :]
```

```
end = t.time()
print('time = ', end-start, sep='')

Ua = np.array([[ua(xi, yi) for yi in y] for xi in x])

av_err = np.max(np.abs(Ua-U_new))
print(f"|Ua-U| = {av_err}")

mx = U_new.max()
mn = U_new.min()
X, Y = np.meshgrid(x, x)
fig = plt.figure(figsize=(12,5.5))
cmap = mpl.cm.get_cmap('RdBu_r')
ax =  fig.add_subplot(1,2,1)
c = ax.pcolor(X, Y, U_new, vmin=mn, vmax=mx, cmap=cmap)
ax.set_xlabel(r"$x_1$", fontsize=14)
ax.set_ylabel(r"$x_2$", fontsize=14)

ax =  fig.add_subplot(1,2,2, projection='3d')
p = ax.plot_surface(X, Y, U_new, vmin=mn, vmax=mx, rstride=3, cstride=3, linewidth=0, cmap=cmap)
ax.set_xlabel(r"$x_1$", fontsize=14)
ax.set_ylabel(r"$x_2$", fontsize=14)

cb = plt.colorbar(p, ax=ax, shrink=0.75)
cb.set_label(r"$u(x, y)$", fontsize=14)

fig1, ax1 = plt.subplots()
cs = plt.imshow(U_new, cmap='inferno')
fig1.colorbar(cs)
plt.show()
```

5.2 Метод матричной прогонки

```
import numpy as np
import math
import sys
import scipy.linalg as sl
import time as t
import matplotlib.pyplot as plt
import matplotlib as mpl
from mpl_toolkits.mplot3d.axes3d import Axes3D


def f(x, y):
        return 5*x + 2*x*y

def left(y):
        return 0.0

def right(y):
        return y+y*y

def bottom(x):
        return 0.0
```

```python
def top(x):
        return x*x+x

def ua(x, y):
        return x*x*y + y*y*x


def sweep(m, C, F):
        Alfa = [np.zeros((m,m))] * m
        beta = np.zeros((m, m))

        Alfa[0] = sl.inv(D)
        beta[0] = np.dot(Alfa[0], F[0])

        for i in range(m-1):
                tmp = sl.inv(D-Alfa[i])
                Alfa[i+1] = tmp
                beta[i+1] = np.dot(tmp, F[i+1] + beta[i])

        X = beta

        for i in range(m-2, -1, -1):
                X[i] = np.dot(Alfa[i], X[i+1]) + beta[i]

        return X

if len(sys.argv) < 2:
        print("N missing")
        exit(1)

n = int(sys.argv[1])
nm1 = n-1
nm2 = n-2
np1 = n+1

h = 1.0/n
h2 = h*h

x = np.linspace(0.0, 1.0, np1)
y = np.linspace(0.0, 1.0, np1)

F = np.zeros((nm1, nm1))
for i in range(nm1):
        F[0, i] += left(y[i+1]) / h2
        F[nm2, i] += right(y[i+1]) / h2
        F[i, 0] += bottom(x[i+1]) / h2
        F[i, nm2] += top(x[i+1]) / h2

        for j in range(nm2):
                F[i, j] += f(x[i+1], y[i+1])

F *= h2

D = np.ones((nm1, nm1))
D = 5 * np.eye(nm1) - sl.triu(sl.tril(D, 1), -1)
```

```
start = t.time()
U = sweep(nm1, D, F).transpose()
end = t.time()
print('time = ', end-start, sep='')

Ua = np.zeros((np1, np1))
for i in range(np1):
for j in range(np1):
Ua[i, j] = ua(x[i], y[j])

#ans = np.max(np.abs(Ua-U))

#print(f"|Ua-U| = {ans}")

x_i = x[1:n]
y_i = y[1:n]

mx = U.max()
mn = U.min()
print(mx, mn)
X, Y = np.meshgrid(x_i, x_i)

fig = plt.figure(figsize=(12, 5.5))
cmap = mpl.cm.get_cmap('RdBu_r')
ax = fig.add_subplot(1,2,1)
c = ax.pcolor(X, Y, U, vmin=mn, vmax=mx, cmap=cmap)
ax.set_xlabel(r"$x$", fontsize=14)
ax.set_ylabel(r"$y$", fontsize=14)

ax = fig.add_subplot(1, 2, 2, projection='3d')
p = ax.plot_surface(X, Y, U, vmin=mn, vmax=mx, rstride=3, cstride=3, linewidth=0, cmap=cmap)
ax.set_xlabel(r"$x$", fontsize=14)
ax.set_ylabel(r"$y$", fontsize=14)

cb = plt.colorbar(p, ax=ax, shrink=0.75)
cb.set_label(r"$u(x, y)$", fontsize=14)

fig1, ax1 = plt.subplots()

cs = plt.imshow(U, cmap='inferno')
fig1.colorbar(cs)
plt.show()
```

6 Список литературы

1. **Пакулина А.Н.** Практикум по методам вычислений. Часть 2. СПб., СПбГУ, 2019. – 113 с.

2. **Самарский А.А, Гулин А.В.** Численные методы математической физики. Наука, 2000. - 310 с.