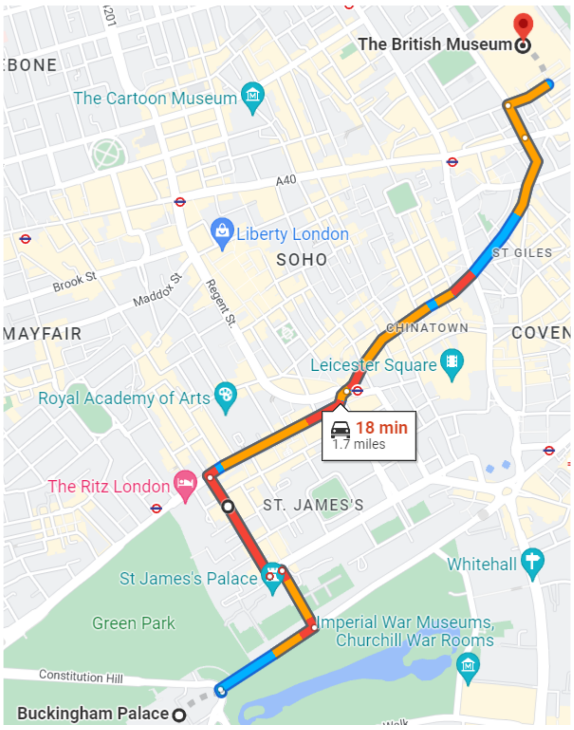
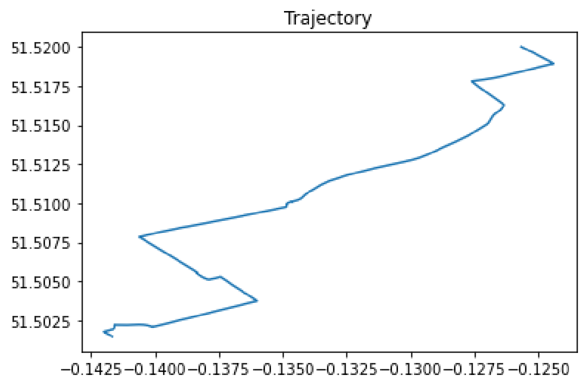
Linear Algebra Assignment 3

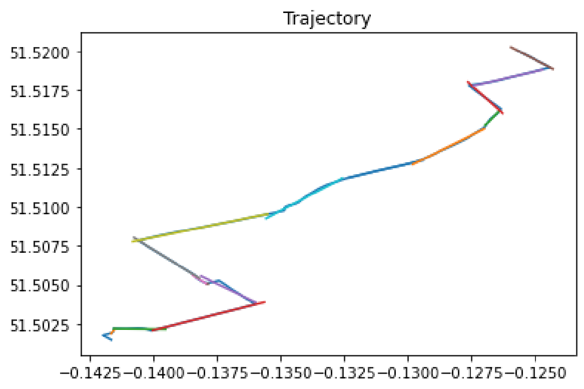
Anastasia Riana 邱慧莉 109006234

Q1:

The route we are using is a route from the British Museum to Buckingham Palace, both located in London, United Kingdom. This route is chosen because both are very popular tourist destinations, are not that far from each other and the route has more than 3 turns. The route has exactly 265 points and we divide it into fourteen line segments. The Google Maps route and the plot can be seen below:







Q2:

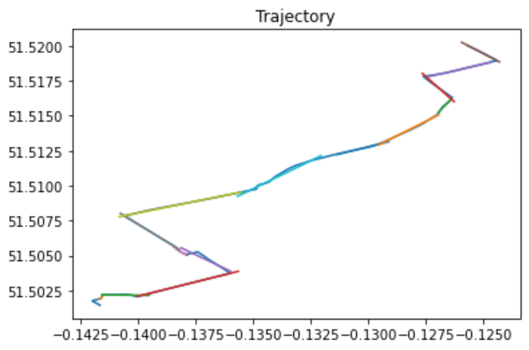
a) The statement describes a linear regression algorithm. Regression is one of the ways to solve a linear system Ax=b with more equations than unknowns. In most cases, the system has no solution because the vector b doesn't belong to C(A). In this case, the linear system is Aw=X and Az=Y.

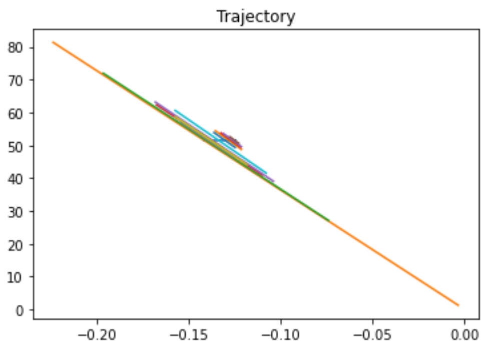
   The equation is derived from how we can use the least square method to find errors and residuals. To get the best result, the best straight line is the aspect that makes the overall error as small as possible, suppose it's e_1 = Aw-X and e_2 = Az-Y. Then, we take squared length of the vector, $\min_w ||e_1|| + \min_z ||e_2||$. If we input both e's, we will get $\min_w ||Aw-X|| + \min_z ||Az-Y||$.

b) *rcond* in *np.linear.lstsq* is a smaller singular value of A matrix. If the function is used for rank determination. singular values will be determined as zero if they are smaller than the *rcond* times the largest singular value of A matrix.

   There are two options of rcond, first is ***rcond = none*** which is used to silence the FutureWarning given if we discard the rcond parameter all together and use the new default. The second option is ***rcond = -1*** which is used to keep using the old behavior.

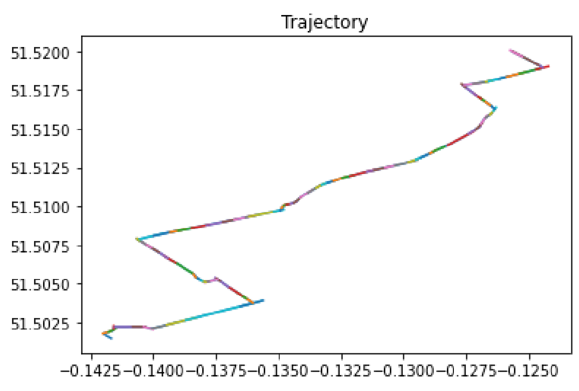
   Result of ***rcond = None (left)*** and ***rcond = 0.1 (right)***





   The main function of ***rcond*** is to zero out small entries of both w and z variables. The way it works is that it will zero out all entries in w and z if the entries fall below ***rcond*** * the largest singular value. Because most of the values of w and z are negative values, hence all entries are going to be zero, leading to the plot being linear.

Q3:



Correctness:

Using numpy.lstsq, we can get the residuals from w and z, then, we add them together. If the sum of both residuals are bigger than the defined error (in this case, it's 10^{-9}), then we append the segment to the plot. Else, it will iterate and check other points. The program can automatically break the trajectory into colorful line segments with an error of each line no more than 10^{-9}. In this implementation, each line is connected but for 2 points. I have tried numerous methods yet it yields the same result, where for those two points, it is going to extend a bit outwards.

Computing Efficiency:

In this implementation, in the first while loop, the lstsq function was called two times, hence the time complexity for this part will be F(2k) time which is the same as F(k) time. Then, in the for loop, the lstsq function was also called two times, hence the time complexity will be F(2k) which is pretty much equivalent to F(k). Combining those together, we will get the total computing efficiency to be F(k) which is linear time.

Compression Efficiency:

In this implementation, the error is 10^{-9} and it segmented the route to be 88 points, which results in all line segments to be connected. The compression ratio will be 88/265 = 33.2%.

Q4:

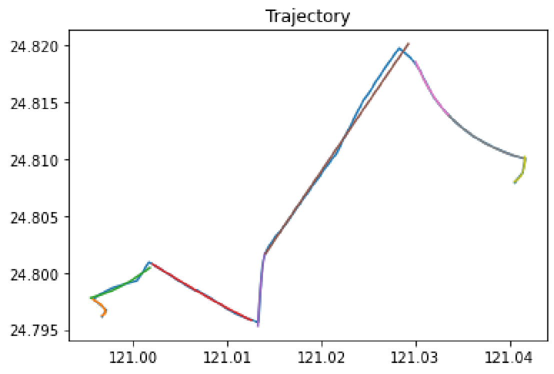
The equation can be derived to be in a matrix form. We can compute t^2, t for x and y and place it in a matrix. For example: suppose we have a point (1,3) and (2,3) as (x,t) and (y,t) respectively. Then, t^2 will be 9 and t will be 3 and we can place it in two matrices.

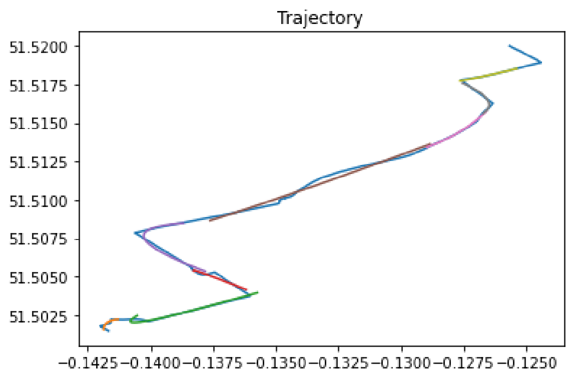
$$A = \left(\begin{array}{ccc|c} 9 & 3 & 1 & 1 \\ \cdots & \cdots & \cdots & \cdots \end{array}\right) \text{ describing x}$$

$$B = \left(\begin{array}{ccc|c} 9 & 3 & 1 & 2 \\ \cdots & \cdots & \cdots & \cdots \end{array}\right) \text{ describing y}$$

Then, using row operations, we will get the reduced row echelon form. In this case, we use list comprehension to get the values of the coefficients of a_1, a_2 and a_3 and b_1, b_2, and b_3. Then, we use the numpy.polyfit function to fit the curve according to the coefficients that we have computed previously. Then, we use the result of As and Bs from polyfit respectively to draw the curve using the function given by the TAs. We collect all the x's and y's and append it to its respective list. And to plot the points, we iterate through the list for the new x's and y's.

Results (left: given route ; right: chosen route):

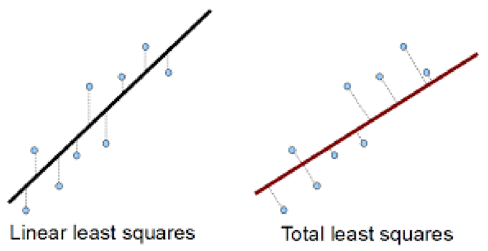




Q5:

Total least square method is a technique of regression that assumes both variables are subject to error, which is often the case in the real world, and is sometimes referred to as orthogonal regression. This method allows us to minimize the sum of squared error between a response variable and a fitted value. In the total least squares method, it is allowed to have errors in both independent and dependent variables. Then, we want to minimize that error and that error will contain observational error and residual.

The difference between total least square method and linear least square method can be seen in this following figure. We can see from the left figure that we take the distance of the plot point until it meets with the regression line. We can consider that it is orthogonal to the x-axis. However, in the right figure, the distance between the line and the plot point must be perpendicular to the equation line which makes it orthogonal towards the regression line.


Linear least squares
Total least squares