Implementing Linear Regression in Numpy

1 Write a function to generate an m+1 dimensional data set, of size n, consisting of m continuous independent variables (X) and one dependent variable (Y) defined as

$$y_i = x_i\beta + e$$

where,

- e is a Gaussuan distribution with mean 0 and standard deviation (σ), representing the unexplained variation in Y
- β is a random vector of dimensionality $m + 1$, representing the coefficients of the linear relationship between X and Y, and
- $\forall i \in [1, n], x_{i0} = 1$

The function should take the following parameters:

- σ: The spread of noise in the output variable
- n: The size of the data set
- m: The number of indepedent variables

Output from the function should be:

- X: An $n \times m$ numpy array of independent variable values (with a 1 in the first column)
- Y: The $n \times 1$ numpy array of output values
- β: The random coefficients used to generatre Y from X

2 Write a function that learns the parameters of a linear regression line given inputs

- X: An $n \times m$ numpy array of independent variable values
- Y: The $n \times 1$ numpy array of output values
- k: the number of iteractions (epochs)
- τ: the threshold on change in Cost function value from the previous to current iteration
- λ: the learning rate for Gradient Descent

The function should implement the Gradient Descent algorithm as discussed in class that initialises β with random values and then updates these values in each iteraction by moving in the the direction defined by the partial derivative of the cost function with respect to each of the coefficients. The function should use only one loop that ends after a number of iterations (k) or a threshold on the change in cost function value (τ).

The output should be an $m + 1$ dimensional vector of coefficients and the final cost function value.

3 Create a report investigating how differen values of n and σ impact the ability for your linear regression function to learn the coefficients, β, used to generate the output vector Y.