|  |  |
| --- | --- |
| **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:** Фам Минь Хиеу

**Группа:** ИУ7И - 42Б

**Преподаватель:** Градов В.М.

*Москва*

*2023 г*

**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.

**Листинг кода**

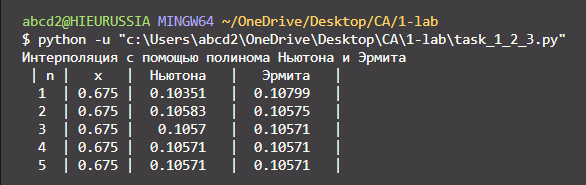
|  |
| --- |
| *def* read\_table(*fname*):      table = []      with open(fname, "r") as f:          table = []          while True:              line = f.readline().strip()              if (len(line)):                  table.append([*float*(x) for x in line.split()])              else:                  break      return table  *def* enter\_x():      print("Введите значение x, по которому вычисляется y : ", end = "")      while True:          try:              val = *float*(input())              break          except:              print("Ввод некорректно! x должно быть числом")      return val  *def* enter\_n(*table*):      print("Введите степень n аппроксимирующих полиномов Ньютона и Эрмита: ", end = "")      flag = 1      while flag:          try:              val = *int*(input())              if val < 0 or val > (len(table) - 1):                  print("n должно быть в интервале [{} , {}]".format(0, len(table) - 1))              else:                  flag = 0          except:              print("Ввод некорректно! n должно быть целым")      return val  *def* get\_data\_from\_user(*fname*):      table = read\_table(fname)      x = enter\_x()      n = enter\_n(table)      return table, n, x |

|  |
| --- |
| *def* find\_st\_end(*table*, *n*, *xo*):      x = [table[i][0] for i in range(len(table))]      if xo < x[0]:          return 0, n      elif xo > x[-1]:          return len(table) - 1 - n, len(table) - 1      else:          i = 0          while x[i] < xo:              i += 1          st = i - n // 2 - 1          end = i + n // 2 + (n % 2) - 1          if end > len(x) - 1:              st -= end - len(x) + 1              end = len(x) - 1          elif st < 0:              end += -st              st = 0          return st, end    *def* diff\_newton(*x*, *y*, *n*):      L = []      L.append(y)      for i in range(n):          curr\_col = L[i]          new\_col = []          for j in range(len(curr\_col) - 1):              new\_elem = round((curr\_col[j + 1] - curr\_col[j]) / (x[j + i + 1] - x[j]), 5)              new\_col.append(new\_elem)          L.append(new\_col)      return L  *def* get\_value(*x*, *diff*, *xo*):      res = 0      for i in range(len(diff)):          term = diff[i][0]          for j in range(i):              term \*= (xo - x[j])          res += term      return res  *def* newton(*table*, *xo*, *n*):      st, end = find\_st\_end(table, n, xo)      x = [table[i][0] for i in range(st, end + 1)]      y = [table[i][1] for i in range(st, end + 1)]      d = diff\_newton(x, y, n)  *# print(d)*      return get\_value(x, d, xo)  *def* diff\_hermite(*x*, *y*, *y1*, *n*):      L = []      L.append(y)      for i in range(n):          curr\_col = L[i]          new\_col = []          for j in range(len(curr\_col) - 1):              if (x[j] == x[j + i + 1]):                  new\_elem = y1[j // 2]              else:                  new\_elem = round((curr\_col[j + 1] - curr\_col[j]) / (x[j + i + 1] - x[j]), 5)              new\_col.append(new\_elem)          L.append(new\_col)      return L  *def* hermit(*table*, *xo*, *n*):      st, end = find\_st\_end(table, n // 2, xo)      table = table[st : end + 1]      x = []      y = []      y1 = []      for i in range (len(table)):          for j in range(2):              x.append(table[i][0])              y.append(table[i][1])          y1.append(table[i][2])      if n / 2 == 1:          x.pop()          y.pop()      d = diff\_hermite(x, y, y1, n)  *# print("hermit tab", d)*      return get\_value(x, d, xo) |

**Результаты работы**

1. Значения y(x) при степенях полиномов Ньютона и Эрмита n= 1, 2, 3, 4, 5 при фиксированном x=0.675 (середина интервала 0.6- 0.75).

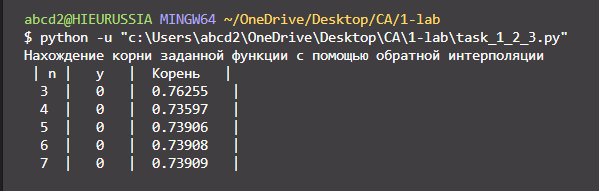
(Вывод программы)



|  |  |  |  |
| --- | --- | --- | --- |
| n | x | п. Ньютона | п. Эрмита |
| 1 | 0.675 | 0.10351 | 0.10799 |
| 2 | 0.675 | 0.10583 | 0.10575 |
| 3 | 0.675 | 0.1057 | 0.10571 |
| 4 | 0.675 | 0.10571 | 0.10571 |
| 5 | 0.675 | 0.10571 | 0.10571 |

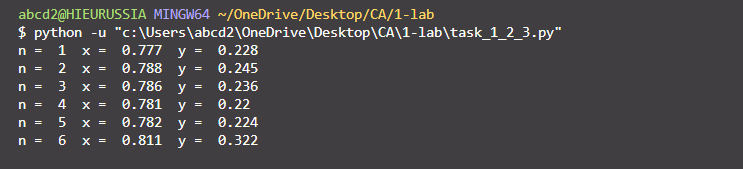
2. Найти корень заданной выше табличной функции с помощью обратной интерполяции, используя полином Ньютона.

(Вывод программы)



|  |  |  |
| --- | --- | --- |
| N | y | Корень |
| 3 | 0 | 0.76255 |
| 4 | 0 | 0.73597 |
| 5 | 0 | 0.73906 |
| 6 | 0 | 0.73908 |
| 7 | 0 | 0.73909 |

3. Решить систему нелинейных уравнений, основываясь на простой идее обратной интерполяции.



**Вопросы при защите лабораторной работы**

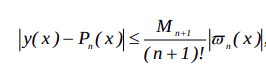
***1. Будет ли работать программа при степени полинома Ньютона n=0?***

Да, будет. Функция вернет значение из таблицы в точке, которая находится ближе ниже к заданному аргументу. Но точность при такой конфигурации будет низкой.

***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 одной точке заданы значения функции и ее первой, второй и третьей производных, а в другой точке заданы значения функции и ее первой производной, то какова будет степень полинома Эрмита, построенного на этих двух точках?***

При данном условии можно построить полиномы Эрмита 0, 1, 2, 3, 4, 5.

***4. Если в одной точке заданы функция и все ее производные, то, что собой представляет полином Эрмита, построенный в этой точке?***

H = y(xo) + (x-xo)y(xo, xo) + (x-xo)2 y(xo, xo, xo) + .....

***5. 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.

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

Выравнивающие переменные – Мы возьмем узлы, которые вблизи заданной точки x = xo, значит и слева и справа от заданный точки. В идеальном

случае, (n + 1) узлов симметрично расположенных относительно значения x.

То есть выравнивающие переменные используются для того, чтобы повысить точность вычисления производной функции.

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

Да потому что все узлы сортируются после ввода

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

Да.

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

Точность при такой конфигурации будет низкой.

**10. Всегда ли можно использовать для обратной интерполяции полином Эрмита?**

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

**11. Предложите алгоритм получения явной зависимости y(x) из неявной функции f(x,y)=0**

Пусть f(x, y) = k.

При фиксируемом значении x = xo. Возьмем несколько значений y. Мы получим таблицу значений y, k. Так с помощью интерполяции находим y = yo при k = 0. Так и далее заполним таблицу x, y.

Дальше с помощью интерполяции востанавляем полином, представляющий через разделенные разности в виде:

