Курсовая работа по вычислительной математике

Леднева Александра, Б01-006

1 Постановка задачи

Для численного решения краевой задачи ОДУ $\frac{\partial^2 u}{\partial x^2} = f(u),\ u(0) = U_1,\ u(L) = U_2$ воспользоваться тремя вариантами трехточечной прогонки (предварительно получив прогоночные соотношения):

а) прямая прогонка (слева направо);

б) обратная прогонка (справа налево);

в) встречные прогонки.

Положить:

$f(u) = \exp \alpha u;\ f(u) = sin(\omega u),\ U_1 = 0,\ U_2 = 1,\ L = 1.$

2 Вывод прогоночных соотношений

Для применения метода прогонки к данной краевой задаче необходимо привести её к виду, удобному для решения. Для этого можно воспользоваться методом конечных разностей.

Разобьём отрезок $[0, L]$ на n равных частей длиной $h = L/n$, обозначим $x_i = ih$ для $i = 0, 1, \ldots, n$. Обозначим $u_i = u(x_i)$ для $i = 0, 1, \ldots, n$ — значения функции u в узлах сетки.

Аппроксимируем вторую производную $u''(x)$ при помощи центральной разности:

$$u''(x_i) = \frac{u_{i+1} - 2u_i + u_{i-1}}{h^2}.$$

Подставляя эту аппроксимацию в исходное уравнение и учитывая, что $f(u_i) = f_i$, получим систему уравнений:

$$\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} = f_i, \qquad i = 1, 2, \ldots, n-1,$$

с граничными условиями $u_0 = U_1$ и $u_n = U_2$. Эту систему можно записать в матричном виде:

$$\frac{1}{h^2}\begin{pmatrix} -2 & 1 & 0 & 0 & \cdots & 0 \\ 1 & -2 & 1 & 0 & \cdots & 0 \\ 0 & 1 & -2 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & -2 \end{pmatrix}\begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{n-1} \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ \vdots \\ f_{n-1} \end{pmatrix} - \begin{pmatrix} U_1/h^2 \\ 0 \\ 0 \\ \vdots \\ U_2/h^2 \end{pmatrix}.$$

Теперь приступим к прогонке. Для решения задачи методом прогонки, сначала необходимо получить прогоночные соотношения для каждого варианта.

а) Прямая прогонка (слева направо)

Вычислим прогоночные коэффициенты α_i и β_i с помощью следующих формул:

$\alpha_1 = -\frac{c_1}{b_1}$

$\beta_1 = \frac{d_1}{b_1}$

$\alpha_i = -\frac{c_i}{b_i + a_i\alpha_{i-1}}$

$\beta_i = \frac{d_i - a_i\beta_{i-1}}{b_i + a_i\alpha_{i-1}}$

где a_i, b_i, c_i - диагональные элементы матрицы системы линейных уравнений, d_i - правая часть уравнения, α_i и β_i - прогоночные коэффициенты.

Найдем решение системы линейных уравнений, используя следующие формулы:

$u_{n-1} = \beta_{n-1}$

$u_i = \alpha_i u_{i+1} + \beta_i$

где u_i - неизвестное значение i-го уравнения системы линейных уравнений.

В данном случае для $f = \exp \alpha u$ или $f = sin(\omega u)$:

$$\begin{aligned} a_i &= 1 \\ b_i &= -2 \\ c_i &= 1 \\ d_i &= h^2 f_i,\ i = 2, \ldots, n-2 \\ d_1 &= h^2 f_1 - U_1,\ d_{n-1} = h^2 f_{n-1} - U_2 \end{aligned}$$

б) Обратная прогонка (справа налево)

Вычислим прогоночные коэффициенты γ_i и δ_i с помощью следующих формул:

$\gamma_{n-1} = -\frac{a_{n-1}}{b_{n-1}}$

$\delta_{n-1} = \frac{d_{n-1}}{b_{n-1}}$

$\gamma_i = -\frac{a_i}{c_i\gamma_{i+1} + b_i}$

$\delta_i = \frac{d_i - c_i\delta_{i+1}}{c_i\gamma_{i+1} + b_i}$

где a_i, b_i, c_i - диагональные элементы матрицы системы линейных уравнений, g_i - правая часть уравнения, γ_i и δ_i - прогоночные коэффициенты.

Найдем решение системы линейных уравнений, используя следующие формулы:

$u_1 = \delta_1$

$u_i = \gamma_i u_{i-1} + \delta_i$

где u_i - неизвестное значение i-го уравнения системы линейных уравнений.

В данном случае для $f = \exp \alpha u$ и $f = sin(\omega u)$:

$$a_i = 1$$

$$b_i = -2$$
$$c_i = 1$$
$$d_i = h^2 f_i,\ i = 2, \ldots, n-2$$
$$d_1 = h^2 f_1 - U_1,\ d_{n-1} = h^2 f_{n-1} - U_2$$

в) Встречные прогонки

Алгоритм метода встречной прогонки состоит из следующих шагов:

1. Разделить систему линейных уравнений на две половины, каждая из которых имеет трехдиагональную матрицу.

2. Применить метод прямой прогонки к первой половине системы линейных уравнений, начиная с левого конца, найти прогоночные коэффициенты α_i и β_i.

3. Применить метод обратной прогонки ко второй половине системы линейных уравнений, начиная с правого конца, найти прогоночные коэффициенты γ_i и δ_i.

4. Найти решение системы линейных уравнений, объединив результаты шагов 2 и 3, используя следующие формулы:

$u_{n/2+1} = \frac{\delta_{n/2+1} + \gamma_{n/2+1}\beta_{n/2+1}}{1 - \gamma_{n/2+1}\alpha_{n/2+1}}$

$u_i = \alpha_i u_{i+1} + \beta_i$ для $i = n/2, ..., 1$

$u_{n/2} = \frac{\beta_{n/2+1} + \alpha_{n/2+1}\delta_{n/2+1}}{1 - \gamma_{n/2+1}\alpha_{n/2+1}}$

$u_i = \gamma_i u_{i-1} + \delta_i$ для $i = n/2+1, ..., n-1$

где u_i - неизвестное значение i-го уравнения системы линейных уравнений.

Так как f_i зависит от значений u_i, то при первой итерации выбираем произвольное значение u_i, чтобы посчитать d_i, а затем повторяем прогонку, используя посчитанные ранее значения u_i. С каждой подобной итерацией получаем все более точное значение u_i, до тех пор, пока разница между u_i, полученным в результате предыдущей прогонки, и u_i, полученным в результате текущей прогонки, не станет меньше установленного ε.

3 Реализация

Все методы были реализованы на языке Python.

3.1 а) Прямая прогонка (слева направо)

```
import numpy as np
import matplotlib.pyplot as plt
import math

alpha0 = 3
omega0 = 3
U1 = 0
U2 = 1
L = 1
n = 1000
eps = 0.0000001
h = L / n

def f1(u0):
    return math.exp(alpha0 * u0)

def f2(u0):
    return math.sin(omega0 * u0)

A = np.zeros(n + 1)
B = np.zeros(n + 1)
C = np.zeros(n + 1)
D = np.zeros(n + 1)

alpha = np.zeros(n + 1)
beta = np.zeros(n + 1)
u = np.zeros(n + 1)
u[0] = U1
u[n] = U2
y = np.ones(n + 1)
while (max(abs(u - y)) > eps):
    for i in range(1, n):
        if (i == 1):#          f2(u[i])          f(u) = sin(omega * u)
            D[i] = f1(u[i]) * h**2 - U1
            A[i] = 0
            B[i] = -2
            C[i] = 1
        elif (i == n - 1):
            D[i] = f1(u[i]) * h**2 - U2
            A[i] = 1
            B[i] = -2
            C[i] = 0
        else:
            D[i] = f1(u[i]) * h**2
            A[i] = 1
            B[i] = -2
            C[i] = 1
        if (i > 1):
            alpha[i] = -C[i]/(A[i] * alpha[i - 1] + B[i])
            beta[i] = (D[i] - A[i] * beta[i - 1])/(A[i] * alpha[i - 1] + B[i])
        elif (i == 1):
            alpha[i] = -C[i]/B[i]
            beta[i] = D[i]/B[i]
    y = u.copy()
    u[n - 1] = beta[n - 1]
```

```
    for i in range(n - 2, 0, -1):
        u[i] = alpha[i] * u[i + 1] + beta[i]
        u[i] = (u[i] + y[i]) / 2

x = np.linspace(0, L, n + 1)
plt.plot(x, u)
plt.xlabel('x')
plt.ylabel('u')
plt.show()
```

3.2 б) Обратная прогонка (справа налево)

```
import numpy as np
import matplotlib.pyplot as plt
import math

alpha0 = 3
omega0 = 3
U1 = 0
U2 = 1
L = 1
n = 1000
eps = 0.0000001
h = L / n

def f1(u0):
    return math.exp(alpha0 * u0)

def f2(u0):
    return math.sin(omega0 * u0)

A = np.zeros(n + 1)
B = np.zeros(n + 1)
C = np.zeros(n + 1)
D = np.zeros(n + 1)

gamma = np.zeros(n + 1)
delta = np.zeros(n + 1)
u = np.zeros(n + 1)
u[0] = U1
u[n] = U2
y = np.ones(n + 1)
while (max(abs(u - y)) > eps):
    for i in range(n - 1, 0, -1):
        if (i == 1):
            D[i] = f1(u[i]) * h**2 - U1
            A[i] = 0
            B[i] = -2
            C[i] = 1
        elif (i == n - 1):
            D[i] = f1(u[i]) * h**2 - U2
            A[i] = 1
            B[i] = -2
            C[i] = 0
        else:
            D[i] = f1(u[i]) * h**2
            A[i] = 1
            B[i] = -2
            C[i] = 1
        if (i < n - 1):
            gamma[i] = -A[i]/(C[i] * gamma[i + 1] + B[i])
```

```
            delta[i] = (D[i] - C[i] * delta[i + 1])/(C[i] * gamma[i + 1] + B[i])
        elif (i == n - 1):
            gamma[i] = -A[i]/B[i]
            delta[i] = D[i]/B[i]
    y = u.copy()
    u[1] = delta[1]
    for i in range(2, n):
        u[i] = gamma[i] * u[i - 1] + delta[i]
        u[i] = (u[i] + y[i]) / 2

x = np.linspace(0, L, n + 1)
plt.plot(x, u)
plt.xlabel('x')
plt.ylabel('u')
plt.show()
```

3.3 в) Встречные прогонки

```
import numpy as np
import matplotlib.pyplot as plt
import math

alpha0 = 3
omega0 = 3
U1 = 0
U2 = 1
L = 1
n = 1000
eps = 0.0000001
h = L / n

def f1(u0):
    return math.exp(alpha0 * u0)

def f2(u0):
    return math.sin(omega0 * u0)

A = np.zeros(n + 1)
B = np.zeros(n + 1)
C = np.zeros(n + 1)
D = np.zeros(n + 1)

gamma = np.zeros(n + 1)
delta = np.zeros(n + 1)
u = np.zeros(n + 1)
u[0] = U1
u[n] = U2
y = np.ones(n + 1)
while (max(abs(u - y)) > eps):
    for i in range(n - 1, n // 2, -1):
        if (i == n - 1):
            D[i] = f1(u[i]) * h**2 - U2
            A[i] = 1
            B[i] = -2
            C[i] = 0
        else:
            D[i] = f1(u[i]) * h**2
            A[i] = 1
            B[i] = -2
            C[i] = 1
        if (i < n - 1):
```

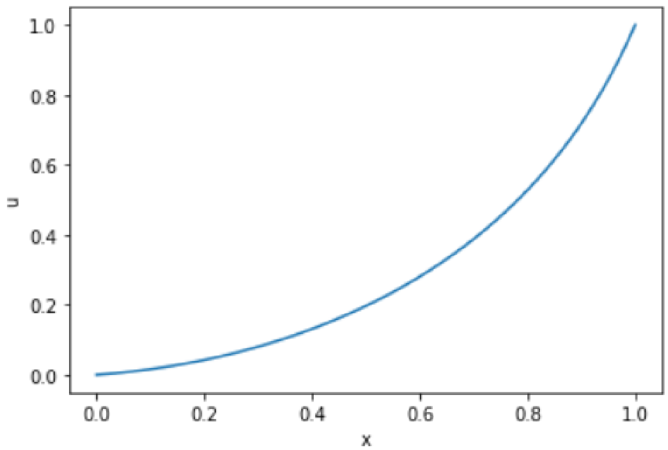
```
                gamma[i] = -A[i]/(C[i] * gamma[i + 1] + B[i])
                delta[i] = (D[i] - C[i] * delta[i + 1])/(C[i] * gamma[i + 1] + B[i])
            elif (i == n - 1):
                gamma[i] = -A[i]/B[i]
                delta[i] = D[i]/B[i]
        for i in range(1, n // 2 + 1):
            if (i == 1):
                D[i] = f1(u[i]) * h**2 - U1
                A[i] = 0
                B[i] = -2
                C[i] = 1
            else:
                D[i] = f1(u[i]) * h**2
                A[i] = 1
                B[i] = -2
                C[i] = 1
            if (i > 1):
                alpha[i] = -C[i]/(A[i] * alpha[i - 1] + B[i])
                beta[i] = (D[i] - A[i] * beta[i - 1])/(A[i] * alpha[i - 1] + B[i])
            elif (i == 1):
                alpha[i] = -C[i]/B[i]
                beta[i] = D[i]/B[i]
        y = u.copy()
        u[n // 2 + 1] = (delta[n // 2 + 1] + gamma[n // 2 + 1] * beta[n // 2 + 1])/
        /(1 - gamma[n // 2 + 1] * alpha[n // 2 + 1])
        u[n // 2] = (beta[n // 2 + 1] + alpha[n // 2 + 1] * delta[n // 2 + 1])/
        /(1 - gamma[n // 2 + 1] * alpha[n // 2 + 1])
        for i in range(n // 2, 0, -1):
            u[i] = alpha[i] * u[i + 1] + beta[i]
            u[i] = (u[i] + y[i]) / 2
        for i in range(n // 2 + 1, n):
            u[i] = gamma[i] * u[i - 1] + delta[i]
            u[i] = (u[i] + y[i]) / 2

x = np.linspace(0, L, n + 1)
plt.plot(x, u)
plt.xlabel('x')
plt.ylabel('u')
plt.show()
```

4 Результаты

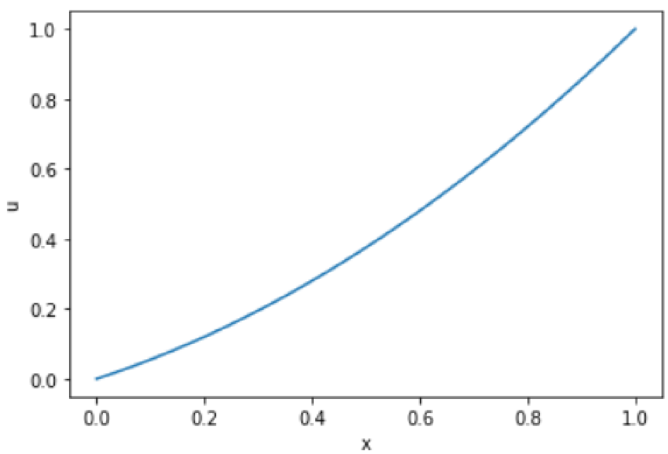
Здесь приведены графики зависимостей $u(x)$ при разных функциях в правых частях исходного уравнения и при разных параметрах α и ω:

1) $f(u) = \exp(\alpha u)$, $\alpha = 3$



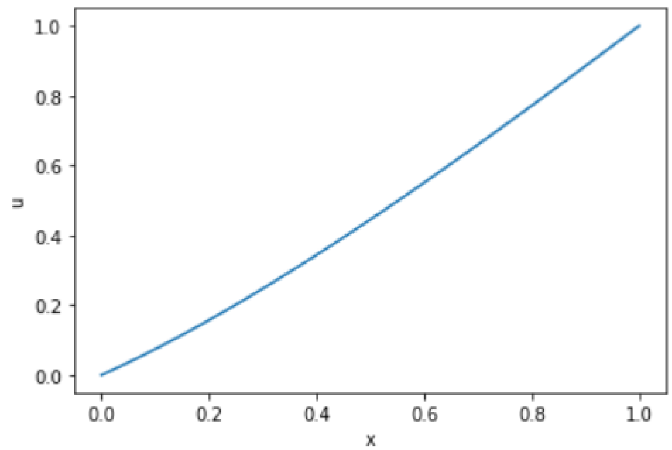
Время работы методов: прямая прогонка — 9.3 секунд, обратная прогонка — 9.8 секунд, встречная прогонка — 7.2 секунд.

2) $f(u) = \exp(\alpha u)$, $\alpha = 0$



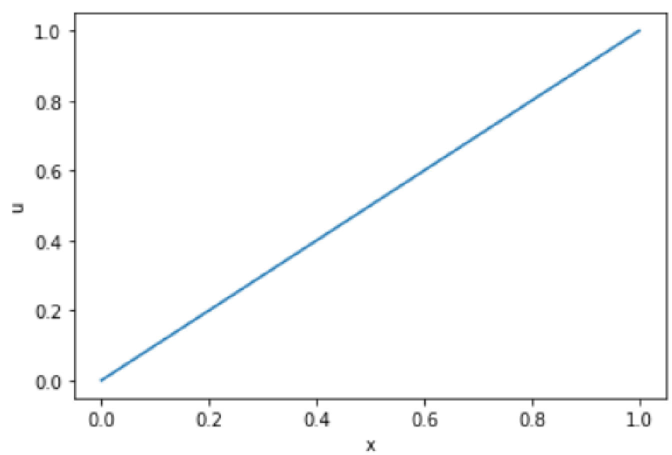
Время работы методов: прямая прогонка — 5.8 секунд, обратная прогонка — 5.5 секунд, встречная прогонка — 2.9 секунд.

3) $f(u) = \exp(\alpha u), \ \alpha = -2$



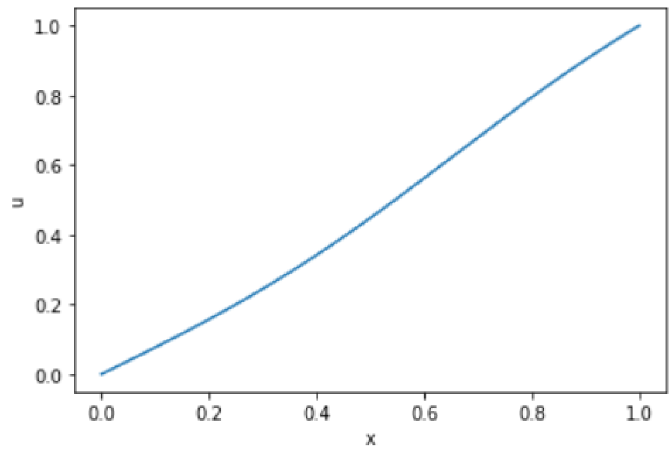
Время работы методов: прямая прогонка — 10.8, обратная прогонка — 8.5 секунд, встречная прогонка — 5.7 секунд.

4) $f(u) = sin(\omega u), \ \omega = 0$



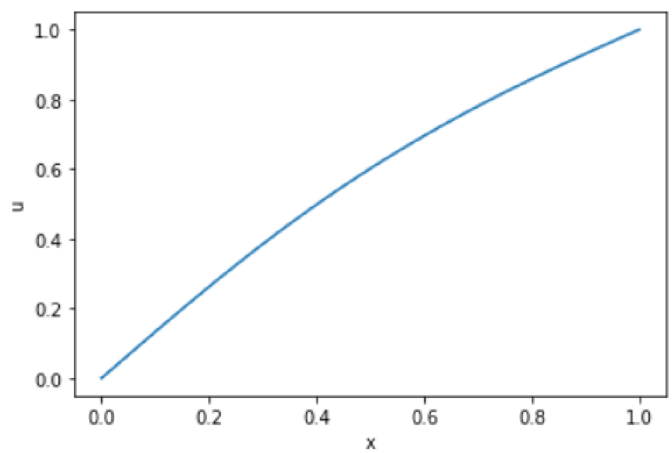
Время работы методов: прямая прогонка — 5.5 секунд, обратная прогонка — 6.1 секунд, встречная прогонка — 2.9 секунд.

5) $f(u) = sin(\omega u),\ \omega = 5$



Время работы методов: прямая прогонка — 10.2 секунд, обратная прогонка — 12.0 секунд, встречная прогонка — 6.3 секунд.

6) $f(u) = sin(\omega u),\ \omega = -3$

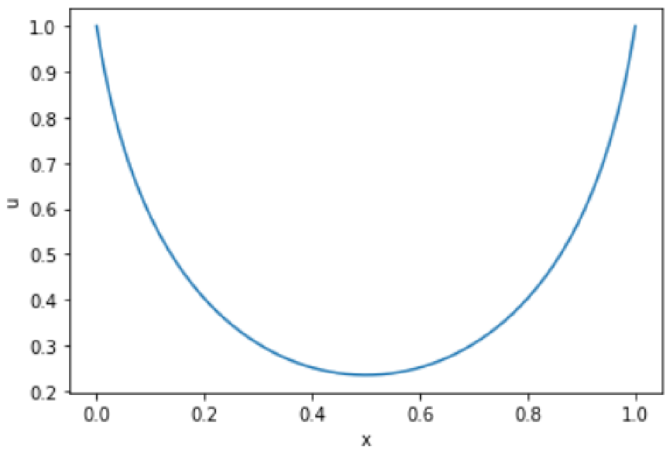


Время работы методов: прямая прогонка — 10.5 секунд, обратная прогонка — 9.5 секунд, встречная прогонка — 5.6 секунд.

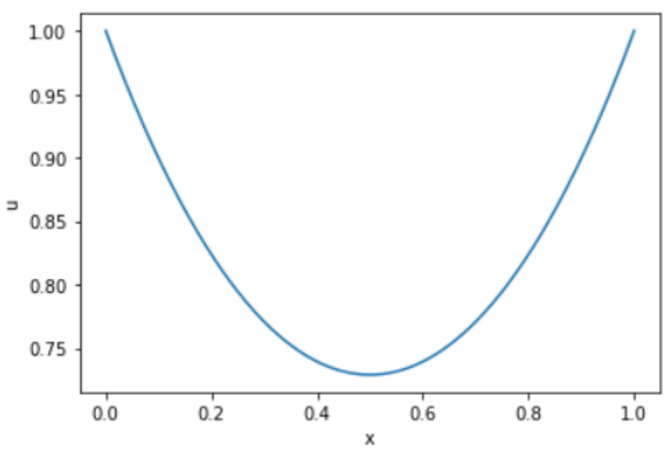
Расхождения в результатах среди методов (попарно): между прямой и обратной прогонкой — по порядку 10^{-6}, между прямой и встречной — по порядку 10^{-5}, между обратной и встречной — по порядку 10^{-5}.

Теперь рассмотрим случай, когда краевые условия являются периодическими, то есть равны: $u(0) = u(L)$ или $U_1 = U_2$, результаты решения краевой задачи:

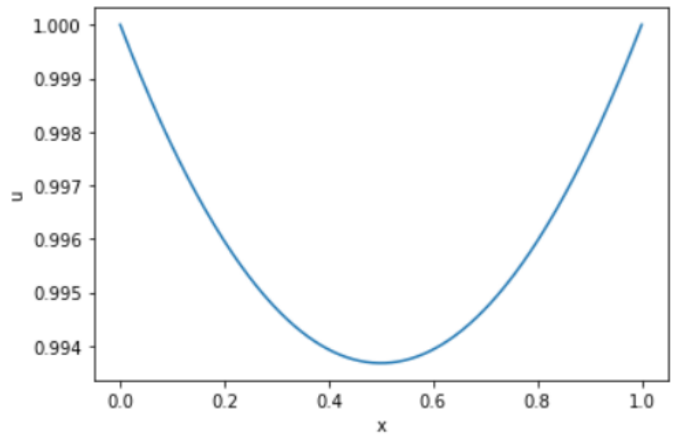
1) $f(u) = \exp(\alpha u)$, $\alpha = 5, U_1 = U_2 = 1$



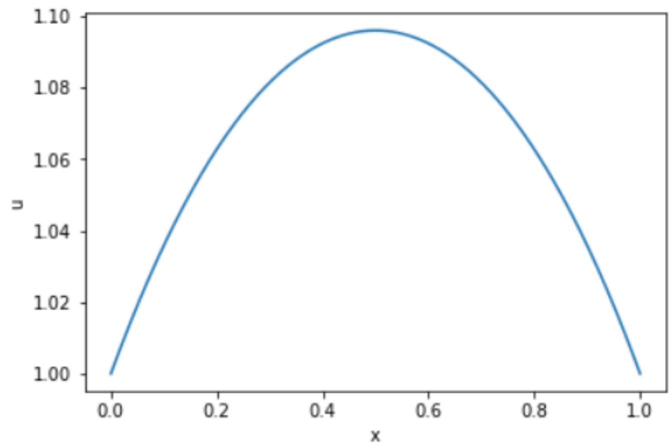
2) $f(u) = \exp(\alpha u)$, $\alpha = 1, U_1 = U_2 = 1$



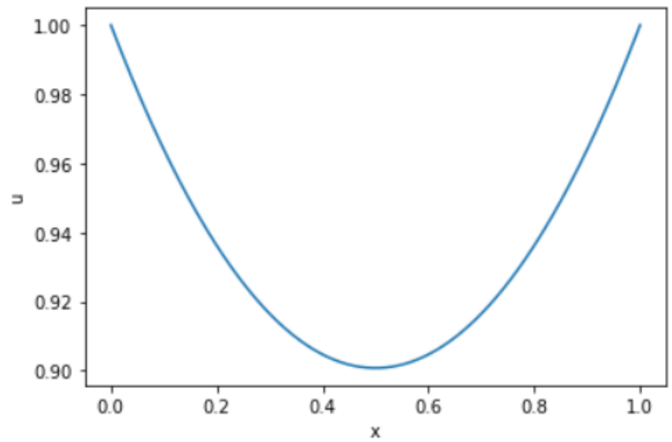
3) $f(u) = \exp(\alpha u),\ \alpha = -3, U_1 \ = \ U_2 \ = \ 1$



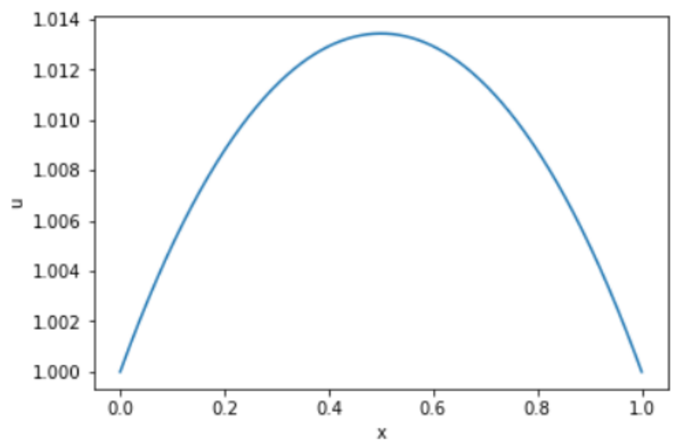
4) $f(u) = \sin(\omega u),\ \omega = 5, U_1 \ = \ U_2 \ = \ 1$



5) $f(u) = \sin(\omega u),\ \omega = 1, U_1 = U_2 = 1$



6) $f(u) = \sin(\omega u),\ \omega = -3, U_1 = U_2 = 1$



5 Исследование на сходимость по сетке

Возьмем краевую задачу $\frac{\partial^2 u}{\partial x^2} = sin(0 \cdot u) = 0$, $u(0) = 0$, $u(1) = 1$, аналитическое решение которой известно:$u(x) = x$. Исследуем все три типа прогонки на сходимость по сетке, то есть снимем зависимость ошибки (то, насколько экспериментальное решение отличается от теоретического) от количества точек сетки (количество точек обратно пропорционально ее шагу: $n = L/h$). Результаты приведены ниже:

а) Прямая прогонка

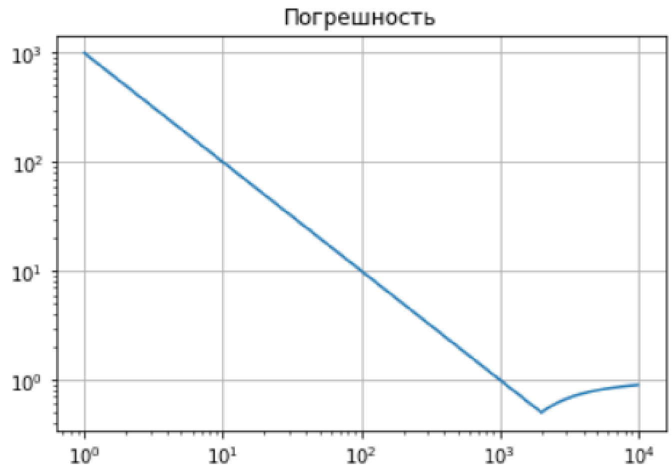


Рис. 1: Error(n)

б) Обратная прогонка

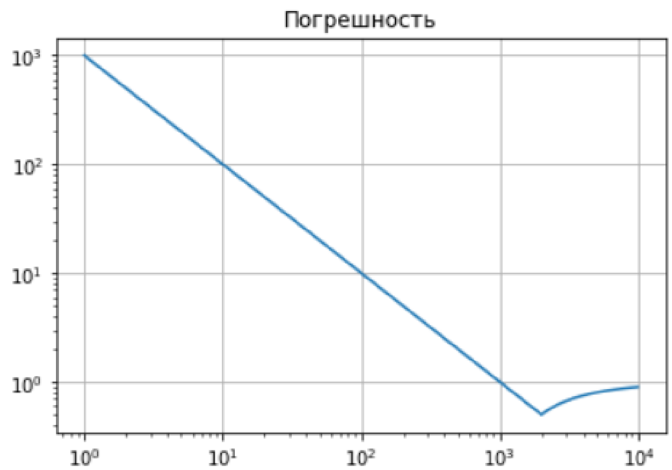


Рис. 2: Error(n)

в) Встречные прогонки

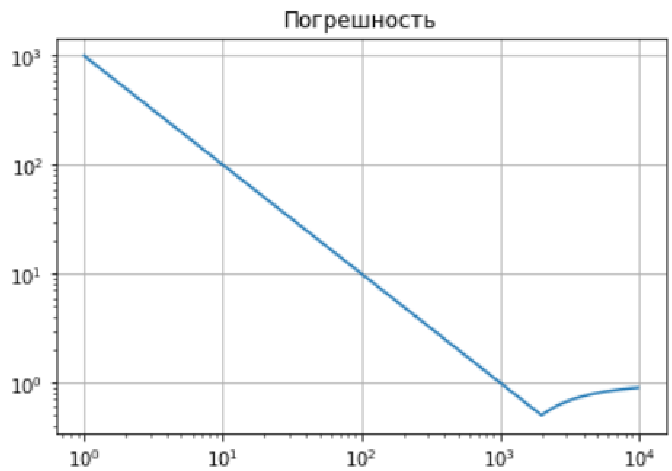


Рис. 3: Error(n)

6 Вывод

Таким образом, были исследованы три варианта трехточечной прогонки: прямая прогонка (слева направо), обратная прогонка (справа налево) и встречные прогонки на примере решения краевой задачи нелинейного дифференциального уравнения второго порядка. Было выяснено, что метод встречных прогонок является наиболее быстрым, но также имеет наибольшее расхождение результата с двумя другими методами при равных иных параметрах.