МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

"ЛЭТИ" ИМ. В.И.УЛЬЯНОВА(ЛЕНИНА)

Кафедра МО ЭВМ

ОТЧЁТ

по учебной практике

Тема: «Аппроксимация множества точек полиномом 3-й степени алгоритмом роя частиц»

Студент гр. 1381 _____ Кагарманов Д. И.

Студент гр. 1381 _____ Исайкин Г. И.

Преподаватель _____ Жангиров Т. Р.

Санкт-Петербург

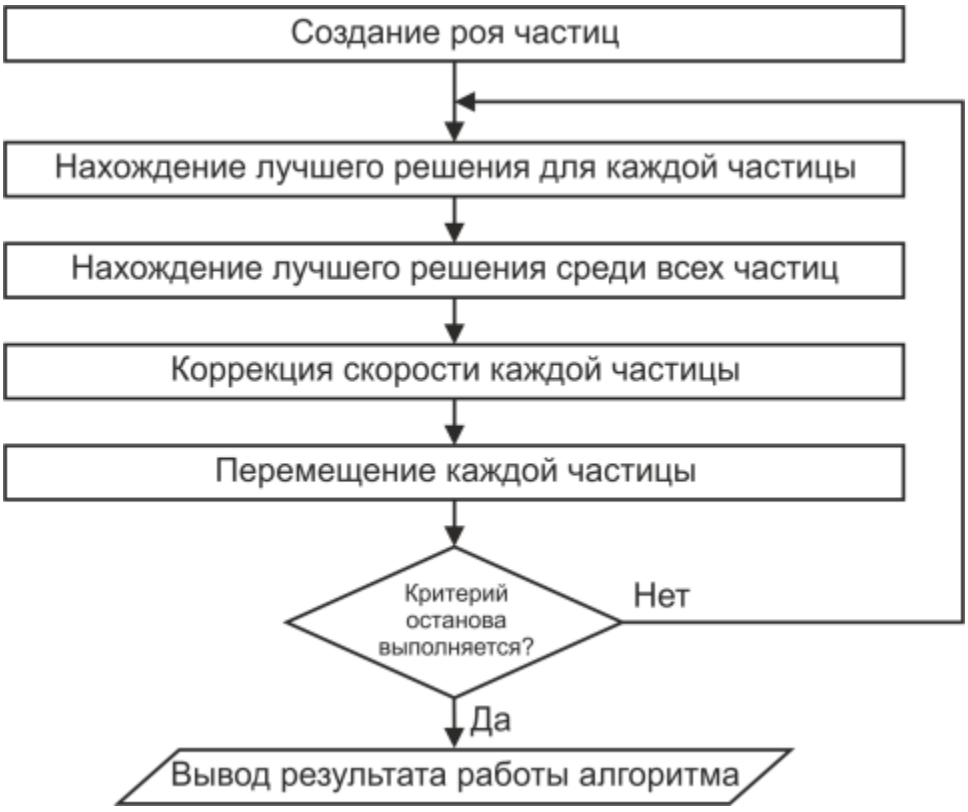
2023

Задание

Пусть имеется набор точек. Необходимо найти коэффициенты *a, b, c, d* полинома вида $ax^3 + bx^2 + cx + d$, чтобы полином наилучшим образом аппроксимировал данное множество точек. В качестве метрики необходимо использовать абсолютное отклонение

Выполнение работы

1. Алгоритм роя частиц



1.1 Создание роя частиц.

Изначально мы имеем множество точек X, которые мы хотим аппроксимировать. Хранится они будут в списке кортежей $X = [(x_1,y_1), \ldots, (x_n, y_n)]$, n – кол-во точек, оно задается пользователем.

Рой частиц в нашем случае будет представлять набор полиномов со случайно сгенерированными начальными коэффициентами a, b, c, d. Частицы храним в списке списков: $S = [[a_1,b_1,c_1,d_1], \ldots, [a_m,b_m,c_m,d_m]]$, m – кол-во частиц, задается пользователем.

Начальная позиция j-ой частицы: $x_j^0 = (a_j^0, b_j^0, c_j^0, d_j^0)$

1.2 Нахождение лучшего решения для каждой частицы.

В качестве целевой функции будем использовать сумму модулей отклонений частицы от всех заданных точек: $\sum_{i=1}^{n} |y_i - s_j| \to min$,

где n – кол-во точек, s_j – j-ая частица.

На каждой итерации алгоритма запоминаем лучшее решение (набор коэффициентов) для каждой частицы.

$p_j^k = (a_j^k, b_j^k, c_j^k, d_j^k)$ – лучшее решение для j-ой частицы на k-ой итерации алгоритма. Храним в списке, исходно инициализируем начальными (случайными) значениями.

1.3 Нахождение лучшего решения для всех частиц.

Среди решений, найденных на предыдущем шаге, находим лучшее.

$b^k = (a^k, b^k, c^k, d^k)$

1.4 Коррекция скорости каждой частицы.

Изначально у каждой j-ой частицы имеется вектор скоростей для каждого коэффициента (задается случайно):

$$v_j = (v_{ja}, v_{jb}, v_{jc}, v_{jd})$$

Классическая коррекция скорости для переменной a частицы j-ой:

$$v_{ja}^{k+1} = v_{ja}^k + \alpha_p r_p \left(p_{ja}^k - a_j^k\right) + \alpha_b r_b \left(b_a^k - a_j^k\right)$$

$r_p, r_b = rand(0, 1)$

α_p, α_b – весовые коэффициенты, подбираются опытным путем для конкретной задачи. Возможно, будет возможность их выбора для удобства тестирования.

При неудовлетворительных результатах сменим коррекцию скорости на канонический алгоритм, при котором весовые коэффициенты не так сильно влияют на сходимость.

1.5 Перемещение частицы

Каждую частицу (то есть коэффициенты каждого полинома) изменяем по формуле:

$$a_j^{k+1} = a_j^k + v_{ja}^{k+1}$$
$$b_j^{k+1} = b_j^k + v_{jb}^{k+1}$$
$$c_j^{k+1} = c_j^k + v_{jc}^{k+1}$$
$$d_j^{k+1} = d_j^k + v_{jd}^{k+1}$$

1.6 Критерий останова.

Превышение некоторого кол-ва итераций, или необходимое значение целевой функции (пределы погрешности). Вектор b^k будет содержать лучшее решение.

2. GUI

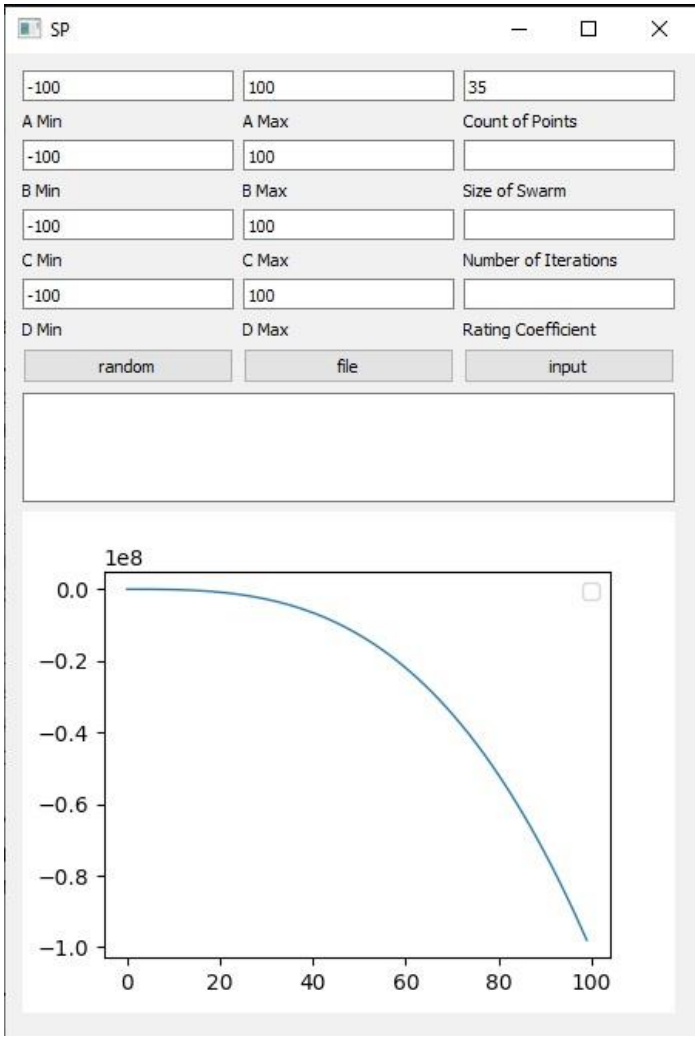


Рисунок 1 – прототип GUI

Для разработки графической оболочки были использованы: библиотеки PyQt5 и matplotlib.

На данный момент предусмотрено: случайная генерация частиц-полиномов со случайными коэффициентами с заданным диапазоном значений; количество аппроксимируемых точек и количество итераций, выполняемых алгоритмом.

Вывод:

Создан план решения задачи и спроектирован макет GUI.