Аппроксимация ломаной траектории

Класс Line

Для начала работы с задачей был создан класс Line в файле line.py. Объекты класса включает различные параметры для представления прямой: k-форма (y = kx + b), общая форма (Ax + By + C = 0) и по двум точкам. Также в классе реализованы различные полезные методы для работы с прямыми: обнаружение пересечения двух прямых, поиск нормали к прямой из точки, определение расстояния от точки до прямой и т.д.

Генерация траектории, класс TrajectoryGenerator

За генерацию эталонной траектории отвечает файл trajectoryGenerator.py с классом TrajectoryGenerator. Первый этап – создание набора отрезков (число выбирается случайно от 4 до 6, длина – от 0.5 до 1.5 м, угол поворота относительно предыдущего – от $-\pi/2$ до $\pi/2$). Для дальнейшей генерации данных необходимо было выделить точки на всей траектории. За разбиение во всём коде отвечает регулируемый параметр дискретизации **ln_seg**, который отражает длину пути, за которую происходит измерение. В визуализации – слайдер Period.

Траектория в точках принимает вид на Рисунке 1.

Генерация данных, класс DataGenerator

Данный класс представлен в файле dataGenerator.py. Основная его задача – зашумление эталонной траектории и генерация данных, имитирующих «измерения». Для этого используется полученная дискретизация траектории.

Зашумление выбрано с нормальным распределением. Математическое ожидание нулевое, а за определение среднеквадратического отклонения отвечает регулируемый параметр **mess –** слайдер Mess.

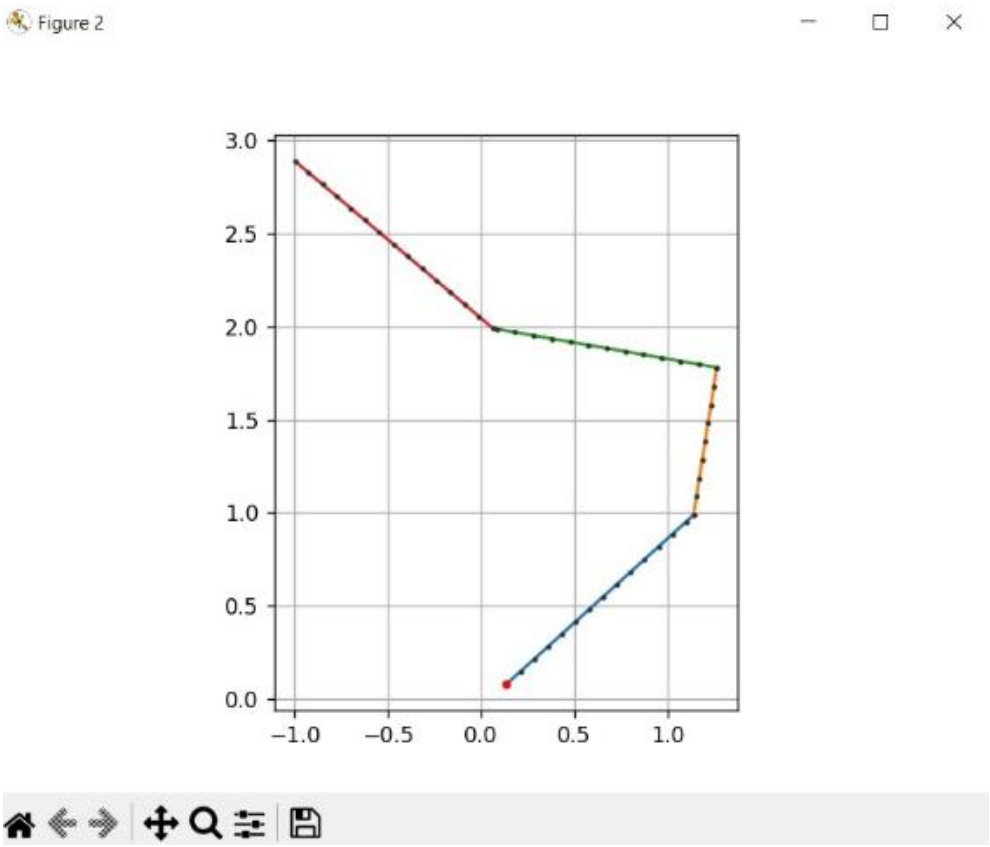

Рисунок 1. Дискретизация траектории (чёрные точки)

Параметр отражает тройное СКО – т.е. такую ширину распределения, что с вероятностью 99.9% величина будет по модулю в его пределах. Поэтому при задании СКО для распределений выбирается треть от параметра mess, умноженная на ln_seg.

Смещение вычисляется для каждой точки по направлению траектории и перпендикулярно ему (Рисунок 2).

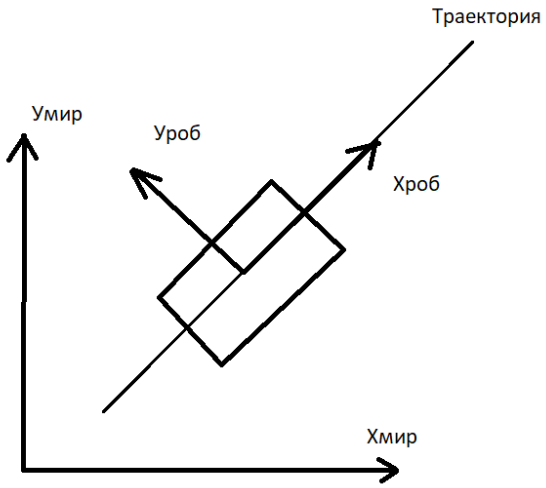

Рисунок 2. Оси зашумления.

Шум вычисляется вдоль осей локализуемого объекта (Хроб О Уроб) и затем пересчитываются в координаты мира (Хмир О Умир). Полученные смещения прибавляются к координатам точек эталонной траектории. На выходе получается датасет из зашумлённых x и y координат (Рисунок 3):

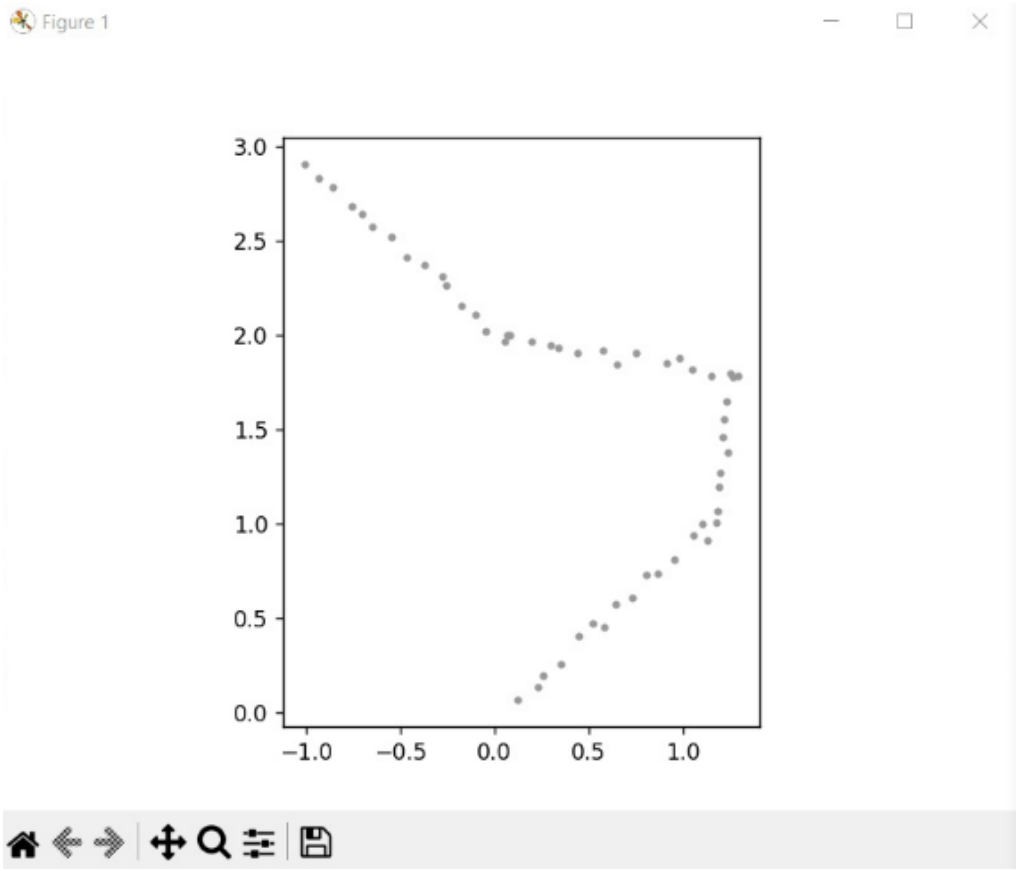

Рисунок 3. Выходной датасет

Решение. Метод наименьших квадратов, класс StraightLineEstimatorLSM

Класс реализован в файле straightLineEstimatorLSM.py. Здесь представлены различные вспомогательные методы и непосредственная реализация алгоритма.

Суть метода состоит в определении функции потерь и работе с ней. В качестве функции потерь выбрана функция:

$$LF(A, C) = \sum_{i=0}^{N} dist_i^2$$

Здесь A, C – параметры прямой в общем виде, аппроксимирующей набор точек до прямой, $dist_i$ – расстояние от i-й точки данных до данной прямой, N – число точек в наборе. Для нахождения оптимальных параметров требуется минимизация функции потерь:

$$\hat{A}, \hat{C} = argmin(LF(A, C))$$

Для каждого отрезка вычисляется своя функция потерь. Для нахождения минимума берутся производные функции двух переменных, которые затем приравниваются нулю. Получив два уравнения, можем решить систему и найти таким образом оптимальные параметры аппроксимирующей прямой. Все вычисления происходили на бумаге.

В коде же представлено квадратное уравнение, которое получено в результате подстановки выраженной величины C, в уравнение производной по A. Его решение даёт два корня, каждый из которых затем подставляется в уравнение производной по C. Таким образом получены две пары значений, которые отражают экстремумы функции потерь. Для определения лучшей (минимизирующей) пары значения подставляются в функцию потерь и берётся пара, дающая наименьшее значение. Таким образом, определяются оптимальные параметры аппроксимации до прямой произвольного набора точек.

Разделение на отрезки

Для применения метода к ломаной траектории необходимо определить края отрезков траектории, иначе весь датасет аппроксимируется до прямой. Здесь появляется регулируемый параметр **tolerance** (слайдер Tolerance) – предельное допустимое расстояние до прямой. Алгоритм разбиения следующий:

1. С самого начала датасета выбираются две точки (срез данных), происходит их аппроксимация до прямой.

2. Вычисляется расстояние от точки, следующей после этого среза, до полученной прямой.

3. Если расстояние не превышает tolerance, точка принимается в текущий срез, затем происходит аппроксимация прямой по трём точкам и оценивается расстояние от следующей (т.е. четвёртой) точки до новой полученной прямой и т.д., пока выполняется условие.

4. Если tolerance превышен, это является сигналом об окончании отрезка ломаной траектории, ставится метка (индекс последнего найденного конца отрезка в данных).

5. Полученная в процессе последняя аппроксимирующая прямая является прямой, аппроксимирующей данный срез, по сути отражающий точки, принадлежащие одному отрезку ломаной траектории, параметры данной прямой сохраняются.

6. Далее алгоритм продолжает те же действия, начиная с индекса последнего конца отрезка, пока не будут пройдены все данные.

Визуализация в непрерывную траекторию

Параметры прямых определены, теперь надо определить координаты концов отрезка на этой прямой для визуализации. Здесь 3 ключевых момента:

1. Начало траектории (красная точка) вычисляется как точка пересечения первой полученной прямой и нормали к ней, проходящий через первую точку из датасета.

2. Промежуточные углы траектории (изломы) имеют координаты точки пересечения двух соседних полученных прямых.

3. Конец траектории (синяя точка) есть точка пересечения последней полученной прямой и нормали к ней, проходящей через последнюю точку датасета.