Техническое задание

1. Цель работы

Разработка скрипта на python, позволяющего рассчитывать радиусы кривизны произвольной полилинии (далее – скрипт)

2. Краткое описание входных данных

Исходными данными для обработки скриптом являются полилинии, т.е. последовательный набор точек (узлов) с известными координатами (заданы в WGS84 UTM34N, в метрах). Полилинии выполнены в формате ESRI Shape. В исходном файле может присутствовать как одна полилиния, так и более одной (чаще всего одна или две).

Как были получены полилинии:

Полилинии были получены ручной оцифровкой снимков участка береговой линии (всего около 70) в разное время. Соответственно, точки (узлы) располагаются на различном расстоянии друг от друга, как линейно, так и вдоль полилинии. Узлы располагались таким образом, чтобы максимально описать форму береговой линии, в том числе небольшие выступы и вогнутости. (Пример на рисунке 1)

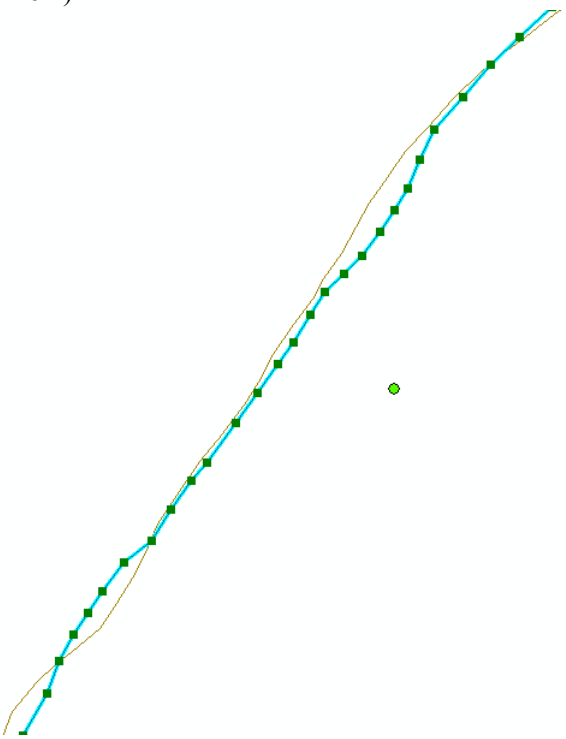

Рисунок 1 – Узлы на полилинии

На полилинии присутствуют вогнутости и выступы с очень различным радиусом кривизны, причем, на этих вогнутостях и выступах могут находиться вогнутости и выступы с меньшим радиусом.(Пример на рисунке 2).

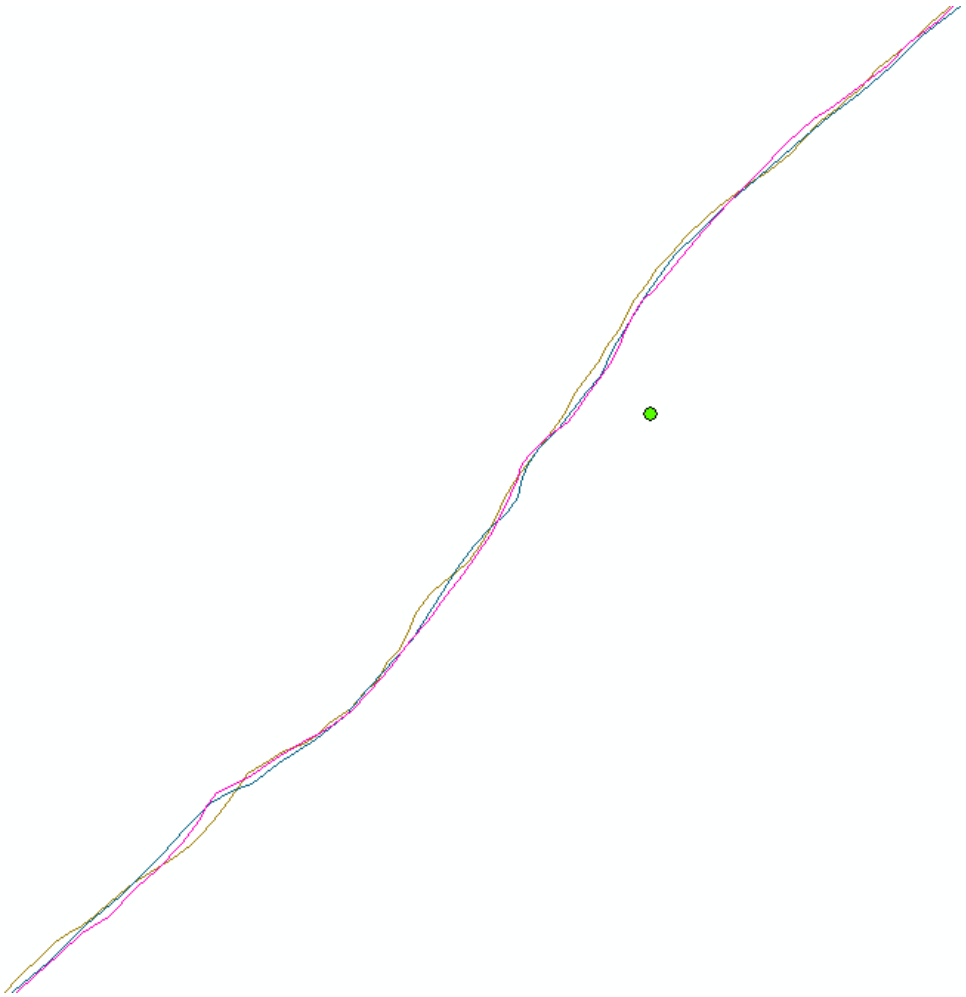

Рисунок 2 – Пример вогнутостей и выступов на трех полилиниях – береговых линиях в разные моменты времени

3. Основные пожелания

Основная идея – получить возможность рассчитать радиус кривизны как больших вогнутостей и выступов, так и малых, которые могут находиться в больших.

3.1 Расчет локальных радиусов кривизны для полилинии. Необходимо получить радиусы кривизны для участков полилинии. Тут, наверное, возможно несколько подходов (это к обсуждению, не является инструкцией):

1. Двигаться вдоль полилинии с определенным шагом и находить кривизну (такой подход уже был реализован в
   https://repository.nced.umn.edu/browser.php?current=keyword&keyword=5&dataset_id=15&folder=237802 (могу выслать эти файлы)
   И, возможно, он единственный имеет право на жизнь.

Это обсуждение также может быть полезно
https://gis.stackexchange.com/questions/127169/how-can-i-calculate-the-radius-or-curvature-of-a-meandering-river

2. Определять локальные экстремумы, воспринимая их как точку для которой надо рассчитать кривизну.
3. Дать возможность ограничивать предельный радиус кривизны, чтобы расчет не проводился для больших вогнутостей и выступов, а проводился только для малых, и наоборот, соответственно. Либо дать возможность игнорировать мелкие вариации в данных

3.2 Дать возможность изменять количество окрестных точек, по которым производится расчет радиуса кривизны и ширину зоны расчета. Поясню: обычно используются соседние 2 точки –

одна справа, другая слева. Хотелось бы иметь возможность варьировать это количество, чтобы захватывать большее или меньшее количество соседних точек, а также варьировать расстояние поиска (например, расчет по максимум 5 соседним точкам, расположенным в радиусе 100 метров; либо, расчет по 4 соседним точкам; либо расчет по точкам расположенным в радиусе 120 метров). Предусмотреть возможность что точек будет не хватать в таком радиусе

3.3 Реализовать расчет вписывания в точки вдоль полилинии окружностей с различным радиусом (задавать радиусы заранее списком) с расчетом RMSE и доли (как мер отклонения участков полилинии от окружностей заданного радиуса). И возможность включить\отключить этот расчет

3.4 Давать комментарии в коде скрипта, чтобы иметь возможность самостоятельно настраивать параметры

3.5 Использовать ту же систему координат в выходных данных, что была во входных (а не новую, условную или расстояние от начала линии). Поясню: расчеты будут проводиться для большого количества полилиний, затем будет проводиться сравнение изменений кривизны во времени и с другими показателями, для чего нужно одно координатное пространство

Выходные данные можно представлять в текстовой табличной форме с разделителями (например «;» или табуляцией). По структуре выходных данных такое предложение:

1. Координата x точки для которой выполнялся расчет (например x)
2. Координата y (например y)
3. Радиус кривизны в точке (например Rk)
4. Размер ширины зоны расчета (например Bandwidth)
5. Количество точек, используемых для расчета (например Count)

6.7.8 и тд. – результаты вписывания окружностей (Можно назвать их R_%указанный радиус%_RMSE и R_%указанный радиус%_ER , например)

Дополнительные пожелания

1. Желательна (но не обязательна) возможность загрузки полилиний непосредственно из формата Shp (думаю есть готовые решения у QGIS). В принципе полилинию я могу задавать и в текстовом виде как нумерованный список (n, x, y), но вручную переводить shp в список довольно времязатратная процедура.
2. Желательна (но не обязательна) возможность пакетной обработки (например, возможность указать директорию, содержащую файлы полилиний)
3. Желательно – дополнительно попробовать сделать расчет характерного радиуса кривизны, т.е. радиус той окружности которая лучше всего впишется
4. Желательно использовать python, поскольку взаимодействие с большинством гис (QGIS; ArcGIS идет через него и у них есть уже готовые блоки кода и библиотеки)

Спасибо. Каждый пункт готов обсуждать, методические предложения математика категорически приветствуются