Computational project 2

**Noe Lomidze**

**CGS approach** vs **MGS approach**

CGS approach: compute vector which is orthogonal to all previously

constructed vectors

MGS approach: compute vector and make all remaining vectors

orthogonal to this vector.

So now about my solution.

I decided to use soccer data, which represents number of goals in each year, scored by FC Barcelona in LaLiga.

Functions.py

In that file I have implemented functions like:

Classic Gram-Schmidt, modified Gram-Schmidt, solving least squares using inverse, using QR factorization and also tested them.

Maybe I’m wrong but I noticed that the built in numpy function

Numpy.linalg.qr() and classic Gram-Schmidt gave me almost the same Q and R matrices, but As I remember, The Gram-Schmidt had better solution in like 0.000000000123 but still, (maybe I’m wrong IDK) but that was interesting.

In my Main.py file I did the linear regression model using least squares,

Time consumed was tiny:  
Time consumed : 0.0003198000049451366

In my second file Main1.py I did almost the same, but instead of linear regression I did just polynomial regression, in my case degree of 2.

Which was a bit smoother than linear regression.

Then I decided to add constraint on least squares but not by my hand, I used “ from scipy.optimize import least\_squares”

**Adding a Smoothness Constraint to a Least Squares Problem**

In the context of polynomial regression, we often want to ensure that the predictions do not change too rapidly, which can lead to overfitting. This is commonly referred to as a “smoothness” constraint.

The smoothness constraint can be enforced by adding bounds on the coefficients of the polynomial. This ensures that the predicted values do not grow too fast or too slow.

About the methods, which one is better and etc. :  
[The Classical Gram-Schmidt (CGS) and Modified Gram-Schmidt (MGS) are both methods used for QR factorization, which is a process of decomposing a matrix into a product of an orthogonal matrix and an upper triangular matrix](https://www.laurenthoeltgen.name/post/gram-schmidt/).

The Classical Gram-Schmidt method is straightforward and easy to understand. [However, it can suffer from numerical instability due to the accumulation of rounding errors in its computations](https://www.laurenthoeltgen.name/post/gram-schmidt/).

On the other hand, the Modified Gram-Schmidt method performs the same computational steps as CGS but in a slightly different order. [This modification improves the numerical stability of the algorithm, making it more reliable for practical use](https://www.laurenthoeltgen.name/post/gram-schmidt/).

In MGS, the components along each previously computed vector are immediately subtracted out of the rest of the columns as soon as they are computed. [This allows MGS to correct errors in each step, leading to more accurate results](https://math.stackexchange.com/questions/3913710/intuitive-explanation-of-why-the-modified-gram-schmidt-is-more-stable-than-the-c).

[In conclusion, while both methods have their uses, the Modified Gram-Schmidt method is generally preferred over the Classical Gram-Schmidt method due to its improved numerical stability](https://www.laurenthoeltgen.name/post/gram-schmidt/).

[However, the choice between CGS and MGS may also depend on the specific requirements of your application](https://www.laurenthoeltgen.name/post/gram-schmidt/" \t "_blank)