БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ  
КАФЕДРА ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ

Лабораторная работа №2

«Способы построения и исследования разностных схем»  
по дисциплине «Численные методы математической физики»

Вариант K-2

Выполнил: Прудникович В.В. студент 3 курс 7 группы

Проверил: Будник А.М. Доцент кафедры вычислительной математики ФПМИ

Для решения краевой задачи вида

на сетке построить разностную схему с весами при с погрешностью аппроксимации не ниже .

Необходимо:

1. Показать, что построенная схема имеет заданный порядок аппроксимации.
2. Исследовать устойчивость полученной разностной схемы по начальным данным, используя метод гармоник.
3. Реализовать данную разностную схему при h = 0.1 и , выбранным из условия устойчивости.
4. Оценить приближенное решение, анализируя погрешность аппроксимации при разных шагах.
5. **Выбор требуемой разностной схемы**

Введём в исходной области G = [0, 1] X [0, 1] прямоугольную сетку. Так как в уравнении фигурирует 2-ая производная по параметру t, то число слоев не может быть меньше 3.

Аппроксимируем имеющиеся дифференциальные операторы разностными:

а также, для удобства безиндексной записи, введем обозначения для операторов, использующих слой ниже или слой выше:

Рассмотрим разностную схему с весами:

В нашем случае необходимо использовать схему с весом = 0,

Тогда .

является краевым условием 3-его рода и аппроксимируется следующим разностным уравнением:

При этом погрешность краевых условий является величиной , если

Тогда , отсюда

1. **Проверка порядка точности.**

Проверим порядок аппроксимации для уравнения и условий.

Условия и выполняются точно на заданной сетке (т.к. узлы сетки лежат ровно на границах заданной области).

Проверим значение величины :

Для того, чтобы величина равнялась величине , вместо возьмём , равную:

) y

Тогда исходная РС примет вид:

Полученная разностная схема аппроксимирует исходную задачу с порядком аппроксимации +)

1. **Исследование устойчивости с помощью метода гармоник.**

Представим исходное уравнение в виде

Расписав выражение , и сгруппировав слагаемые по T, получим:

= 0

Разделим всё на получившийся общий множитель

Рассмотрим характеристическое ур-ие :

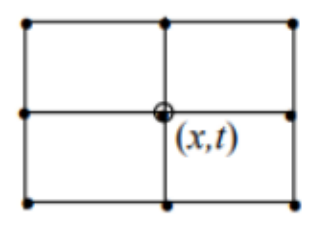
Достаточным условием устойчивости является выполнение неравентсв: , где , корни вышеуказанного уравнения. По т. Виета Тогда выполнение неравенств возможно только при D :

В нашем случае заданный вес , тогда условие устойчивости выполняется при .

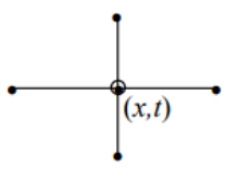
1. **Реализация выбранной разностной схемы.**

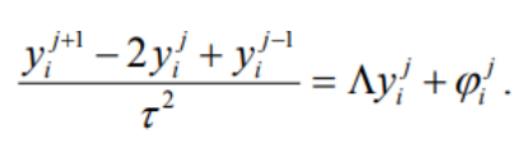
По условию задачи разностная схема является весовой с весом . Как уже было сказано ранее, для аппроксимации 2-ой производной по t потребуется использовать 3 слоя.

Весовая схема для уравнения колебания:

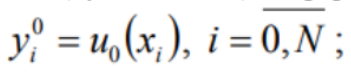


Конкретный случай для :



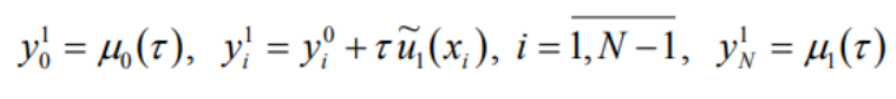


Так как нам необходимо использовать 3 временных слоя на каждом шаге, то первые 2 мы должны задать заранее, воспользовавшись условиями исходной задачи.

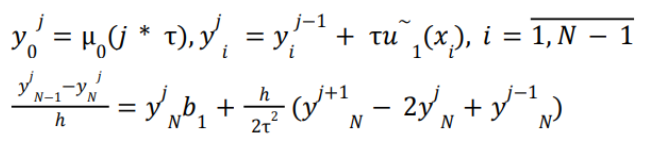


Формула для элементов 0-ого слоя:

Формула для элементов 1-ого слоя:



Заполнив первые 2 слоя, начинаем заполнять все оставшиеся:



Листинг программы

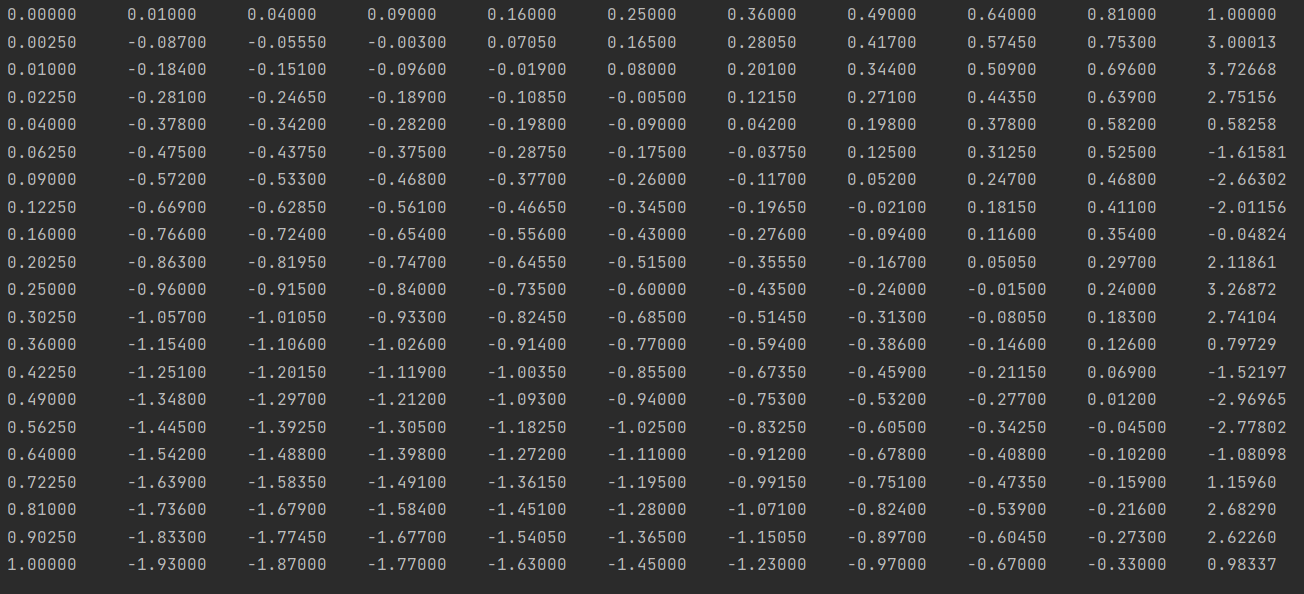
**import numpy as np  
def null\_l(i, h):  
 return (i \* h)\*\*2  
def toFixed(numObj, digits=0):  
 return f"{numObj:.{digits}f}"  
def u\_new(i, j, h, tau, matrix):  
 res = matrix[j - 1][i] + tau\*((i\*h)\*\*2 + (tau/2)\*((matrix[0][i+1] +matrix[0][i-1] - 2\*matrix[0][i])/(h\*\*2))-2)  
 return res  
h = 0.1  
tau = 0.05  
matrix = np.zeros((int(1/tau)+1,int(1/h)+1))  
for i in range(int(1/h)+1):  
 matrix[0][i] = null\_l(i, h)  
for i in range (1, int(1/tau)+1):  
 matrix[i][0] = null\_l(i, tau)  
for j in range(1,int(1/tau)+1):  
 for i in range(1, int(1/h)):  
 matrix[j][i] = u\_new(i, j, h, tau, matrix)  
matrix[1][10] = 3 + (tau)\*\*3  
for i in range(2,21):  
 matrix[i][10] = 2\*tau\*tau/(h\*\*2)\*(matrix[i - 1][9] - matrix[i - 1][10]) -matrix[i - 1][10]\*2\*tau\*tau/h +2\*matrix[i - 1][10] - matrix[i - 2][10]  
for i in range(21):  
 line = ''  
 for j in range(11):  
 line += str(toFixed(matrix[i][j], 5)) + ' \t'  
 print(line)  
print("\n")  
h = 0.001  
tau = 0.0005  
matrix = np.zeros((int(1/tau)+1,int(1/h)+1))  
for i in range(int(1/h)+1):  
 matrix[0][i] = null\_l(i, h)  
for i in range (1, int(1/tau)+1):  
 matrix[i][0] = null\_l(i, tau)  
for j in range(1,int(1/tau)+1):  
 for i in range(1, int(1/h)):  
 matrix[j][i] = u\_new(i, j, h, tau, matrix)  
 matrix[1][int(1/h)] = 3 + (tau)\*\*3  
for i in range(2,int(1/tau)+1):  
 matrix[i][int(1/h)] = 2\*tau\*tau/(h\*\*2)\*(matrix[i - 1][int(1/h)-1] - matrix[i - 1][int(1/h)]) - matrix[i - 1][int(1/h)]\*2\*tau\*tau/h +2\*matrix[i - 1][int(1/h)] - matrix[i - 2][int(1/h)]  
i = 0  
while i < int(1/tau)+1:  
 line = ''  
 j = 0  
 while j < int(1/h)+1:  
 line += str(toFixed(matrix[i][j], 5)) + ' \t'  
 j = j + 100  
 print(line)  
 i = i + 100**

1. **Анализ результатов:**

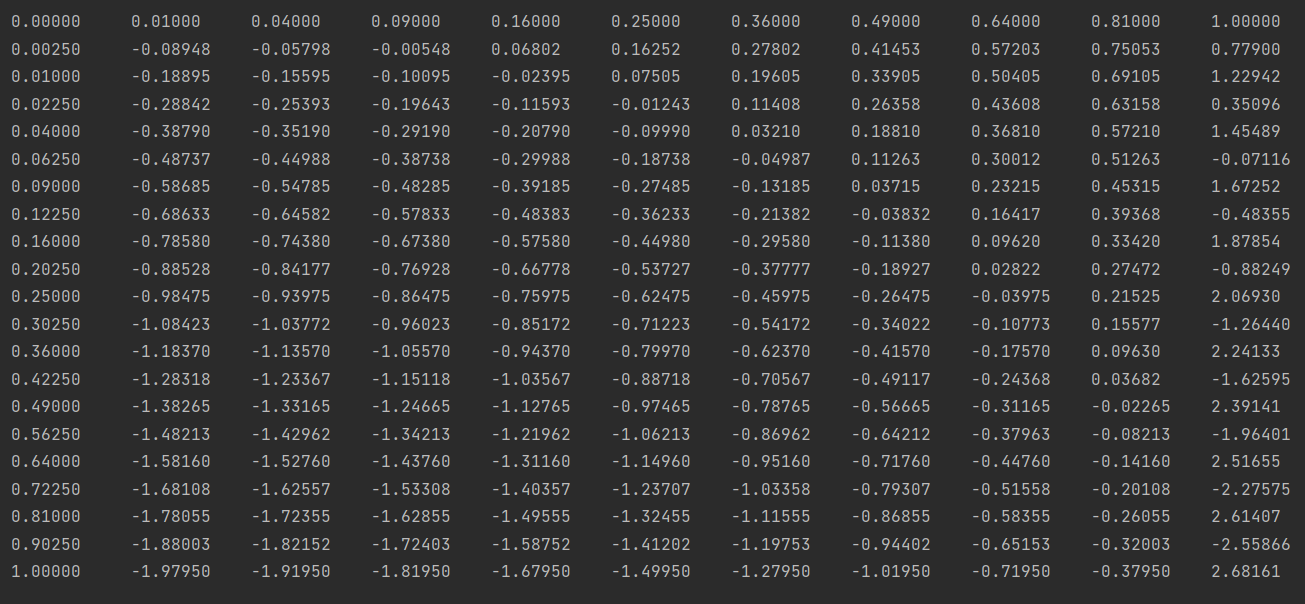
Рассмотрим значения функции, полученные с помощью аппроксимации её разностной схемой с различными шагами.

Также стоит учесть, что для устойчивости следует выбирать .

**h = 0.1, = 0.05,**



**h = 0.001, = 0.0005:**



**Можно заметить, значения для внутренних узлов схем отличаются не более, чем на O() , где h - значение шага для верхнего случая. Если рассматривать значения функции для шагов h = 0.001, = 0.0005, как точные, то можно сделать вывод, что порядок аппроксимации схемы в первом случае = O(+)**