Санкт-Петербургский Государственный университет ИТМО
Факультет програмной инженерии и компьютерной техники

Лабораторная работа №5 по дисциплине "Вычислительная математика"

Интерполяция функции

Вариант №7

Работу выполнила:
Д. А. Карасева
Группа: P3217
Преподаватель:
Т. А. Малышева

Санкт-Петербург
2024

Содержание

1. Цель работы

Решить задачу интерполяции, найти значения функции при заданных значениях аргумента, отличных от узловых точек. Выполнить программную реализацию *многочлена Лагранжа, многочлена Ньютона с конечными и разделенными разностями.*

Исходные данные задаются тремя способами: a) в виде набора данных (таблицы x,y), пользователь вводит значения с клавиатуры; b) в виде сформированных в файле данных (подготовить не менее трех тестовых вариантов); c) на основе выбранной функции, из тех, которые предлагает программа, например, sin x. Пользователь выбирает уравнение, исследуемый интервал и количество точек на интервале (не менее двух функций). Сформировать и вывести таблицу конечных разностей; Вычислить приближенное значение функции для заданного значения аргумента, введенного с клавиатуры, указанными методами. Сравнить полученные значения; Построить графики заданной функции с отмеченными узлами интерполяции и интерполяционного многочлена Ньютона/Гаусса (разными цветами); Программа должна быть протестирована на различных наборах данных, в том числе и некорректных. Проанализировать результаты работы программы.

2. Описание методов. Расчетные формулы

Многочлен Лагранжа

Интерполяция – способ нахождения промежуточных значений величины по имеющемуся дискретному набору известных значений. Пусть в ходе эксперимента при изменении входной величины x (x0, x1, x2,..., xn) получены значения функции y=f(x) (y0, y1, y2.....yn). При этом во многих случаях аналитическое выражение функции y(x) не известно и получить его по таблице ее значений в большинстве случаев невозможно. Поэтому вместо нее строят другую функцию, которая легко вычисляется и имеет ту же таблицу значений (совпадает с ней в точках x0, x1, x2,..., xn), что и f(x). Нахождение приближенной функции называется интерполяцией, а точки x0, x1, x2, ..., xn – узлами интерполяции. Интерполирующую функцию ищут в виде полинома n степени.

Интерполяционный полином Лагранжа имеет вид:

$$P_n(x) = \sum_{i=0}^{n} y_i * L_n(x), L_n(x) -$$

$$L_n(x) = \frac{(x-x_0)...(x-x_{i-1})...(x-x_n)}{(x_i-x_0)...(x_i-x_{i-1})...(x_i-x_n)} = \prod_{k=o}^{n} \frac{(x-x_k)}{(x_i-x_k)}$$

Следовательно

$$P_n(x) = \sum_{i=0}^{n} y_i * \prod_{k=o}^{n} \frac{(x-x_k)}{(x_i-x_k)}$$

Многочлен Ньютона

Если узлы интерполяции равноотстоящие по величине, так что $x_i + 1–x = h = const$, где h – шаг интерполяции, т.е. xi=x0+nh, то интерполяционный многочлен можно записать в форме, предложенной Ньютоном. Интерполяционные полиномы Ньютона удобно использовать, если точка интерполирования находится в начале таблицы - первая интерполяционная формула Ньютона или конце таблицы – вторая формула.

Интерполирующий полином ищется в виде

$$P_n(x) = a_0 + a_1(x-x_0) + a_2(x-x_0)(x-x_1) + ... + a_n(x-x_0)...(x-x_{n-1})$$

Построение многочлена сводится к определению коэффициентов a_i. При записи коэффициентов пользуются конечными разностями.

3. Вычислительная часть

x	y	X_1	X_2
0.5	1.5320	0.751	0.651
0.55	2.5356	0.751	0.651
0.6	3.5406	0.751	0.651
0.65	4.5462	0.751	0.651
0.7	5.5504	0.751	0.651
0.75	6.5559	0.751	0.651
0.8	7.5594	0.751	0.651

Построить таблицу конечных разностей

N	0	1	2	3	4	5	6
x_i	0.5	0.55	0.6	0.65	0.7	0.75	0.8
y_i	1.532	2.5356	3.5406	4.5462	5.5504	6.5559	7.5594
Δy_i	1.0036	1.005	1.005	1.0042	1.005	1.003	
$\Delta^2 y_i$	0.00139	0.00059	-0.00139	0.0013	-0.002		
$\Delta^3 y_i$	-0.00079	-0.0019	0.0027	-0.003			
$\Delta^4 y_i$	-0.0011	0.0047	-0.006				
$\Delta^5 y_i$	0.0059	-0.0108					
$\Delta^6 y_i$	-0.0166						

Вычислить значения функции для аргумента $X_1 = 0.751$, используя формулу Ньютона

Сравним значение аргумента X_1 с серединой данного отрезка $\frac{(0.5+0.8)}{2} = 0.65$. $X_1 > 0.65 \rightarrow$ воспользуемся второй формулой Ньютона.

$$t = \frac{x - x_n}{h} = \frac{0.751 - 0.8}{0.05} = -0.98$$

$f(x) \sim P_n(x) = y_n + t\Delta y_{n-1} + \frac{t(t+1)}{2!} * \Delta^2 y_{n-2} + ... + \frac{t(t+1)...(t+n-1)}{n!}\Delta^n y_0 = 7.5594 + -0.98 * 1.003 + \frac{-0.98(-0.98+1)}{2!} * (-0.002) + ... + \frac{-0.98(-0.98+1)...(-0.98+6-1)}{6!} * (-0.0166) \sim 6.57603286985$

Вычислить значения функции для аргумента $X_2 = 0.651$, используя формулу Гаусса

Сравним значение аргумента X_2 с серединой данного отрезка $\frac{(0.5+0.8)}{2} = 0.65$. $X_2 > 0.65 \rightarrow$ воспользуемся первой формулой Гаусса.

$$t = \frac{(x - x_0)}{h} = \frac{(0.651 - 0.65)}{0.05} = 0.2$$

$f(x) \sim P_n(x) = y_0 + \Delta y_0 + \frac{t^2}{2!}\Delta^2 y_{-1} + \frac{(t+1)^3}{3!}\Delta^3 y_{-1} + ... + \frac{(t+n-1)...(t-n)}{(2n)!}\Delta^{2n} y_{-n} = 4.5462 + (0.02)(1.0042) + \frac{(0.02)(0.02-1)}{2} * (-0.0014) + \frac{(0.02+1)(0.02)(0.02-1)}{6} * (0.0027) + \frac{}{(0.02+1)(0.02)(0.02-1)(0.02-2)} 24 * (0.0047) + \frac{(0.02+2)(0.02+1)(0.02)(0.02-1)(0.02-2)}{120} * (-0.0107) + \frac{(0.02+2)(0.02+1)(0.02)(0.02-1)(0.02-2)(0.02-3)}{720} * (-0.0166) \sim 4.56629484$

4. Листинг программы

Репозиторий на GitHub

Функция, отвечающая за реализацию многочлена Ньютона с конечными разностями

```
def finite_diff_newton(self, x):
        x_values = self.x
        y_values = self.y

        if (x_values[0] < x < x_values[-1]):
            if x < (x_values[0] + x_values[-1]) / 2:
                differences = [y_values]
                for i in range(1, self.n):
                    prev_diff = differences[i - 1]
                    curr_diff = [(prev_diff[j + 1] - prev_diff[j]) for j in
                    differences.append(curr_diff)

                h = x_values[1] - x_values[0]
                u = (x - x_values[0]) / h
                result = y_values[0]

                for i in range(1, self.n):
                    term = differences[i][0]
                    for j in range(i):
                        term *= (u - j)
                        term /= (j + 1)
                    result += term
            else:
                differences = [y_values[::-1]]
                for i in range(1, self.n):
                    prev_diff = differences[i - 1]
                    curr_diff = [(prev_diff[j] - prev_diff[j + 1]) for j in
                    differences.append(curr_diff)

                h = x_values[-1] - x_values[-2]
                u = (x - x_values[-1]) / h
                result = y_values[-1]

                for i in range(1, self.n):
                    term = differences[i][0]
                    for j in range(i):
                        term *= (u + j)
                        term /= (j + 1)
                    result += term

        elif x < x_values[0]:
            h = x_values[1] - x_values[0]
            f0 = y_values[0]
            df0 = (y_values[1] - y_values[0]) / h
```

```
        result = f0 + df0 * (x - x_values[0])
    elif x > x_values[-1]:
        h = x_values[-1] - x_values[-2]
        fn = y_values[-1]
        dfn = (y_values[-1] - y_values[-2]) / h

        result = fn + dfn * (x - x_values[-1])
```

Функция, отвечающая за реализацию многочлена Лагранжа

```
def lagrange_polynomial(self, x_value):
    y_value = 0
    for i in range(self.n):
        term = 1
        for j in range(self.n):
            if i != j:
                term *= (x_value - self.x[j]) / (self.x[i] - self.x[j])
        y_value += self.y[i] * term
    return y_value
```

Функция, отвечающая за реализацию многочлена Ньютона с разделенными разностями

```
def divided_differences(self):
    dd = np.zeros((self.n, self.n))
    dd[:, 0] = self.y
    for j in range(1, self.n):
        for i in range(self.n - j):
            dd[i, j] = float((dd[i + 1, j - 1] - dd[i, j - 1])
            / (self.x[i + j] - self.x[i]))
    return dd

def newton_polynomial(self, x_value):
    dd = self.divided_differences()
    y_value = self.y[0]
    term = 1
    for j in range(1, self.n):
        term *= (x_value - self.x[j - 1])
        y_value += dd[0, j] * term
    return y_value
```

5. Примеры и результаты работы программы

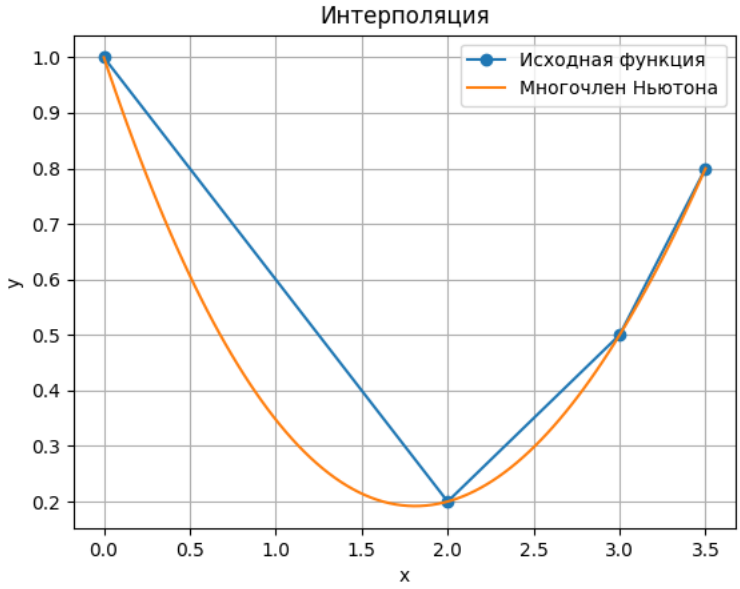

Рисунок 5.1

Выберите способ задания данных:
1. Ввод данных с клавиатуры
2. Загрузка данных из файла
3. Выбор функции
Введите номер способа: >? 2
Введите имя файла: >? 3.txt
Таблица разделенных разностей:
[[1. -0.4 0.23333333 -0.00952381]
[0.2 0.3 0.2 0.]
[0.5 0.6 0. 0.]
[0.8 0. 0. 0.]]
Введите значение аргумента для вычисления: >? 0.1
Приближенное значение функции (многочлен Лагранжа) для x = 0.1: 0.9104190476190477
Приближенное значение функции (многочлен Ньютона, разделенные разности) для x = 0.1: 0.9104190476190476
Приближенное значение функции (многочлен Ньютона, конечные разности) для x = 0.1: 0.9168937500000001

6. Вывод

В результате выполнения лабораторной работы я изучила методы интерполяции функции, смогла программно реализовать многочлены Ньютона и Лагранжа на языке Python.