Московский авиационный институт

(национальный исследовательский университет)

Институт информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №7

по курсу “Численные методы”

Студент: Пясковский Н.Л.

Группа: М8О-409Б-19

Руководитель: Пивоваров Д.Е.

Оценка: _______________

Москва 2022

1) 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 $U(x, y)$. Исследовать зависимость погрешности от сеточных параметров h_x, h_y.

2) Вариант

8.

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = -2\frac{\partial u}{\partial x} - 3u,$$

$$u(0, y) = \cos y,$$

$$u(\frac{\pi}{2}, y) = 0,$$

$$u(x,0) = \exp(-x)\cos x,$$

$$u(x, \frac{\pi}{2}) = 0.$$

Аналитическое решение: $U(x, y) = \exp(-x)\cos x \cos y$.

3) Теория

Уравнение Пуассона является классическим примером уравнения эллиптического типа

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

или уравнение Лапласа при $f(x,y) \equiv 0$.

Здесь функция $u(x,y)$ имеет различный физический смысл, а именно: стационарное, независящее от времени, распределение температуры, скорость потенциального (безвихревого) течения идеальной (без трения и теплопроводности) жидкости, распределение напряженностей электрического и магнитного полей, потенциала в силовом поле тяготения и т.п.

Если на границе Γ расчетной области $\overline{\Omega} = \Omega + \Gamma$ задана искомая функция, то соответствующая *первая краевая* задача для уравнения Лапласа или Пуассона называется *задачей Дирихле*

$$\begin{cases} \dfrac{\partial^2 u}{\partial x^2} + \dfrac{\partial^2 u}{\partial y^2} = f(x, y), \quad (x, y) \in \Omega; & (5.42) \\ u(x, y)\big|_{\Gamma} = \varphi(x, y), \quad (x, y) \in \Gamma. & (5.43) \end{cases}$$

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

$$\begin{cases} \dfrac{\partial^2 u}{\partial x^2} + \dfrac{\partial^2 u}{\partial y^2} = f(x, y), \quad (x, y) \in \Omega; & (5.44) \\ \left.\dfrac{\partial u(x, y)}{\partial n}\right|_{\Gamma} = \varphi(x, y), \quad (x, y) \in \Gamma. & (5.45) \end{cases}$$

При этом n – направление внешней к границе Г нормали.

Более приемлемой является координатная форма краевого условия (5.45)

$$\frac{\partial u}{\partial x}\cos(\hat{n, i}) + \frac{\partial u}{\partial y}\cos(\hat{n, j}) = \varphi(x, y),$$

где $\cos(\hat{n, i})$, $\cos(\hat{n, j})$ – направляющие косинусы внешнего вектора единичной нормали к границе Г, i и j орты базисных векторов.

Наконец третья краевая задача для уравнения Пуассона (Лапласа) имеет вид

$$\begin{cases} \dfrac{\partial^2 u}{\partial x^2} + \dfrac{\partial^2 u}{\partial y^2} = f(x, y), \quad (x, y) \in \Omega; \\ \left.\dfrac{\partial u(x, y)}{\partial n}\right|_{\Gamma} + \alpha \left.u\right|_{\Gamma} = \varphi(x, y), \quad (x, y) \in \Gamma. \end{cases}$$

Рассмотрим краевую задачу для уравнений Лапласа или Пуассона (5.42), (5.43) в прямоугольнике $x \in [0, l_1], y \in [0, l_2]$, на который наложим сетку

$$\omega_{h_1, h_2} = \{x_i = ih_1, i = \overline{0, N_1}; \quad y_j = jh_2, j = \overline{0, N_2}\}. \quad (5.46)$$

На этой сетке аппроксимируем дифференциальную задачу во внутренних узлах с помощью отношения конечных разностей по следующей схеме (вводится сеточная функция u_{ij}, $i = \overline{0, N_1}$, $j = \overline{0, N_2}$;):

$$\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h_1^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h_2^2} + O\left(h_1^2 + h_2^2\right) = f(x_i, y_j), \quad (5.47)$$
$$i = \overline{1, N_1 - 1}, \quad j = \overline{1, N_2 - 1}$$

которая на шаблоне (5.5) имеет второй порядок по переменным x и y, поскольку шаблон центрально симметричен.

СЛАУ имеет 5-диагональный вид и решается с помощью методов ЛА

Рассмотрим разностно-итерационный метод Либмана численного решения задачи Дирихле (5.42), (5.43). Для простоты изложения этого метода примем $h_1 = h_2 = h$, тогда из схемы (5.47) получим (k-номер итерации)

$$u_{i,j}^{(k+1)} = \frac{1}{4}[u_{i+1,j}^{(k)} + u_{i-1,j}^{(k)} + u_{i,j-1}^{(k)} + u_{i,j+1}^{(k)} - h^2 \cdot f_{i,j}], \quad f_{i,j} = f(x_i, y_j),$$
$$i = \overline{1, N_1 - 1}, \quad j = \overline{1, N_2 - 1}. \tag{5.48}$$

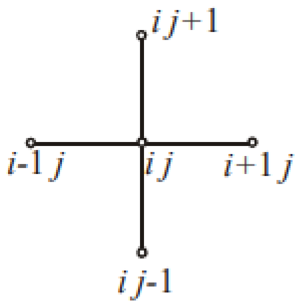


Рис. 5.5. Центрально-симметричный шаблон для уравнения Лапласа

Видно, что для использования полученной формулы необходимо найти значения $u_{i,j}^{(0)}$ на нулевой итерации. Для этого на каждой координатной линии используем линейные интерполяции граничных значений. Полученные значения подставляем в формулу и получаем распределение $u_{i,j}^{(1)}$. Это распределение опять же подставляется в формулу, после чего получаем распределение $u_{i,j}^{(2)}$ и тд.
Процесс Либмана прекращается при условии:

$$\|u^{(k+1)} - u^{(k)}\| \le \varepsilon, \quad \|u^{(k)}\| = \max_{i,j} |u_{i,j}^{(k)}|$$

Рассмотрим метод Зейделя, являющийся улучшенной версией метода Либмана. Ращличие состоит в том, что в формулу подставляются не только значения с предыдущей итерации, но и уже полученные значения с текущей.

Таким образом мы получаем формулу:

$$u_{i,j}^{(k+1)} = \frac{1}{4}[u_{i+1,j}^{(k)} + u_{i-1,j}^{(k+1)} + u_{i,j+1}^{(k)} + u_{i,j-1}^{(k+1)} - h^2 f_{i,j}]$$

Стоит отметить, что скорость сходимости метода Здейделя выше чем у метода Либмана.
Метод верхней релаксации отличается наличием свободного параметра с, который определяет величину смещения каждой компоненты в зависимости от ее положения на предыдущем шаге:

$$u_{i,j}^{(k+1)} = (1 - \mathrm{c})u_{i,j}^{(k)} + \mathrm{c}\frac{1}{4}[u_{i+1,j}^{(k)} + u_{i-1,j}^{(k+1)} + u_{i,j+1}^{(k)} + u_{i,j-1}^{(k+1)} - h^2 f_{i,j}]$$

Скорость сходимости метода верхней релаксации зависит от выбора параметра с, и при верном выборе она может быть значительно быстрее скоростей сходимости остальных методов.

4) Программа

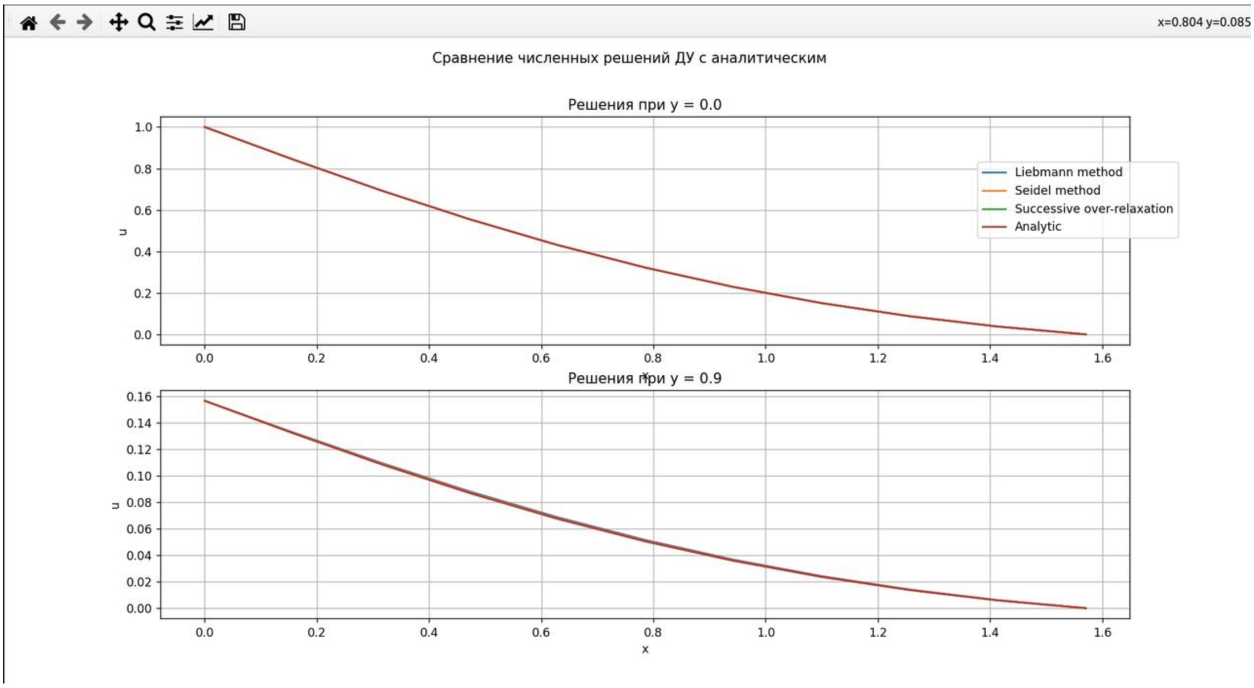
```
import copy

import numpy as np
from matplotlib import pyplot as plt
import warnings

Nx = 10  # количество отрезков x
Ny = 10  # количество отрезков y
lx = np.pi / 2  # правая граница координат x
ly = np.pi / 2  # правая граница координат y
hx = lx / Nx  # шаг сетки по x
hy = ly / Ny  # шаг сетки по y
bx = 2  #
by = 0  #
c = 3  #
eps = 0.001  # точность вычислений
max_iterations = 1000  # предельное количество итераций


# аналитическое решение
def analytic_solution(x, y):
   return np.exp(-x) * np.cos(x) * np.cos(y)


# условия
def phi1(y):
   return np.cos(y)


def phi2(y):
   return 0


def phi3(x):
   return np.exp(-x) * np.cos(x)


def phi4(x):
   return 0


# метод простых итераций (метод Либмана)
def liebmann_method():
   u = np.zeros((Nx + 1, Ny + 1))

   u[0, 0] = phi1(0)
   u[-1, -1] = phi2(-hy)

   for i in range(1, Nx):
      u[i, 0] = phi3(i * hx)
      u[i, -1] = phi4(i * hx)
```

```
    for j in range(1, Ny):
        u[0, j] = phi1(j * hy)
        u[-1, j] = phi2(j * hy)
        for i in range(1, Nx):
            u[i, j] = u[0, j] + (u[-1, j] - u[0, j]) / lx * i * hx

    k = 0
    while True:
        k += 1
        if k > max_iterations:
            warnings.warn("Max number of iterations exceeded!")
            break

        u_prev = copy.deepcopy(u)

        for j in range(1, Ny):
            for i in range(1, Nx):
                u[i, j] = (-(u_prev[i + 1, j] + u_prev[i - 1, j]) - hx ** 2 * (u_prev[i, j + 1] + u_prev[i, j - 1]) / (
                            hy ** 2) - bx * hx * (u_prev[i + 1, j] - u_prev[i - 1, j]) / 2 - by * hx ** 2 * (
                                   u_prev[i, j + 1] - u_prev[i, j - 1]) / (2 * hy)) / (
                                  c * hx ** 2 - 2 * (hy * hy + 1 * hx ** 2) / (hy ** 2))

        norm = np.linalg.norm(u - u_prev, np.inf)
        if norm <= eps:
            break

    print("liebmann_method: k =", k)
    return u


# метод простых итераций с верхней релаксацией, omega - параметр релаксации
def sor_method(omega):
    u = np.zeros((Nx + 1, Ny + 1))

    u[0, 0] = phi1(0)
    u[-1, -1] = phi2(-hy)

    for i in range(1, Nx):
        u[i, 0] = phi3(i * hx)
        u[i, -1] = phi4(i * hx)

    for j in range(1, Ny):
        u[0, j] = phi1(j * hy)
        u[-1, j] = phi2(j * hy)
        for i in range(1, Nx):
            u[i, j] = u[0, j] + (u[-1, j] - u[0, j]) / lx * i * hx

    k = 0
    while True:
        k = k + 1
        if k > max_iterations:
            warnings.warn("Max number of iterations exceeded!")
            break
```

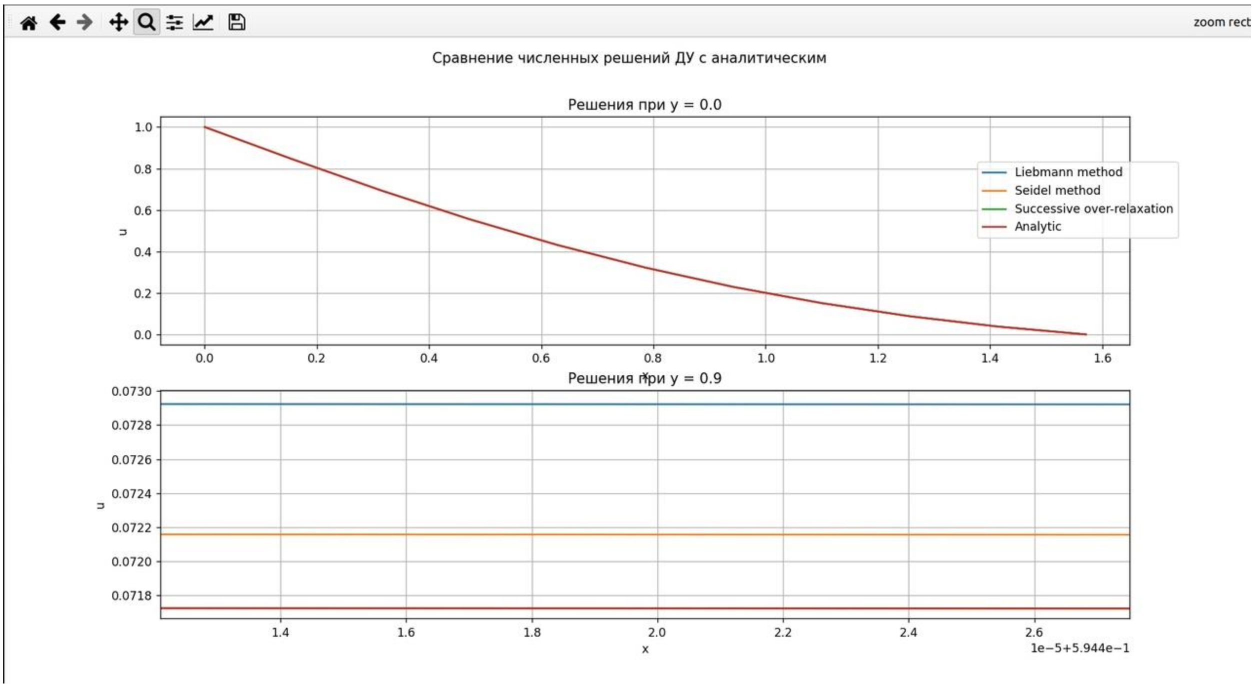
```
        u_prev = copy.deepcopy(u)

        for j in range(1, Ny):
            for i in range(1, Nx):
                u[i, j] = ((-(u_prev[i + 1, j] + u[i - 1, j]) - 1 * hx ** 2 * (u_prev[i, j + 1] + u[i, j - 1]) / (
                            hy ** 2) - bx * hx * (u_prev[i + 1, j] - u[i - 1, j]) / 2 - by * hx ** 2 * (
                                    u_prev[i, j + 1] - u[i, j - 1]) / (2 * hy)) / (
                                   c * hx ** 2 - 2 * (hy ** 2 + 1 * hx ** 2) / (hy ** 2))) * omega + (1 - omega) * \
                          u_prev[i, j]

        norm = np.linalg.norm(u - u_prev, np.inf)
        if norm <= eps:
            break

    if omega == 1:
        print("seildel_method: k =", k)
    else:
        print("sor_method: k =", k)

    return u


# вывод решения в виде графиков
def show_result(y_axis, x_axis, u1, u2, u3):
    fig, ax = plt.subplots(2)
    fig.suptitle('Сравнение численных решений ДУ с аналитическим')
    fig.set_figheight(15)
    fig.set_figwidth(16)
    y = 0
    for i in range(2):
        ax[i].plot(x_axis, u1[:, y], label='Liebmann method')
        ax[i].plot(x_axis, u2[:, y], label='Seidel method')
        ax[i].plot(x_axis, u3[:, y], label='Successive over-relaxation')
        ax[i].plot(x_axis, [analytic_solution(x, y_axis[y]) for x in x_axis], label='Analytic')
        ax[i].grid(True)
        ax[i].set_xlabel('x')
        ax[i].set_ylabel('u')
        ax[i].set_title(f'Решения при y = {y / Ny}')
        y += Ny - 1

    plt.legend(bbox_to_anchor=(1.05, 2), loc='upper right', borderaxespad=0)
    plt.show()

    fig = plt.figure(num=1, figsize=(19, 12), clear=True)
    ax = fig.add_subplot(1, 1, 1, projection='3d')
    fig.suptitle('Аналитическое решение')
    xgrid, ygrid = np.meshgrid(x_axis, y_axis)
    ax.plot_surface(xgrid, ygrid, analytic_solution(xgrid, ygrid))
    ax.set(xlabel='x', ylabel='y', zlabel='u')
    fig.tight_layout()
    plt.show()


# вывод изменения ошибки со временем
```

```
def show_inaccuracy(y_axis, x_axis, u):
    inaccuracy = np.zeros(Nx + 1)
    for i in range(Nx + 1):
        inaccuracy[i] = np.max(np.abs(u[i] - np.array([analytic_solution(x_axis[i], y) for y in y_axis])))

    plt.figure(figsize=(14, 8))
    plt.plot(x_axis[1:], inaccuracy[1:], 'violet', label='Ошибка')
    plt.legend(bbox_to_anchor=(1.05, 1), loc='upper right', borderaxespad=0.)
    plt.title('График изменения ошибки')
    plt.xlabel('y')
    plt.ylabel('error')
    plt.grid(True)
    plt.show()


#
def main():
    u1 = liebmann_method()
    u2 = sor_method(1)  # метод Зейделя при omega = 1
    u3 = sor_method(1.5)

    y_axis = np.zeros(Ny + 1)
    for j in range(Ny + 1):
        y_axis[j] = j * hy

    x_axis = np.zeros(Nx + 1)
    for i in range(Nx + 1):
        x_axis[i] = i * hx

    show_result(y_axis, x_axis, u1, u2, u3)

    show_inaccuracy(y_axis, x_axis, u1)


if __name__ == '__main__':
    main()
```

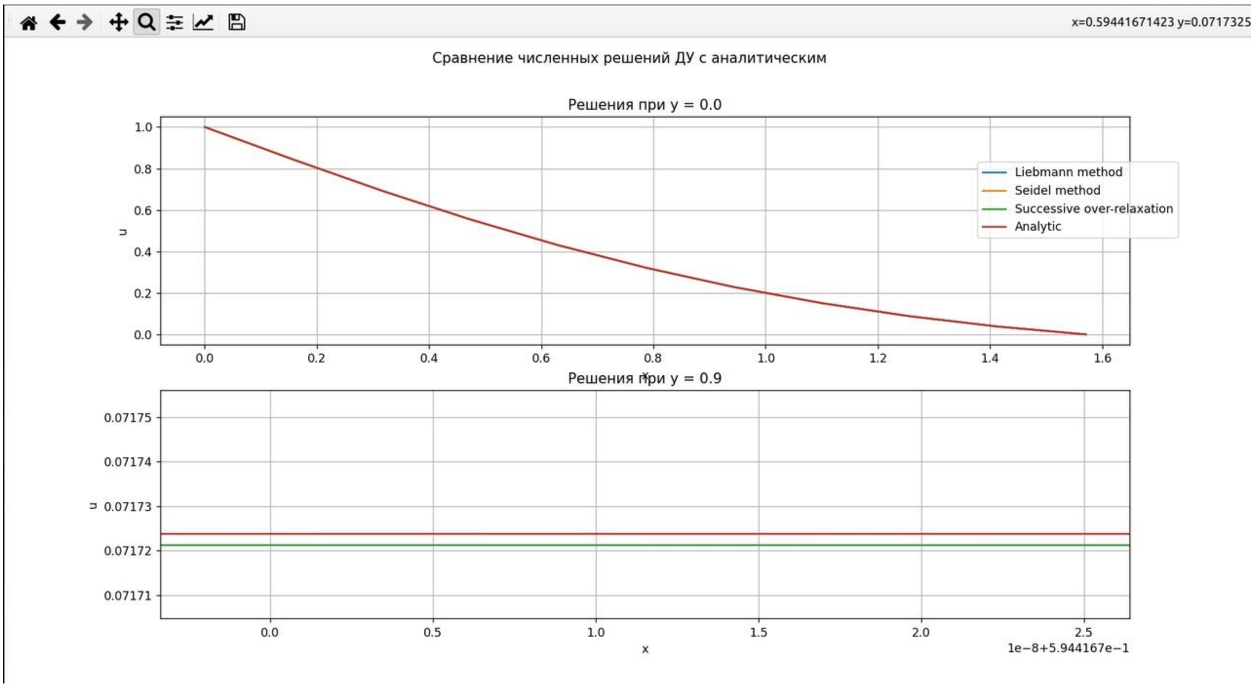
5) Результат



x=0.804 y=0.085
Сравнение численных решений ДУ с аналитическим
Решения при y = 0.0
Liebmann method
Seidel method
Successive over-relaxation
Analytic
u
x
Решения при y = 0.9
u
x



zoom rect
Сравнение численных решений ДУ с аналитическим
Решения при y = 0.0
Liebmann method
Seidel method
Successive over-relaxation
Analytic
u
x
Решения при y = 0.9
u
x
1e−5+5.944e−1



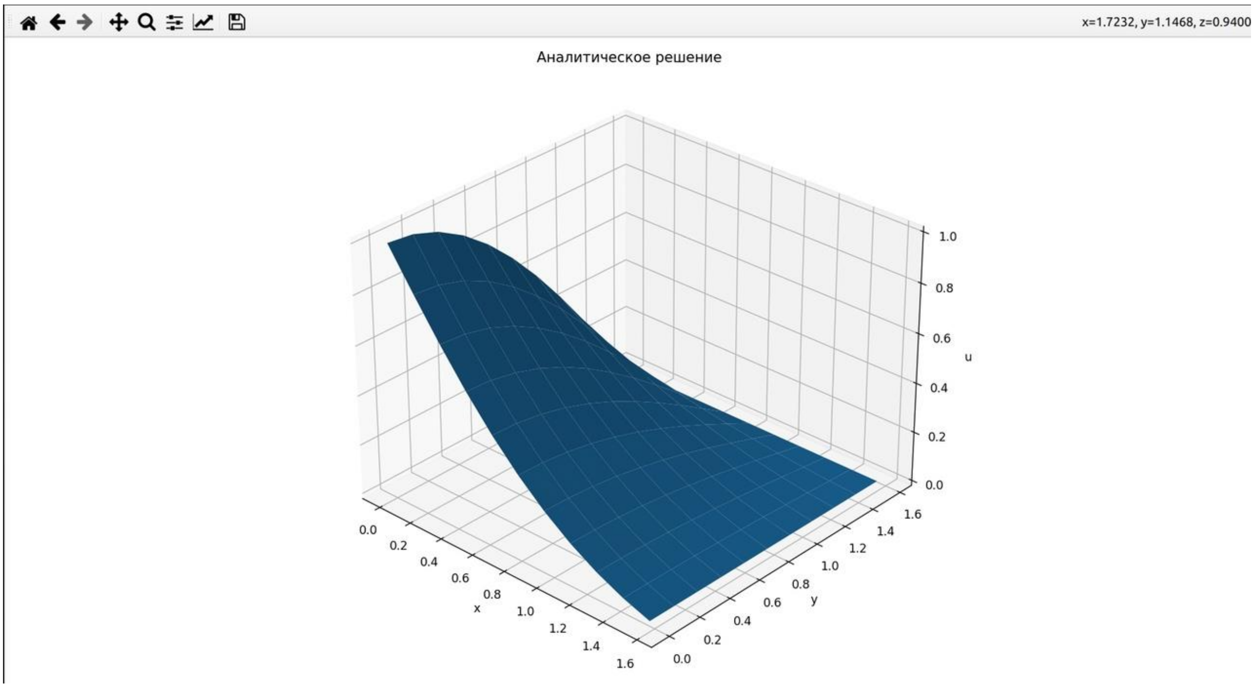


График изменения ошибки во времени для метода Либмана:

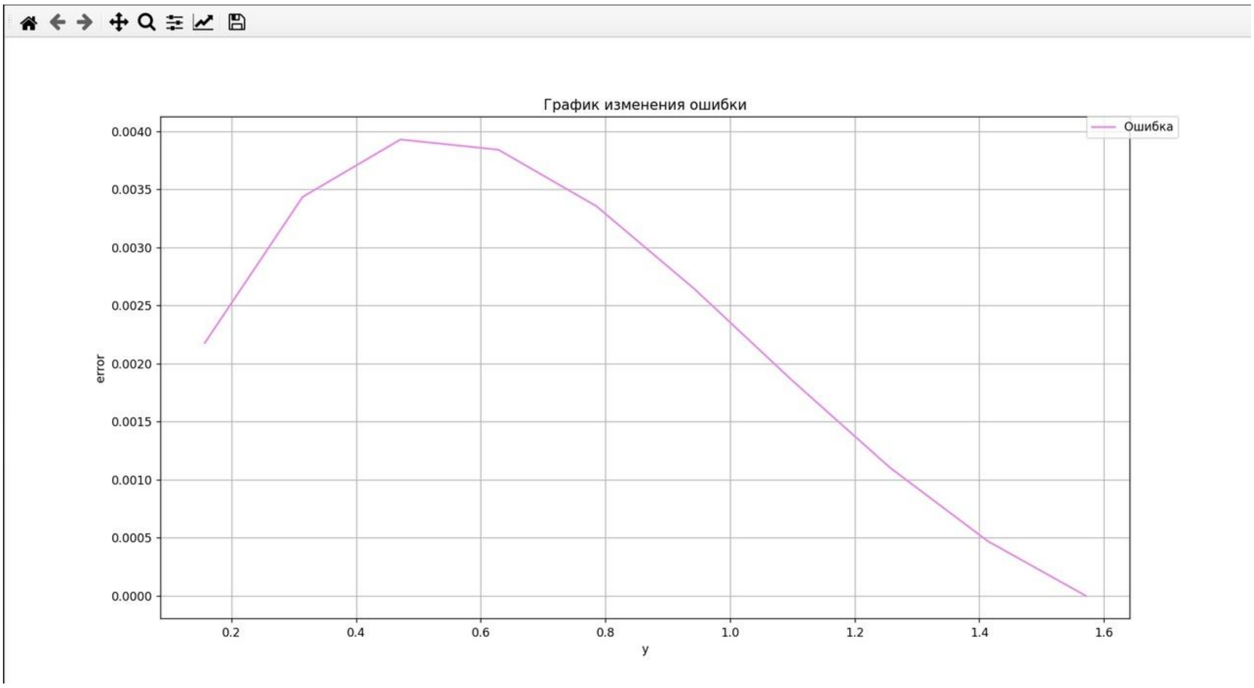


График изменения ошибки
Ошибка
error
y
0.0040
0.0035
0.0030
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6