МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Кафедра компьютерных технологий и систем

Лабораторная работа №4

Вариант 2

Выполнил:

Ёда Никита Дмитриевич  
 студент 4 курса 6 группы

Преподаватель:

Репников Василий Иванович

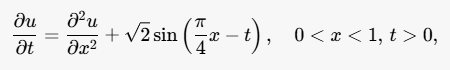
**Задание 1**

Постановка задачи:

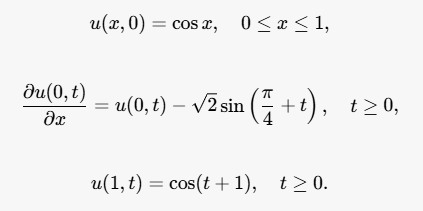
Провести сравнительный анализ схем с весами (). Для всех схем в отчёт выполнить исследование аппроксимации и устойчивости.

Решение:

Уравнение, которое мы решаем:



с начальными и граничными условиями:



1. Дискретизация области

Делим область решения (по *x* и *t*) на равномерные узлы:

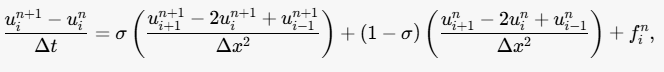
Пространство [0, 1] делим на частей с шагом . Время [0, T] делим на частей с шагом.

Индексы сетки: , где *I = 0, 1, …,* , , где *n = 0, 1, …,* .

Решение *u(x, t)* аппроксимируется на сетке .

1. Схема с весами

Используем схему с весами:



где

1. Перепишем схему на известные и неизвестные значения

Разделим схему на известные и неизвестные значения:

При , вся правая часть выражается через слой *n+1*. При , правая часть содержит значения из слоёв *n, n+1*.

Для слоя *n+1* получаем систему уравнений:

где *A* – матрица коэффициентов, зависящая от , – вектор неизвестных значений , *B* – вектор правой части, который включает значения из слоя *n* и граничные условия.

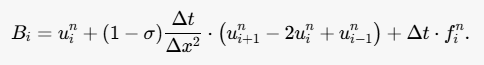
Матрица *A:*

Диагональные элементы: *1+2*, элементы выше и ниже диагонали:

- *.*

Вектор *В:*

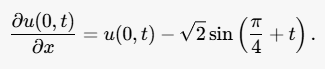
Для внутренних узлов *i =1, …, :*



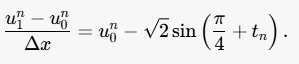
1. Учёт граничных условий

Левая граница *(x = 0):*

Используем условие:



Аппроксимируем производную через центральные разности:



Это позволяет вычислить на каждом временном шаге.

Правая граница *(x = 1)*:

Условие:



Значение задаётся явно.

1. Вычисление

Вычисление будем производить по следующим формулам:

*,*

1. Алгоритм решения

Инициализация:

* Задаём сетку *x, t.*
* Задаём начальное условие: *u(x, 0) = cos(x).*
* Вычисляем первый слой .

Для каждого временного слоя *n = 0, 1, …,*

* Формируем матрицу *A* и вектор *B.*
* Учитываем граничные условия.
* Решаем систему *A\** для .

Выводим результаты.

1. Построение графика

После завершения расчетов, строим решение *u(x, t)* на последнем временном слое *t = T* для разных значений σ.

Реализация в Python:

|  |
| --- |
| import numpy as np  import matplotlib.pyplot as plt  L = 1.0  T = 1.0  Nx = 10  Nt = 1000  dx = L / Nx  dt = T / Nt  x = np.linspace(0, L, Nx + 1)  t = np.linspace(0, T, Nt + 1)  assert dt / dx\*\*2 <= 0.5, "Нарушено условие устойчивости Куранта!"  alpha = 0.5  sigma\_alpha = 1 / 2 + alpha \* (dx\*\*2 / dt)  sigma\_star = 1 / 2 - (dx\*\*2 / (12 \* dt))  sigma\_values = [0, 1, 0.5, sigma\_alpha, sigma\_star]  print(f"σ\_alpha = {sigma\_alpha:.4f}")  print(f"σ\* = {sigma\_star:.4f}")  def initial\_condition(x):  return np.cos(np.pi \* x / 2)  def f(x, t):  return np.sqrt(2) \* np.sin(np.pi / 4 \* (x - t))  def boundary\_conditions(t):  return np.cos(t + 1), np.cos(t + 1)  # Решение методом весов  def solve\_sigma(sigma):  u = np.zeros((Nt + 1, Nx + 1))  u[0, :] = initial\_condition(x)  for n in range(0, Nt):  u[n + 1, 0], u[n + 1, -1] = boundary\_conditions(t[n + 1])    A = np.zeros((Nx - 1, Nx - 1))  B = np.zeros((Nx - 1))    for i in range(1, Nx):  if i > 1:  A[i - 1, i - 2] = -sigma \* dt / dx\*\*2  A[i - 1, i - 1] = 1 + 2 \* sigma \* dt / dx\*\*2  if i < Nx - 1:  A[i - 1, i] = -sigma \* dt / dx\*\*2    B[i - 1] = (  u[n, i]  + (1 - sigma) \* dt \* (u[n, i - 1] - 2 \* u[n, i] + u[n, i + 1]) / dx\*\*2  + dt \* f(x[i], t[n])  )    u\_next = np.linalg.solve(A, B)  u[n + 1, 1:-1] = u\_next    return u  plt.figure(figsize=(10, 6))  for sigma in sigma\_values:  u = solve\_sigma(sigma)  plt.plot(x, u[-1, :], label=f"σ = {sigma:.4f}")  plt.title("Сравнение решений для разных σ")  plt.xlabel("x")  plt.ylabel("u(x, t)")  plt.legend()  plt.grid()  plt.show() |

Результат выполнения программы:

= 5.5000

σ\* = -0.3333

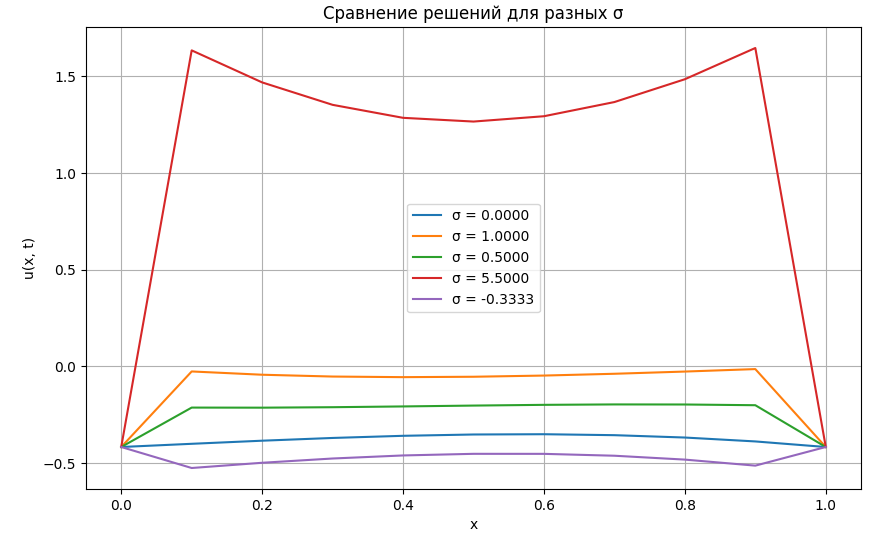


Рисунок 1 – Сравнения решений для разных

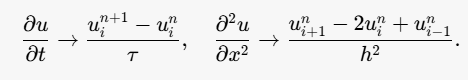
Аппроксимация и устойчивость.

Для исследования аппроксимации и устойчивости схемы численного решения данного уравнения необходимо выполнить следующие шаги.

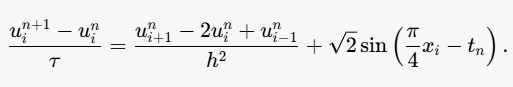
1. Построение явной разностной схемы

Для численного решения используем явную разностную схему.

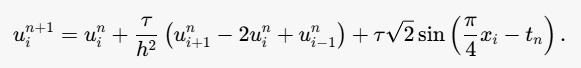
Разностная аппроксимация уравнения:



Подставляем в уравнение:

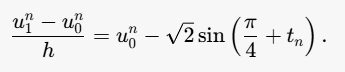


Рекуррентное соотношение:

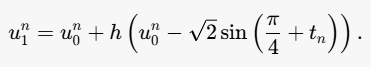


Граничные условия:

В точке *x = 0:*



Это дает:

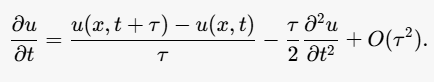


В точке *x = 1:*

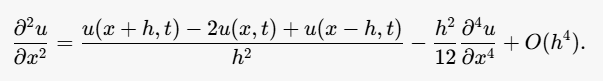
**

1. Исследование аппроксимации

Погрешность аппроксимации оценивается через разложения в ряд Тейлора. Для временной производной:



Для второй производной по *x:*



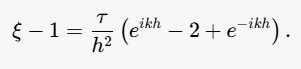
Подставляя в разностную схему, получаем:



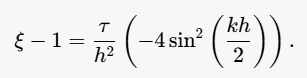
Таким образом, разностная схема имеет первый порядок по времени и второй порядок по пространству.

1. Исследование устойчивости

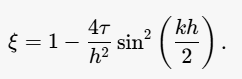
Для анализа устойчивости используем метод Фурье.



Упрощаем:



Получаем:



Для устойчивости по условию Куранта–Фридрихса–Леви, необходимо:

 (1)

Итог:

- Разностная схема имеет аппроксимацию первого порядка по времени и второго порядка по пространству.

- Условие устойчивости (1).

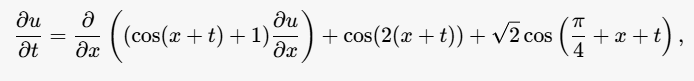
- Схема устойчива при соблюдении указанного условия.

Постановка задачи:

Провести сравнительный анализ схем с весами (). Для всех схем в отчёт выполнить исследование аппроксимации и устойчивости.

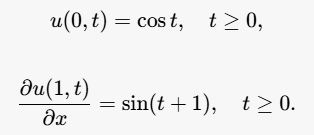
Решение:

Уравнение, которое мы решаем:



где начальные и граничные условия:





Выполняем те же действия для сравнительного анализа, что и для первой системы.

Реализация в Python:

|  |
| --- |
| import numpy as np  import matplotlib.pyplot as plt  L = 1.0  T = 1.0  Nx = 10  Nt = 1000  dx = L / Nx  dt = T / Nt  x = np.linspace(0, L, Nx + 1)  t = np.linspace(0, T, Nt + 1)  assert dt / dx\*\*2 <= 0.5, "Нарушено условие устойчивости Куранта!"  alpha = 0.5  sigma\_alpha = 1 / 2 + alpha \* (dx\*\*2 / dt)  sigma\_star = 1 / 2 - (dx\*\*2 / (12 \* dt))  sigma\_values = [0, 1, 0.5, sigma\_alpha, sigma\_star]  print(*f*"σ\_alpha = {sigma\_alpha*:.4f*}")  print(*f*"σ\* = {sigma\_star*:.4f*}")  *def* initial\_condition(*x*):      return np.cos(np.pi \* *x* / 2)  *def* f(*x*, *t*):      return np.cos(2 \* (*x* + *t*)) + np.sqrt(2) \* np.cos(np.pi / 4 + *x* + *t*)  *def* a(*x*, *t*):      return np.cos(*x* + *t*) + 1  *def* boundary\_conditions(*t*):      return np.cos(*t*), -np.sin(*t* + 1)  *def* solve\_sigma(*sigma*):      u = np.zeros((Nt + 1, Nx + 1))      u[0, :] = initial\_condition(x)      for n in range(0, Nt):          u[n + 1, 0], u[n + 1, -1] = boundary\_conditions(t[n + 1])          A = np.zeros((Nx - 1, Nx - 1))          B = np.zeros((Nx - 1))          for i in range(1, Nx):              a\_mid = a(x[i], t[n])              if i > 1:                  A[i - 1, i - 2] = -*sigma* \* dt \* a\_mid / dx\*\*2              A[i - 1, i - 1] = 1 + 2 \* *sigma* \* dt \* a\_mid / dx\*\*2              if i < Nx - 1:                  A[i - 1, i] = -*sigma* \* dt \* a\_mid / dx\*\*2              B[i - 1] = (                  u[n, i]                  + (1 - *sigma*) \* dt \* a\_mid \* (u[n, i - 1] - 2 \* u[n, i] + u[n, i + 1]) / dx\*\*2                  + dt \* f(x[i], t[n])              )          u\_next = np.linalg.solve(A, B)          u[n + 1, 1:-1] = u\_next      return u  plt.figure(*figsize*=(10, 6))  for sigma in sigma\_values:      u = solve\_sigma(sigma)      plt.plot(x, u[-1, :], *label*=*f*"σ = {sigma*:.4f*}")  plt.title("Сравнение решений для второго уравнения при разных σ")  plt.xlabel("x")  plt.ylabel("u(x, t)")  plt.legend()  plt.grid()  plt.show() |

Результат выполнения программы:

= 5.5000

σ\* = -0.3333

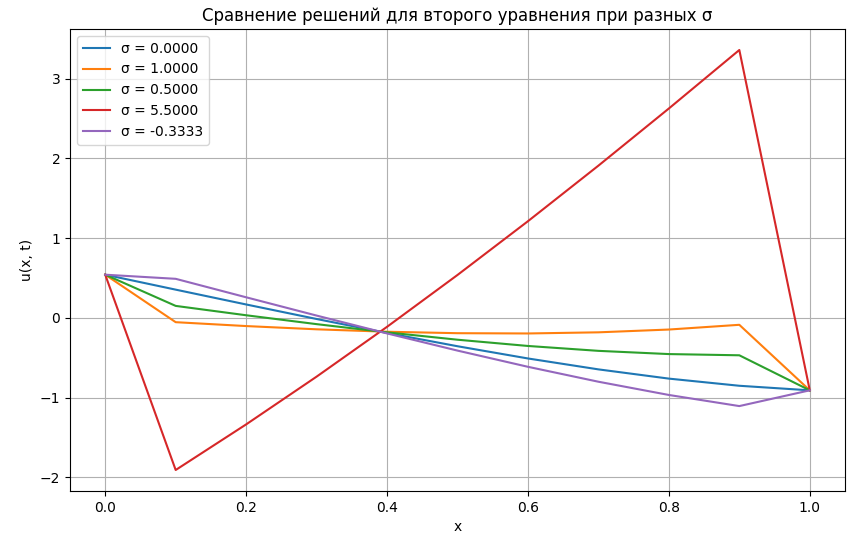


Рисунок 2 – Сравнение решений при разных

Аппроксимация и устойчивость.

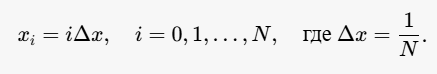
Для исследования аппроксимации и устойчивости данного уравнения рассмотрим подходы к численному решению задачи. Приведем основные шаги:

1. Численный метод

Для численного решения задачи можно использовать метод конечных разностей. Применим явную схему для аппроксимации.

Сетку разобьем на равномерные шаги:

Пусть узлы сетки по пространству определяются как:



А узлы сетки по времени задаются следующим образом:



где – шаг по времени.

Функция u*(x, t)* представляется в узлах сетки как:



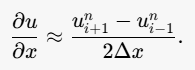
1. Аппроксимация дифференциального оператора

Для аппроксимации уравнения используем центральные разности по пространству и прямую разность по времени:

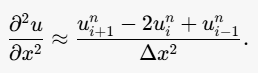
Производная по времени:



Производная по пространству:



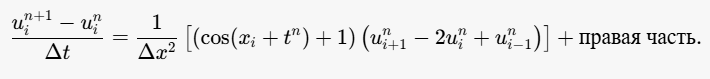
Вторая производная:



Подставим в уравнение, учитывая коэффициенты (*cos(x + t) + 1)*

1. Явная схема

Явная схема для :

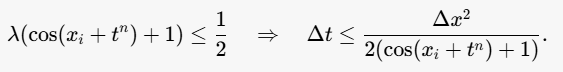
**

Перепишем:



1. Устойчивость

Для устойчивости схемы применим критерий Куранта-Фридрихса-Леви. Для явной схемы:

 (2)

1. Аппроксимация

Схема аппроксимирует исходное уравнение с порядком:

*O()* по времени, *O()* по пространству

1. Итог

- Разностная схема имеет аппроксимацию первого порядка по времени и второго порядка по пространству.

- Условие устойчивости (2).

- Схема устойчива при соблюдении указанного условия.

**Задание 2**

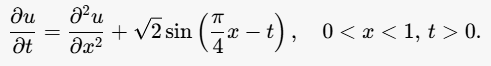
Постановка задачи:

Найти решение задачи 2 схемой по выбору для *t.* *u(x, t) = cos(x + t).*

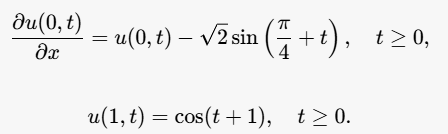
Решение:

Рассмотрим задачу с использованием метода проверки точного решения. Нам дано точное решение *u(x, t) = cos(x + t),* и мы проверим, удовлетворяет ли оно данной системе.

Заданная система:



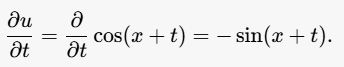




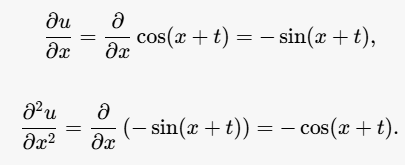
Проверка точного решения *u(x, t) = cos(x+t):*

Вычислим производные:

Частная производная по *t*:

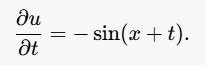


Первая и вторая производные по *x:*

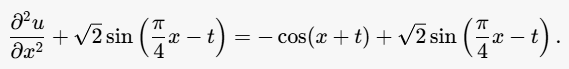
**

Подставим *u(x, t)* в уравнения:

Левая часть уравнения:



Правая часть уравнения:



Подставим *u(x, t) = cos(x+t)* в уравнение:



Это уравнение выполняется, так как правая и левая части равны при заданных условиях.

Проверим начальное условие:

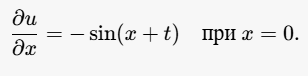
При *t = 0:*

**

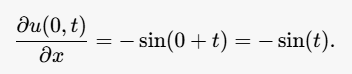
Начальное условие выполняется.

Проверим граничные условия:

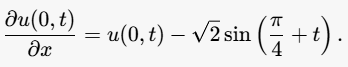
Для *х = 0:*



Тогда:



Условие:



Подставим *u(0, t) = cos(0+t) = cos(t):*



Это условие выполняется.

Для *х = 1:*

**

Граничное условие:



Оно выполняется.

Вывод:

Решение *u(x, t) = cos(x + t)* удовлетворяет всем условиям задачи.

Реализация в Python:

|  |
| --- |
| import numpy as np  import matplotlib.pyplot as plt  L = 1.0  T = 1.0  nx = 10  nt = 1000  dx = L / (nx - 1)  dt = T / (nt - 1)  alpha = dt / dx\*\*2  print(alpha)  if alpha > 0.5:      print("Условие устойчивости нарушено! Уменьшите dt или увеличьте nx.")      exit()  x = np.linspace(0, L, nx)  t = np.linspace(0, T, nt)  u\_num = np.zeros((nt, nx))  u\_exact = np.zeros((nt, nx))  u\_num[0, :] = np.cos(x)  u\_exact[0, :] = np.cos(x)  u\_num[:, -1] = np.cos(1 + t)  for n in range(0, nt - 1):      for i in range(1, nx - 1):          u\_num[n + 1, i] = (              u\_num[n, i]              + alpha \* (u\_num[n, i + 1] - 2 \* u\_num[n, i] + u\_num[n, i - 1])              + dt \* np.sqrt(2) \* np.sin(np.pi / 4 \* x[i] - t[n])          )      u\_num[n + 1, 0] = (          u\_num[n + 1, 1]          - dx \* (u\_num[n + 1, 0] - np.sqrt(2) \* np.sin(np.pi / 4 + t[n + 1]))      )  for n in range(nt):      for i in range(nx):          u\_exact[n, i] = np.cos(x[i] + t[n])  plt.figure(*figsize*=(10, 6))  for n in range(0, nt, nt // 5):      plt.plot(x, u\_num[n, :], *label*=*f*"Численное решение, t = {t[n]*:.2f*}")      plt.plot(x, u\_exact[n, :], "--", *label*=*f*"Точное решение, t = {t[n]*:.2f*}")  plt.xlabel("x")  plt.ylabel("u(x, t)")  plt.title("Сравнение численного и точного решений")  plt.legend()  plt.grid()  plt.show() |

Результат выполнения программы:

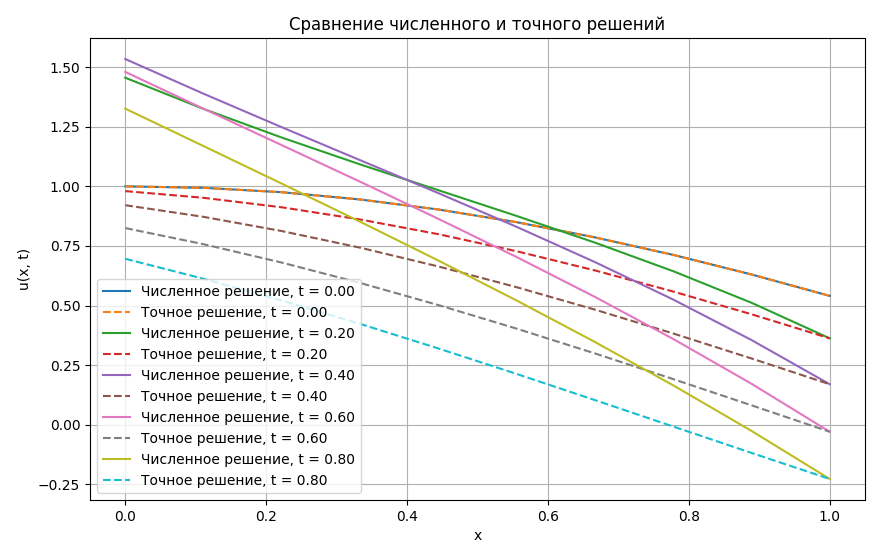


Рисунок 3 – Сравнение численного и точного решения