|  |  |
| --- | --- |
| **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 № 1***

**Тема:** Построение и программная реализация алгоритма полиномиальной интерполяции табличных функций.

**Студент:** Динь Вьет Ань

**Группа:** ИУ7И - 44Б

**Преподаватель:** Градов В.М.

*Москва*

*2022г*

**Цель работы:** Получение навыков построения алгоритма интерполяции таблично заданных функций полиномами Ньютона и Эрмита.

**Исходные данные**

1. Таблица функции и её производных

|  |  |  |
| --- | --- | --- |
| X | Y | Y` |
| 0 | 1 | -1 |
| 0.15 | 0.838771 | -1.14944 |
| 0.30 | 0.655336 | -1.29552 |
| 0.45 | 0.450447 | -1.43497 |
| 0.60 | 0.225336 | -1.56464 |
| 0.75 | -0.018310 | -1.68164 |
| 0.90 | -0.278390 | -1.78333 |
| 1.05 | -0.552430 | -1.86742 |

2. Степень аппроксимирующего полинома Ньютона n или количество узлов для полинома Эрмита.

3. Значение аргумента, для которого выполняется интерполяция.

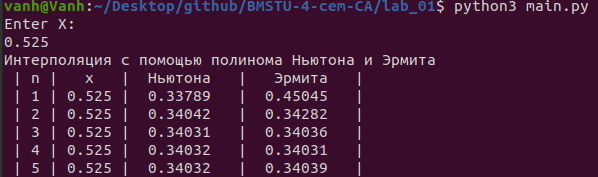
**Код программы**

|  |
| --- |
| import math  def parse\_table():'  table = []  data\_file = open("data.txt", "r")  line = [float(value) for value in data\_file.read().split()]  data\_file.close()  if len(line) == 0 or len(line) % 3:  print("Invalid input")  exit(0)  for i in range (len(line) // 3):  table.append([line[i \* 3], line[i \* 3 + 1], line[i \* 3 + 2]])  table.sort()  count = len(table) - 1  arr\_x = []  arr\_y = []  for i in range(len(table)):  arr\_x.append(table[i][0])  arr\_y.append(table[i][1])  return table, arr\_x, arr\_y  def input\_x():  print("Enter X: ")  flag = 0  x = 0  while flag == 0:  x = input()  try:  val = float(x)  flag = 1  except ValueError:  print("Some error! Try again")  return float(x)  def find\_x0\_xn(data, power, arg):  index\_x = 0  while arg > data[index\_x][0]:  index\_x += 1  index\_x0 = index\_x - power // 2 - 1  index\_xn = index\_x + (power // 2) + (power % 2) - 1  if index\_xn > len(data) - 1:  index\_x0 -= index\_xn - len(data) + 1  index\_xn = len(data) - 1  elif index\_x0 < 0:  index\_xn += -index\_x0  index\_x0 = 0  return index\_x0, index\_xn  def div\_diff(x, y, node):  pol = []  for i in range(node):  pol.append([0] \* (node + 1))  for i in range(node):  pol[i][0], pol[i][1] = x[i], y[i]  i = 2  new\_node = node - 1  while i < (node + 1):  j = 0  while j < new\_node:  pol[j][i] = round((pol[j + 1][i - 1] - pol[j][i - 1]) \  / (pol[i - 1][0] - pol[0][0]), 5)  j += 1  i += 1  new\_node -= 1  return pol  def polinom(x, y, node, arg):  pol = div\_diff(x, y, node)  y = pol[0][1]  i = 2  while i < node + 1:  j = 0  p = 1  while j < i - 1:  p \*= (arg - pol[j][0])  j += 1  y += pol[0][i] \* p  i += 1  return y  def hermit\_interpolate(data, node, arg, coords\_x):  x0, xn = find\_x0\_xn(data, node // 2, arg)  data = data[x0 : xn + 1]  pol = []  if node == 0:  return pol, x0  for i in range(2 \* len(data)):  pol.append([0] \* (node + 2))  i = 0  for j in range(len(data)):  pol[i][0] = data[j][0]  pol[i][1] = data[j][1]  pol[i][2] = data[j][2]  i += 1  pol[i][0] = data[j][0]  pol[i][1] = data[j][1]  i += 1  i = 2  for j in range(len(pol) - 1):  if j % 2 == 1:  pol[j][i] = (pol[j][1] - pol[j + 1][1]) \  / (pol[j][0] - pol[j + 1][0])  i = 3  new\_node = node - 2  while i < len(pol):  j = 0  while j < new\_node:  pol[j][i] = round((pol[j + 1][i - 1] - pol[j][i - 1]) \  / (pol[i - 1][0] - pol[0][0]), 5)  j += 1  i += 1  new\_node -= 1  return pol, x0  def result\_polynomial(pol, node, arg): #here is function to calculate value for given x  y = pol[0][1]  i = 2  while i < node + 1:  j = 0  p = 1  while j < i - 1:  p \*= (arg - pol[j][0])  j += 1  y += pol[0][i] \* p  i += 1  return y  def main():  data, coords\_x, coords\_y = parse\_table()  x = input\_x()  arr\_n = [1, 2, 3, 4, 5]  print("Интерполяция с помощью полинома Ньютона и Эрмита\n",  "| n | x | Ньютона | Эрмита |")  for n in arr\_n:  print(" |{:^3}|{:^7}|".format(n, round(x, 4)), end="")  flag = False  for i in range(0, len(data)):  if x == data[i][0]:  print("{:^12}|{:^12}|".format(round(data[i][1], 5), round(data[i][1], 5)))  flag = True  if not flag:  x0, xn = find\_x0\_xn(data, n, x)  ax = coords\_x[x0 : xn + 1]  ay = coords\_y[x0 : xn + 1]  if len(ax):  my\_root = polinom(ax, ay, n + 1, x)  print("{:^12}|".format(round(my\_root, 5)), end="")    # полином Эрмита  pol, x0 = hermit\_interpolate(data, n, x, coords\_x)  my\_root2 = 0  if not n:  my\_root2 = data[x0][1]  else:  my\_root2 = result\_polynomial(pol, n, x)  print("{:^12}|".format(round(my\_root2, 5)), end="\n")  print("Обратная инерполяция с помощью полинома Ньютона\n",  "| n | y | Корень |")  for n in arr\_n:  print(" | {} | {} |".format(n, 0), end="")  flag = False  for i in range(0, len(data)):  if 0 == data[i][1]:  print("{:^11}|".format(data[i][0]))  flag = True  if not flag:  n\_data = []  for i in range(len(data)):  n\_data.append([data[i][1], data[i][0], data[i][2]])  n\_data.sort()  coords\_y.clear()  coords\_x.clear()  for i in range(len(n\_data)):  coords\_x.append(n\_data[i][0])  coords\_y.append(n\_data[i][1])  x0, xn = find\_x0\_xn(n\_data, n, 0)  ax = coords\_x[x0 : xn + 1]  ay = coords\_y[x0 : xn + 1]  if len(ax):  my\_root = polinom(ax, ay, n + 1, 0)  print(" {} |".format(round(my\_root, 5)))  if \_\_name\_\_ == "\_\_main\_\_":  main() |

**Результаты работы**

1. Значения y(x) при степенях полиномов Ньютона и Эрмита n= 1, 2, 3, 4, 5 при фиксированном x=0.525 (середина интервала 0.45- 0.60). Результаты свести в таблицу для сравнения полиномов.

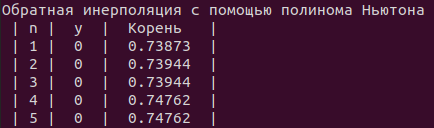
(Вывод программы)



|  |  |  |  |
| --- | --- | --- | --- |
| n | x | п. Ньютона | п. Эрмита |
| 1 | 0.525 | 0.33789 | 0.34282 |
| 2 | 0.525 | 0.34042 | 0.34282 |
| 3 | 0.525 | 0.34031 | 0.34036 |
| 4 | 0.525 | 0.34032 | 0.34031 |
| 5 | 0.525 | 0.34032 | 0.34042 |

2. Найти корень заданной выше табличной функции с помощью обратной интерполяции, используя полином Ньютона.

(Вывод программы)



|  |  |  |
| --- | --- | --- |
| n | x | Корень |
| 1 | 0 | 0.73873 |
| 2 | 0 | 0.73944 |
| 3 | 0 | 0.73944 |
| 4 | 0 | 0.74762 |
| 5 | 0 | 0.74762 |

**Вопросы при защите лабораторной работы**

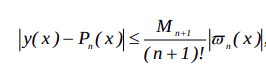
***1. Будет ли работать программа при степени полинома n=0?***

Да, работать программа с такой степенью будет, как расчет интерполяции с помощью полинома Ньютона, так и с помощью полинома Эрмита. Функция вернет значение из таблицы в точке, которая находится ближе к заданному аргументу. Но точность при такой конфигурации будет низкой.  
 Расчет корня с помощью обратной интерполяции тоже будет работать - это значение аргумента из данной таблицы, в которой функция принимает значение ближе к нулю.

***2. Как практически оценить погрешность интерполяции? Почему сложно применить для этих целей теоретическую оценку?***

Практически оценить погрешность интерполяции можно при помощи оценки первого отброшенного члена в полиноме Ньютона. При этом в полиноме остаются члены, которые больше заданной погрешности расчетов.

Теоретическую погрешность многочлена Ньютона можно оценить с помощью формулы (где используются производные данной функции):

, где  - максимальное значение производной интерполируемой функции, а также 

Именно поэтому теоретическую погрешность сложно оценить.

***3. Если в двух точках заданы значения функции и ее первых производных, то полином какой минимальной степени может быть построен на этих точках?***

При данном условии можно построить полиномы Эрмита 0, 1, 2 и 3 степени а полиномы Ньютона - 0 и 1 степени. Минимальная степень равна 0.

***4. В каком месте алгоритма построения полинома существенна информация об упорядоченности аргумента функции (возрастает, убывает)?***

Информация об упорядоченности аргумента функции существенна при выборе приближенного интервала значений (из (n+1) узлов, которые по возможности расположены симметрично относительно заданного аргумента).

Если аргумент будет неупорядоченным, то значение функции получится с низкой точностью или совсем неверным.

***5. Что такое выравнивающие переменные и как их применить для повышения точности интерполяции?***

Если функция, а точнее ее разделенные разности, значительно меняются на нескольких интервалах, то интерполяция обобщенным многочленом обычно не будет точной для дифференцирования данной функции. Поэтому для таких функций используется квазилинейная интерполяция, которая производится при помощи выравнивающих переменных. То есть выравнивающие переменные используются для того, чтобы повысить точность вычисления производной функции.

***6. Будет ли работать ваша программа при произвольном неупорядоченном расположении узлов в исходной таблице?***

Да потому что все узлы сортируются после ввода

***7. Принципиально ли для корректной работы вашего алгоритма, чтобы узлы были расположены по возрастанию?***

Да.

***8. Что будет происходить с точностью интерполяции по мере продвижения от центра к краям таблицы?***

Точность при такой конфигурации будет низкой.

***9. Можно ли использовать для обратной интерполяции полином Эрмита?***

Нет, потому что при обратной интерполяции мы рассматриваем функцию x(y) с y = 0. Если используемся полином Эрмита, нужно знать производные этой функции.