Вычислительная математика

Раевский Григорий, группа 3.2

04.03.2024

Отчет по лабораторной работе 1

Содержание

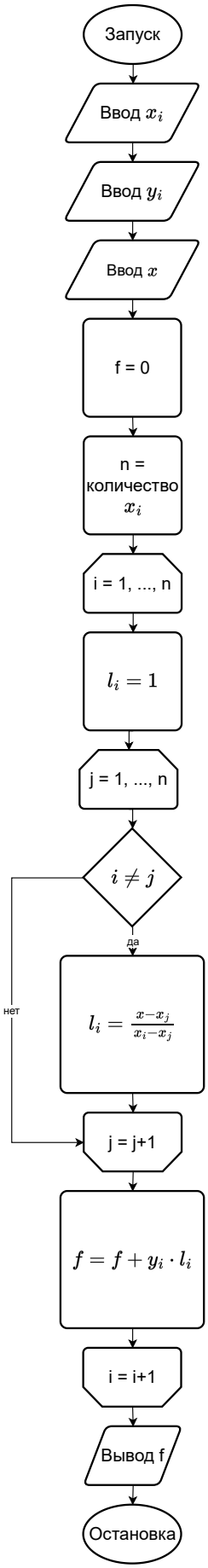
Важно

Задача была сдана на CodeTest до изменения вариантов с полиномом Чебышева (2 марта), на практике рекомендовали уведомить об этом в отчете.

Описание численного метода

Метод интерполяции Лагранжа — метод аппроксимации, который в данном случае используется для нахождения значения интерполяционного полинома в точке x. Метод основан на построении интерполяционного полинома в форме Лагранжа, который проходит через все заданные точки. Полином состоит из суммы произведений значения функции в точке x на базисный полином в этой точке. Полином Лагранжа для $x_1, x_2, ..., x_n$ и $y_1, y_2, ..., y_n$ точек выражается, как $f(x) = \sum_{i=1}^{n} y_i * l_i(x)$, где $l_i(x)$ - $i-$й базисный полином. $l_i(x)$ определяется, как $l_i(x) = \prod_{j=1}^{n} \frac{x-x_j}{x_i-x_j}$, при этом $i \neq j$. При этом, базисный полином равен 1 в точке x_i, а в точках x_j он равен нулю.

Диаграмма


Запуск
Ввод x_i
Ввод y_i
Ввод x
f = 0
n = количество x_i
i = 1, ..., n
$l_i = 1$
j = 1, ..., n
$i \neq j$
да
нет
$l_i = \frac{x - x_j}{x_i - x_j}$
j = j+1
$f = f + y_i \cdot l_i$
i = i+1
Вывод f
Остановка

Листинг кода

```
def interpolate_by_lagrange(x_axis, y_axis, x): # x_axis, y_axis - interpolation nodes, x -
    interpolation point
    result = 0 # result of interpolation

    for i in range(len(x_axis)):
        poly_i = 1 # Lagrange multiplier

        for j in range(len(x_axis)):
            if i != j: # to avoid zero division
                poly_i *= (x-x_axis[j])/(x_axis[i]-x_axis[j])

        result += poly_i * y_axis[i] # final interpolation result for this iteration
    return result
```

Листинг 1: Python example

Примеры работы программы

Стандартный случай

stdin:

```
3
-1.0 0.0 1.0
-1.0 0.0 1.0
0.5
```

stdout:

```
0.5
```

Программа отработала штатно, выдав значение интерполированной функции

Граничные условия

stdin:

```
2
-1.0 1.0
-1.0 1.0
-1.0
```

stdout:

```
-1.0
```

Программа вывела значение '-1.0' в точности, так как интерполяция в точке одного из узлов возвращает значение этого узла

Нарушение уникальности узлов

stdin:

```
3
0.0 0.0 0.0
0.0 1.0 2.0
0.5
```

Программа не сможет корректно отработать, так как нарушает условие интерполяции Лагранжа ($\forall i, j x_i \neq x_j$)

Интерполяция вне диапазона

stdin:

```
4
-1.0 -0.5 0.5 1.0
1.0 0.5 -0.5 -1.0
2.0
```

stdout:

```
-2.0
```

Программа посчитала значение функции в точке 2.0, однако эта точка лежит за пределами узлов интерполирования. Из-за этого результат может быть некорректным.

Одна точка

stdin:

```
1
0.0
1.0
0.0
```

stdout:

```
1.0
```

Программа вернет значение функции в единственной точке интерполяции.

Выводы

Программа стабильно отрабатывает на данных, если они удовлетворяют условиям возможности интерполяции Лагранжа. Однако она не способна выполнить расчет, если входные данные нарушают эти требования (например, если значения x у узлов интерполяции неуникальны, программа не отработает должным образом из-за деления на ноль).

Метод Лагранжа — это очень простой, но при этом мощный и удобный инструмент для аппроксимации различных функций на основе заданных данных. Он хорошо работает с небольшими данными. Однако из-за его высокой алгоритмической сложности (алгоритмическая сложность интерполяции Лагранжа - $O(n^2)$, где n - количество точек из-за вложенного цикла), возможности проявления феномена Рунге (и соответственно, численной неустойчивости из-за чувствительности полиномов высокой степени к изменениям входных данных) могут использоваться другие алгоритмы. Например, метод интерполяции Ньютона дает возможность добавлять новые точки, не пересчитывая весь полином, а метод кубических сплайнов позволяет делать более гладкую аппроксимацию.