Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

Лабораторная работа № 14

«Аппроксимация граничных условий второго рода в методе конечных разностей на примере уравнения теплопроводности»

Выполнил:

студент гр. 953506

Кондрашов И.Д.

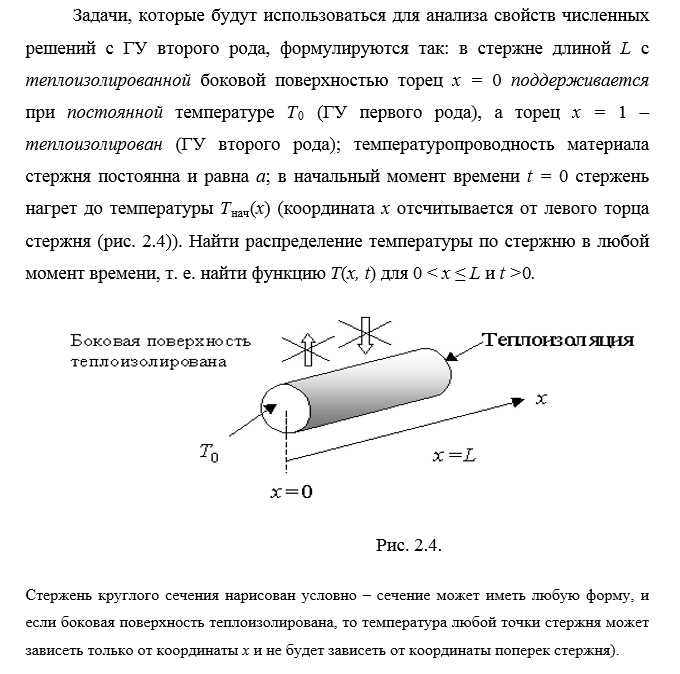
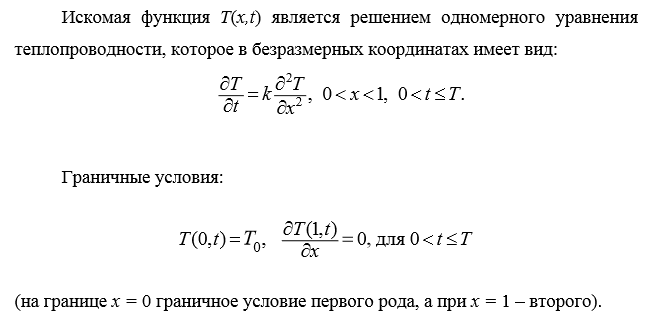
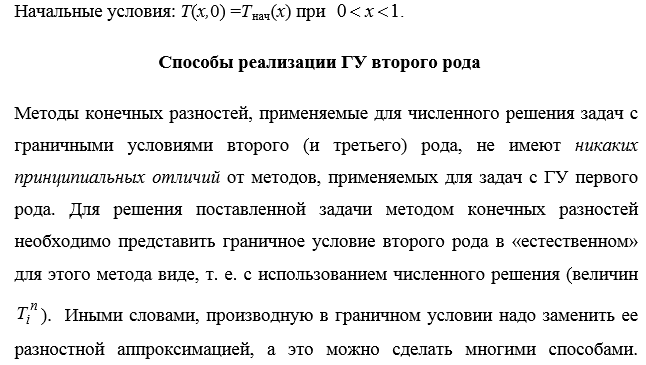
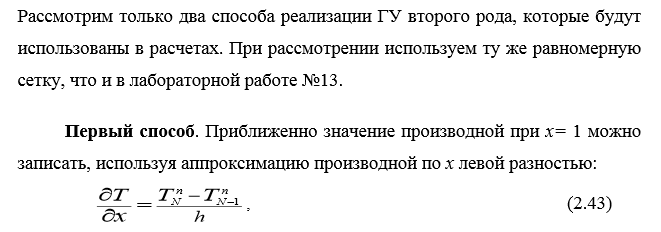
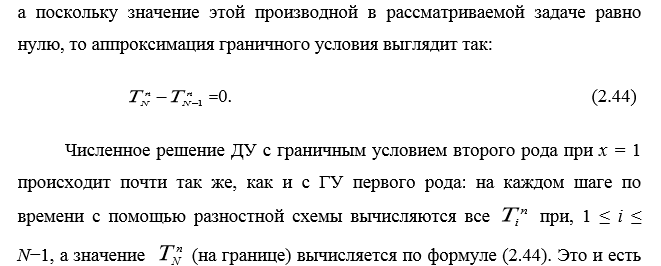
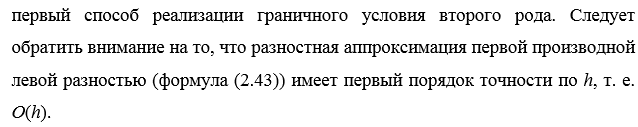
Проверил:

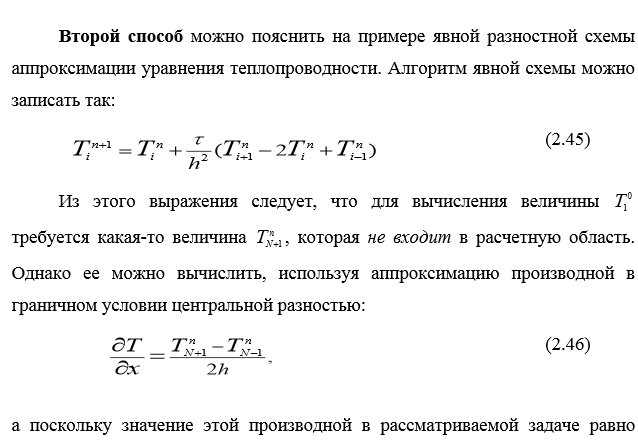
Анисимов В.Я.

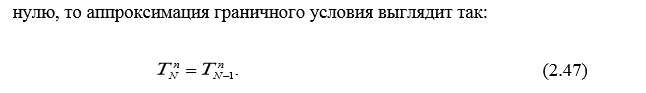
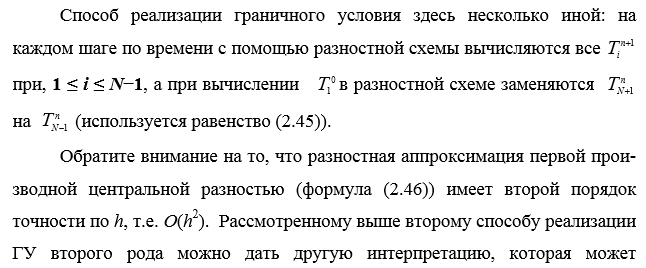
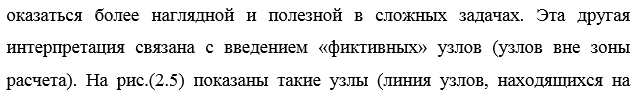
Минск, 2021

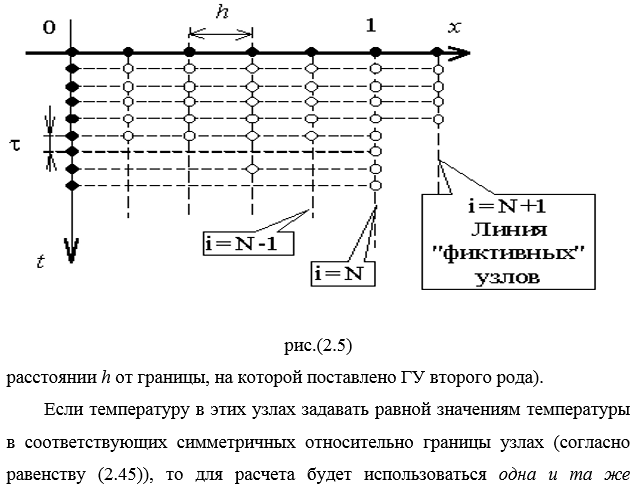
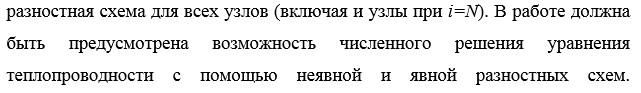
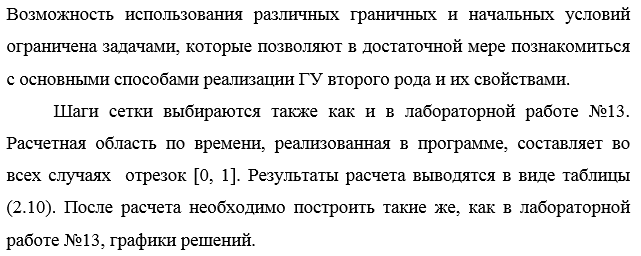
**Цель работы**

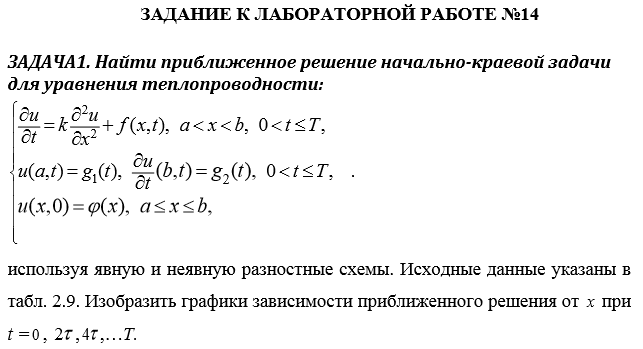
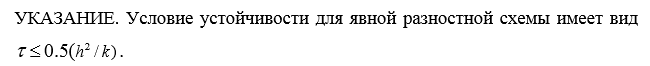
Ознакомиться с наиболее часто применяемыми способами аппроксимации граничных условий второго рода (граничных условий (ГУ) Неймана) в методе конечных разностей (на примере ГУ для одномерного нестационарного уравнения теплопроводности).

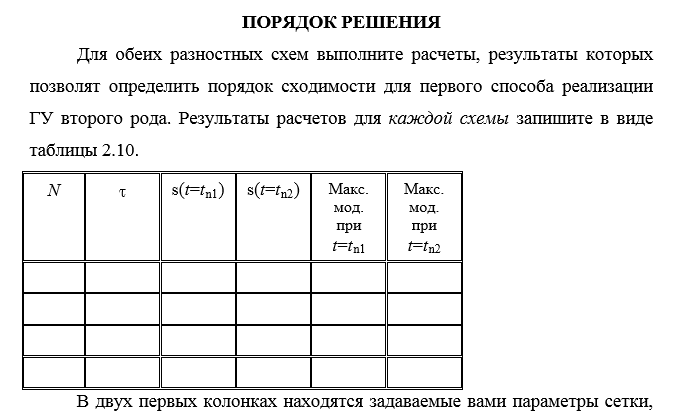
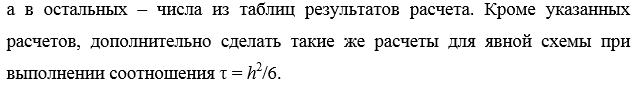
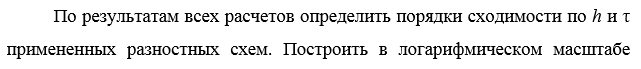
**Краткие теоретические сведения**     

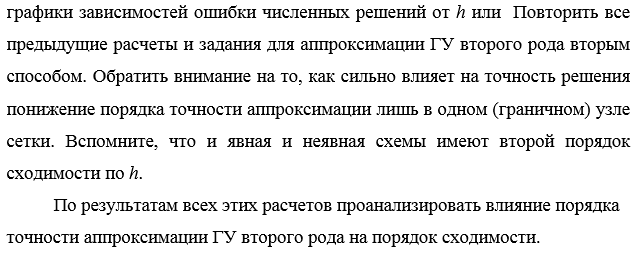




**Результаты**

1. Неявная разностная схема (1 - ый способ)

Chart, line chart

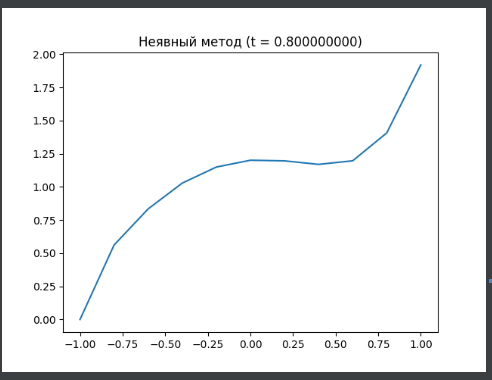
Description automatically generated

Chart, line chart

Description automatically generated

**Chart, line chart

Description automatically generated**

****

Text

Description automatically generated with low confidence

2. Явная разностная схема (1 – ый способ)

Chart, line chart

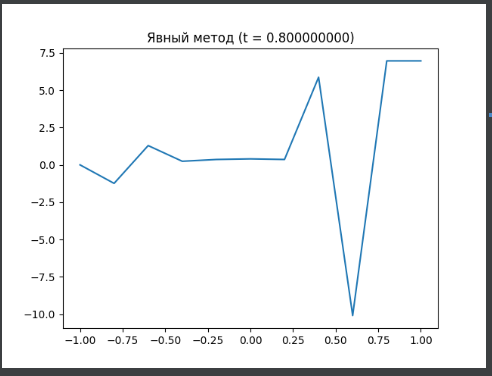
Description automatically generated

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated



Text

Description automatically generated

3. Явная разностная схема (2 - ой способ)

Chart

Description automatically generated

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

Text

Description automatically generated

**Листинг кода**

import sympy  
import numpy  
from matplotlib import pyplot  
from prettytable import PrettyTable  
from sympy import true  
  
x = sympy.Symbol('x')  
start = -1  
finish = 1  
y\_arrays = []  
k = 0.2  
T = 1  
Y2 = []  
g1 = lambda t: 0  
g2 = lambda t: 0  
f = lambda x: 1 - x\*\*2  
data\_table = []  
  
  
def solve(tau, h, t, last\_layer = None):  
 fx = 1  
 A = []  
 B = []  
 Y = []  
 i = 0  
 if true:  
 pass  
 xi = start  
 while xi < finish - 1e-5:  
 if t != 0:  
 a = -k \* tau  
 b = h\*\*2 + 2 \* k \* tau  
 c = -k \* tau  
 d = tau \* h\*\*2 \* fx + h\*\*2 \* last\_layer[i]  
  
 if i == 0:  
 Ai = - c / b  
 Bi = (d - a \* g1(t)) / b  
 else:  
 Ai = -c / (b + a \* A[-1])  
 Bi = (d - a \* B[-1]) / (b + a \* A[-1])  
  
 A.append(Ai)  
 B.append(Bi)  
 else:  
 Y.append(f(xi))  
 xi += h  
 i += 1  
  
 if t == 0:  
 Y.append(finish \*\* 2)  
 return Y  
  
 Y = [0] \* 11  
  
 Y[0] = g1(t)  
 Y[-2] = (A[-2] \* h \* g2(t) + B[-2]) / (1 - A[-2])  
 Y[-1] = Y[-2] + 4\*h # yb  
  
 while i > 1:  
 i -= 1  
 Y[i] = A[i] \* Y[i + 1] + B[i]  
  
 return Y  
  
  
def solve2(tau, h, t, j):  
 Y = [0] \* 11  
 fx = 1  
 xi = start  
  
 if j == 0:  
 i = 0  
 while xi < finish:  
 Y[i] = f(xi)  
 i += 1  
 xi += h  
 else:  
 Y[0] = g1(t)  
 for i in range(1, 10):  
 Y[i] = Y2[j - 1][i] + tau / h \*\* 2 \* (Y2[j - 1][i + 1] - 2 \* Y2[j - 1][i] + Y2[j - 1][i - 1]) #+ tau \* fx  
 xi += h  
  
 Y[-2] = Y2[j - 1][-2] + tau / h \*\* 2 \* (2 \* h \* g2(t) - Y2[j - 1][-2] + Y2[j - 1][-3]) #+ tau \* fx  
 Y[-1] = Y[-2] + 2 \* h \*g2(t)  
  
 Y2.append(Y)  
  
  
def implicit\_function(h, tau):  
 y\_arrays.clear()  
 t = 0  
 for i in range(0, 10):  
 if i == 0:  
 y\_arrays.append(solve(tau, h, t))  
 else:  
 y\_arrays.append(solve(tau, h, t, y\_arrays[-1]))  
 print(t)  
 if t == 0:  
 t = 2 \* tau  
 else:  
 t \*= 2  
 if t > T:  
 break  
  
  
def explicit\_function(h, tau):  
 t = 0  
 Y2.clear()  
 for i in range(0, 10):  
 solve2(tau, h, t, i)  
 print(t)  
 if t == 0:  
 t = 2 \* tau  
 else:  
 t \*= 2  
 if t > T:  
 break  
  
  
def print\_grafic(array, h, method, t):  
 x = numpy.arange(start, finish + h, h)  
 pyplot.plot(x, array)  
 pyplot.title(method + " метод (t = " + t + ")")  
 pyplot.show()  
  
  
def create\_table():  
 table = PrettyTable(["N", "tau", "S(t=2tau)", "S(t=4tau)", "Mmod(t=2tau)", "Mmod(t=4tau)"])  
 for data in data\_table:  
 table.add\_row(data)  
 print(table)  
  
  
def my\_round(value):  
 return "{:.5f}".format(value)  
  
  
def main():  
 h = (finish - start) / 10  
 tau = 0.5 \* h \*\* 2 / k  
 tau\_test = tau  
 N = h  
 j = 1  
 for i in range(0, 4):  
 implicit\_function(h, tau\_test)  
 max\_2\_tau = 0  
 for u in y\_arrays[1]:  
 if max\_2\_tau < u:  
 max\_2\_tau = u  
  
 max\_4\_tau = 0  
 for u in y\_arrays[2]:  
 if max\_4\_tau < u:  
 max\_4\_tau = u  
  
 data\_table.append([N, my\_round(tau\_test), my\_round(y\_arrays[1][j]), my\_round(y\_arrays[2][j]), max\_2\_tau, max\_4\_tau])  
 N \*= 2  
 j += 2  
 tau\_test /= 4  
  
 create\_table()  
  
 implicit\_function(h, tau)  
 t = 0  
 for array in y\_arrays:  
 print\_grafic(array, h, "Неявный", "{t:.9f}".format(t=t))  
 if t == 0:  
 t = 2 \* tau  
 else:  
 t \*= 2  
  
 data\_table.clear()  
 tau\_test = tau  
 N = h  
 j = 1  
 for i in range(0, 4):  
 explicit\_function(h, tau\_test)  
 max\_2\_tau = 0  
 for u in Y2[1]:  
 if max\_2\_tau < u:  
 max\_2\_tau = u  
  
 max\_4\_tau = 0  
 for u in Y2[2]:  
 if max\_4\_tau < u:  
 max\_4\_tau = u  
  
 data\_table.append([N, my\_round(tau\_test), my\_round(Y2[1][j]), my\_round(Y2[2][j]), max\_2\_tau, max\_4\_tau])  
 N \*= 2  
 j += 2  
 tau\_test /= 4  
  
 create\_table()  
 explicit\_function(h, tau)  
 t = 0  
 for array in Y2:  
 print\_grafic(array, h, "Явный", "{t:.9f}".format(t=t))  
 if t == 0:  
 t = 2 \* tau  
 else:  
 t \*= 2  
  
 tau = h \*\* 2 / 6  
 data\_table.clear()  
 tau\_test = tau  
 N = h  
 j = 1  
 for i in range(0, 4):  
 implicit\_function(h, tau\_test)  
 max\_2\_tau = 0  
 for u in y\_arrays[1]:  
 if max\_2\_tau < u:  
 max\_2\_tau = u  
  
 max\_4\_tau = 0  
 for u in y\_arrays[2]:  
 if max\_4\_tau < u:  
 max\_4\_tau = u  
  
 data\_table.append([N, my\_round(tau\_test), my\_round(y\_arrays[1][j]), my\_round(y\_arrays[2][j]), max\_2\_tau, max\_4\_tau])  
 N \*= 2  
 j += 2  
 tau\_test /= 4  
  
 create\_table()  
 explicit\_function(h, tau)  
 t = 0  
 for array in Y2:  
 print\_grafic(array, h, "Явный", "{t:.9f}".format(t=t))  
 if t == 0:  
 t = 2 \* tau  
 else:  
 t \*= 2  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

**Выводы**

В данной лабораторной работе я ознакомился с наиболее часто применяемыми способами аппроксимации граничных условий второго рода (граничных условий (ГУ) Неймана) в методе конечных разностей (на примере ГУ для одномерного нестационарного уравнения теплопроводности). Нашёл приближенное решение начально-краевой задачи для уравнения теплопроводности. Построил графики зависимости приближенного решения от *х* при