Дисциплина: Численные методы

Лабораторное задание №4

Вариант 19

**Отчёт**

Тема: «Интерполирование и приближение функций»

Выполнил:  
студент 3 курса 62 группы  
Дорохов М.В.

Проверила:  
старший преподаватель  
Фролова О.А.

**1. Постановка задачи**

Сглаживание функции, заданной таблицей значений в неравноотстоящих точках, с помощью многочлена первой степени, построенного по трём ближайшим точкам методом наименьших квадратов.

**Входные параметры:**

* X – вектор значений аргументов в порядке возрастания;
* Y – вектор значений функции в порядке возрастания аргумента;
* N – количество значений функции.

**Выходные параметры:**

* YY – вектор сглаженных значений функции;
* IER – индикатор ошибки:
  + IER = 0 – сглаженные значения отличаются от исходных;
  + IER = 1 – сглаженные значения совпадают с исходными;
  + IER = 2 – во входном векторе X нарушен порядок возрастания.

**2. Алгоритм решения**

**Цель программы:**Сглаживание значений функции, заданной таблично, с использованием линейной аппроксимации по трём ближайшим точкам методом наименьших квадратов.

**Основные этапы алгоритма:**

1. Чтение входных данных:
   * Считывание количества точек N и значений векторов X и Y из файла.
2. Проверка порядка возрастания X:
   * Проверка, что X[i] > X[i-1] для всех i. Если условие нарушено, устанавливается IER = 2, и выполнение завершается.
3. Сглаживание:
   * Для первой точки: YY[0] = Y[0].
   * Для последней точки: YY[N-1] = Y[N-1].
   * Для внутренних точек (i = 1 до N-2):
     + Используются три точки: (X[i-1], Y[i-1]), (X[i], Y[i]), (X[i+1], Y[i+1]).
     + Строится многочлен первой степени методом наименьших квадратов.
     + Вычисляется YY[i] как значение этого многочлена в точке X[i].
4. Проверка совпадения:
   * Сравниваются векторы YY и Y с учетом погрешности 1e-12.
   * Если все YY[i] равны Y[i], устанавливается IER = 1.
   * Иначе IER = 0.
5. Вывод результатов:
   * Выводятся векторы X, Y, YY и значение IER.

**3. Используемые формулы**

Для сглаживания внутренней точки i используются следующие выражения:

* – расстояние между X[i] и предыдущей точкой;
* – расстояние между X[i] и следующей точкой;
* Сглаженное значение:

Для крайних точек:

Эта формула представляет аналитическое решение задачи минимизации суммы квадратов отклонений для трёх точек, адаптированное для вычисления значения в средней точке.

**4. Реализация в коде**

Реализация выполнена на языке Rust с использованием структуры Smoother, которая инкапсулирует данные и методы для чтения, сглаживания и вывода результатов. Визуализация данных реализована с помощью библиотеки plotters.

**Структура Smoother**

**Поля структуры:**

* x: Vec<f64> — вектор значений аргументов (в порядке возрастания).
* y: Vec<f64> — вектор исходных значений функции.
* yy: Vec<f64> — вектор сглаженных значений функции.
* n: usize — количество точек данных.
* ier: i32 — индикатор ошибки:
  + 0 — сглаженные значения отличаются от исходных.
  + 1 — сглаженные значения совпадают с исходными (в пределах погрешности (10^{-12})).
  + 2 — нарушен порядок возрастания в векторе x.

**Методы структуры:**

1. new() -> Self  
   Создаёт новую структуру Smoother с пустыми векторами и начальными значениями n = 0, ier = 0.
2. read\_data(&mut self, filename: &str) -> Result<(), Box<dyn Error>>  
   Считывает данные из файла:
   * Первая строка файла содержит количество точек n.
   * Следующие n строк содержат пары значений x и y, разделённые пробелом.
   * Обрабатывает ошибки, такие как отсутствие файла или неверный формат данных.
3. smooth(&mut self)Выполняет сглаживание данных:
   * Если n < 3, копирует y в yy и завершает работу.
   * Проверяет порядок возрастания в x. Если порядок нарушен, устанавливает ier = 2 и завершает работу.
   * Для крайних точек (i = 0 и i = n-1) присваивает yy[i] = y[i].
   * Для внутренних точек (i от 1 до n-2) вычисляет сглаженное значение yy[i] по формуле из пункта 3.
   * Сравнивает yy и y с точностью (10-12). Если они совпадают, устанавливает ier = 1, иначе ier = 0.
4. print(&self)  
   Выводит в консоль:
   * Вектор x.
   * Вектор y.
   * Значение ier.
   * Если ier != 2, также выводит вектор yy.

**Функция plot\_data**

**Прототип:**fn plot\_data(smoother: &Smoother, output\_file: &str) -> Result<(), Box<dyn Error>>

**Описание:**Создаёт график с исходными и сглаженными данными, сохраняя его в PNG-файл:

* Проверяет, что ier != 2. Если условие не выполнено, выводит сообщение об ошибке и завершает работу.
* Создаёт область рисования размером 800x600 пикселей.
* Определяет диапазон по оси Y с запасом ±0.01 для улучшенной видимости.
* Рисует график с заголовком "Результат Сглаживания", осями и сеткой.
* Отображает исходные точки как красные кружки (радиус 2 пикселя), исходную функцию как красную линию и сглаженную функцию как синюю линию.
* Добавляет легенду с подписями: "Исходные точки", "Исходная функция", "Сглаженная функция".

1. **Тестирование**

Тесты представляют собой наборы данных, каждый из которых включает:

* Количество точек N.
* Набор пар значений (x, y), где x — аргументы, а y — соответствующие значения функции.
* В некоторых тестах указано уравнение, описывающее идеальную зависимость y от x (например, y = x или y = x + 1).

Эти тесты предназначены для проверки различных аспектов работы алгоритма:

* Обработка идеально линейных данных.
* Обнаружение ошибок, таких как нарушение порядка возрастания аргументов x.
* Сглаживание данных с минимальными, умеренными или переменными отклонениями от линейной зависимости.

**Тест 1.**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **X** | **0.0** | **1.0** | **2.0** | **3.0** | **4.0** | **5.0** |
| **Y** | **1.0** | **2.0** | **3.0** | **4.0** | **5.0** | **6.0** |
| **YY** | **1.0** | **2.0** | **3.0** | **4.0** | **5.0** | **6.0** |

**IER = 1.**

****

Данные представляют собой идеально линейную зависимость y = x + 1. Аргументы x упорядочены по возрастанию с шагом 1.0. Этот тест проверяет способность алгоритма корректно обрабатывать данные, которые уже соответствуют линейной модели. Ожидается, что сглаженные значения будут совпадать с исходными, что может быть индикатором корректной работы алгоритма в идеальных условиях.

**Тест 2.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **X** | **0.0** | **2.0** | **1.0** | **3.0** |
| **Y** | **1.0** | **2.0** | **3.0** | **4.0** |
| **YY** | **-** | **-** | **-** | **-** |

**IER = 2.**

В этом тесте аргументы x не упорядочены по возрастанию: последовательность x = [0.0, 2.0, 1.0, 3.0] **нарушает требование алгоритма о строгом возрастании значений x**. Тест предназначен для проверки способности программы выявлять такие ошибки и корректно сигнализировать о них, например, устанавливая соответствующий код ошибки (например, IER = 2), без выполнения сглаживания.

**Тест 3.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **X** | **-1.0** | **1.0** | **2.0** | **3.0** | **5.0** |
| **Y** | **-1.0** | **1.0001** | **1.9996** | **3.003** | **5.0** |
| **YY** | **-1.00000** | **0.99986** | **2.00090** | **3.00090** | **5.00000** |

**IER = 0.**

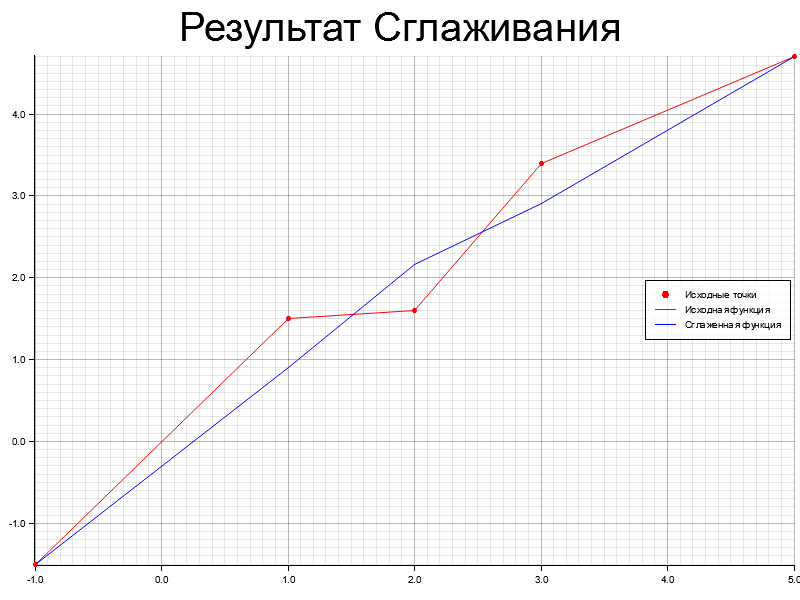
****

Данные очень близки к идеальной линейной зависимости y = x, но содержат минимальные отклонения (например, 1.0001 вместо 1.0, 1.9996 вместо 2.0). Этот тест оценивает, как алгоритм распознаёт и обрабатывает почти идеальные данные. Если отклонения находятся в пределах заданной погрешности, сглаженные значения могут совпадать с исходными, что проверяет точность алгоритма.

**Тест 4.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **X** | **-1.0** | **1.0** | **2.0** | **3.0** | **5.0** |
| **Y** | **-1.5** | **1.5** | **1.6** | **3.4** | **4.7** |
| **YY** | **-1.00000** | **0.99986** | **2.00090** | **3.00090** | **5.00000** |

**IER = 0.**

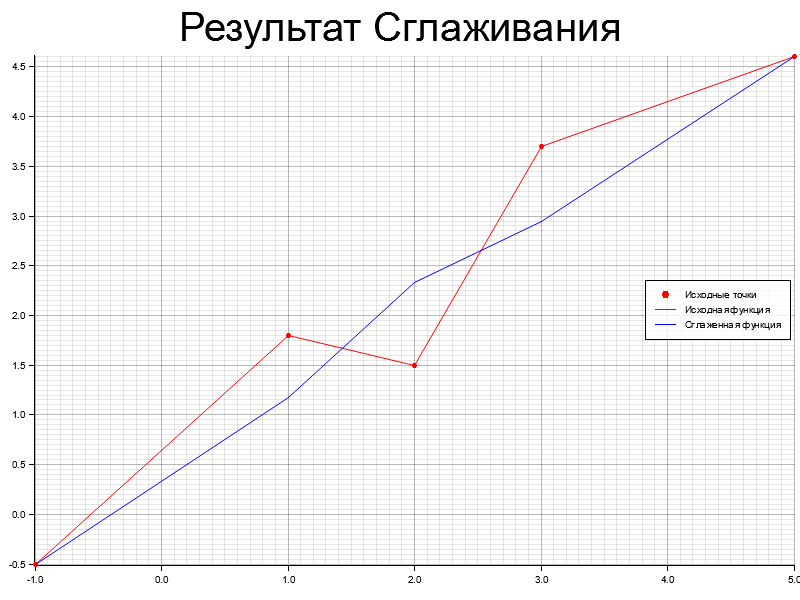
****

Данные имеют умеренные отклонения от идеальной зависимости y = x (например, -1.5 вместо -1.0, 1.5 вместо 1.0). Тест предназначен для проверки способности алгоритма сглаживать данные с заметными, но не чрезмерными отклонениями. Ожидается, что сглаженные значения будут отличаться от исходных, демонстрируя работу метода наименьших квадратов.

**Тест 5.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **X** | **-1.0** | **1.0** | **2.0** | **3.0** | **5.0** |
| **Y** | **-0.5** | **1.8** | **1.5** | **3.7** | **4.6** |
| **YY** | **-0.50000** | **1.17857** | **2.33333** | **2.95000** | **4.60000** |

**IER = 0.**

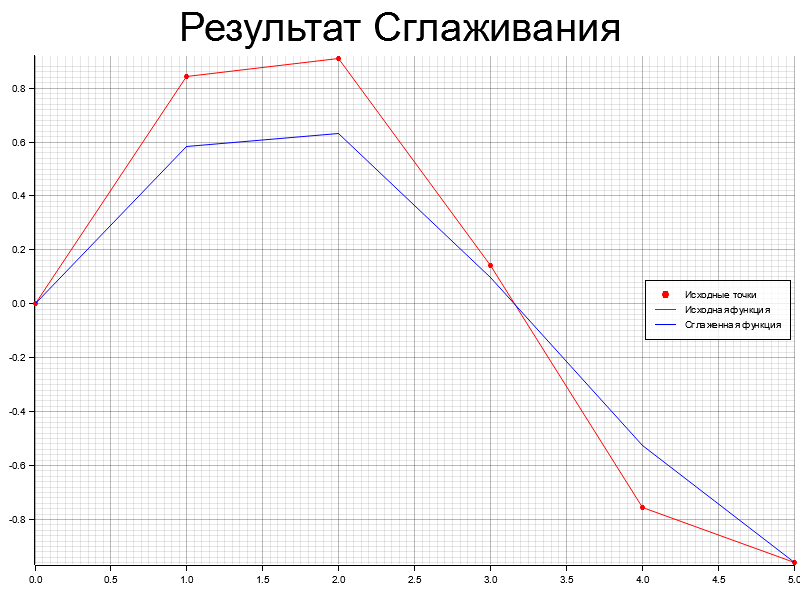
****

Данные содержат переменные отклонения от y = x, имитирующие шум с разной амплитудой (например, -0.5 вместо -1.0, 1.8 вместо 1.0). Этот тест проверяет, как алгоритм справляется с неравномерными отклонениями, моделирующими реальные шумные данные. Ожидается, что сглаживание скорректирует значения, приближая их к линейной зависимости.

**Тест 6.**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **X** | **0.0** | **1.0** | **2.0** | **3.0** | **4.0** | **5.0** | **6.0** |
| **Y** | **0.0** | **0.8415** | **0.9093** | **0.1411** | **-0.7568** | **-0.9589** | **-0.2794** |
| **YY** | **0.00000** | **0.58360** | **0.63063** | **0.09787** | **4.60000** | **-0.52487** | **-0.95890** |

**IER = 0.**

****

Поскольку функция y=sin(x)нелинейна и имеет колебания, сглаженные значения будут отличаться от исходных. Алгоритм сглаживания, использующий линейную аппроксимацию по трём точкам, попытается заменить локальные участки данных прямыми линиями. Это приведёт к:

* Сглаживанию пиков (например, около x=2.0, где y≈0.9093).
* Уменьшению резкости спадов (например, около x=5.0, где y≈−0.9589).