# Лабораторная работа №4 по теме: «. Решение конечно-разностным методом краевой задачи для обыкновенного дифференциального уравнения второго порядка»

## Подготовил студент Михайлов Денис

## группы Б8117-02.03.01

**Постановка задачи:**

Получить решение конечно-разностным методом краевой задачи для обыкновенного дифференциального уравнения второго порядка.

**Дано:**

С краевыми условиями:

Где

**Ход решения:**

Решим краевую задачу с помощью трёхточечной разностной схемы:

Где:

Коэффициенты уравнения:

Для решения системы воспользуемся методом монотонной прогонки:

* Шаг 1: находим
* Шаг 2: находим и , где
* Шаг 3: Получаем
* Шаг 4: Получаем решение , где
* Наши краевые условия:

Найдем решение при данном шаге , где первый столбец – точное решение, второй столбец – решение с помощью разностной схемы:

1.0                      1

0.9975062344139651       0.9275072401294253

0.9900990099009901       0.9226945307927693

0.9779951100244499       0.9132779728510031

0.9615384615384615       0.8995958657248821

0.9411764705882353       0.8820922215088052

0.9174311926605504       0.8612878576744636

0.8908685968819599       0.8377496260752673

0.8620689655172413       0.8120608643482495

0.8316008316008315       0.7847955013809811

0.8                      0.7564973756229075

0.7677543186180422       0.7276654367407908

0.7352941176470588       0.6987447499933076

0.7029876977152899       0.6701226903455486

0.6711409395973154       0.6421294154642898

0.64                     0.6150416129163984

0.6097560975609756       0.5890885739531365

0.5805515239477503       0.564459797535912

0.5524861878453039       0.5413135257329309

0.5256241787122208       0.5197858212458872

0.5                      0.5

Преобладание диагональных элементов (условие и условие выполняются) обеспечивает устойчивость метода прогонки относительно погрешностей округления. Так как разностная схема является частью метода, то значит и сама схема устойчива относительно погрешности округления.

Сама трехточечная разностная схема обладает первым порядком точности.

**Код программы**

from math import exp

def isP(x):

    return -(pow(x, 2) + 1)

def isQ(x):

    return -2\*x

def isR(x, h):

    return (isP(x)/2) \* h

def isA(x, h):

    return (1/(h\*\*2)) \* (abs(isR(x, h)) + exp(-abs(isR(x, h))) - isR(x, h))

def isC(x, h):

    return (1/(h\*\*2)) \* (abs(isR(x, h)) + exp(-abs(isR(x, h))) + isR(x, h))

def isB(x, a, c):

    return isQ(x) - a - c

def func(x):

    return 2\*(3\*pow(x, 2) - 1) / ((x\*\*2) + 1)\*\*3

def MMS(h, N):

    X = [h\*i for i in range(N+1)]

    d = [func(x) for x in X]

    a = [isA(x, h) for x in X]

    c = [isC(x, h) for x in X]

    b = [isB(X[i], a[i], c[i]) for i in range(N+1)]

    a[0] = c[N] = 0

    b[0], b[N] = isB(X[0], 0, c[0]), isB(X[N], a[N], 0)

    A, B = [], []

    A.append(1)

    B.append(0)

    for i in range(1, N+1):

        z = a[i]\*A[i-1] + b[i]

        A.append(-c[i]/z)

        B.append((d[i] - a[i]\*B[i-1])/z)

    Y = [0 for i in range(N+1)]

    Y[N] = 0.5

    for i in range(N-1, -1, -1):

        Y[i] = A[i] \* Y[i+1] + B[i]

    Y[0] = 1

    return Y