Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Кафедра информатики

Лабораторная работа № 12

по дисциплине «Методы численного анализа»

по теме «Решение краевых задач методом разностных аппроксимаций»

Вариант 12

Выполнил:

студент гр. 853502

Шаплыко Н. А.

Проверил:

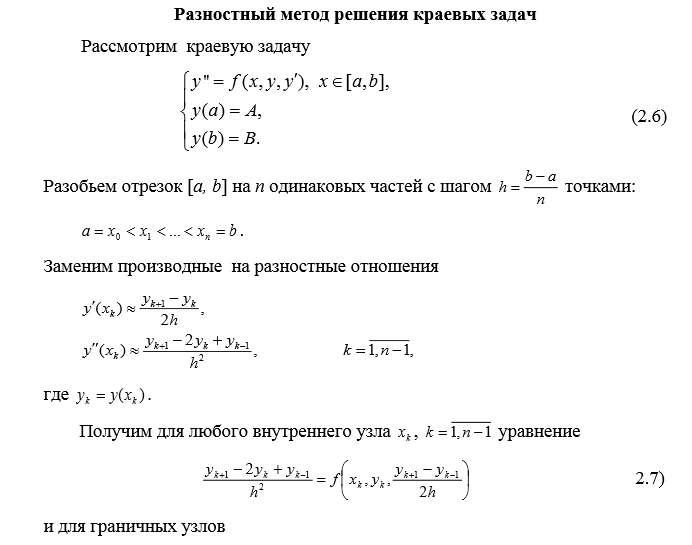
Анисимов В.Я.

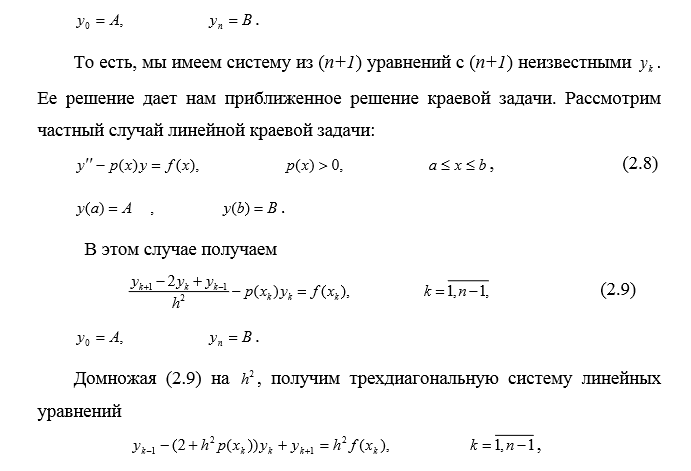
Минск, 2020

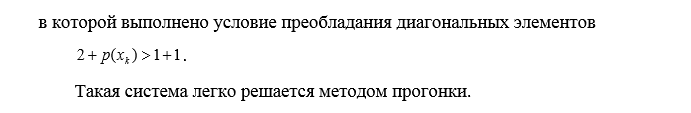
**Цель работы**

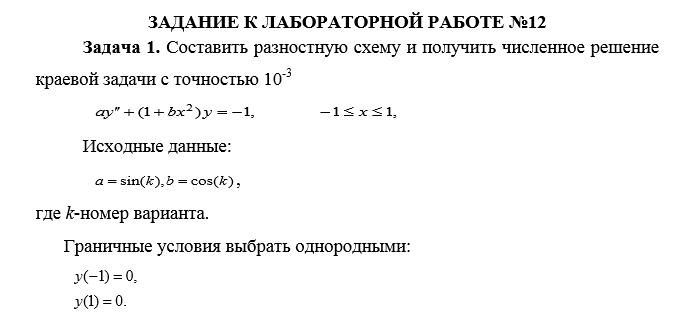
1. Изучить метод разностных аппроксимаций, составить алгоритм метода и программу их реализации, получить численное решение заданной краевой задачи;
2. Составить алгоритм решения краевых задач указанными методами, применимыми для организации вычислений на ПЭВМ;
3. Составить программу решения краевых задач по разработанному алгоритму;
4. Выполнить тестовые примеры и проверить правильность работы программ.

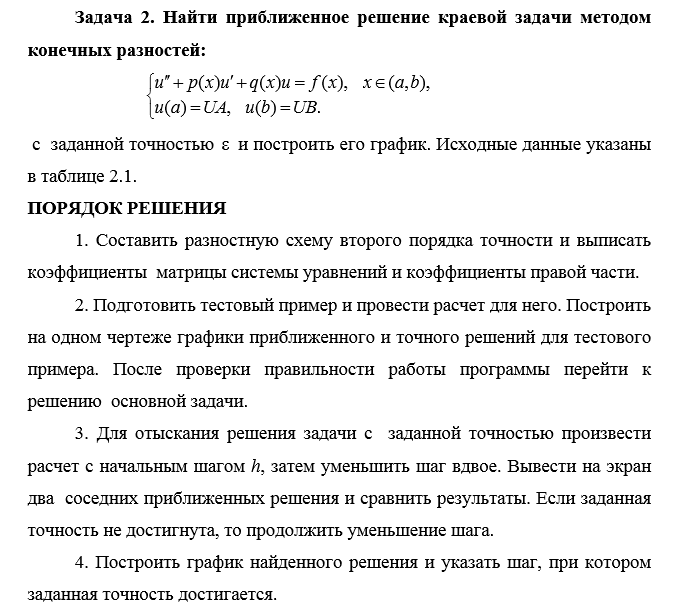
**Краткие теоретические сведения**

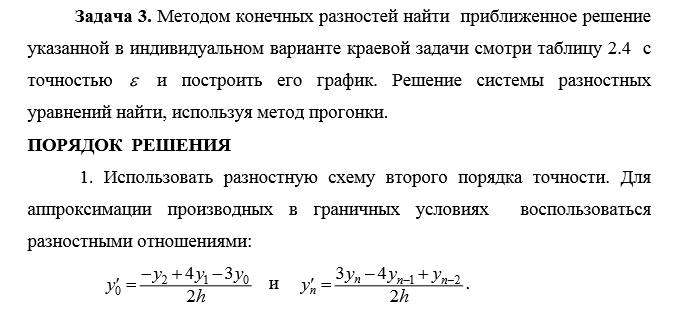


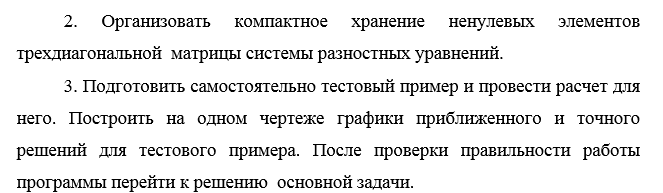


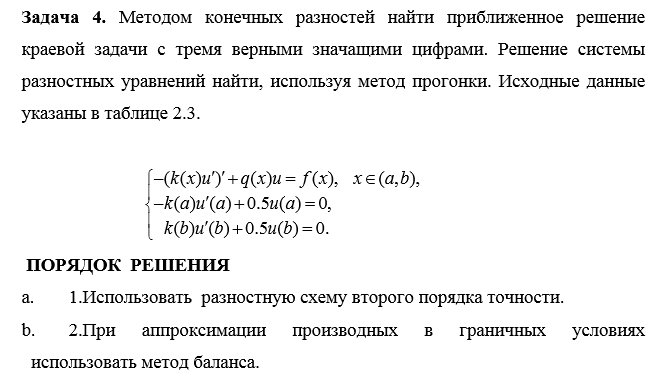












**Результаты**

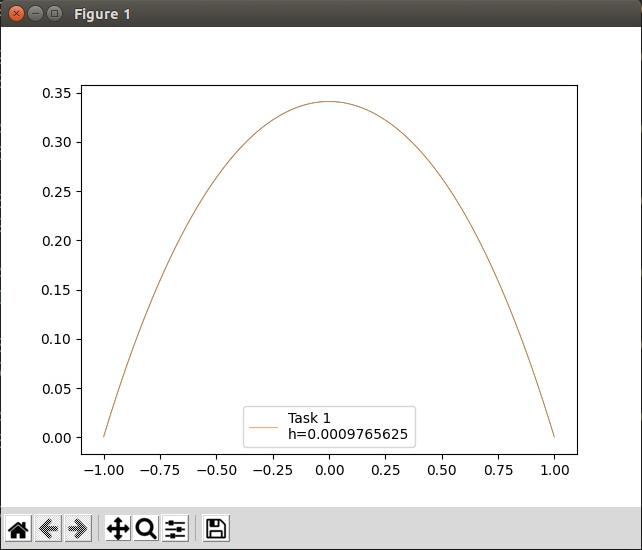


Рисунок 1. Решение задачи №1

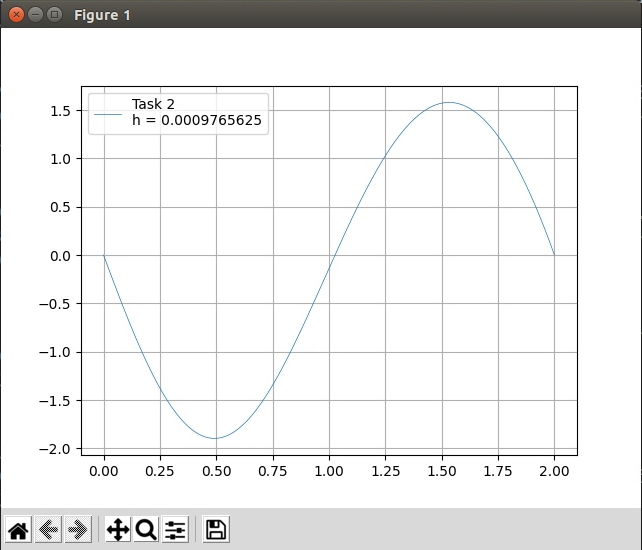


Рисунок 2. Решение задачи №2

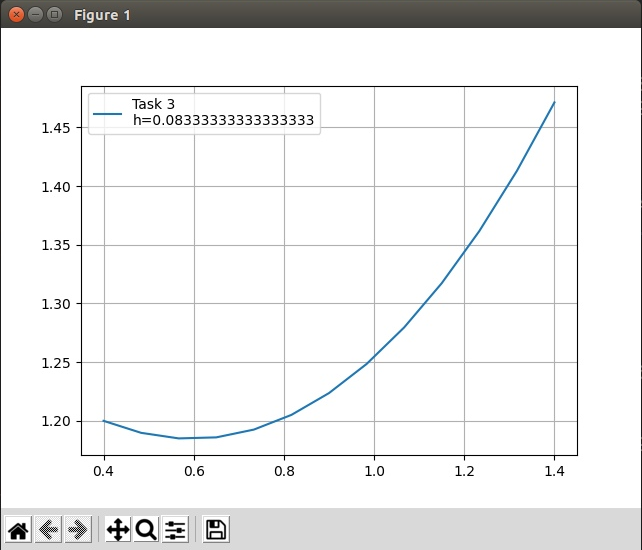


Рисунок 3. Решение задачи №3

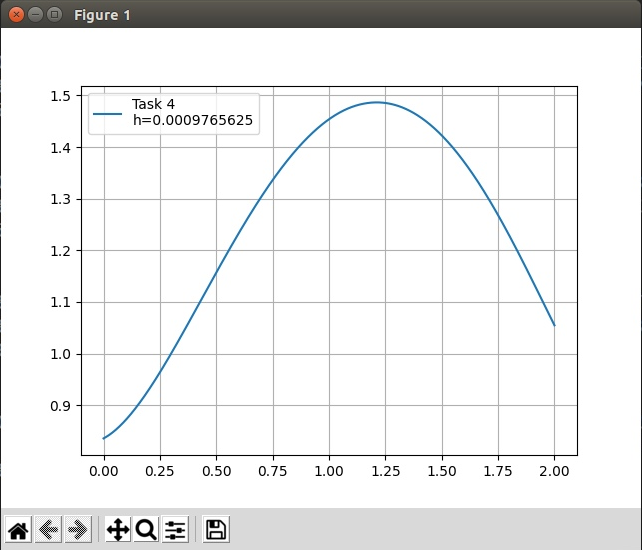


Рисунок 4. Решение задачи №4

**Выводы**

В ходе работы:

* изучил метод разностных аппроксимаций
* составил алгоритм метода и программу их реализации
* получил численное решение заданной краевой задачи
* составил алгоритм решения краевых задач указанными методами, применимыми для организации вычислений на ПЭВМ
* составил программу решения краевых задачпо разработанному алгоритму
* выполнил тестовые примеры и проверить правильность работы программ.
* Были изучены методы разностных аппроксимаций со вторым порядком точности для краевых задач первого и третьего рода.
* Для приближения краевых условий использовались правые и левые формулы конечных разностей второго порядка и метод баланса (приближение при помощи метода центральных прямоугольников).

**Листинг кода**

import numpy as np

import math

from matplotlib import pyplot as plt

def sweep\_method(a, b, c, d):

AlphaS = [-c[0] / b[0]]

BetaS = [d[0] / b[0]]

GammaS = [b[0]]

n = len(d)

result = [0 for i in range(n)]

for i in range(1, n - 1):

GammaS.append(b[i] + a[i] \* AlphaS[i - 1])

AlphaS.append(-c[i] / GammaS[i])

BetaS.append((d[i] - a[i] \* BetaS[i - 1]) / GammaS[i])

GammaS.append(b[n - 1] + a[n - 1] \* AlphaS[n - 2])

BetaS.append((d[n - 1] - a[n - 1] \* BetaS[n - 2]) / GammaS[n - 1])

result[n - 1] = BetaS[n - 1]

for i in reversed(range(n - 1)):

result[i] = AlphaS[i] \* result[i + 1] + BetaS[i]

return result

def check\_eps(current, prev, eps):

eps\_t = max([math.fabs(current[i \* 2] - prev[i]) for i in range(len(prev))])

print('Check: ', eps\_t)

if eps\_t > eps:

return False

return True

def deep\_copy(system):

return [item for item in system]

def get\_h(a, b, n):

return (b - a) / n

def task1():

iteration\_count = 0

prev = []

current = []

eps = 1e-3

h = 0

n = 32

A = -1

B = 1

first = 0

last = 0

while True:

h = get\_h(A, B, n)

points = list(np.linspace(A, B, n + 2))

print(points)

del points[0]

del points[-1]

a = [1 for x in points]

b = [-(2 + (h \*\* 2) \* (1 + math.sin(12) \* (x \*\* 2)) / math.cos(12)) for x in points]

c = [1 for x in points]

d = [-(h \*\* 2) for x in points]

d[0] = d[0] - a[0] \* first

d[-1] = d[-1] - c[-1] \* last

current = sweep\_method(a, b, c, d)

if iteration\_count != 0 and check\_eps(current, prev, eps):

break

print('h = ', h)

prev = deep\_copy(current)

iteration\_count += 1

n \*= 2

print("\nh = ", h)

print("Количество итераций = ", iteration\_count)

print("Количество точек = ", n)

print("y = ", current)

plt.plot(points[::2], prev, linewidth=0.5)

points.insert(0, A)

points.append(B)

current.insert(0, first)

current.append(last)

plt.plot(points, current, linewidth=0.5, label='Task 1\nh={}'.format(h))

plt.legend()

plt.show()

def task2():

A = 0

B = 2

n = 8

first = 0

last = 0

eps = 0.01

test = False

iteration\_count = 0

prev = list()

while True:

h = get\_h(A, B, n)

points = list(np.linspace(A, B, n + 2))

del points[0]

del points[-1]

a = [(1 - (-0.5 + math.sin(x)) \* h / 2) for x in points]

b = [(8 / (1 + 0.25 \* (x \*\* 2))) \* (h \*\* 2) - 2 for x in points]

c = [(1 + (-0.5 + math.sin(x)) \* h / 2) for x in points]

d = [5 \* (1 - x \*\* 2) \* (h \*\* 2) for x in points]

if test:

a = [1 / (h \*\* 2) - math.tan(x) / (2 \* h) for x in points]

b = [-2 / (h \*\* 2) + math.cos(x) \*\* 2 for x in points]

c = [1 / (h \*\* 2) + math.tan(x) / (2 \* h) for x in points]

d = [0 for x in points]

d[0] = d[0] - a[0] \* first

d[-1] = d[-1] - c[-1] \* last

current = sweep\_method(a, b, c, d)

if iteration\_count != 0 and check\_eps(current, prev, eps):

break

print("\nh = ", h)

print("y = ", current)

print("prev = ", prev)

prev = deep\_copy(current)

iteration\_count += 1

n \*= 2

print("\nh = ", h)

print("Количество итераций = ", iteration\_count)

print("Количество точек = ", n)

print("y = ", current)

if test:

points = list(range(A, B, h))

y = [-(-10 + math.cos(math.sin(1))) \* math.sin(math.sin(x)) / math.sin(math.sin(1)) + math.cos(math.sin(x)) for

x in points]

plt.plot(points, y, label="Точно решение", linewidth=0.5)

plt.legend()

points.insert(0, A)

points.append(B)

current.insert(0, first)

current.append(last)

plt.plot(points, current, label="Task 2\nh = {}".format(h), linewidth=0.5)

plt.legend()

plt.grid(True)

plt.show()

def task3():

A = 0.4

B = 1.4

n = 6

eps = 0.1

first = 2

last = 5

iteration\_count = 0

points = []

current = []

prev = []

h = 0

while True:

h = get\_h(A, B, n)

points = list(np.linspace(A, B, n + 1))

del points[0]

a = [(1 / h \*\* 2) + 0.5 \* x / (2 \* h) for x in points]

b = [-(2 / h \*\* 2) + 1 for x in points]

c = [(1 / h \*\* 2) + (-0.5 \* x / (2 \* h)) for x in points]

d = [2 for x in points]

d[0] = d[0] - a[0] \* 1.2

a[-1] = a[-1] - c[-1] / (h + 3)

b[-1] = b[-1] + 4 \* c[-1] / (h + 3)

d[-1] = d[-1] - c[-1] \* 3.2 \* h / (h + 3)

c[-1] = 0

current = sweep\_method(a, b, c, d)

if iteration\_count != 0 and check\_eps(current, prev, eps):

break

print("\nh = ", h)

print("y = ", current)

print("prev = ", prev)

prev = deep\_copy(current)

iteration\_count += 1

n \*= 2

print("\nh = ", h)

print("Количество итераций = ", iteration\_count)

print("Количество точек = ", n)

print("y = ", current)

plt.figure()

points.insert(0, A)

current.insert(0, 1.2)

plt.plot(points, current, label='Task 3\nh={}'.format(h))

plt.legend()

plt.grid(True)

plt.show()

def task4():

A = 0

B = 2

C = 1.125

k1 = 1.8

k2 = 0.6

q1 = 6.5

q2 = 7.8

eps = 0.001

n = 16

iteration\_count = 0

prev = []

points = []

current = []

while True:

h = get\_h(A, B, n)

points = list(np.linspace(A, B, n))

a = [-k1 / h \*\* 2 if x < C else -k2 / h \*\* 2 for x in points[1:-1]]

b = [2 \* k1 / h \*\* 2 + q1 if x < C else 2 \* k2 / h \*\* 2 + q2 for x in points[1:-1]]

c = [-k1 / h \*\* 2 if x < C else -k2 / h \*\* 2 for x in points[1:-1]]

d = [8 \* x \* (2.5 - x) for x in points[1:-1]]

a.insert(0, 0)

b.insert(0, k1 / h + 0.5)

c.insert(0, -k1 / h)

d.insert(0, 0)

a.append(-k2 / h)

b.append(k2 / h + 0.5)

c.append(0)

d.append(0)

current = sweep\_method(a, b, c, d)

if iteration\_count != 0 and check\_eps(current, prev, eps):

break

print("\nh = ", h)

print("y = ", current)

print("prev = ", prev)

prev = deep\_copy(current)

iteration\_count += 1

n \*= 2

print("\nh = ", h)

print("Количество итераций = ", iteration\_count)

print("Количество точек = ", n)

print("y = ", current)

plt.plot(points, current, label="Task 4\nh={}".format(h))

plt.legend()

plt.grid(True)

plt.show()

def main():

task1()

task2()

task3()

task4()

if \_\_name\_\_ == "\_\_main\_\_":

main()