Цифровая олимпиада «Волга–IT`20»

Дисциплина «Системное программирование (C++)»
Задание финального этапа

Имеется:

Кривая с данными, считанными во время бурения. Данные считываются по всей длинне траектории бура с определенными интервалами по времени. Данные предоставляются в виде JSON файла в формате:

```
[
 {
 md: 132.121212,
 data: 10.857
 },
 {
 md: 132.5483,
 data: 5.123
 },
 ...
]
```

md - текущая длинна пробуренной скважины, не может быть отрицательной, равномерно увеличивается

data - величина некоего показателя, считанного для текущей точки на траектории скважины (по условию задачи не может быть отрицательной

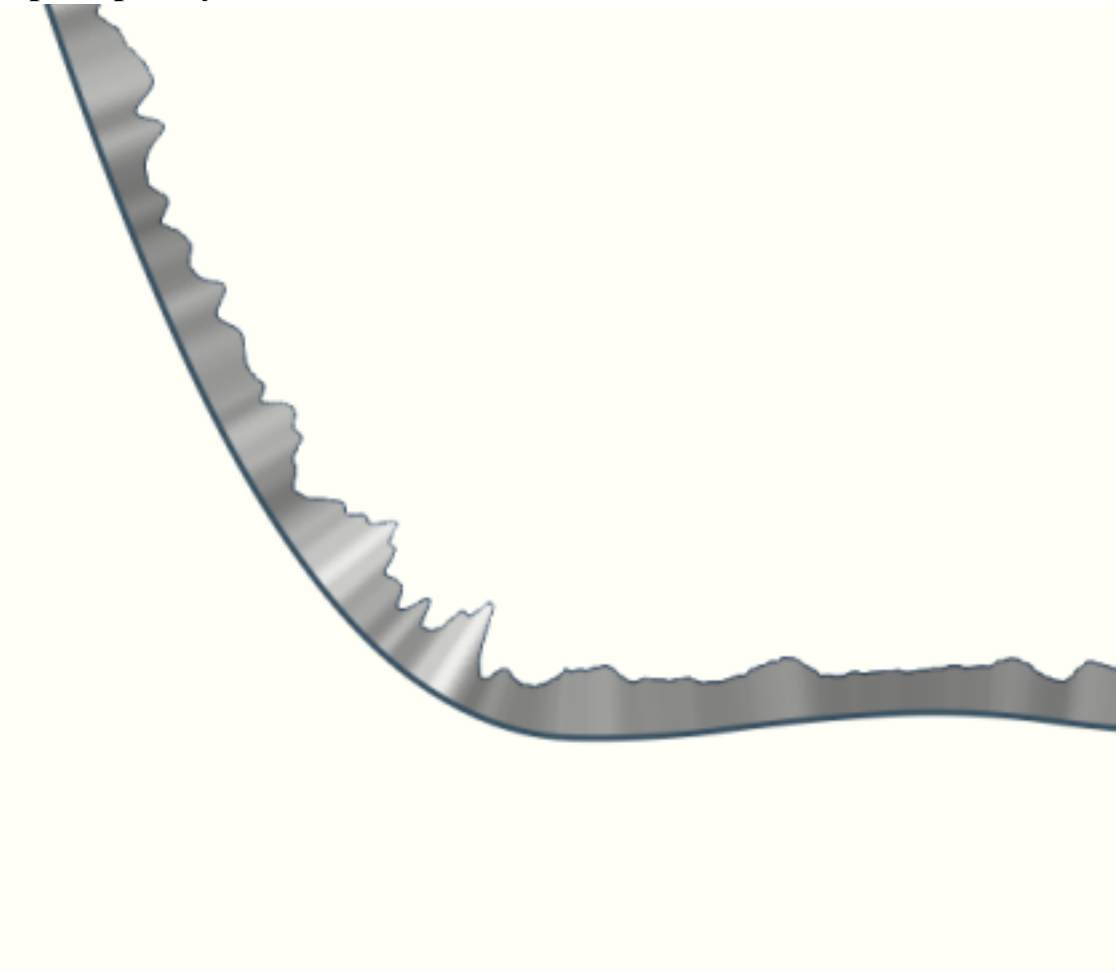
Необходимо:

Нарисовать представленную кривую вдоль траектории. Для этого необходимо методом интерполяции найти точку на траектории, соответствующую md точки кривой данных и взять точку на нормали к этой точке с длинной отрезка, соответствующей значению data для этой точке.

Дополнительно:

Сделать заливку от кривой данных до кривой траектории линейным градиентом в интервале data [0, max], где max - максимальное значение кривой на всей длине. 0% соответствует цвету rgb(255, 0, 0), а 100% соответствует цвету rgb (0,0,255) (50% соответствует цвету rgb(127, 0, 127))

Пример визуализации:



Подсказки по коду

В рамках первого тура необходимо было построить траекторию с

использованием метода минимальной кривизны dogleg. Одна из возможный

реализаций вычисления dogleg:

```
double calcDogLeg(double prevInclinationRad, double inclinationRad,
 double prevAzimutRad, double azimutRad)
 {
 const auto u1 = utils::makeUnitVectorBySurveyAngles(prevAzimutRad,
prevInclinationRad);  const auto u2 =
utils::makeUnitVectorBySurveyAngles(azimutRad, inclinationRad);

 const double cosValue = dot(u1, u2);
 const double angleBetweenVectorsRad = std::acos(clamp(-1.0, 1.0,
cosValue));
 return angleBetweenVectorsRad;
 }
```

Одна из возможных реализаций вычисления смещения deltaX, deltaY, deltaTvd соседних точек траектории (для простоты граничные случаи не учитываются):

```
ReconstructedSurveyStationPosition
reconstructSurveyStationPosition(double deltaMeasureDepth,
  double  azimuth1Rad,  double  inclination1Rad,   double  azimuth2Rad,
double inclination2Rad,  double dogLegRad)
 {
 /**
 * Coordinates of the tangent *unit* vector at the first point.
 */
 const double x1 = std::sin(azimuth1Rad) * std::sin(inclination1Rad);
const double y1 = std::cos(azimuth1Rad) * std::sin(inclination1Rad);
const double z1 = std::cos(inclination1Rad);

 /**
 * Coordinates of the tangent *unit* vector at the second point.
 */
 const double x2 = std::sin(azimuth2Rad) * std::sin(inclination2Rad);
const double y2 = std::cos(azimuth2Rad) * std::sin(inclination2Rad);
const double z2 = std::cos(inclination2Rad);

 const double k = 0.5 * deltaMeasureDepth * std::tan(0.5 * dogLegRad) / (0.5 *
dogLegRad);

 ReconstructedSurveyStationPosition result;
```

```
result.deltaEastWest = k * (x1 + x2);
result.deltaNorthSouth = k * (y1 + y2);
result.deltaTrueVerticalDepth = k * (z1 + z2);

return result;
}
```