­­­Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет Программной Инженерии и Компьютерной Техники

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



Лабораторная работа №5 по дисциплине  
«Вычислительная математика»

"Интерполяция функции"

Выполнил: Сафронов Егор Михайлович

Группа: P3213

Преподаватель: Малышева Татьяна Алексеевна

г. Санкт-Петербург

2022

Цель работы:

Решить задачу интерполяции, найти значения функции при заданных значениях аргумента, отличных от узловых точек.

Задание лабораторной работы:

1. Вычислительная реализация задачи:  
   a) Используя первую или вторую интерполяционную формулу Ньютона, первую или вторую интерполяционную формулу Гаусса вычислить значения функции при данных значениях аргумента (для значения Х1 и Х2, см. табл. 1–4).   
   b) Построить таблицу конечных разностей.
2. Программная реализация задачи:  
   a) Исходные данные задаются в виде: а) набора данных (таблицы x, y), б) на основе выбранной функции (например, ).   
   b) Вычислить приближенное значение функции для заданного значения аргумента, введенного с клавиатуры, указанными методами (см. табл.5).  
   c) Построить графики заданной функции с отмеченными узлами интерполяции и интерполяционного многочлена Ньютона/Гаусса (разными цветами).

Вычислительная реализация:

|  |  |
| --- | --- |
| X | Y |
| 1,10 | 0,2234 |
| 1,25 | 1,2438 |
| 1,40 | 2,2644 |
| 1,55 | 3,2984 |
| 1,70 | 4,3222 |
| 1,85 | 5,3516 |
| 2,00 | 6,3867 |

X\_1 = 1.875 X\_2 = 1.575

Таблица разностей для многочлена Ньютона:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№** |  |  |  |  |  |  |  |  |
| 0 | 1,1000 | 0,2234 | 1,0204 | 0,0002 | 0,0132 | -0,0368 | 0,0762 | -0,1313 |
| 1 | 1,2500 | 1,2438 | 1,0206 | 0,0134 | -0,0236 | 0,0394 | -0,0551 |  |
| 2 | 1,4000 | 2,2644 | 1,0340 | -0,0102 | 0,0158 | -0,0157 |  |  |
| 3 | 1,5500 | 3,2984 | 1,0238 | 0,0056 | 0,0001 |  |  |  |
| 4 | 1,7000 | 4,3222 | 1,0294 | 0,0057 |  |  |  |  |
| 5 | 1,8500 | 5,3516 | 1,0351 |  |  |  |  |  |
| 6 | 2,0000 | 6,3867 |  |  |  |  |  |  |

Необходимо применять интерполирование вперед, поскольку интересующие нас точки лежат в левых половинах соответствующих отрезков.

Таблица разностей для многочлена Гаусса:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№** |  |  |  |  |  |  |  |  |
| -3 | 1,1000 | 0,2234 | 1,0204 | 0,0002 | 0,0132 | -0,0368 | 0,0762 | -0,1313 |
| -2 | 1,2500 | 1,2438 | 1,0206 | 0,0134 | -0,0236 | 0,0394 | -0,0551 |  |
| -1 | 1,4000 | 2,2644 | 1,0340 | -0,0102 | 0,0158 | -0,0157 |  |  |
| 0 | 1,5500 | 3,2984 | 1,0238 | 0,0056 | 0,0001 |  |  |  |
| 1 | 1,7000 | 4,3222 | 1,0294 | 0,0057 |  |  |  |  |
| 2 | 1,8500 | 5,3516 | 1,0351 |  |  |  |  |  |
| 3 | 2,0000 | 6,3867 |  |  |  |  |  |  |

Необходимо применять первую формулу интерполирования Гаусса, поскольку и

Примеры работы программы:

**Пример 1:**

Text

Description automatically generated Text, calendar

Description automatically generated

Графики:

Chart, line chart

Description automatically generated

**Пример 2 (функция с синусом):**

Text

Description automatically generated

Графики:

Chart

Description automatically generated

Реализация расчета:

func LagrangeInterpolation(inputSeries [][]float64, size int, x float64) \*m.Series {  
 var res\_y float64 = 0  
 var up float64 = 1  
 var down float64 = 1  
  
 for i := 0; i < size; i++ {  
 for j := 0; j < size; j++ {  
 if j != i {  
 up \*= x - inputSeries[j][0]  
 down \*= inputSeries[i][0] - inputSeries[j][0]  
 //fmt.Printf("up \*= (%f - %f)\n", x, inputSeries[j][0])  
 //fmt.Printf("down \*= (%f - %f)\n\n", inputSeries[i][0], inputSeries[j][0])  
 }  
 }  
 res\_y += (up / down) \* inputSeries[i][1]  
 up = 1  
 down = 1  
 //fmt.Printf("res\_y += (%f) \* %f\n", up/down, inputSeries[i][1])  
 //fmt.Printf("res\_y = %f\n==========\n", res\_y)  
 }  
  
 fmt.Printf("L(%f) = %f\n", x, res\_y)  
  
 interpolatedSeries := m.NewSeries()  
 interpolatedSeries.Add(m.MakeValue(x, res\_y))  
 interpolatedSeries.Add(m.MakeValue(x, res\_y))  
 return interpolatedSeries  
}  
  
func NewtonInterpolation(inputSeries [][]float64, size int, x float64) \*m.Series {  
 counter := 0  
 for i := 0; i < size; i++ {  
 if inputSeries[i][0] > x {  
 counter = i  
 break  
 }  
 }  
 betweenRight := 0  
 betweenLeft := 0  
 if counter > 0 {  
 betweenRight = counter  
 betweenLeft = counter - 1  
 } else {  
 betweenRight = counter + 1  
 betweenLeft = counter  
 }  
  
 tableSize := size - 1  
 t := 0.0  
 coef := 1.0  
 res\_y := 0.0  
 h := inputSeries[1][0] - inputSeries[0][0]  
 table = make([][]float64, tableSize)  
 for i := 0; i < tableSize; i++ {  
 table[i] = make([]float64, tableSize-i)  
 }  
  
 for columns := 0; columns < tableSize; columns++ {  
 for raws := 0; raws < tableSize-columns; raws++ {  
 if columns == 0 {  
 table[columns][raws] = inputSeries[raws+1][1] - inputSeries[raws][1]  
 } else {  
 table[columns][raws] = table[columns-1][raws+1] - table[columns-1][raws]  
 }  
 }  
 }  
   
 // if in left part  
 if math.Abs(inputSeries[betweenLeft][0]-x) <= math.Abs(inputSeries[betweenRight][0]-x) {  
 t = (x - inputSeries[betweenLeft][0]) / h  
   
  
 res\_y += inputSeries[betweenLeft][1] \* coef  
 coef = t  
 res\_y += table[0][betweenLeft] \* coef  
 coef = 1  
 for i := 1; i < tableSize-betweenLeft; i++ {  
 for j := 0; j <= i; j++ {  
 coef \*= t - float64(j)  
 }  
 coef /= float64(Factorial(i + 1))  
 res\_y += table[i][betweenLeft] \* coef  
 coef = 1  
 }  
 } else {  
 t = (x - inputSeries[betweenRight][0]) / h  
  
 res\_y += inputSeries[betweenRight][1] \* coef  
 coef = t  
 res\_y += table[0][betweenRight-1] \* coef  
 coef = 1  
 for i := 1; i < betweenRight; i++ {  
 for j := 0; j <= i; j++ {  
 coef \*= t + float64(j)  
 }  
 coef /= float64(Factorial(i + 1))  
 res\_y += table[i][betweenRight-i-1] \* coef  
 coef = 1  
 }  
 }  
  
 fmt.Printf("N(%f) = %f\n", x, res\_y)  
  
 interpolatedSeries := m.NewSeries()  
 interpolatedSeries.Add(m.MakeValue(x, res\_y))  
 interpolatedSeries.Add(m.MakeValue(x, res\_y))  
 return interpolatedSeries  
  
}

Выводы по работе:

Реализованы различные интерполяции (с помощью многочленов Лагранжа и Ньютона). Также проведен расчет с помощью многочлена Гаусса. Для данного варианта, и данных точках, результаты интерполяции практически не разнятся от метода к методу.