Final Project Report

Modélisation des incertitudes, mécanique probabiliste (MIFO)

Adélaïde Allemand
Muhammad Moeeze Hassan
Kokou Attiogbe

Submitted to
D. Clouteau, F. Gatti, F. Lopez-Cabbalero, A. Fau

Université Paris Saclay
20 Feb 2022

Contents

List of Figures

Chapter 1

Task 1: Generate the dataset

For each input variable of the problem (A_1, A_2, A_3, L, E, and P), we want to generate N samples folloging some prescribed distributions as given in Figure 1.1.

X	$p_X(x)$	μ_X	$\frac{\sigma_X}{\mu_X}$
$A_1[m^2]$	Lognormal	$7.5 \cdot 10^{-3}$	0.1
$A_2[m^2]$	Lognormal	$1.5 \cdot 10^{-3}$	0.1
$A_3[m^2]$	Lognormal	$5.0 \cdot 10^{-3}$	0.1
$E[MPa]$	Lognormal	70.0	0.05
$L[m]$	Lognormal	9.0	0.05
$P[kN]$	Gumbel max	350.0	0.1

Figure 1.1: Distributions followed by the input variables

For A_1, A_2, A_3, L and E:

- For each variable, we compute its parameters λ_S and ζ_S from its values of μ_S and σ_S.
- We generate 5 times N independent normal random variables following $\mathcal{N}(0,1)$. We obtain vectors S_i, $i = 1, 5$.
- For each of them, we perform a Rosenblatt transformation, meaning we do $X_i = \exp(\lambda_S + \zeta_S S_i)$.

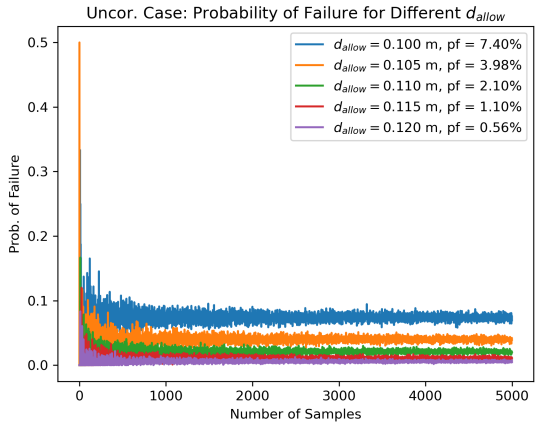
For P:

- We compute α_S and s_0 from μ_S and σ_S.
- We generate N independent normal random variables following $\mathcal{U}(0,1)$. We obtain a vector S_6.
- We perform a Rosenblatt transformation, meaning we do $X_6 = s_0 - \alpha_S \cdot \ln(-\ln S_6)$.

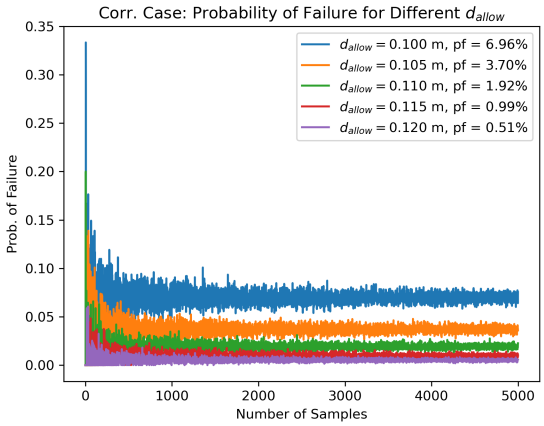
For each of the N combinations of input variables, we compute the displacement d value in a deterministic way.

Then, we compute the limit state function g for each value of d and the performance function I for each value of g. We can thus compute the probability of failure as follows : $\hat{p}_f = \frac{1}{N}\sum_{i=1}^{N} I(d_i)$. The plots are shown in Figure 1.2a and Figure 1.2b.

We also compute the statistical mean $\mu_d = \frac{1}{N}\sum_{i=1}^{N} d_i$ and the statistical standard deviation $\sigma_d = \frac{1}{N}\sum_{i=1}^{N}(d_i - \mu_d)^2$. Results are shown on Figure 1.3a and Figure 1.3b, where we see that a certain amount of samples is necessary to estimate accurately the probability of failure ($N \geq 100$). Results are compared with and without A_1 and L being correlated.

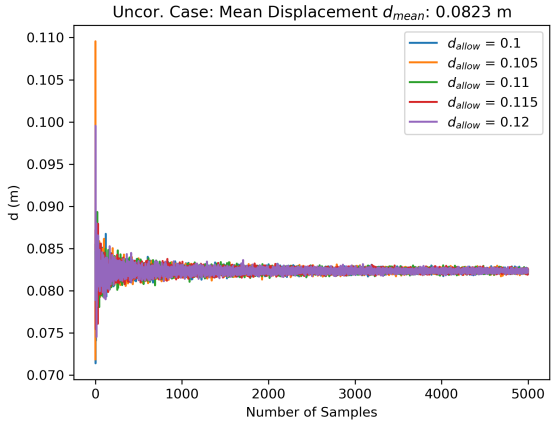


(a) Probability of Failure for LDB1 Dataset

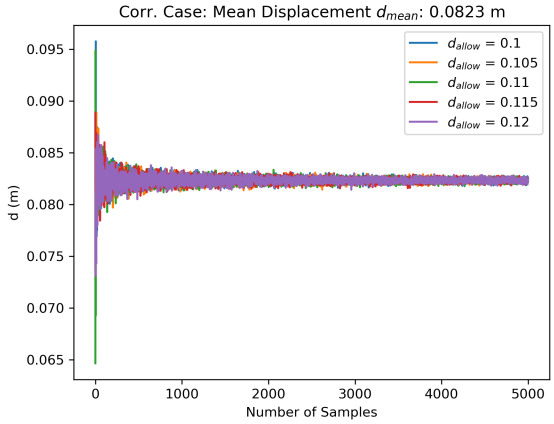


(b) Probability of Failure for LDB2 Dataset ($\rho = 0.2$)

Figure 1.2: Probability of failure



(a) Mean Displacements for LDB1 Dataset



(b) Mean Displacements for LDB2 Dataset ($\rho = 0.2$)

Figure 1.3: Mean displacement

We can also notice that the displacement d follows a certain distribution, which is not the same when A_1 and L are correlated (see Figure 1.4).

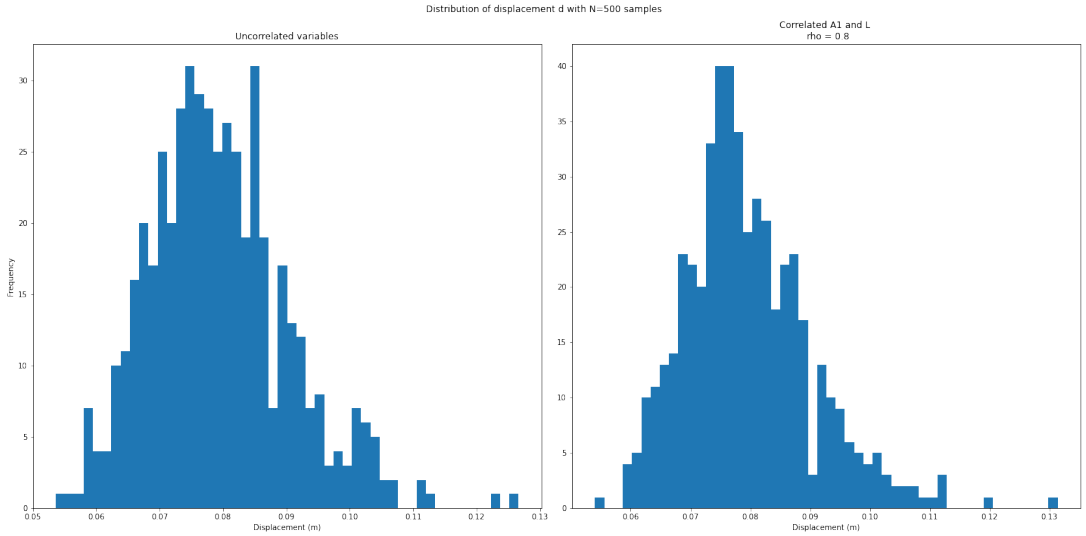


Figure 1.4: Distribution of d

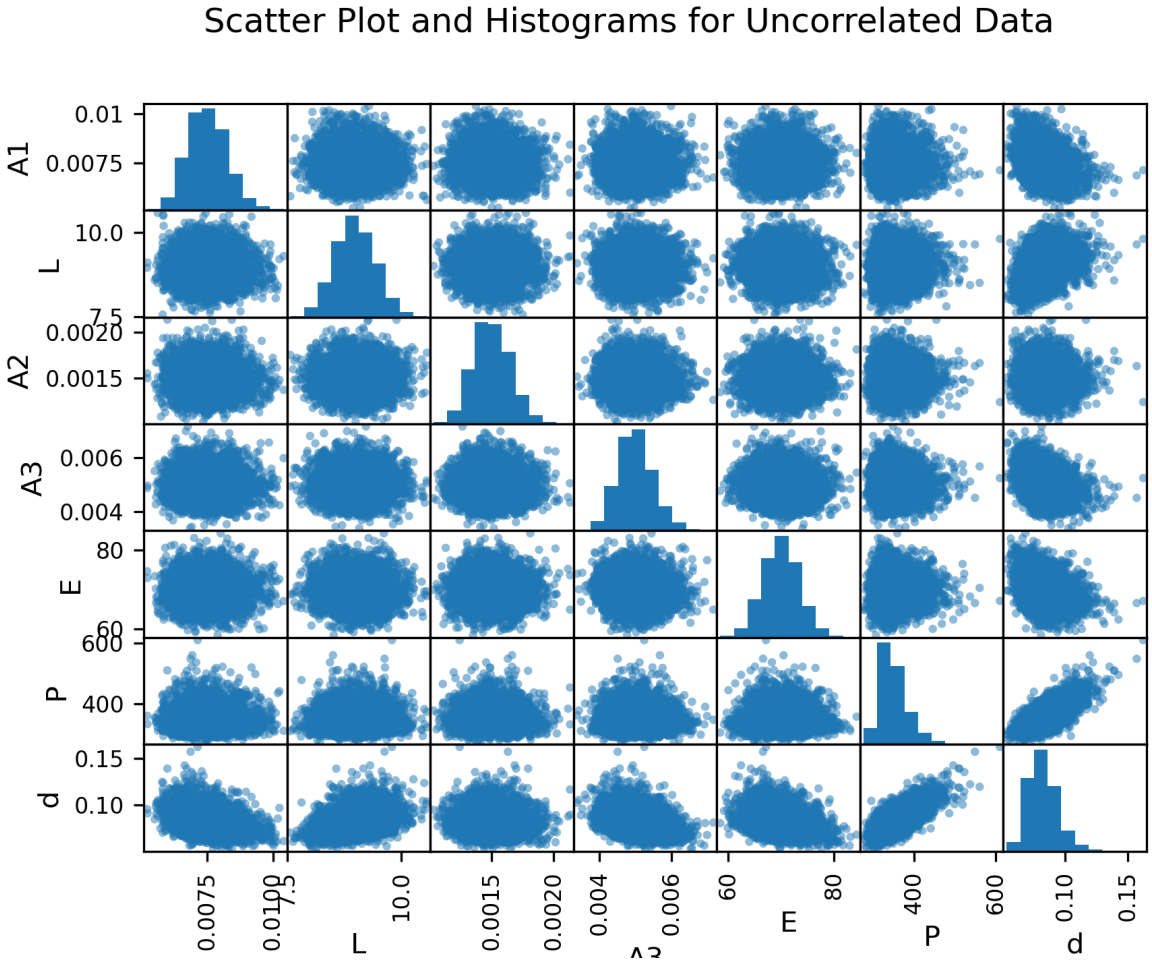


Figure 1.5: Scatter Plots between Variables in Uncorrelated Data Case

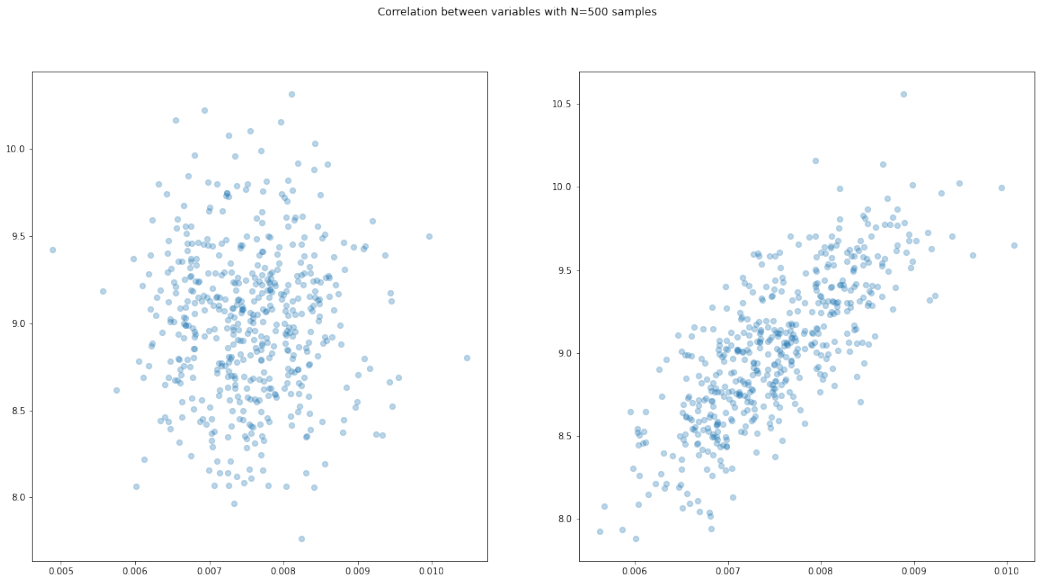


Figure 1.6: Scatter Plots between Variables in Correlated A1 and L variables ($\rho = 0.8$)

From the plots in the Figure 1.5, we can see that the histograms are following a Log Normal Distribution for all variables except for P and d. For P, the distribution is set as Gumbel.

Additionally, from the Figure 1.6, we can see that the scatter between A1 and L are correlated which leads to the circular-like scatter to turn into an elliptical one. It is to be noted that, unlike asked in the question, we used a $\rho = 0.8$ just to make the visualization easier. For the rest of the report, it is set as ρ.

Chapter 2

Task 2: Basic statistics on the dataset

Pearson Correlation was found using `pandas` library in `python`. Using it, the results are shown in the Figure 2.1. We can see that for the correlation between A_1 and L is equal to $\rho = 0.2$ which we set in the chapter 1.

```
Correlation Matrix for Uncorrelated Data:

          A_1         L       A_2       A_3         E         P         D
A_1  1.000000 -0.015401 -0.012175  0.000626  0.003658  0.008574 -0.340450
L   -0.015401  1.000000  0.004058  0.006824  0.001834  0.013950  0.364242
A_2 -0.012175  0.004058  1.000000 -0.018763 -0.021919  0.002338 -0.000435
A_3  0.000626  0.006824 -0.018763  1.000000  0.002835 -0.000680 -0.341867
E    0.003658  0.001834 -0.021919  0.002835  1.000000 -0.010420 -0.356897
P    0.008574  0.013950  0.002338 -0.000680 -0.010420  1.000000  0.711880
D   -0.340450  0.364242 -0.000435 -0.341867 -0.356897  0.711880  1.000000

================================================

Correlation Matrix for Correlated Data:

          A_1         L       A_2       A_3         E         P         D
A_1  1.000000  0.208200  0.001508 -0.019886 -0.018376  0.000141 -0.259386
L    0.208200  1.000000 -0.008894 -0.003889 -0.005104  0.002778  0.297434
A_2  0.001508 -0.008894  1.000000 -0.002497  0.004412 -0.010091 -0.032207
A_3 -0.019886 -0.003889 -0.002497  1.000000 -0.006469  0.009187 -0.345505
E   -0.018376 -0.005104  0.004412 -0.006469  1.000000  0.017986 -0.335027
P    0.000141  0.002778 -0.010091  0.009187  0.017986  1.000000  0.734301
D   -0.259386  0.297434 -0.032207 -0.345505 -0.335027  0.734301  1.000000

================================================
```

Figure 2.1: Correlation Matrix for LDB1 and LDB2

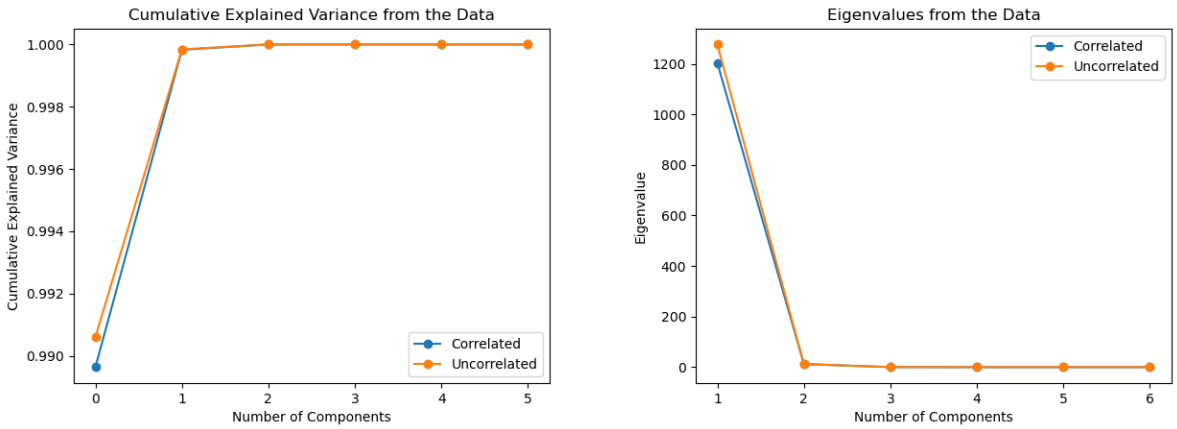
Chapter 3

Task 3: PCA

PCA analysis was done using the `sklearn.decomposition.PCA`. Following plots were plotted to observe the effectiveness of PCA. It is to be noted that there is not much variation between the datasets $LDB1$ and $LDB2$ plots.

- **Cumulative Explained Variance:** It can be seen from the Figure 3.1a that the explained variance converges to nearly 1.0 for only two components. This highlights the fact that the variance can be explained by only two input variables in the original dataset.
- **Eigenvalues**: Figure 3.1b confirms that the eigenvalues for two components are enough for explaining the dataset variance in both cases.
- **Coefficients for PCA Components**: For the first three components of PCA, the coefficients were plotted against the original variables in in order to see which variable is associated with each of these three PCA components (Figure 3.2). We can see that the first component is associated with P; this makes perfect sense.
- **Biplots**: Biplots for the first and second components of PCA reduced data set along with the directions of spread for all the components. The length of arrows are scaled by the magnitude of the variance.

In conclusion, only two components are required to effectively explain the dataset. The first component is associated with P, which is the loading, and this makes perfect sense (as loading is the most obvious parameter that increases or decreases the displacement). The variance for this component is 0.99: this shows a high dependency. The second component is Young's Modulus, explained with a ratio of 0.01.



(a) Cumulative Variance for LDB1 and LDB2 Datasets

(b) Eigenvalues for LDB1 and LDB2 Datasets

Figure 3.1: Cumulative Variance and Eigenvalues

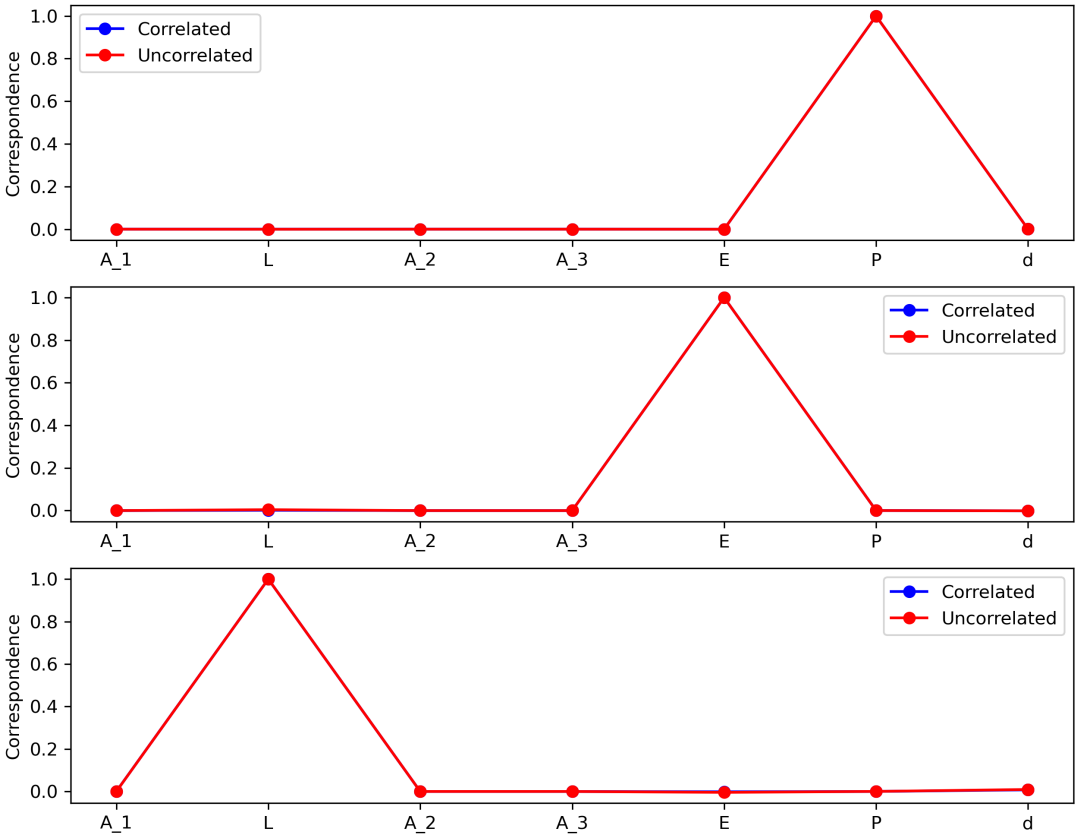
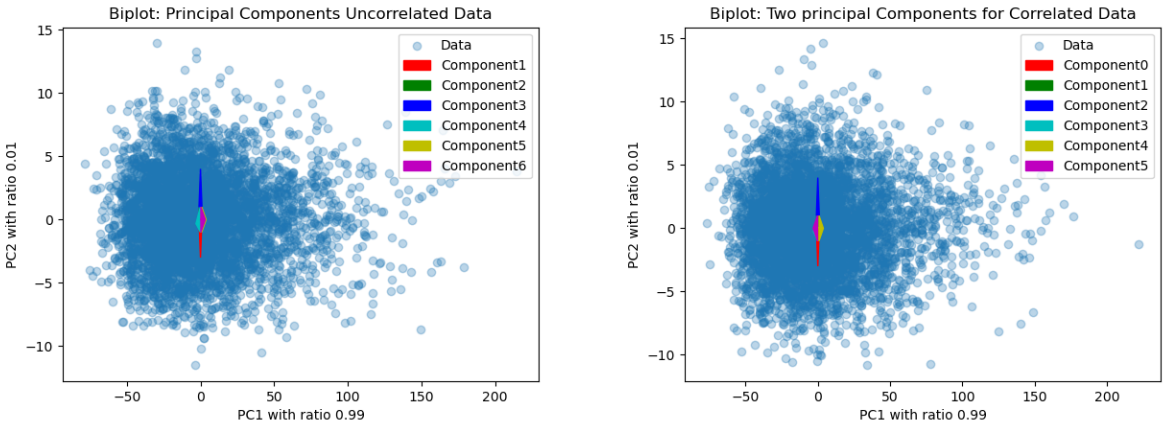


Figure 3.2: Component of PCA vs the Original Variables from LDB1 and LDB2 Datasets



(a) Biplot with Direction of Components for LDB1 Datasets

(b) Biplot with Direction of Components for LDB2 Datasets

Figure 3.3: Biplots

Chapter 4

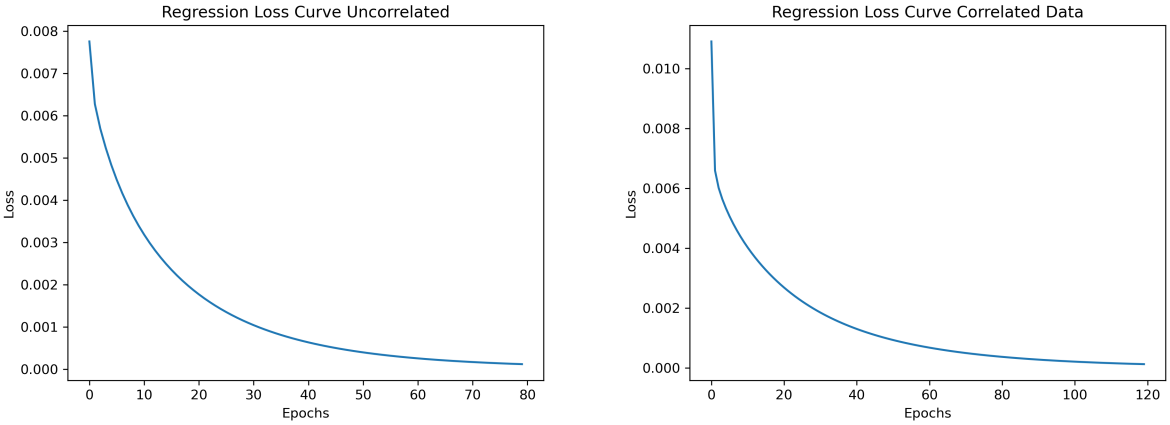
Task 4: Design a metamodel

The meta-model chosen was to be an Artificial Neural Network (ANN). The network was created using `sklearn` library, and its neural network library `MLPRegressor` (Multi-Layer Perceptron Regressor) was used to train on the dataset with the following initial parameters.

- **Hidden Layers**: $(256, 128, 64)$
- **Activation**: `relu`
- **Solver**: `adam`
- **Learning Rate**: `adaptive`
- **Alpha**: 0.01
- **Early Stopping**: `True`

As it can be seen from the parameters, one way to stop **overfitting**, was to pass `early_stopping` to stop the training as soon as the `validation_score` stopped improving (the validation dataset is a fraction of the training dataset, `default = 0.1`).

The Regression loss curve is shown in Figure 4.1a and Figure 4.1b for the two datasets `LDB1` and `LDB2`.



(a) Regression Loss Curve for LDB1

(b) Regression Loss Curve for LDB2

Figure 4.1: Regression Loss Curves

The metrics R^2 score, $RMSE$, and ϵ_m for the two datasets are shown in Figure 4.2 and Figure 4.3. We can see nearly $R^2 = 1$ and very small errors in both the cases.

```
Uncorrelated Data
==================================

R-squared score: 0.9960033728097237
R-squared score from Sklearn Metrics: 0.9960033728097237
R-squared score from Sklearn Metrics on Train Data: 0.996420936516582
RMSE: 0.0007395636342332756
RMSE from Sklearn Metrics: 0.0007395636342332756
epsilon: 0.23196910714683594
```

	Predicted	Original
0	0.100319	0.099852
1	0.081089	0.080603
2	0.072986	0.073507
3	0.084626	0.084041
4	0.080388	0.080482

Figure 4.2: Metrics and Example Comparison between Original and Predicted Data for LDB1 Dataset

It can also be seen that R^2 and $RMSE$ from `sklearn.metrics` and $MLPRegressor$ score are equal: this validates our implementation.

Hyper-Parameter Tuning

For finding the optimal hyperparameters for the ANN, it is generally proposed to do a grid search over parameters, and to update them based on the best score curves in the `MLPRegressor` attributes. A rough grid search yielded the results shown Figure 4.4 and Figure 4.5. We can see a slight improvement when larger first and second layers were chosen (`hidden layer: (1000, 500)`) over a denser network (`256, 128, 64`), whereas results didn't change a lot. Without any hypothesis, it can be said that the correlation was such that it didn't need a denser network to be approximated with the meta-model.

```
Correlated Data
================================

R-squared score: 0.9954452492919619
R-squared score from Sklearn Metrics: 0.9954452492919619
R-squared score from Sklearn Metrics on Train Data: 0.9962240890783548
RMSE: 0.0007557209003381558
RMSE from Sklearn Metrics: 0.0007557209003381558
epsilon: -0.1727388265590489
```

	Predicted	Original
0	0.080337	0.080603
1	0.075289	0.076299
2	0.090289	0.090484
3	0.070357	0.070472
4	0.078053	0.078556

Figure 4.3: Metrics and Example Comparison between Original and Predicted Data for LDB2 Dataset

```
{'activation': 'relu', 'alpha': 0.005, 'hidden_layer_sizes': (1000, 500), 'learning_rate': 'adaptive'}
R-squared score: 0.9991616867548785
R-squared score from Sklearn Metrics: 0.9991616867548785
RMSE: 0.0003387130211432493
RMSE from Sklearn Metrics: 0.0003387130211432493
epsilon: 0.21271846804989794
```

	Predicted	Original
0	0.100124	0.099852
1	0.080929	0.080603
2	0.074062	0.073507
3	0.084385	0.084041
4	0.080452	0.080482

Figure 4.4: Metrics and Example Comparison between Original and Predicted Data for LDB1 Dataset after Grid Search

```
{'activation': 'relu', 'alpha': 0.005, 'hidden_layer_sizes': (1000, 500), 'learning_rate': 'adaptive'}
R-squared score: 0.9976894924980603
R-squared score from Sklearn Metrics: 0.9976894924980603
RMSE: 0.0005382484883130868
RMSE from Sklearn Metrics: 0.0005382484883130868
epsilon: -0.3859970173850037
```

	Predicted	Original
0	0.080162	0.080603
1	0.075919	0.076299
2	0.089889	0.090484
3	0.070677	0.070472
4	0.078240	0.078556

Figure 4.5: Metrics and Example Comparison between Original and Predicted Data for LDB2 Dataset after Grid Search