**Задача №2**

Пример задачи, рассчитанной на студентов общих групп.

Рассчитано на 8-х студентов. Каждый из них должен написать и отладить свою функцию, написать функцию main(), из которой обратиться ко всем остальным функциям, написанных студентами из его подгруппы.

Для решения задачи необходимы знания функций ввода-вывода, операторов цикла, умение обрабатывать одномерные массивы, алгоритм решения нелинейного уравнения (метод деления отрезка пополам), аппроксимации функции методом наименьших квадратов, умение генерировать псевдослучайные числа.

Предлагается выполнить машинный эксперимент, в ходе которого провести обработку симулированных экспериментальных данных с целью определения характерного времени радиоактивного распада некоторого вещества и оценить интервал времени, на котором процесс распада можно описывать линейной функцией.

1. **Подготовка данных для эксперимента**.

Известно, что процесс распада радиоактивных материалов описывается функцией

**.**

где - это начальное количество радиоактивного вещества, 𝜷-

характерное время распада (decay\_time), t – текущее время.

Принимая требуется табулировать эту функцию, где t

изменяется на интервале [start\_time, end\_time] (в расчетах

принимается start\_time=0), с шагом step и записать полученные

значения времени в одномерный массив time[100], а значения

функции в одномерный массив radioactivity[100] размера N<=100.

Затем необходимо наложить на полученные значения

радиоактивности шум с амплитудой 0,05 который нужно получить,

используя генератор псевдослучайных чисел. Для этого, сначала

сформируем псевдослучайные числа в интервале [-1, +1] с точностью

5 знаков после запятой. Что бы получить

нужную амплитуду, умножим каждое случайное число из интервала

[-1, +1] на 0,05.

Предполагается, что полученный массив значений: табулированная

функция плюс шум, - это и есть данные, полученные

экспериментальным путем.

В этой части задачи предлагается написать две функции.

Прототип функции

**int experiment (double \*radioactivity, double \*time, double start\_time, double end\_time, double step)** - передаем указатель на массив значений функции, массив значений аргументов, границы интервала start\_time и end\_time, шаг измерений step. В результате получаем массив значений функции radioactivity(t), массив значений аргументов time и их количество N, которое возвращает функция. Значения start\_time, end\_time, и step вводятся с экрана как исходные данные в функции main().

Для моделирования данных эксперимента введем константу, определяющую характерное время распада, которую подставим в формулу

формирования массива **radioactivity.** Ее значения дается как исходные данные и определяется в этой функции.

Прототип функции

**void add\_noise (double \*radioactivity, int N)** - передаем указатель на массив значений функции и количество измерений, в теле функции, формируем N псевдослучайных чисел и суммируем их с элементами массива radioactivity.

Результатом выполнения этой части задачи получаем массив значений функции radioactivity(t), с наложенным на них шумом.

1. **Моделирование обработки измерений**.

На входе имеем массив экспериментальных значений radioactivity и массив аргументов time, состоящий из N элементов. Из физического смысла, предполагается, что процесс может быть описан функцией

экспоненциальной

где decay\_time – характерное время распада,

или

линейной

**- ,**

где decay\_rate коэффициент затухания.

Требуется аппроксимировать экспериментальные данные функциями и , используя метод наименьших квадратов, который заключается в минимизации суммы квадратов отклонений (невязки) и найти характерное время распада decay\_time в первом случае и коэффициент затухания decay\_rate во второй модели, которые должны быть описаны в функции main().

В точке минимума функции частная производная по независимому параметру (decay\_time в первом случае и decay\_rate во втором) равна нулю. Таким образом, нам необходимо продифференцировать суммы квадратов отклонений, приравнять их к нулю и решить соответствующие уравнения.

В результате решения уравнений будут найдены коэффициенты decay\_time с некоторой, наперед заданной точностью Ɛ (precision, которая описывается и вводится как исходные данные в функции main()) и decay\_rate.

Находим сумму квадратов невязок и

**=**

**=**

**= 0**

**= 0**

Находим частные производные и после нехитрых преобразований получим следующие два уравнения.

**= 0**

**() = 0**

Для нахождения decay\_time необходимо обратиться к функции решения нелинейного уравнения методом деления отрезка пополам. Интервал, на котором существует единственный корень вводится в функции как исходные данные с экрана (interval1 и interval2).

Для нахождения decay\_rate необходимо провести вычисления. Решение выглядит следующим образом:

decay\_rate =

Решая вторую часть задачи необходимо написать три функции.

Прототип функции

**double** **nonlinear\_equation (double \*radioactivity, double \*time, int N, double precision)** - передаем указатель на массив экспериментальных данных и массив аргументов, количество элементов массива и точность решения нелинейного уравнения Ɛ (precision) . Функция возвращает значение decay\_time, который является корнем нелинейного уравнения, решенного методом деления отрезка пополам. Эта функция, в свою очередь, обращается к функции, в которой вычисляется значение функции, ноль которой мы ищем в текущей точке интервала

Прототип функции

**double model(double \*radioactivity, double \*time, int N, double point)** - передаем указатель на массив экспериментальных данных и массив аргументов, количество элементов массива, значение, в котором необходимо найти значение функции, возвращает значение функции в

данной точке.

Третья функция предназначена для нахождения decate\_rate путем решения линейного уравнения по приведенной выше формуле.

Прототип функции

**double linear\_equation (double \*radioactivity, double \*time, int N)** - передаем указатель на массив экспериментальных данных и массив аргументов, количество элементов массива, возвращаем значение decate\_rate.

1. **Анализ точности моделей.**

В этой части задачи предлагается провести сравнение экспоненциальной и линейной аппроксимаций на предмет адекватности описания экспериментальных данных.

Для этого надо рассчитать значения среднеквадратичного отклонения и , найденных аппроксимирующих функций и “экспериментальных” данных.

**=**

**=**

на разных временных интервалах. Предполагается, что на небольших временных интервалах обе функции адекватно описывают экспериментальные данные, а на больших, линейная модель не может быть использована.

Значения среднеквадратичного отклонения второй модели интерполяции всегда больше чем первой в силу физики процесса радиоактивного распада, но степень различия между интерполяционными моделями сильно зависит от соотношения между временем распада и интервалом проведения измерений. Если время проведения измерений много меньше характерного времени распада, то обе модели достаточно адекватно описывают процесс, если же время измерений того же порядка или больше характерного времени распада, то адекватна только экспоненциальная модель.

Предлагается приблизительно найти интервал, начиная с которого линейная модель перестает работать.

Для этого найдем decay\_time и decay\_rate на 10 точках, затем, будем увеличивать количество точек на 1, для нового интервала вычислять характерное время и коэффициент затухания, сравнивать среднеквадратичные отклонения обоих моделей, до тех пор, пока не найдем количество точек (M), при котором среднеквадратичное отклонение линейной модели превысит среднеквадратичное отклонение экспоненциальной модели более чем в два раза.

[time\_start, ] и будет искомый нами интервал.

Для реализации этой части задачи необходимо написать три функции, в которых:

- найти среднеквадратичное отклонение для экспоненциальной и линейной модели на разных интервалах и найти точку, в которой среднеквадратичное отклонение линейной модели превысит среднеквадратичное отклонение экспоненциальной модели в два раза.

Прототип функции

**double precision\_analysis (double \*radioactivity, double \*time, int N)** - передаем указатель на массив экспериментальных и массив аргументов, количество элементов массива. Функция возвращает значение времени, начиная с которого линейная модель не адекватно описывает процесс распада (time\_differences).

Для вычисления среднеквадратичных отклонений эта функция обращается к функциям dev\_exp и dev\_linear.

- найти среднеквадратичное отклонение для экспоненциальной модели на заданном интервале.

Прототип функции

**double dev\_exp(double \*radioactivity, double \*time, int M, double decay\_time)** - передаем указатель на массив экспериментальных данных, массив аргументов, количество элементов массива, границы интервала и значение decay\_time. Функция возвращает среднеквадратичное отклонение для экспоненциальной модели deviation1.

- найти среднеквадратичное отклонение для линейной модели на заданном интервале

Прототип функции

**double dev\_linear(double \*radioaktivity, double \*time, int M, double decay\_rate)** - передаем указатель на массив экспериментальных и массив аргументов, количество элементов массива, границы интервала и значение decay\_rate. Функция возвращает среднеквадратичное отклонение для линейной модели deviation2.

**Исходные данные**

В функции main

|  |  |
| --- | --- |
| start\_time | 0. |
| end\_time | 9.9 |
| step | 0.1 |
| precision | 0.0001 |
|  |  |

В функции experiment

double const betta = 3.5

В функции nonlinear\_equation

interval1 0.2

interval2 8.2

**РЕЗУЛЬТАТЫ РАБОТЫ ПРОГРАММЫ**

|  |  |
| --- | --- |
| decay\_time | 3.5 ± 0.05 |
| decay\_rate | 8.152 ± 0.05 |
| time\_differences | 4.1 : 5.1 |
| Разброс результатов обусловлен разными наборами случайных чисел, которые изменяются при каждом запуске программы | |

**Данные для тестирования программы**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Вариант 1 | | | | Вариант 2 | | | |
| Исходные данные | | Результат решения | | Исходные данные | | Результат решения | |
| start\_time | 0.0 | decay\_time | 3.0±0.05 | start\_time | 0.0 | decay\_time | 4.0±0.05 |
| end\_time | 9.9 | end\_time | 9.9 |
| step | 0.1 | step | 0.1 |
| precision | 0.001 | decay\_rate | 7.751±0.05 | precision | 0.001 | decay\_rate | 8.546±0.05 |
| const betta | 3.0 | const betta | 4.0 |
| interval1 | 0.2 | time\_differences | 3.3 : 4.3 | interval1 | 0.2 | time\_differences | 4.5 : 5.9 |
| Interval2 | 8.2 | Interval2 | 8.2 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Вариант 3 | | | | Вариант 4 | | | |
| Исходные данные | | Результат решения | | Исходные данные | | Результат решения | |
| start\_time | 0.0 | decay\_time | 4.5±0.05 | start\_time | 0.0 | decay\_time | 3.7±0.05 |
| end\_time | 9.9 | end\_time | 9.9 |
| step | 0.1 | step | 0.1 |
| precision | 0.001 | decay\_rate | 8.921±0.05 | precision | 0.001 | decay\_rate | 8.317±0.05 |
| const betta | 4.5 | const betta | 3.7 |
| interval1 | 0.2 | time\_differences | 5.5 : 6.4 | interval1 | 0.2 | time\_differences | 4.3 : 5.4 |
| Interval2 | 8.2 | Interval2 | 8.2 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Вариант 5 | | | | Вариант 6 | | | |
| Исходные данные | | Результат решения | | Исходные данные | | Результат решения | |
| start\_time | 0.0 | decay\_time | 5.0±0.05 | start\_time | 0.0 | decay\_time | 5.3±0.05 |
| end\_time | 9.9 | end\_time | 9.9 |
| step | 0.1 | step | 0.1 |
| precision | 0.001 | decay\_rate | 9.319±0.05 | precision | 0.001 | decay\_rate | 9.625±0.05 |
| const betta | 5.0 | const betta | 5.3 |
| interval1 | 0.2 | time\_differences | 5.7 : 7.1 | interval1 | 0.2 | time\_differences | 5.8 : 7.7 |
| Interval2 | 8.2 | Interval2 | 8.2 |