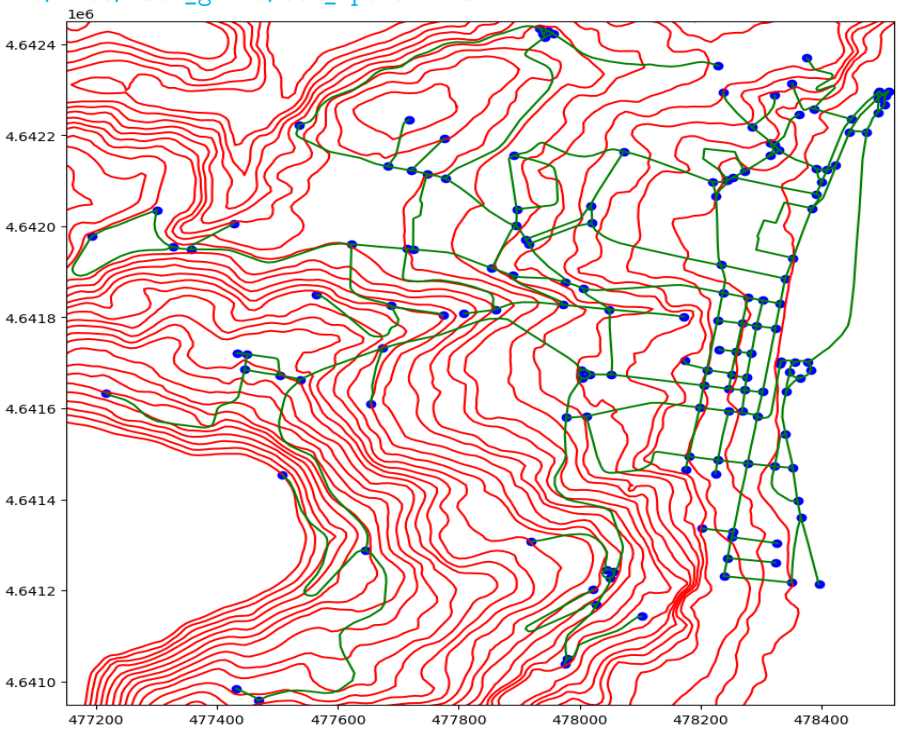
Graph Laplacian method to estimate points elevation within adjacent contour lines

8 décembre 2023

The goal of this project is to estimate an elevation model for an Open Street Map graph.

The python library OSMNX (https://osmnx.readthedocs.io/en/stable/) provides users with a graph encoding the road network of a given area : the edges come with Shapely *LineString* objects representing (stretches of) roads and the nodes correspond to the intersections.

Given a set of contour lines, also encoded as *LineString* objects, one can compute all the point intersections between OSM edges and contour lines thanks to the powerful Geopandas *overlay* function : https://geopandas.org/en/stable/docs/user_guide/set_operations.html



The picture above represents some graph network (the harbor of Ajaccio in Corsica), OSM edges are drawn in green, OSM intersections in blue and contour lines in red.

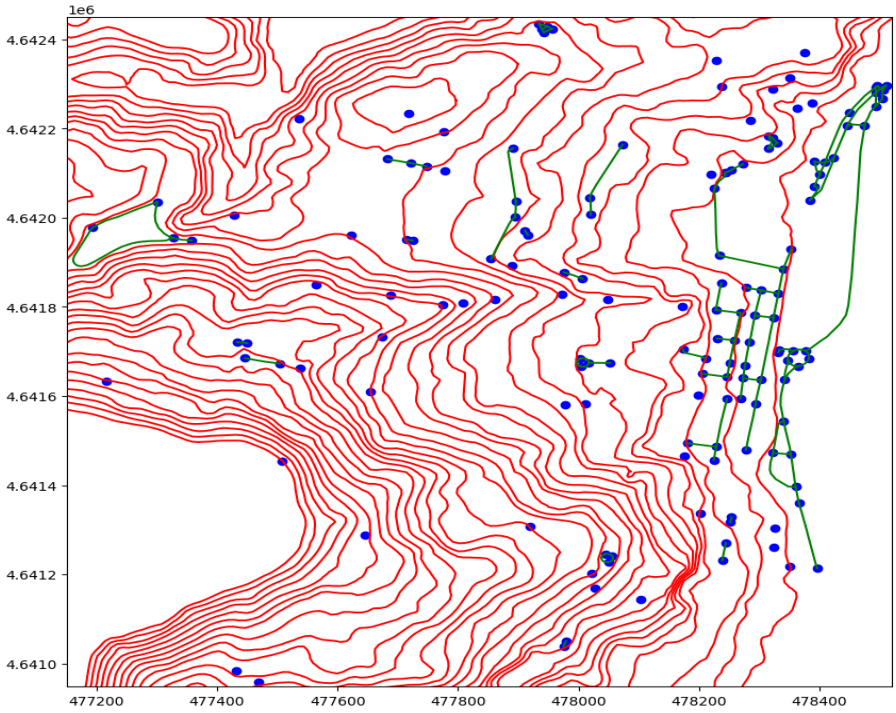
We now need to estimate the nodes elevations :

(i) Given an OSM edge that intersects several contour lines, the different intersection points provide us with a piece-wise linear interpolation of the elevation profile. For it to be complete, we need to know the altitude at the begin (resp. at the end) of the edge, i.e. at some node point.

(iii) Some edges are entirely lying within adjacent contour lines and we can not extract any elevation data directly. However, given some OSM nodes elevation estimation, we can (and will) modelize the elevation profile along such edges by the unique affine curve which agrees with the nodes values at its extremities.

How should we choose an elevation distribution ? We can not know for sure what's going one between adjacent contour lines. Let's make the following assumption : road network have been designed to minimize the total elevation difference on the road network. Even though if might not be entirely true, it sill makes sense : in order to connect two areas across a mountain range, roads rather go over passes than peaks. Most of past and present civilizations worked hard to build dense road networks but were lazy enough to do it in a smart way.

Let's take a look at the graph obtained by removing edges that intersect contour lines :



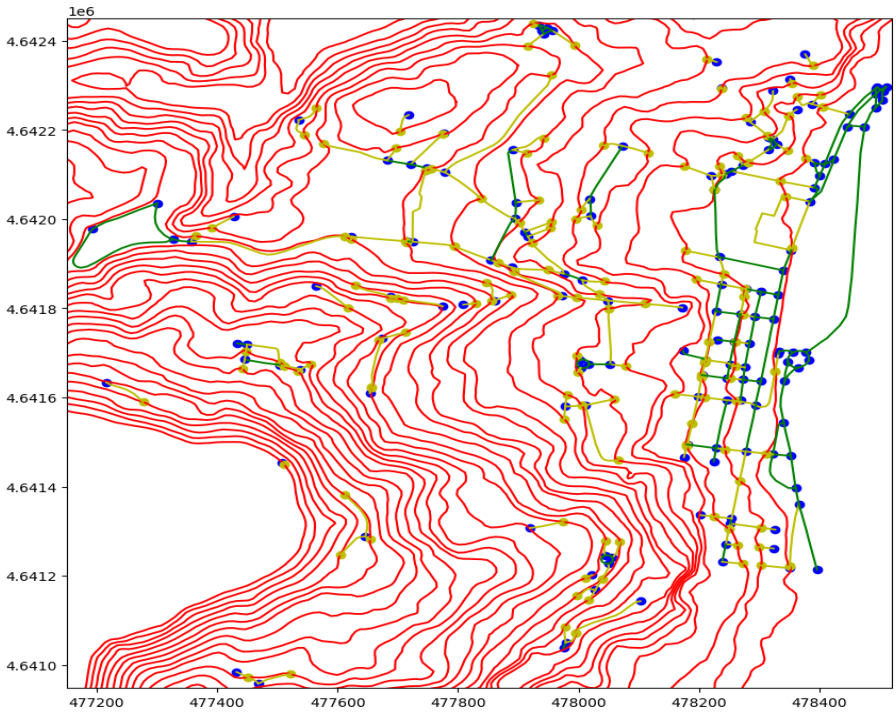
Let (V, E) be the nodes and the edges of this subgraph. We are looking for

a an elevation configuration $(v_1, \cdots, v_n) \in \mathbb{R}^{|V|}$ that minimizes the functional

$$L(v_1, \cdots, v_n) = \sum_{(i,j)\in E} (v_i - vj)^2$$

It turns out that this operator is pretty standard in graph theory : it is the *laplacian* of a graph. It is a quadratic form on the free vector space generated by the nodes of the graph. Its kernel (which is also the set of minimizing elements since the form is positive) is the set of elevation configurations $(v_1, \cdots, v_n) \in \mathbb{R}^{|V|}$ such that $v_i = v_j$ if $(i, j) \in E$, i.e. constant on each connected component of the graph.

Such a configuration is far from being suitable because we did not take into account any of the surrounding elevation data. Let's complete the graph as follows : for each edge that intersects at least one contour line and connecting u to v, add the closest intersection to u (resp. v) to the graph nodes and connect it to u (resp. to v) as shown in the figure below (the new nodes and edges are drawn in yellow).



Let's minimize the laplacian of this new graph but only over the OSM nodes variables since the elevation of the newly added intersection nodes is already known. In a more formal way, let $V_1 = V$ be the OSM nodes, V_2 the intersections nodes, $(c_1, \cdots, c_m) \in \mathbb{R}^{|V_2|}$ their elevations and E the edges of the updated graph. We want to minimize

$$\widetilde{L}(v_1, \cdots, v_n) = L(v_1, \cdots, v_n, c_1, \cdots, c_m)$$

$$= \sum_{(i,j)\in E, i\in V_1, j\in V_1} (v_i - v_j)^2 + \sum_{(i,j)\in E, i\in V_1, j\in V_2} (v_i - c_j)^2 + \sum_{(i,j)\in E, i\in V_2, j\in V_1} (c_i - v_j)^2 + \sum_{(i,j)\in E, i\in V_2, j\in V_2} (c_i - c_j)^2$$

$$= \langle Av, v\rangle + \langle B, v\rangle + C$$

which reduces to the sum of a quadratic form, a linear form and a constant.

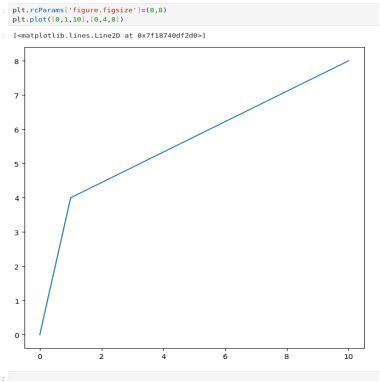
If the affine subspace of $\mathbb{R}^{|V1|+|V2|}$ defined by $v_{n+j} = c_j$ for $j = 1, \cdots, m$ does not intersect the null space of L, then the minimizing vector $v^\star = (v_1^\star, \cdots, v_n^\star)$ will yield a non trivial elevation configuration. The latter condition is equivalent to the existence of two different elevations among the intersection nodes.

Geometrically speaking : given a connected component of the first graph obtained by removing edges intersecting contour lines, if all the neighboring contour lines share the same elevation then there is no natural way to estimate the component's nodes elevations.

This method has one major drawback : it does not take into account the distance between nodes. Let's look at the following situation :

a (resp. c) are intersection nodes with elevations equal to 0 (resp. 8) and b is an OSM node whose elevation is to be estimated, $d(a, b) = 1$ and $d(b, c) = 9$.

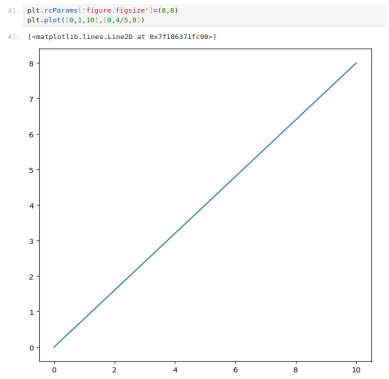
According to the previous formula, let v be the elevation of node b, then minimizing $(v - 0)^2 + (v - 8)^2$ yields $v = 4$. Let's draw the elevation profile following the road $a \to b \to c$:



This curve is not the elevation profile one would expect but why ?

Intuitively, we care more about minimizing the elevation gap between a and b rather than the one between a and c because b is closer to a. We can formalize our intuition by introducing weights : the closer two nodes i and j are, the more attention we want to pay to the term $(v_i - v_j)^2$ in the sum we're minimizing.

Let's divide each such term by the distance between the two nodes : the weighted version of the previous optimization problem is now minimizing $\frac{(v-0)^2}{1}+\frac{(v-8)^2}{9}$ i.e. $v=\frac{4}{5}$ which yields the more "natural" curve shown below :



Minimizing the weighted sum

$$\sum_{(i,j)\in E, i\in V_1, j\in V_1}\frac{(v_i-v_j)^2}{d(i,j)}+\sum_{(i,j)\in E, i\in V_1, j\in V_2}\frac{(v_i-c_j)^2}{d(i,j)}+\sum_{(i,j)\in E, i\in V_2, j\in V_1}\frac{(c_i-v_j)^2}{d(i,j)}+\sum_{(i,j)\in E, i\in V_2, j\in V_2}\frac{(c_i-c_j)^2}{d(i,j)}$$

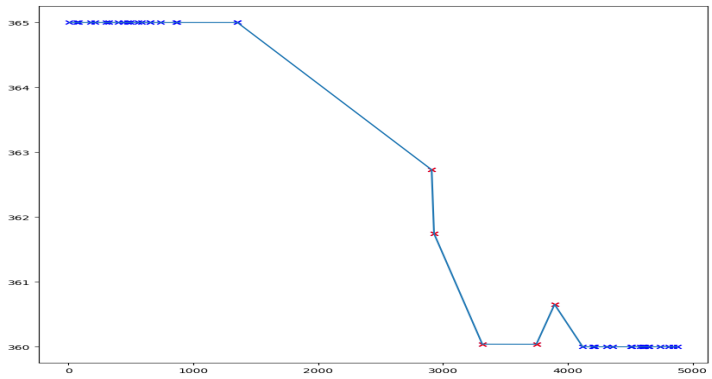
yield more accurate nodes elevations.

Once we gathered all the estimated nodes elevations, we can compute an elevation profile for each edge by concatenating the start node elevation, the intersections with the contour lines and the end node elevation.

It is always hard to measure the accuracy of an elevation model, one way to do it would be to inspect the elevation profile along some path (see the notebook https://github.com/pierremennesson/contours_lines/blob/main/visualizing_elevations.ipynb) and see if the curves don't look to irregular, i.e. with unlikely sharp variations.

The thing is that these curves *are supposed not to* be irregular as we kind of tried to minimize the irregularity. In order to assess the validity of our model, one should measure the elevation at some key points with a device but even these might have some measurement errors (by the way, the reader might want to check https://github.com/pierremennesson/garmin_elevation_process a project that also aims at computing precise elevation profiles but from cycling navigation data).

The author chosed as a test case (among others) a portion he's used to cycle known as *route de Belgodère* in Corsica. This road stretch is going downhill but with a very small slope, goes slightly up and then keep going down as shown below.



The red crosses correspond to estimated nodes elevations and the blue ones to contour lines intersection. If, in the graph obtained by removing intersecting edges, the 5 OSM nodes (in red) were forming an isolated chain, adding the neighboring intersection nodes would coerce the optimal configuration into removing the bump.

This is because the bumpy node is the extremity of other edges, small country roads the author never cycled or even paid attention. All he can say is that the estimation agrees with how he perceived the physical reality and that he would gladly take any skeptical reader for a beautiful ride in order to witness the different test cases in the beautiful island that is Corsica. Below is the itinerary (notice that the bump does not appear in the Google Maps elevation profile) :

https://www.google.com/maps/dir/42.5617624,8.8934139/42.5453795,+8.9313689/@42.5527862,8.9017123,15z/data=!3m1!4b1!4m7!4m6!1m0!1m3!2m2!1d8.9313689!2d42.5453795!3e1?entry=ttu

Remarks. *(i) When first diving into this problem, the author wanted to tackle the problem following these steps :*

(a) order contour lines by inclusion, i.e. find some tree structure on the contour lines.

(b) given some punctured plane portion obtained by taking the polygon delimited by a contour line and by removing the parts within its children contour lines, one can try to solve the Dirichlet problem for harmonic functions with the boundary conditions of being constant on the parent/children contour lines.

On top of being computationally expensive, it requires a tree structure on the contour lines, structure that, in the best scenario, is also computationally expensive when it's not impossible if one is dealing with open contour lines.

Nevertheless, the author was happily surprised to see the discrete graph equivalent formulation appearing naturally after having chosen other directions.

*(ii) We first introduced the optimization problem as "*minimizing the total elevation difference on a road network*". The introduction of the inverse distance weights reformulate the optimization problem into "*minimizing the mean slope along edges*" which is obvioulsy related yet slightly different.*

(iii) On top of being more computationally feasible than the approach described in (i), the graph discrete analog is likely to be more accurate : the 2 dimensional spatial elevation distribution within two adjacent contour lines might be really irregular and far from being harmonic. Nevertheless, the 1 dimensional elevation profile along edge, which correspond to roads built by lazy/smart humans are likely to follow some minimizing elevation gain/slope principle as the one stated in (ii). This is why we set choose weights to be the inverse of the distances along the OSM edges instead of the true physical distances between OSM nodes (think of hairpin turns).