|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ «Информатика и системы управления»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Лабораторная работа № 2**

|  |  |
| --- | --- |
| **Тема «Многомерная интерполяция. Алгоритм и программа построения интерполяционного полинома Ньютона»**  **Дисциплина: Вычислительные алгоритмы**  **Студент Кузин Антон**  **Группа ИУ7-42Б**  **Оценка (баллы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Преподаватель Градов В.М.** |  |

Москва.

2020 г.

**Цель работы:**

Целью данной лабораторной работы было реализовать алгоритм многомерной интерполяции с использованием.

**Условие:**

**Входные данные:** пункт меню и необходимые аргументы; текстовый файл, содержащий значения функции.

**Выходные данные:** результат интерполяции или сообщение о невозможности получения результата.

**Алгоритм:**

Изначально выбирается конфигурация, на данном этапе проверяется, находятся ли x в таблице и y, в противном случае выводится сообщение, о том, что экстраполяция запрещена.

После чего производится интерполяция по фиксированному x для значений y, входящих в конфигурацию, таким образом выделяется массив значений некой функции u(y), а затем производится интерполяция по получившимся значениям.

**Двумерный полином Ньютона**

class Table:

def \_\_init\_\_(self, x, y, z):

self.x = x

self.y = y

self.z = z

class Fy:

def \_\_init\_\_(self, x: [], y: []):

self.arg = x

self.val = y

def read(filename):

file = open(filename, "r")

y = []

z = []

x = list(map(float, file.readline()[1:].split()))

while 1:

line = file.readline()

if line == '':

break

nums = line.split()

y.append(float(nums[0]))

z\_line = []

for i in range(1, len(nums)):

z\_line.append(float(nums[i]))

z.append(z\_line)

file.close()

return x, y, z

def get\_borders(args, val, power):

pos = -1

for i in range(1, len(args)):

if args[i] > args[i - 1]:

if args[i] >= val > args[i - 1]:

pos = i

break

if args[i] < args[i - 1]:

if args[i] <= val < args[i - 1]:

pos = i

break

if pos == -1:

return None, None

left\_border = pos - (power // 2) - 1

right\_border = pos + (power // 2 + power % 2)

if left\_border < 0:

left\_border = 0

right\_border = power + 1

elif right\_border > len(args):

left\_border = len(args) - power

right\_border = len(args)

return left\_border, right\_border

def chose\_configuration(xy: Fy, val, power, shift):

left\_border, right\_border = get\_borders(xy.arg, val, power)

if left\_border is None:

return None, None

work\_array\_x = []

poly\_coefs = [[0 for j in range(power - i + 1)] for i in range(power + 1)]

for i in range(left\_border, right\_border):

work\_array\_x.append(xy.arg[i])

poly\_coefs[0][i - left\_border] = xy.val[i - shift]

return work\_array\_x, poly\_coefs

def get\_poly\_coefs(args, poly\_coefs, power):

for i in range(1, power + 1):

for j in range(power - i + 1):

poly\_coefs[i][j] = (poly\_coefs[i - 1][j + 1] - poly\_coefs[i - 1][j]) / (args[j + i] - args[j])

return poly\_coefs

def calculate\_val(args, poly\_coefs, arg, power):

res = poly\_coefs[0][0]

mult = 1

for i in range(power):

mult \*= (arg - args[i])

res += mult \* poly\_coefs[i + 1][0]

return res

def interpolate(xy: Fy, arg, power, shift):

work\_array\_x, poly\_coefs = chose\_configuration(xy, arg, power, shift)

if work\_array\_x is None:

return None

poly\_coefs = get\_poly\_coefs(xy.arg, poly\_coefs, power)

return calculate\_val(xy.arg, poly\_coefs, arg, power)

def interpolate\_3d(wt: Table, x, y, nx, ny):

left\_border, right\_border = get\_borders(wt.y, y, ny)

if left\_border is None:

return

temp\_res = []

for i in range(left\_border, right\_border):

tr = interpolate(Fy(wt.x, wt.z[i]), x, nx, 0)

if tr is None:

return

temp\_res.append(tr)

return interpolate(Fy(wt.y, temp\_res), y, ny, left\_border)

def f(x, y):

return x \* x + y \* y

def form\_new\_table\_f():

name = input("Введите имя файла, в который будет сохранена таблица: ")

file = open(name, "w")

arg\_min = float(input("Введите минимальное значение аргумента: "))

arg\_max = float(input("Введите максимальное значение аргумента: "))

step = float(input("Введите шаг: "))

file.write('x')

i = arg\_min

while i < arg\_max:

file.write(" %.3f" % i)

i += step

file.write('\n')

i = arg\_min

while i < arg\_max:

file.write("%.3f" % i)

j = arg\_min

while j < arg\_max:

file.write(' %.3f' % f(i, j))

j += step

file.write("\n")

i += step

file.close()

cur\_table = Table(None, None, None)

file\_name = 'two\_sqr'

while 1:

print("\n\nМеню: ")

print("1. Интерполяция")

print("2. Выбрать файл")

print("3. Сформировать файл по заданной функции")

print("4. Выход")

ch = int(input("\nВаш выбор: "))

if ch < 0 or ch > 5:

pass

elif ch == 1:

if len(file\_name) == 0:

print("Файл не выбран")

continue

if cur\_table.x is None or len(cur\_table.x) == 0:

table = read(file\_name)

x = float(input("Введите X для интерполяции: "))

y = float(input("Введите Y для интерполяции: "))

nx = int(input("Введите степень полинома по Х: "))

ny = int(input("Введите степень полинома по Y: "))

res = interpolate\_3d(cur\_table, x, y, nx, ny)

if res is None:

print("Экстраполяция запрещена")

continue

print("Результат интерполяции: %.3f" % res)

elif ch == 2:

file\_name = input("Введите название файла: ")

x, y, z = read(file\_name)

cur\_table = Table(x, y, z)

elif ch == 3:

form\_new\_table\_f()

elif ch == 4:

break