МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Кафедра компьютерных технологий и систем

Лабораторная работа №3

**“Исследование устойчивости разностных схем”**

Вариант 2

Выполнил:

Ёда Никита Дмитриевич  
 студент 4 курса 6 группы

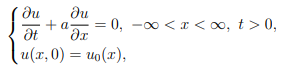
Преподаватель:

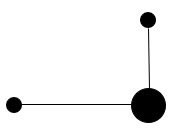
Репников Василий Иванович

**Задание 1**

Постановка задачи:

1. Исследовать устойчивость РС спектральным методом.
2. Исследовать устойчивость РС с помощью принципа максимума.
3. Выполнить машинную реализацию РС и проверить найденное условие устойчивости.



**

Решение:

По шаблону построим разностную схему:







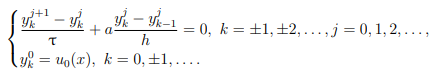
Вычислим погрешность аппроксимации разностной схемы



Исследование устойчивости разностной схемы спектральным методом.



Запишем разностную схему в индексной форме:



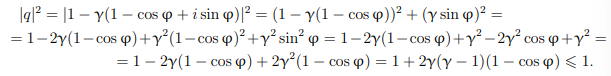
Подставим в разностное уравнение:







По спектральному методу необходимо выполнение условия :

**

Отсюда:

**

Т.к. *1-cos>0, ,* то:



При выполнении этих условий разностная схема будет устойчива, при *a=1:*

**

Исследование устойчивости разностной схемы:

В качестве точки для исследования устойчивости возьмём точку . Перепишем аппроксимацию основного уравнения переноса:





Проверим выполнения условий устойчивости:





При *а = 1*:



Реализация в Python:

|  |
| --- |
| import numpy as np  import matplotlib.pyplot as plt  import seaborn as sns  def generate\_grids(left\_border, right\_border, num\_x\_points, upper\_bound, num\_t\_points):      h = (right\_border-left\_border) / num\_x\_points      nodes\_x = np.linspace(left\_border, right\_border, num\_x\_points+1)      tau = upper\_bound / num\_t\_points      nodes\_t = np.linspace(0, upper\_bound, num\_t\_points+1)      print('h =', h)      print('tau =', tau)      return nodes\_x, nodes\_t, h, tau  def u(x, t, a, u\_0):      return u\_0(x-a\*t)  a = 1  def u\_0(x):      return x\*\*2  def diff\_scheme\_solve(nodes\_x, nodes\_t, h, tau, u\_0, a):      gamma = a \* tau / h        y = np.zeros((len(nodes\_x), len(nodes\_t)))      for k in range(len(nodes\_x)):          y[k, 0] = u\_0(nodes\_x[k])      for k in range(len(nodes\_x)-1):          for j in range(len(nodes\_t)-1):              y[k, j+1] = (1-gamma) \* y[k, j] + gamma \* y[k+1, j]      return y  nodes\_x, nodes\_t, h, tau = generate\_grids(0, 1, 5, 0.25, 5)  y = diff\_scheme\_solve(nodes\_x, nodes\_t, h, tau, u\_0, a)  plt.figure(figsize=(16, 8))  # Создаем одну область для всех графиков  for j, t in enumerate(nodes\_t):      plt.plot(nodes\_x[:-1], y[:-1, j], label=f'numerical solution (t={round(t, 2)})')      plt.plot(nodes\_x, u(nodes\_x, t, a, u\_0), label=f'exact solution (t={round(t, 2)})')  plt.grid(True)  plt.xlabel('x')  plt.ylabel('u(x,t)')  plt.title('Аппроксимация для всех временных шагов')  plt.legend()  plt.show() |

Результат выполнения программы:

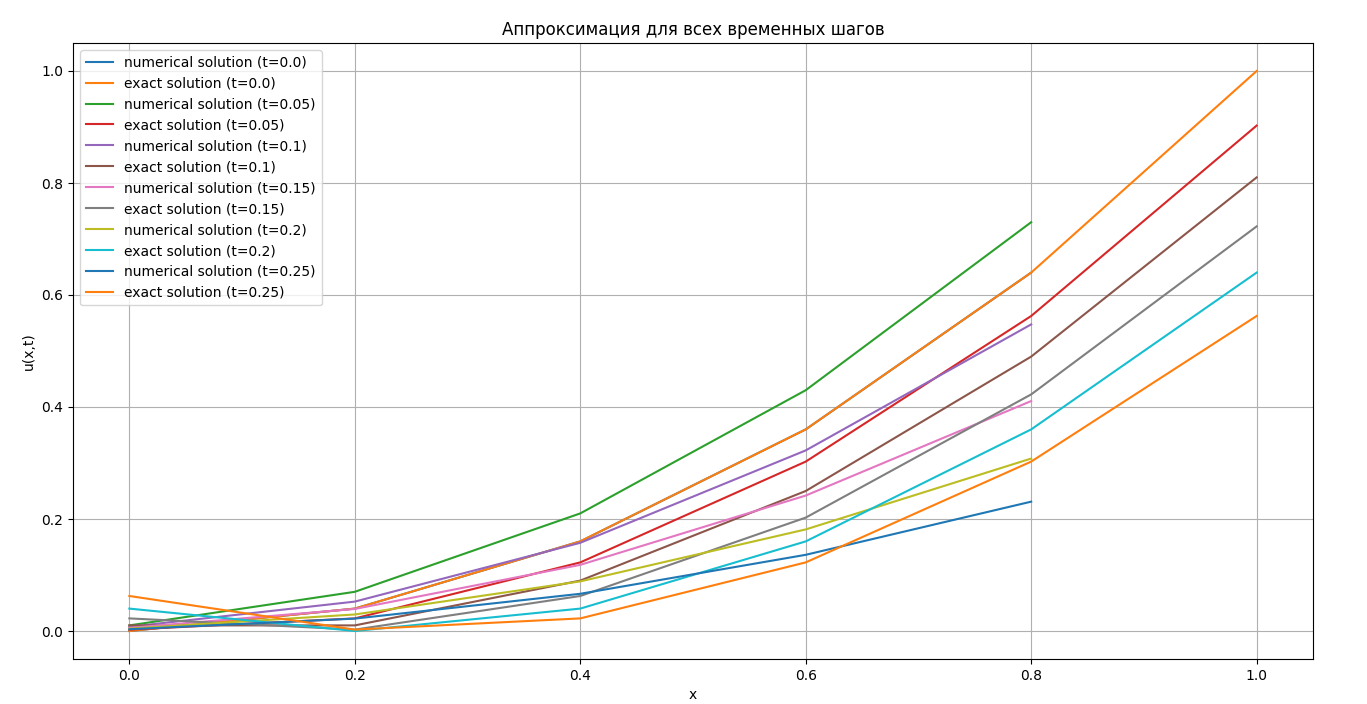
**

Рисунок 1 – Аппроксимация для всех временных шагов