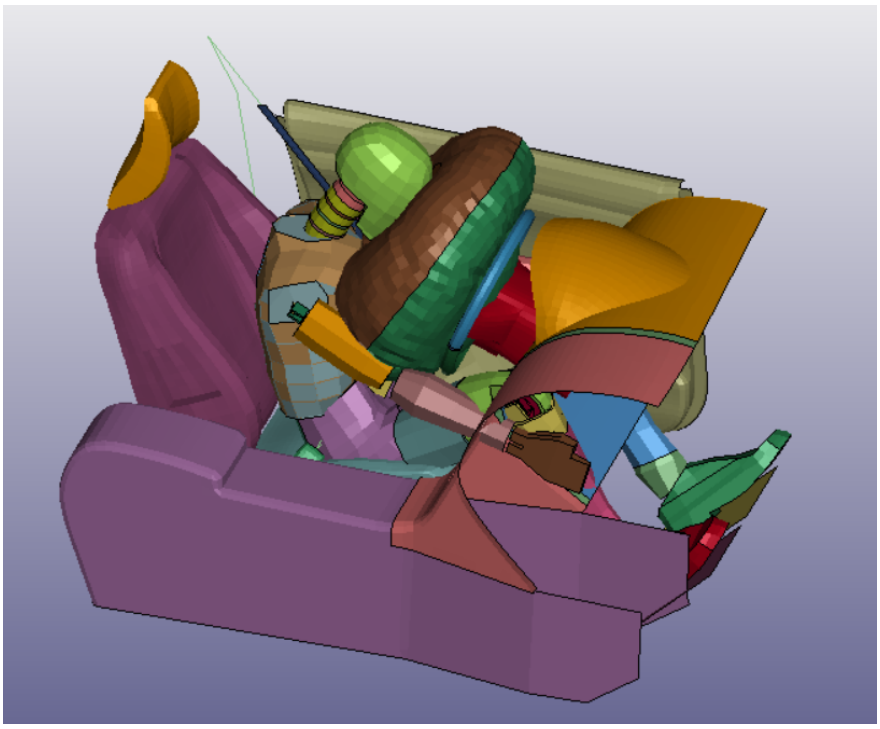
CHALMERS
UNIVERSITY OF TECHNOLOGY



Applicability of using machine learning to improve computational efficiency of combined pre-crash and crash simulations

Master's thesis in Automotive engineering

Peifeng Wang & Qiang Xu

DEPARTMENT OF MECHANICS AND MARITIME SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2022
www.chalmers.se

Master's thesis 2022

Applicability of using machine learning to improve computational efficiency of combined pre-crash and crash simulations

Peifeng Wang
Qiang Xu

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mechanics and Maritime Sciences
Chalmers University of Technology
Gothenburg, Sweden 2022

Applicability of using machine learning to improve computational efficiency of combined pre-crash and crash simulations
Peifeng Wang
Qiang Xu

Supervisor: Ekant Mishra, Karl-Johan Larsson, Johan Iraeus
Examiner: Johan Iraeus

Master's Thesis 2022:67
Department of Mechanics and Maritime Sciences
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone: + 46 (0)31-772 1000

Applicability of using machine learning to improve computational efficiency of combined pre-crash and crash simulations

Peifeng Wang
Qiang Xu
Department of Mechanics and Maritime Sciences
Chalmers University of Technology

Abstract

Vehicle occupant safety is commonly evaluated in dynamic finite element simulations. Common finite element occupant models are human body models (HBM), active human body models (AHBM), and dummy models in pre-crash and crash simulations are time-consuming, and a simulation may cost dozens of hours or even days, which will affect economic efficiency. Machine learning offers a potentially time-efficient approach with high accuracy. This thesis project aimed to compare finite element (FE) simulations and machine learning (ML) prediction results to determine the applicability and efficiency of the ML method for pre-crash and crash simulations. The Hybrid III fast finite element model was always used as an occupant model in combined pre-crash (braking or steering) and crash simulations. Six parameters related to occupant kinematics and safety system design were varied. Pre-pretensioner force, activation time of pre-pretensioner, braking level, braking duration, steering level, and steering duration in 200 FE simulations. From these FE simulations head kinematics, rib strains and seatbelt forces were extracted as output. Using the commercially available ML software, LUNAR, corresponding curves were predicted with algorithms that LUNAR suggested. Then the prediction accuracy was evaluated by calculating mean square error (MSE), root mean square error (RMSE), mean absolute error (MAE) and mean absolute percentage error (MAPE) between actual FE- simulation results and ML-model predictions. The conclusion shows that ML can provide a method that saves more than 19% of the time to obtain highly accurate results for head node displacement, rib strain and belt force prediction in both braking and steering simulations, while for head node acceleration especially in in-crash phase, it has low accuracy.

Keywords: FE, ML, Hybrid III fast model, Pre-crash and crash simulations, DOE, LUNAR, Algorithms, Interpolation, Error evaluation, Time reduction.

Acknowledgements

We sincerely thank Autoliv and Chalmers University of Technology for providing us with the precious and valuable opportunity for this thesis and for the company's kind support for us. First, we would like to grant oceans of thanks to our respected supervisor to Ekant Mishra, who taught us a lot and guided our thesis project from beginning to end. Our gratitude also goes to Karl-Johan Larsson and Johan Iraeus. They also provided a lot of support and helpful advice during our thesis project. There are other teachers and friends in the company and school who also helped us a lot. We could not have completed this thesis without their assistance.

Peifeng Wang, Qiang Xu, Gothenburg, August 2022

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

AE	Autoencoder
AEBS	The advanced emergency braking system
AHBM	Active human body model
AI	Artificial intelligence
ATD	Anthropometric test device
CAS	Collision avoidance system
DL	Deep learning
DOE	Design of experiment
FE	Finite element
FEM	Finite element method
FFNN	Feedforward neural networks
GPR	Gaussian process regression
HBM	Human body model
HIC	Head injury criteria
HPC	High-performance computer
MAE	Mean absolute error
MAPE	Mean absolute percentage error
ML	Machine learning
MSE	Mean square error
PCA	Principal component analysis
POD	Proper orthogonal decomposition
RBF	Radial basic function
RER	Random forest regression
ROM	Reduced order modelling
RMSE	Root mean square error
SVM	Support vector machine
THUMS	Total human model for safety

Nomenclature

Below is the nomenclature of indices, sets, parameters, and variables that have been used throughout this thesis.

Indices

i	Indices for clustering samples
j	Indices for cluster centers
k	Indices for clusters

Sets

a	Set of cluster centers
X	Set of original dimensional samples
Y	Set of samples after dimensionality reduction
D	Dataset of all samples
S	A training set of samples in the dataset
V	The validation set of samples in the dataset
T	The test set of samples in the dataset

Parameters

k	The number of clusters
n	The original number of dimensions
$n^{'}$	The target number of dimensions
m	The number of n- dimension samples
C	The covariance matrix of the samples
m_{min}	The minimum number of test points

p	The number of given parameters.
t	could be 1 or 2

Variables

x_i	The ith sample
a_j	The jth cluster center
c_k	The kth cluster
x_{ni}	The ith dimension's value of the nth sample
x_{mi}	The ith dimension' value of the mth sample
y_i	The ith value of the original data
$\hat{y}_i$	The ith value of the prediction
z_i	The ith sample of DOE generation
$d(x_i, x_j)$	The inter-site distance between the ith and jth sample
d_{ij}	The inter-site distance between the ith and jth sample

Operations

$\cup$	Calculate the union of two sets
$\cap$	Calculate the intersection of two sets

Contents

List of Figures

List of Tables

1

Introduction

1.1 Background

1.1.1 Traffic crashes

In modern society, traffic collisions are happening every day all over the world. Road traffic crashes are a global problem, because the economic losses and the casualties caused by the crashes cannot be neglected. Twenty years ago, up to 50 million people are injured in road traffic crashes every year, and because of road traffic crashes, 1.2 million people lost their lives [1]. The numbers grow over time, according to WHO 2014, 1.25 million people died because of road traffic crashes which means more than 3400 people per day [2]. Also from WHO 2014, in the figure 1.1 below, it shows that road traffic crashes took 24.4% of all injury-related deaths globally [2].

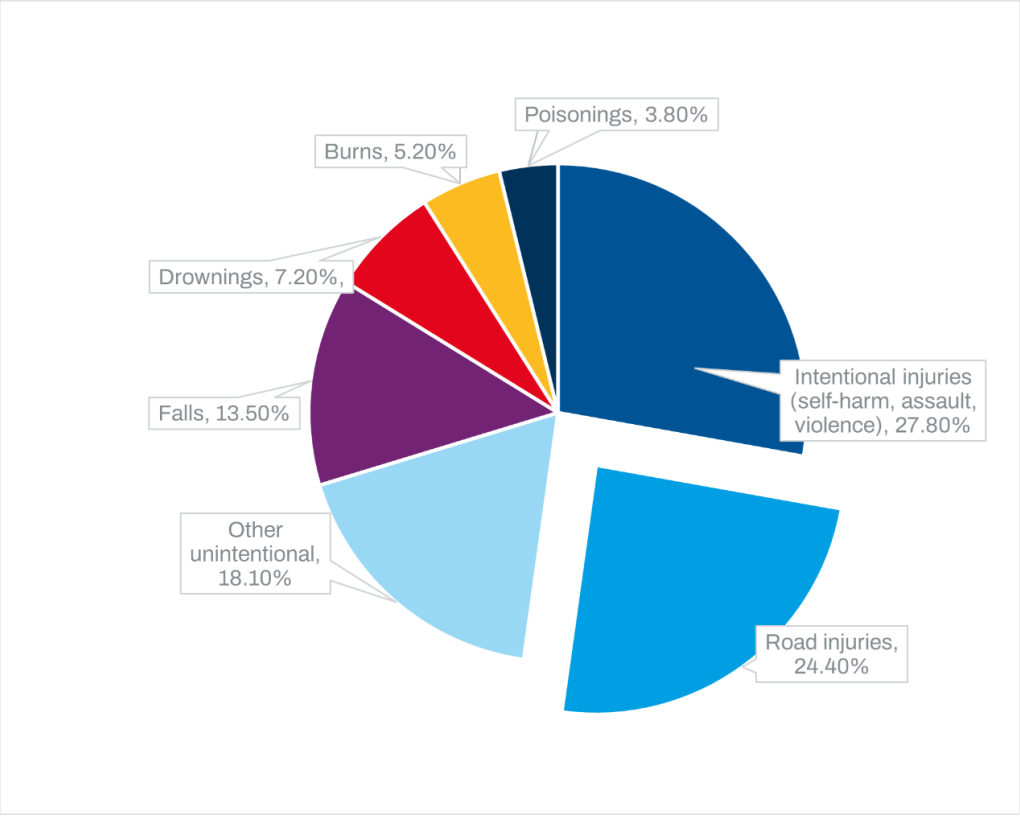


Figure 1.1: Global Mortality from All Injuries, 2012

Road traffic crashes are a problem that the whole world needs to pay attention to. So how to avoid it or reduce the losses it may cause is a big question worth researching.

1.1.2 Passive safety devices

In order to avoid or reduce the damage to the occupants in the crash, many kinds of passive safety devices have been invented. Seatbelts and airbags are two common passive safety devices.
The seatbelt is a kind of vehicle safety devices which is designed to protect the occupants against harmful movement during a crash or an emergency stop. The two-point belt was firstly made and used in the early 1800s [3]. A two-point belt, as its name, is attached at two endpoints laterally. The two-point seat belt, which is also called a lap belt, wraps around the occupant's hip to restrain the occupant on the seat. It was tried on aircraft in the 1900s [4], and it can still be seen on passenger planes today. However, in the event of an accident, the force will naturally concentrate on the waist of the member, which may cause injury. In 1959, the three-point seat belt was invented [5]. The general three-point seat belt consists of three parts: webbing, retractor and fixing mechanism, shown in figure 1.2. The three-point seatbelt is fastened from the shoulders and the same side of the waist to the other side of the waist, so the energy from moving body of occupants can be distributed to their shoulders, chest and pelvis. It is an improved production of two-point seat belts, so it is safer and more comfortable for the occupants [4] [5].

Figure 1.2: The three-point seatbelt[6]

Nowadays the three-point belt usually contains retractors and pretensioners. The retractor controls the seatbelt webbing. The role of the seatbelt pretensioners is to restrain the occupant on the seat by tightening the seat belt before the collision to give occupant a safer contact with airbags. Seat belts are made of extremely strong synthetic fibers. After the seat belt is fastened, the retractor with a load-limit device will automatically tighten it. When the vehicle has an emergency braking, a frontal collision or a rollover, the occupant will stretch the seat belt quickly and violently. At this moment, the retractor with a load-limit device can instantly jam the seat belt, making the occupant close to the seat, reducing the risk of injury in a collision. An electric pre-pretensioner module can be in retractor systems now. The pre-pretensioner could give a force in pre-crash scenarios to tighten the seat belt before a crash and to adjust the occupants' position to decrease the risk of injury due to occupant out-of-position [7]. By properly using seatbelts, 50% to 80% of all deaths of unbelted occupants in road traffic collisions could have been prevented [8].

The airbag system is also a passive safety protection system that works in conjunction with seat belts to provide occupants with effective crash protection. The airbag was first proposed in 1953. Mostly it contains airbags, sensors and electric control devices. Frontal airbags can be installed on the driver's and the other passenger's sides. The airbag in drivers' side is usually installed in the steering wheel and the airbag in passengers' side is usually installed inside the platform in front of the passenger, as shown in figure 1.3. During crashes, there will be a huge impact force on the occupant's chest and head. And this force could be distributed by the airbags. Apart from that, the airbag can also prevent the occupant's head from colliding with the other body parts which may cause more injuries. During a frontal crash, airbag can protect occupants effectively. Even if the seatbelts aren't fastened, airbag also could decrease the risks of injury.

Figure 1.3: One kind of frontal airbags[9]

1.1.3 Avoidance measures

More safety equipment is being developed to ensure the occupants' safety. A Collision Avoidance System (CAS) is an advanced driver assistance system which is designed for preventing or reducing the severity of a collision [10]. It could contain a forward collision warning system, an advanced emergency braking system (AEBS), etc. It may reduce the crash speed or prevent vehicles from crashing. Using this system, when a potential collision object such as a pedestrian, another vehicle, an obstacle is detected, an alarm will sound or other measures will be taken to avoid the crash. For the past decade, many automatic collision avoidance systems have already been on the market and are being heavily used [11].

In daily life, when people encounter traffic crashes, they usually choose to brake or steer or both. Therefore, it is reasonable to use braking and steering as measures in pre-crash scenarios.

1.1.4 Crash test and simulations

Crash testing is a destructive test that generally evaluates crashworthiness and crash compatibility of various modes of transportation or associated systems and components.

Figure 1.4: Frontal impact test[12]

It has many types, such as frontal-impact tests, side-impact tests and sled testing, etc. The frontal-impact crash test is the most common crash test, figure 1.4 is an example. A sled test is a cost-effective way of testing components like airbags and seats. It is simpler in structure than vehicles. All these tests can either be performed in real life or virtually as a computer simulation. A computer model is also a cost-effective way since the economic cost of the computer model is much lower than the cost of a full crash test (facilities, test fields).

1.1.5 Finite element simulation and machine learning

To prevent a traffic crash, braking or steering or a combination of both may be taken. This may influence the position and posture of the occupants due to inertial forces. The human body model (HBM) or active human body model (AHBM) are common but expensive choices for finite element simulation in pre-crash and crash scenarios. The calculation of an active HBM with active muscles or HBMs is complicated and time-consuming. Dozens of hours of analysis for a model are not cost-effective. Results also could be obtained in a shorter period using machine learning, so machine learning may reduce the number of FE simulations needed. It could be time-efficient way for the calculations and predictions of finite element analysis. Previously, machine learning models have been used to predict FE results. Machine learning may offer a time-efficient and economical approach to such analyses.

One previous study evaluated the use of ML models in building surrogates to predict HBM responses and injury metrics. They found that, for surrogate models of the Hybrid III FAST model, linear models are more suitable for training sizes less than 60. However, tree-based boosting methods performed better for more training samples up to 120 [13].

In another project, the nodal kinematic time histories of a FE human body model were predicted using surrogate ML models trained on in-crash FE simulations data [14]. Gaussian process regression (GPR), random forest regression (RFR), and feed-forward neural networks (FFNN) were paired with principal component analysis (PCA) and autoencoder (AE) for dimensionally reduction. Then the error for unseen test data was evaluated by using mean absolute error and mean Euclidean distance error for varied sample sizes. The results showed that PCA-ML had a much higher error and variance than AE-ML, and RFR was less accurate compared to GPR and FFNN for a simulation size of 50 [14]. Furthermore, another previous study used a different way of machine learning to train models to reduce the computational time by using the LUNAR software [15]. They used the SAFER THUMS v9.0 without muscle activation in a simple sled setup and with active muscles in a complex pre-crash braking and crash scenario. Then by using the LUNAR software, ML models were built and trained. Kinematics, forces, and strains from FE and ML methods were compared, and the accuracy of prediction and error between the two methods was determined. The errors were the mean error between two obtained curves like head acceleration plots, lumbar Z-force plots, lumbar Y-moment plots, and rib and brain strains from FE and ML methods. For the first simple sled setup, using machine learning in LUNAR could improve the time efficiency of calculations by 74.6%, while the error was between 0.7% and 11.5%. When the same ML method was also applied to the scenarios of pre-crash braking and crash, time reduction reached about 96,8%, while the mean error varied between 6,9% and 76,1%, which means that the machine learning method has no practical effect [15].

Most previous studies used AHBM and HBM. Unlike AHBM and HBM, the Hybrid III fast model dummy model is a simpler but faster model. Machine learning may also have the potential to provide a time-efficient approach to it while maintaining high prediction accuracy. So that the successful methods applied to the Hybrid III Fast Model Dummy Model may be also used for the HBMs.

1.2 Objective

This project aims to develop a method for time-efficient analysis of occupant pre-crash kinematics and in-crash injury risks by training a model using machine learning in LUNAR. By comparing the results and computing time in the two methods, the applicability and efficiency of the ML method can be determined.

1.3 Limitations

a. The hybrid III fast model, a dummy model is used in this project. It's a rather simple model compared to AHBM and HBM. Although it requires much less time and computing resources than AHBM and HBM, it is rather simple so that the kinematics based on simple model analysis may not be very accurate.
b. Machine learning algorithms can be separated into many types. In each branch, there are numerous algorithms that have both advantages and disadvantages. Various data contain distinguishable features, and the most important thing is to locate the most appropriate algorithm. In this thesis, LUNAR software is used. The types of algorithms that LUNAR provides are limited, so the algorithm used only can be selected from LUNAR.

2

Theory

2.1 Finite element method

Finite element method (FEM) is a method to solve differential equations numerically that arise in engineering and mathematical modeling. This method is often used in some traditional engineering fields. After decades of development and refinement, this method has been extended to numerical modeling of physical systems in various engineering areas such as heat conduction, fluid dynamics and electromagnetism [16].

In automotive industry, compared to the full-scale crash test above, FEM has a lot of advantages. The main advantages of FEM include fewer hardware prototypes, faster and lower cost [17].

To evaluate the occupant's safety, generally, surrogates for human occupants like Anthropometric Test Devices (ATD) or dummies are used to represent humans in crash tests. This can also be done virtually using dummy models or Human Body Models in Finite Element (FE) simulations.

Different from dummy models, a Human Body Model (HBM) is more complex. A HBM is more refined and more anthropomorphic that it can be used to study the body's muscle activity, body posture, and reactions. For further, injury mechanisms and injury criteria can then be predicted. Computational human body models help reduce traffic injuries and fatalities and they are widely applied in automotive crash safety research industry [18]. For instance, total human model for safety (THUMS) is a kind of HBMs as shown in figure 2.1 [19].

Active human body model (AHBM) is FE HBM with active muscle control, like SAFER THUMS v9.0 with active muscles [20]. AHBM can more realistically predict the state of humans in a pre-crash scenario, because when humans realize that a crash is about to happen, muscles will contract or expand to control the movement of limbs and other body parts.

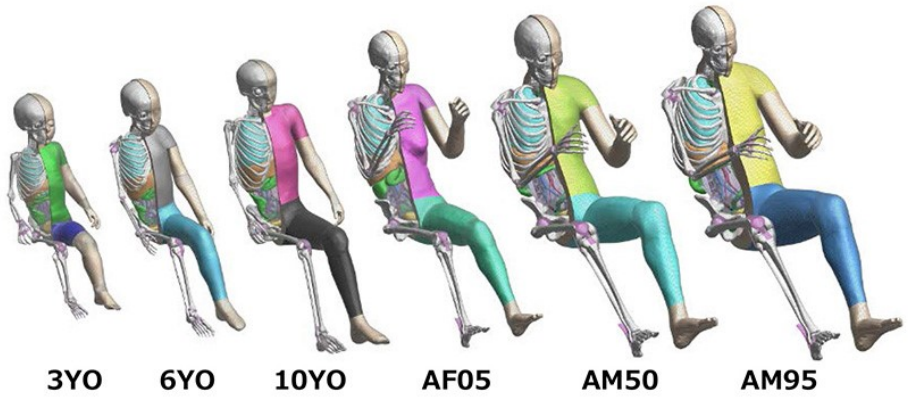


Figure 2.1: THUMS version 4: occupant models [21]

Compared to HBM or AHBM, dummy model seems to be much simpler. And because the dummy model has a simpler structure and fewer elements, the time consumption required for each simulation is much lower. This is a simple and effective method when the research subject does not need to reproduce the real human body in too much detail.

2.2 Hybrid III FAST model

In frontal crash tests, the Hybrid III model is often used for the evaluation of automotive safety restraint systems. It is a kind of anthropomorphic test device, also a dummy model. For instance, Hybrid III 50th Male FE model is the most widely used crash test dummy model all over the world [22]. It is also called Hybrid III M50 model or Hybrid III 50th percentile male crash test dummy since it could represent the average body size of an adult male. It is often used to evaluate vehicle's safety restraint systems in frontal crash tests. It is shown as figure 2.2.

The Hybrid III FAST model, used in this project, is a simplified dummy FE model, which is a semi-deformable FE model, shown in figure 2.3. It has lower element counts than the Hybrid III M50 models used in development work and is designed to be computationally inexpensive while providing a dummy-like behavior in terms of restraint system loading and kinematics during crash simulations. This Hybrid III FAST model can simulate the occupants' kinematics during pre-crash maneuvers and crash scenarios. The displacement, velocity, acceleration of critical nodes on the model, strains of specific elements, seatbelt forces and more data or indicators can be studied. Then some injury risks like head injury criteria (HIC) could be calculated.

Figure 2.2: Hybrid III 50th Male FE [22]

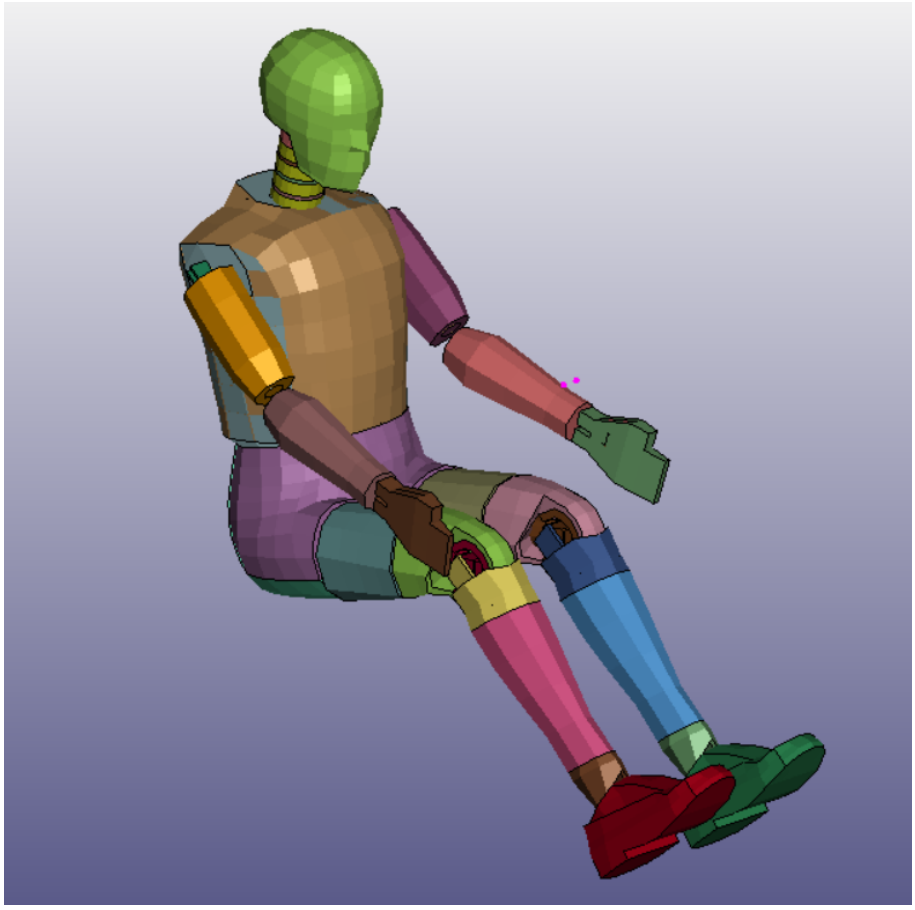


Figure 2.3: Hybrid III FAST model

Although simple dummy models such as the Hybrid III FAST model consume much less time per simulation than HBM, each model still costs dozens of minutes or hours. In this situation, machine learning may provide a way to reduce the amount of FE simulations needed in order to save time.

2.3 Machine learning

2.3.1 Concept

Machine learning is becoming more and more popular in technical fields. It's fundamental for many science fields like data science. Besides, machine learning has also been used in many other fields such as speech recognition, face recognition, autopilot, medical diagnosis, etc [23].
Machine learning is one part of artificial intelligence (AI). The artificial intelligence is to make decisions and predictions for different problems based on huge amount of data. Huge amount of high-quality data is usually necessary for artificial intelligence. The core of artificial intelligence is the machine learning algorithms. Algorithms directly affect how the data is used and to make the computer become more intelligent without additional manually programming. Deep learning is a branch of machine learning. Neural networks are the basic structure of deep learning. Machine learning excluded deep learning is usually called as tradition machine learning. It could be divided into two parts: Supervised learning and Unsupervised learning. Connections of them are shown below in Figure 2.4 [24].

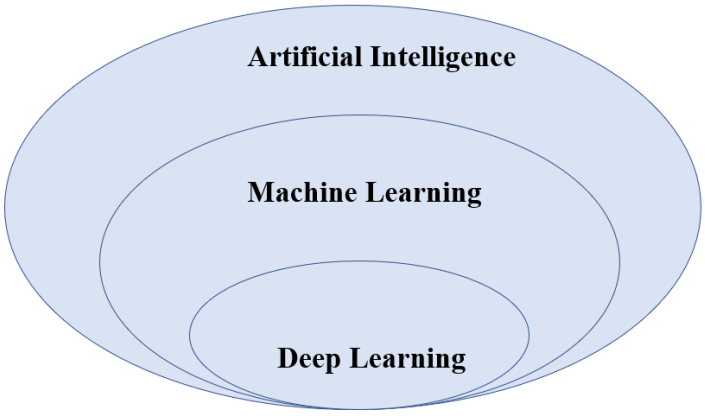


Figure 2.4: Relationships between AI, ML and DL

For supervised learning and unsupervised learning, there are many algorithms which are widely used in this field such as Random Forest, Bootstrapped, Logistic regression, Support Vector Machines (SVM) etc. With the development of machine learning industry, there are many open-source machine learning frameworks are provided by individuals or companies. Engineers can easily construct machine learning

systems easily using these frameworks [25].

2.3.2 Supervised learning

The most important feature of supervised learning is that all the data consists of two parts: input and output. Supervised learning is trying to find a function which can predict output variable(s) based on input variable(s) [26].
Supervised learning is characterized by both input and result. The input data is named as "training data". It's a pattern of (x, y) sample points, where x is the independent variable and y is the dependent variable (result wanted). Each set of training data has a clear identification or result such as "spam" and "non-spam" in the anti-spam system, "1", "2", "3", "4" in handwritten digits classification etc.

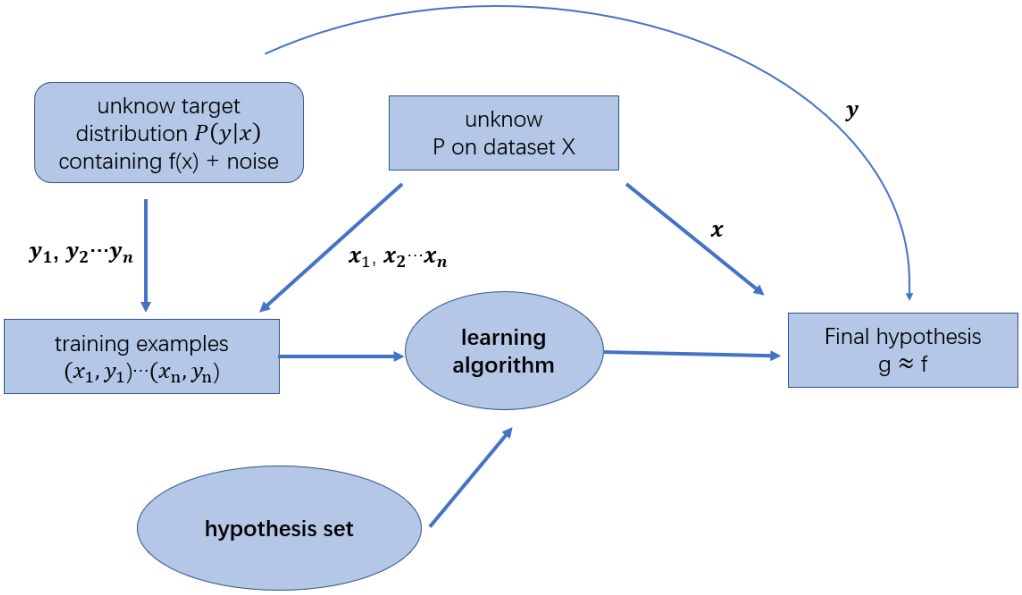


Figure 2.5: The pipeline of supervised learning

A model is obtained through the learning process and a functional relationship between y and x is obtained, or a conditional probability model which tells the probability of y occurring under the premise of x. The model is built by learning from the training data, comparing the difference between prediction and the true data. To minimize the difference between prediction and true data, the model should be able to adjust itself to reach a satisfying accuracy [27]. To use supervised learning, data y is necessary. The most common application of it is regression and classification. Common algorithms include Decision Tree, Support Vector Machine, etc. Pipeline of supervised learning is shown as figure 2.5.

2.3.3 Unsupervised learning

Unlike to dataset of supervised learning, there are many cases when data is not labeled. It cost much to do labeling manually when the dataset has a big size.

Alternative is to use computer do the label step instead of humans. This is called unsupervised learning, which can predict targets using un-labeled training data and solve many problems. One important feature of unsupervised learning is that all samples are used as input. There is no output of the original data which is different from supervised learning [28]. There are two commonly used scenarios for unsupervised learning: clustering and dimensionality reduction.

2.3.3.1 Clustering

In its basic form the clustering problem is defined as the problem of finding groups of data points in a given dataset. They are also called as clusters. It's desired that samples belonging to the same cluster are more similar than other samples [29]. K-means is one of the most common clustering algorithms. It's shown as figure 2.6.

K-means needs to specify the parameter k, which is used to divide n data into k clusters. The k clusters obtained after division need to have a high degree of similarity. Goal of this algorithm is that all samples are divided into k clusters according to a certain standard. This algorithm is processed in a continuous cycle. The algorithm steps are:

Step 1: Select k(the number set before) samples as centers of k clusters. $a = a_1, a_2, a_3, \ldots a_k$. This selection can be done by different methods. The parameter k can affect much of the final performance. Choosing a proper value of parameter k is essential for K-means algorithm.
Step 2: Go through all samples except sample which are set as centers. For each sample x_i, classify it into the closest one of k clusters based on the distance from x_i to cluster centers a. How the distance is calculated can be designed for different problems.
Step 3: Recalculate the cluster center of each cluster c_k, the new cluster center should be the same as the centroid of all samples within this cluster. New cluster centers $a_j = \frac{1}{|c_k|} \sum_{x \in c_k} x$.
Step 4: Repeat previous step 2 and step 3, until obtaining a specific condition. The condition is usually set by the programmer, numbers of literation of step 2 and 3 or one threshold of error change are usually used.

Advantages of k-means: Principle of k-means is simple. It can reach an acceptable local optimum in most cases. It's not sensitive to the size of datasets, usually the scalability would improve with bigger datasets. The algorithm is not too complex.

Disadvantages of k-means: The number of clusters need to be set manually first. It's difficult to find out a proper number without prior knowledge. Because k-means reaches to local optimum, the initial selection of cluster centers can bring big effect to the final performance, different selecting methods can lead to huge differences. Sensitive to outliers. Works bad for imbalanced datasets.

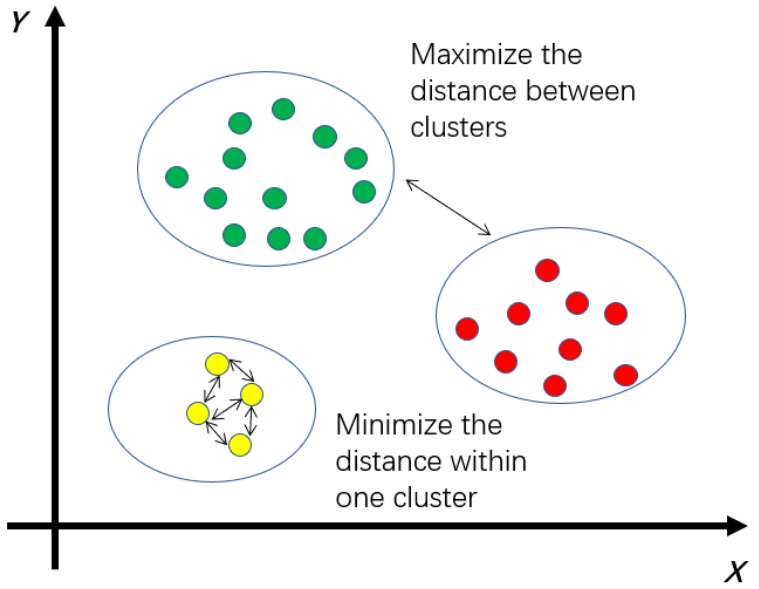


Figure 2.6: K-means example

2.3.3.2 Dimensionality reduction

Number of independent variables could be over 10 or even 100 in some cases. They could provide with rich information but requires more effort in data preparation. Also, such huge number of variables brings high complexity, leads to huge computing cost and analyzing difficulties in finding connections between independent and dependent variables undoubtedly. Connections between independents variables are also not wanted for analysis. If variables are analyzed separately, most information will be lost. Reducing variables blindly would lose much useful information, leads to errors. So, how to reduce the number of variables is vital for data preparation. It should be able to minimize the number of variables and remain the most important information at the same time. This process is called dimensionality reduction. It reserves vital information and remove useless data. In practice, it's designed to reduce much time and computing cost with acceptable data loss.

Proper orthogonal decomposition(POD), also known as PCA or Karhunen-Loeve expansion, is a classical feature extraction and data representation algorithm which is commonly applied in many physical fields[30].

POD algorithm steps:

Input: n dimensional sample set X=$(x_1, x_2, ..., x_n)$,dimension number to reduce to: $n^{'}$. Number of samples is m.

Output: The sample set Y after dimensionality reduction.

Step1: Do center process to all samples. $x_{ni} = x_{ni} - \frac{1}{m}\sum_{mi=1}^{m} x_{mi}$

Step2: Calculate the covariance matrix of the samples. $C = \frac{1}{m}XX^T$

Step3: Calculate the eigenvalues and corresponding eigenvectors of the covariance

matrix.
Step4: According to the size of the corresponding eigenvalue, the eigenvectors are arranged into a matrix from top to bottom. And the first k rows should be used to build matrix P.
Step5: Y=PX is the data with $n^{'}$ dimension after decreasing dimensionality.

2.3.4 Dataset split

Dataset are the basis of AI, and it's usually separated to achieve the goal of model building, training and testing as shown in figure 2.7. Generally, there are three types of data sets:

Training set: The training set is used to train the machine learning model. Get the model's parameters.

Validation set: It is used to test the performance of the model (this is the same as the test set), but the model parameters can be adjusted according to the test results, mainly the hyperparameters (this is different from the test set). The validation set is a middleware between the training set and the test set. If there is no validation set, there will be a problem: With only a training set and a test set, it is impossible to evaluate the model before using the test set. However, it's preferred to evaluate the model before the final model is determined. Then adjust the model parameters (especially the hyperparameters) according to the evaluation results until the model is considered as good enough. Lastly use the test set to test which is only to see the approximate generalization ability of the model. The test set should never be used to update the model parameters. It is for this purpose that a validation set is partitioned. The validation set is still a training set to a certain extent, because the model's parameters are changed according to the results of the validation set.

Test set: It is used to evaluate the generalization ability of the final model that has been trained but cannot be used for model parameter tuning. For a trained model, it is necessary to know its generalization ability (accuracy on new samples). The real generalization ability should be calculated on all potential samples except the original sample set. But it is impossible to get all potential samples in practice. So, the alternative is to divide a test set as an approximation from the original sample set. Use this set to approximate the generalization ability as the generalization ability on all samples. There shouldn't be any intersection between the training set and the validation set. The test set should only be used to test the generalization ability of the final model and shouldn't be used to update the model parameters. In other words, only the final determined model should use the test set for evaluation. The model should not be modified once it's fed with the data of test set. If the accuracy (that is, the error) obtained on the test set is used as the basis for further training of the model, the test set is treated as a training set rather than a test set anymore.

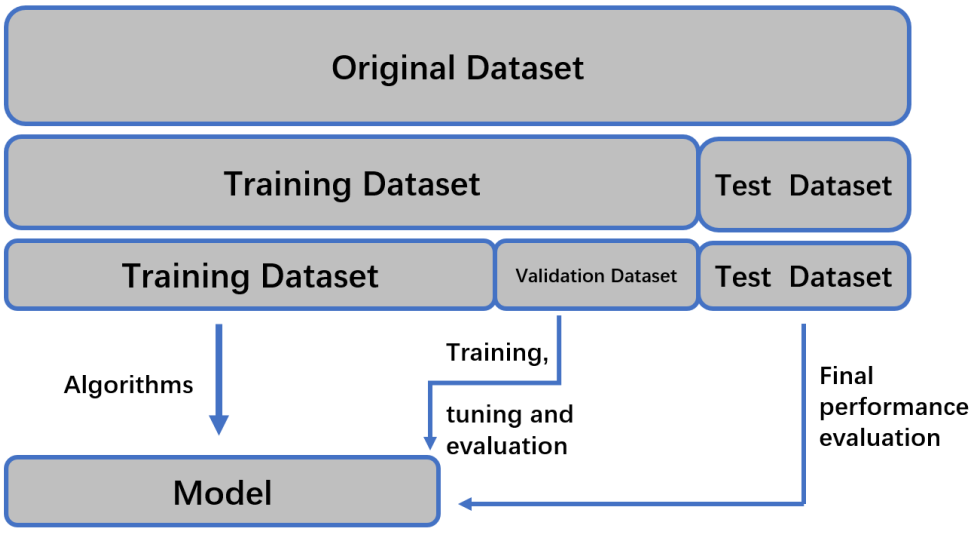


Figure 2.7: Data split for ML

Generally, there are three ways to divide the original data set: hold-out method, cross-validation method and bootstrap method. Here hold-out method is used. In this way, the dataset D is directly divided into three mutually exclusive sets, one of which is used as training set S, one as validation set V, and one as test set T. T, V and S satisfy: $D = S \cup T \cup V, S \cap T \cap V = 0$. After training the model with S, set V is used for the parameter selection configuration of the model. Lastly use set T to evaluate performance as an evaluation of the generalizing ability of the model. Dataset T is also called held-out data. Issues to be aware of using hold-out method: The partitioning of the training/validation/test set should be as consistent as possible with the data distribution to avoid the impact on the result, because of the extra bias made by the data dividing process.
After the sample ratio of the training/validation/testing set is given, there are still many ways to divide the initial data set D, which may affect the results of model evaluation. Therefore, the results obtained by a single use of the hold-out method are often not stable and reliable. When using the hold-out method, several random divisions and repeated experimental evaluations are generally used to obtain the average value as the evaluation result.
It is desirable to evaluate the performance of the model trained with dataset D, but the hold-out method needs to divide D into training/ validation/test sets, which leads to a dilemma: if the training set S contains the majority of samples, the trained model may be similar to the model trained with D. Because dataset T is relatively small, the evaluation results may not be stable and accurate; If the test set T contains more samples, the difference between the training set S and dataset D is big then the model is different from the model trained with D. There may be a big error compared with the model trained with dataset D which reduces the fidelity of the evaluation results. Therefore, it is common to use about 2/3 4/5 of the samples for training and the remaining samples as testing.
Defects of hold-out method: In the process of dividing a dataset into training set, validation set and test set: even if the numbers of samples divided into each subset

are the same at each time, the final division results may differ a lot since all samples are different from each other. In some cases when the dataset is small, it's highly possible that the final performance of the model can be very different because of different subsets even with the exact same hyperparameters. It highly possible that in the case of a small dataset, the final training and testing results will fluctuate greatly for different subsets. It's because the obtained optimal hyperparameters are different because training set and the validation set are divided differently.

For each dataset in this work:

X-data: 100 sets of varying parameters of FE models will be split into three parts: training part, validation part and testing part. It will be done by DOE method introduced in coming sections.
Y-data: There are 100 curves corresponding to 100 FE-simulations. They are split into training dataset with 60 curves, validation dataset with 20 curves and test dataset with 20 curves. This step is carried with a function in Python script.

2.3.5 Evaluation of regression model

Regression model is a predictive modeling technique to study the relationship between dependent variable (target variable) and independent variable (predictor variable). It is commonly used in predictive analysis, time series modeling, and discovering causal relationships between variables. There are several metrics widely used for model's performance evaluation [31], shown in table 2.1:

Mean Absolute Error (MAE): It's the average of the absolute values of the deviations of all label values from the regression model predictions. It has an advantage that it can intuitively reflects the deviation between the predicted result of the regression model and the actual result. It can reflect the size of the actual forecast error accurately. Meanwhile, there is no positive and negative cancellation caused by different error signs.

Mean Absolute Percentage Error (MAPE): It's an improvement of MAE that takes the proportion of absolute error relative to the true value into consideration. The advantage is that the error value and ratio between the predicted value and the actual obtained value are both considered.

Mean Square Error (MSE): Although MAE can measure the performance of the regression model well, the existence of the absolute value makes the MAE function not smooth and cannot be derived at some points. Changing the absolute value in MAE to the square of the residual to avoid such problems and calculate the MSE. It can zoom in the values with large prediction bias. It can be useful comparing the stability of different forecasting models.

Root Mean Square Error (RMSE): It's a square root operation based on the MSE [32].

Table 2.1: Metrics used for model's performance evaluation.

Metric	Formula
MAE	$\frac{1}{n}\sum_{i=1}^{n}\lvert y_i - \hat{y}_i\rvert$
MAPE	$\frac{100\%}{n}\sum_{i=1}^{n}\frac{\lvert y_i - \hat{y}_i\rvert}{y_i}$
MSE	$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
RMSE	$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$

2.4 LUNAR software

2.4.1 Introduction of LUNAR

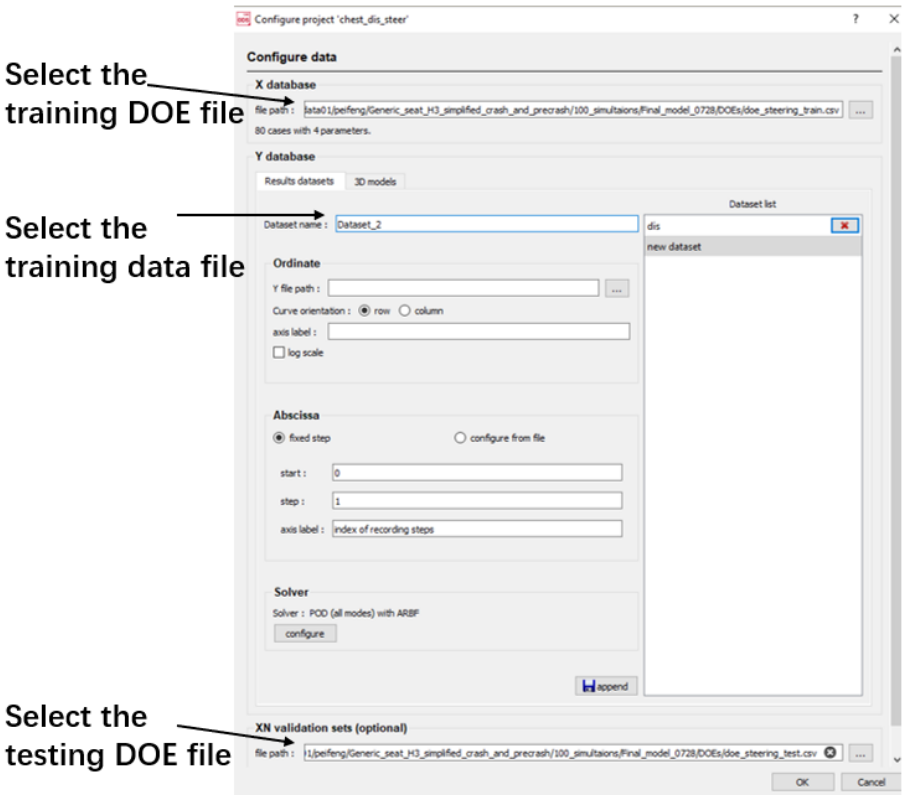


Figure 2.8: Interface of LUNAR for importing data

LUNAR processes data to investigate the relation between independent variables and dependent target (usually curves). LUNAR will make an initial prediction first and then use its tool, reduced-order-modeling (ROM) simulation, to improve the predicting accuracy. There are many different algorithms provided by LUNAR

within ROM. Scenarios where these algorithms are best suited are real-time prediction problems.

2.4.2 LUNAR algorithms

A big advantage of LUNAR comparing to other AI frameworks is that LUNAR is a fully packaged software which can provide users a convenient operation interface. With proper data prepared according to its requirements, importing the data is done by selecting right files in Lunar's interface, shown in figure 2.8.
The Lunar provides many algorithms to creative a machine learning models based on different needs. Some of them are shown in the figure 2.9. This figure is copied from LUNAR official help notes.

METHODS	SOLVER	Type of use
Interpolation methods (direct interpolation)	• Kriging	• Time response • Scalar response
	• RBF • ARBF (adaptive RBF)	• Time response • Scalar response
	• InvD	• Time response • Scalar response
Reduced Order Modelling (ROM) method (decomposition, reduction, reconstruction by interpolation)	• POD Rbf • POD Krigging • POD ARBF (default) • POD InvD	• Animation • Time response • Scalar response
	• Clustering Rbf • Clustering Krigging • Clustering ARBF	• Images • Animations • Quick for big data (>1e+6)
	• FFT Rbf • FFT Krigging • FFT ARBF • FFT InvD	• Periodic time response
Clustering Method (classification method)	• SVM	• Time response • Scalar response

Figure 2.9: Methods and solvers provided by LUNAR

With so many algorithms provided, which to choose for each dataset is a problem. The selection should depend on the size of dataset, CPU performance and the required accuracy. In this work, accuracy and time-cost are considered as the two most important features. However, the time cost is always very small compared to the time of FE-simulations regardless which algorithm is selected. Accuracy is the only factor which affect the algorithm selection then.

2.4.3 ROM

Reduced order model is a simplification of complex algorithms which investigate the relations between independent variables and target curves. ROM only require proper datasets to learn and predict. There is no other prior requirements. ROM consists of three parts:
Decomposition: Split the system up in multiple simpler systems.
Reduction: Compress/reduce the volume of data (filtering noise).
Reconstruct: Build the system back up again after solving the "simple" systems.
Lunar provides many different ROM methods: POD Rbf, FFT ARBF, POD Kriging, etc.

2.4.4 User script

User script is one function of LUNAR which provide users with a way to run Quasar scripts easily through interface. There are already more than 10 scripts prepared by LUNAR which achieve different goals. Users can change values of arguments of the script through the corresponding interface. It's shown as figure 2.10.

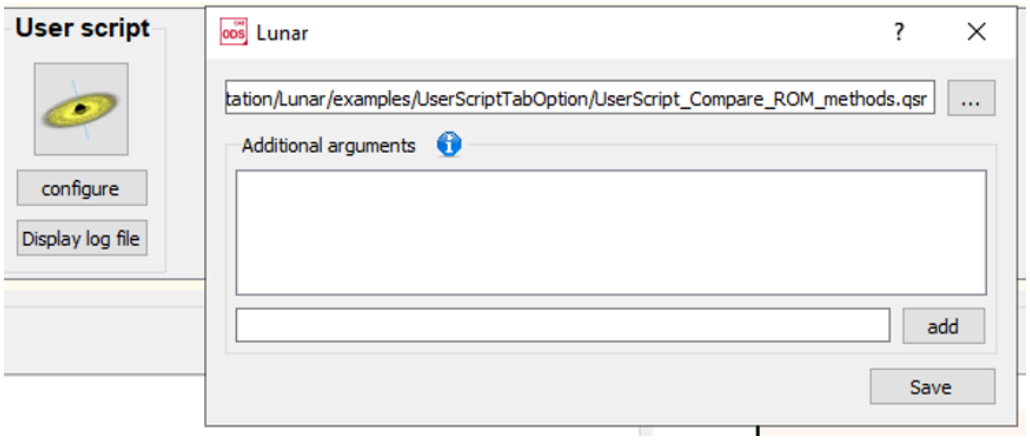


Figure 2.10: Interface of using script in LUNAR

Compare-ROM-Methods is one user script which can help users find out the best algorithm and parameters set for each dataset. This script automatically tests many prediction methods (POD, Direct) on X-Validation points loaded by the configuration tab (not the sensitivity tab) and compares the predicted results with the imported curves. After the script run, there will be one log file generated. It will propose the best method and parameter set in the log file.

2.4.5 Method of DOE

Design of experiment (DOE) is to design space for every parameter interested. In this project a method called the optimal Latin hypercube technique is used to generate DOE matrix for varying parameters.
Below figure 2.11 shows three methods of DOE. Assume that they have the same number of varying parameters, p (in this example figure it is 9). And these parameters are going to distributed in the same limited space. The first left figure gives a standard way, in this way, these points are distributed at only three levels. It is very simple so that it is easy to fit the data by using a quadratic model. But it is uncertain that if the real model is more nonlinear. The middle figure shows a method called random Latin hypercube technique. It can be seen that these points are randomly distributed in the design space. Compared to the previous one, it has more levels. But the disadvantage is that the distribution is uneven. The right figure shows the optimal Latin hypercube technique. The distributed 9 points represent 9 levels of every factor. And they are uniformly distributed within the limited space. Compared to the above two methods, this method may provide the best opportunity to fit real model with proper function.

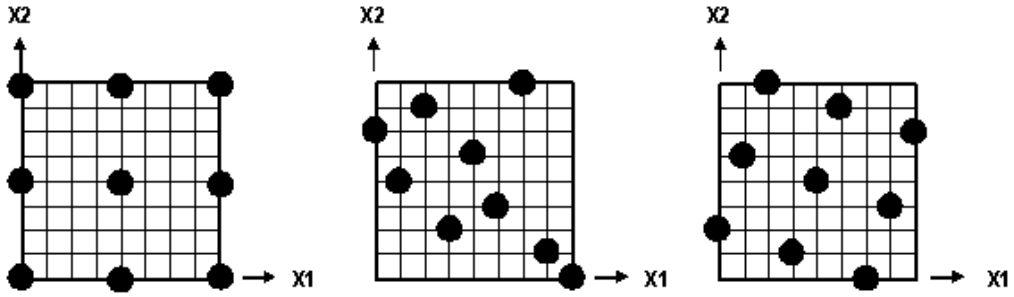


Figure 2.11: Three kinds of distribution

If a design maximizes the minimum inter-site distance, the design is named as maximum distance design. It can be calculated by,

$$\min_{1\leq i,j\leq n, i\neq j} d(z_i, z_j) \tag{2.1}$$

$$d(z_i, z_j) = d_{ij} = [\sum_{k=1}^{m} (|z_{ik} - z_{jk}|)^t]^{\frac{1}{t}} \tag{2.2}$$

t can be 1 or 2, and $d(x_i, x_j)$ represents the distance between x_i and x_j (2 sample points).
The minimum number of test points m_{min} could be calculated from the number of given parameters p,

$$m_{min} = 2p + 1 \tag{2.3}$$

3

Methodology

3.1 Workflow

The workflow can be summarized as figure 3.1. The process of FE simulation and ML prediction will be described in detail in the next chapters.

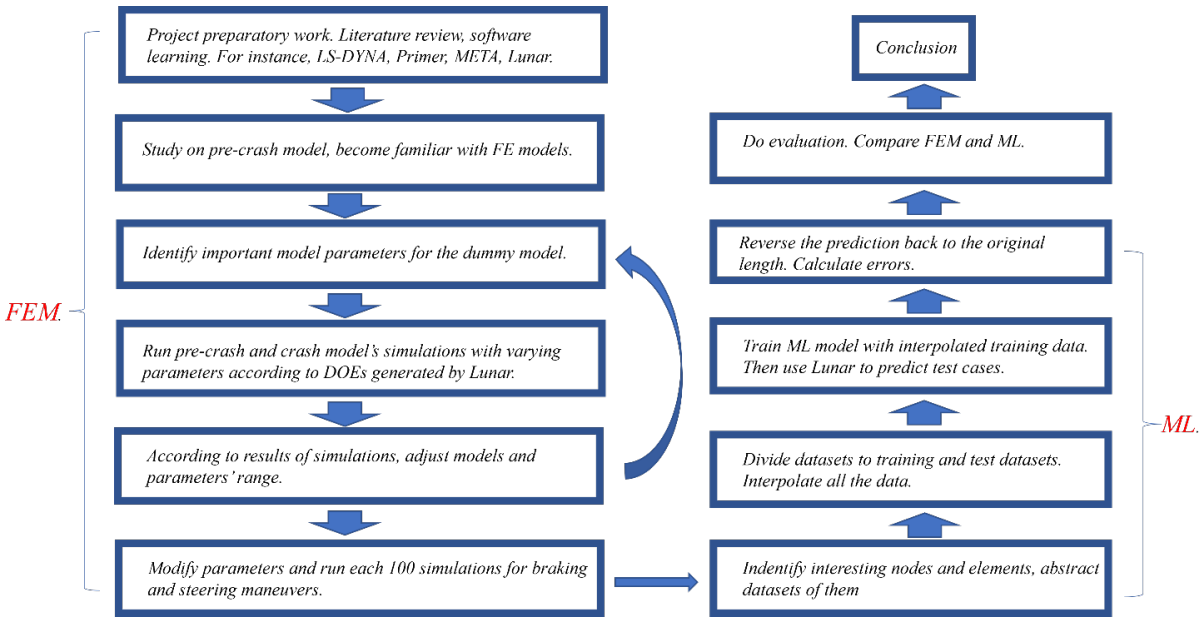


Figure 3.1: Workflow

3.2 FE model preparation

The Hybrid III FAST model was used as the occupant model, representing an average male. It was positioned in the driver position in a generic vehicle environment model. The vehicle model consists of generic models of seatbelt, steering wheel, seat, door inner panel, instrument panel, center console and footrests. As shown in figure 3.2.

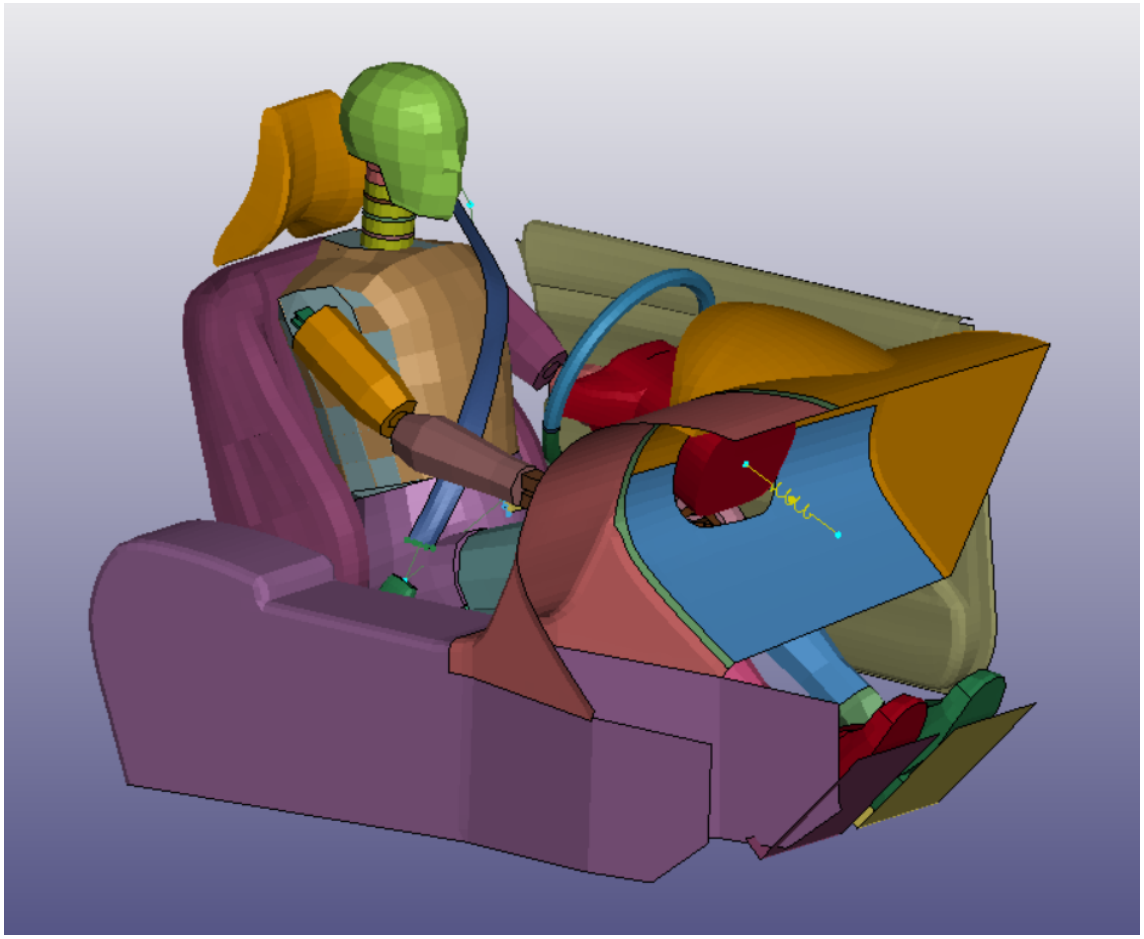


Figure 3.2: Pre-crash and crash model

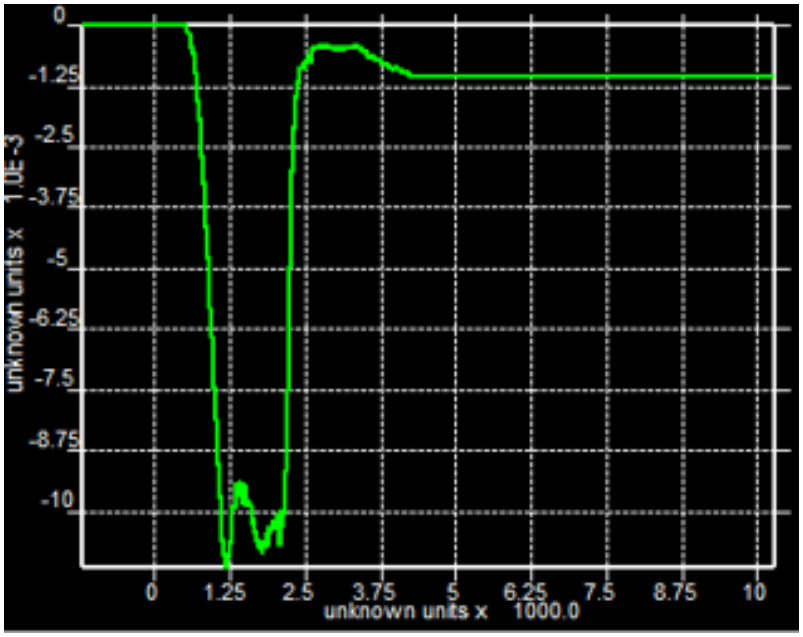


Figure 3.3: Braking pulse curve with braking level 1 g (Y axis label: acceleration $[mm/ms^2]$, X axis label: time [×1000 ms])

This model is used in combined pre-crash and crash simulations. In one example case, the pre-crash phase consists of a braking pulse with a peak of about 1 g and a steering pulse with a peak of about 0.01 g as shown in Figures 3.3 and 3.4 below. In figure 3.3, the acceleration stays at 0 until 200 ms. The first 200 ms corresponds to 300 ms to 500 ms in timeline in figure 3.7. This period is left for activation of pre-pretensioner force. It is the same for steering pulses. Then the acceleration drops to 1 g and oscillates around it. The acceleration stays around 1 g for approximately 1000 ms and then returns to a lower value and gradually stabilizes.

Almost the same with braking pulse, the difference between steering and braking pulses is that the acceleration of steering is on Y axis which is perpendicular to braking acceleration, and the acceleration is first positive and then negative with same time and absolute value.

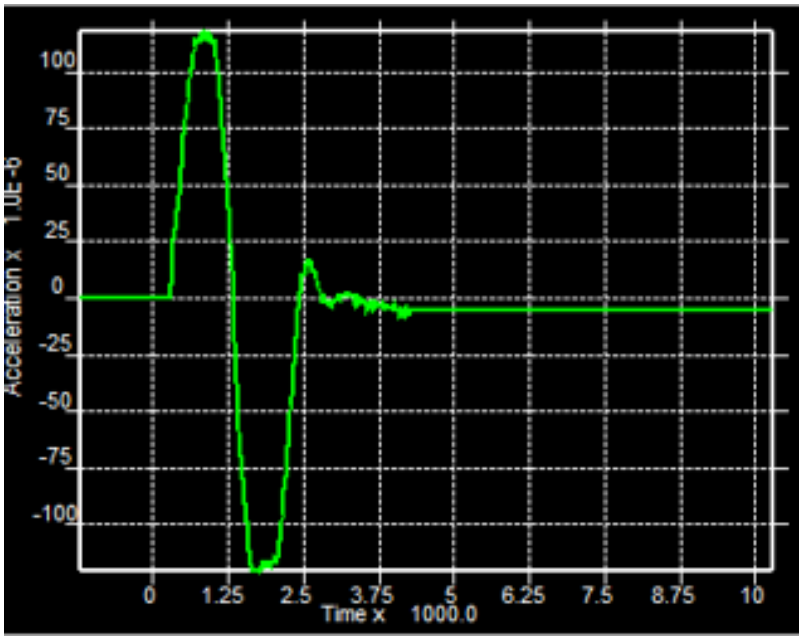


Figure 3.4: Steering pulse curve with steering level 0.01 g (Y axis label: acceleration $[mm/ms^2]$, X axis label: time [×1000 ms])

As shown in timeline in figure 3.7, from 350th ms to 500th ms, this pre-pretensioner force should be activated at a specific time. The time depends on the parameter called pre-pretensioner force's activation time before avoidance. If this parameter is –150 ms then the activation time is 500 minus 150 equals 350 ms. In figure 3.5, after activation, the belt force will gradually increase over a certain period of time and finally reach the desired value, and in this case it is 500 N.

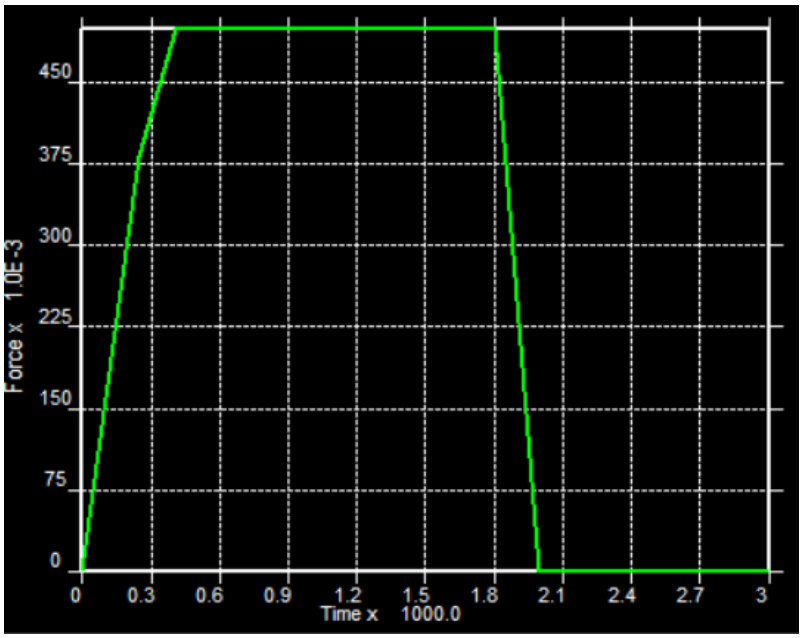


Figure 3.5: Belt force (Y axis label: belt force [kN], X axis label: time [×1000 ms])

The crash phase consists of a frontal crash pulse having a peak of about 27 g as shown in figure 3.6 below. Then, the pre-crash braking or steering starts followed by frontal crash at a delta V of 60 km/h for 150 ms.

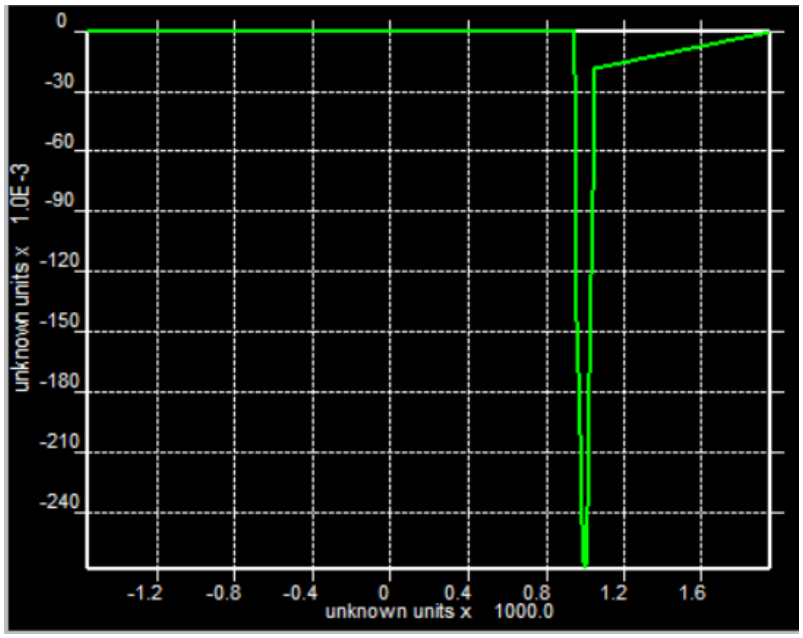


Figure 3.6: Crash pulse (Y axis label: acceleration [mm/ms^2], X axis label: time [×1000 ms])

In this timeline, shown in below figure 3.7, it can be seen that the total simulation time also included an initial 300 ms of gravity loading at the beginning of the simulation. After the gravity loading, from 300th milliseconds to 300+T0 milliseconds,

this period T0 is called the duration of avoidance measures. It could be mainly steering or braking. And from 350th milliseconds to 500th milliseconds, pre-pretensioner force should be activated from a specific time point, which is called activation time of pre-pretensioner force.

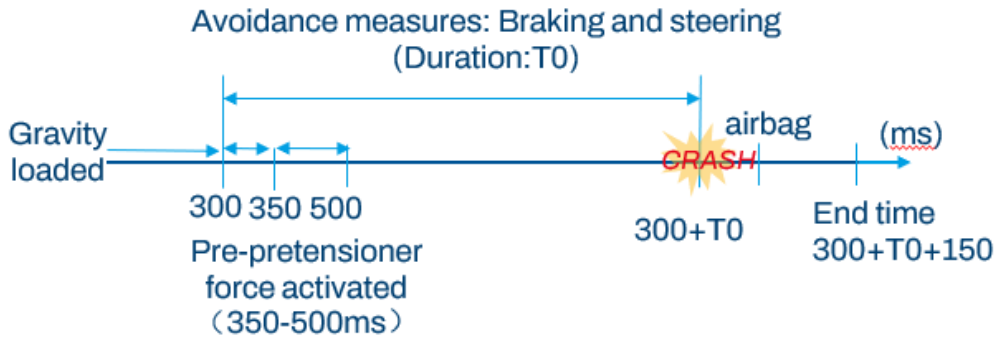


Figure 3.7: Timeline of the entire simulation

Then pre-pretensioner will be activated to provide force to tighten seatbelts. At the same moment when the pre-pretensioner force is activated, avoidance measures are going to be taken. 10 ms after the crash, the airbag will be triggered. Then simulation will end after 140 ms.

3.3 Simulations of FE models

The FE model used is the pre-crash and crash model which contains the Hybrid III FAST model and the vehicle's several inner parts introduced in above chapters. From actual tests and several FE simulation tests with different extreme values, the proper parameters' range list is obtained as table 3.1.

3.3.1 Parameter space

The parameter space for the simulations is divided into two parts and each part has 100 simulations. In next chapters, one part is called braking simulations, the other is steering simulations. It means that in braking simulations, the varying parameters are pre-pretensioner force, pre-pretensioner activation time before avoidance, duration of braking and braking level. When it comes to steering simulations, the varying parameters are pre-pretensioner force, pre-pretensioner activation time before avoidance, duration of steering and steering level. 100 simulations each were run for braking+crash and steering+crash simulations.

Table 3.1: Range list of varying parameters.

Number	Parameter name	Minimum value	Maximum value
1	Pre-pretensioner force [N]	0	500
2	Pre-pretensioner force's activation time before avoidance [ms]	-150	0
3	Duration of avoidance (braking or steering) [ms]	200	1500
4	Braking level [g]	0	1.1
5	Steering level [g]	0	0.7

3.3.2 DOE

According to this range list , above table 3.1, 100-DOE could be generated from Lunar. In this project the Optimal Latin Hypercube technique introduced in previous section was chosen to generate the DOEs.

For each 100 simulations, there are 4 parameters to vary. When n is 4, the minimum amount of test points for DOE should be 9.

So 100 simulations are absolutely enough. Generated DOEs is attached in Appendix 1, 100 DOE matrix for braking+crash simulations and steering+crash simulations.

These parameters are modified directly in key files of this pre-crash and crash model, shown in figure 3.8. For forces and braking and steering levels, scaling factors are set to change the value. PPTFRC is a scaling factor for pre-pretensoner force. BREAKG and STEERG are Scaling factors for braking level and steering level.

CRASHNI is to adjust the duration time for steering and braking. That is T0 in timeline. PRET_T is the activation time of pre-pretensioner force.

A Python script was written (Included in the Appendix 2) which updates these parameters in key files of 200 models for braking and steering automatically. All simulations were run in the LS-Dyna solver (version mpp971_s_R9.3.1_139027) in high-performance computers (HPC).

```
*KEYWORD
*PARAMETER
$ Scaling factor for ppt force
R PPTFRC     0.6833
$ Start of breaking (applied after settling)
R BREAKINI       0.
$ Start of breaking (applied after settling)
R STEERINI       0.
$ Breaking acceleration in g's
R BREAKG       0.01
$ Steering acceleration in g's
R STEERG     0.1864
$ Delta velocity during crash
R DV             65.
$ Duraton (ms) for dummy settling (precrash applied after this time)
R SETTLING      300.
$ T0 for crash (ms)
R CRASHINI   423.232
$ Relative trigg time for airbag (measured from T0) in ms
...
$ Lock time belt system
R RET_LOCK       640.
R LAP_LOCK       640.
R PRET_T    481.8182
$
```

Figure 3.8: Modifying parameters in key files

3.4 Machine learning prediction

Data is transferred between FE-simulations (binout files), LUNAR and Python script. Figure 3.9 shows the overall pipeline of using machine learning in this thesis.

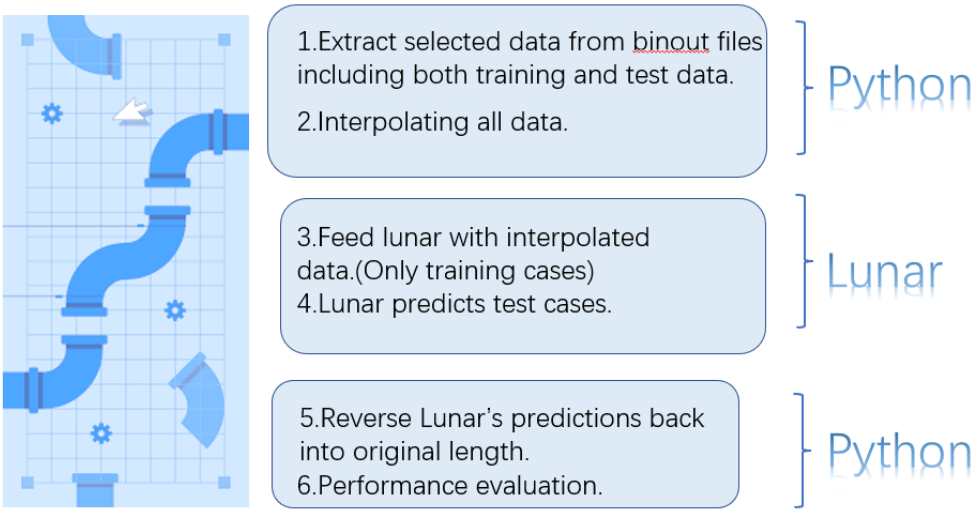


Figure 3.9: Pipeline of using machine learning

3.4.1 Collection and preparation for the dataset

The quality of the data can directly determine the effect of the final machine learning model. A good data set can provide many benefits in later machine learning procedures. The first step in preparing the data is to exclude error data. Otherwise, it's very hard to correct it later because it's hidden in the overall data. Therefore, all simulations are inspected to ensure they are conducted correctly before further machine learning steps. There are 100 simulations of braking scenarios and another 100 for steering scenarios. Information about if the simulation is run successfully is stored in the 'run.key.info' file. One Python script is created to do this work: read the 'run.key.info' file corresponding to each simulation and print out which simulation fails. The script tells that there is only one braking simulation is failed for some unknown reason. Data of 199 simulations left is used in later work.

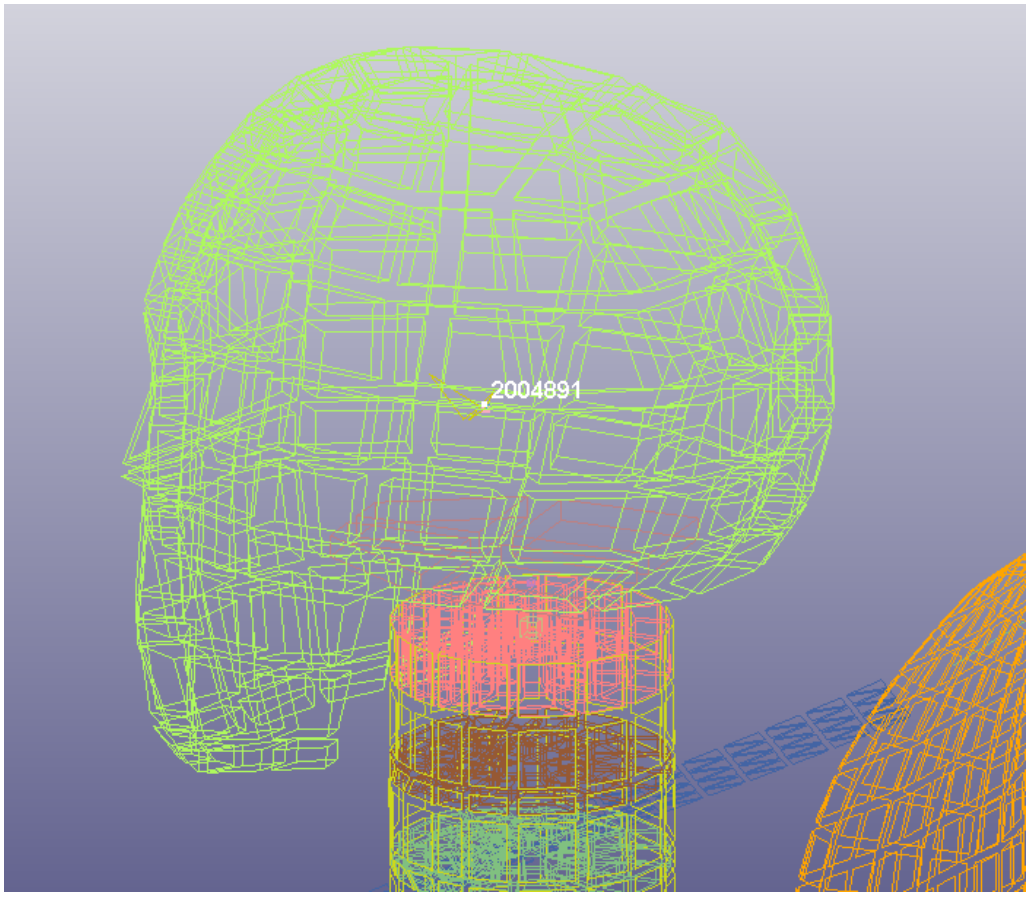


Figure 3.10: Selected head node in this model

To investigate head's kinematics, the center node of the head, node 2004891 as shown in figure 3.10, is selected as investigating target. Two features of this node are focused: displacement and acceleration of this node. So, there are four datasets in this part of work: Node's displacement data of braking and steering simulations, node's acceleration data of braking and steering simulations. Because when the braking level changes, the steering level remains 0.01 g. The acceleration is mainly on the X axis, and the acceleration on the Y axis is very small and can be ignored.

In braking simulations, both displacement and acceleration are data in the X-axis direction. In the steering simulations, the steering level is varied between 0 and 0.7g, so both the acceleration and displacement on the X and Y axes are considered.
Head node acceleration and displacement curves for data of braking simulation, data in the x direction is only considered. The reason is that both the car and occupant barely move in y-axis direction during the crash. There is minimal steering in the whole process.
For data of steering simulation, data of both x-axis direction and y-axis direction is considered. The reason is that the occupant would move in y-axis direction during steer before the crash and in x-axis direction during the crash. So, for both acceleration and displacement, there are three set of curves are predicted: curve in x-axis direction, y-axis direction and total curve.
The maximum acceleration and displacement are also vital for predicting head node's kinematics. In addition to predicting time history curves, peak values of them are also predicted.
Shoulder belt forces from all simulations are also extracted.
To extract the strain data, for every simulation, a separate d3part file is generated to store the strain data for the rib parts exclusively, shown in figure 3.11. Then the maximum strain curve of each simulation is selected and stored from the d3part files manually. There are some cases where one curve is significantly higher than others which are considered noisy data here. In order to make the data accurate and representative, the significantly higher curve is ignored, and the second highest curve is selected as the maximum curve. Shown in figure 3.12.

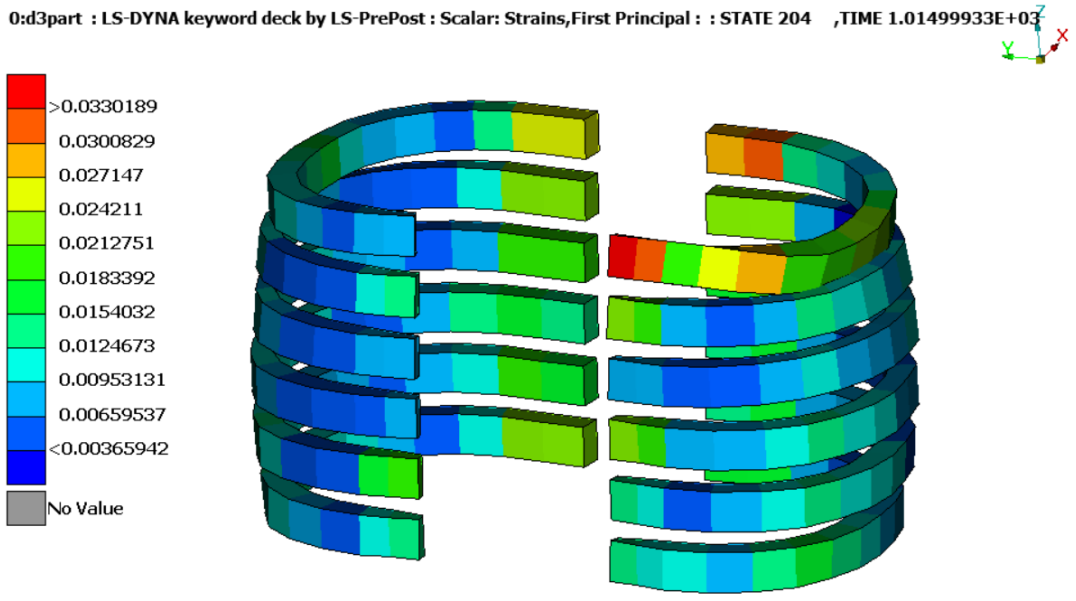


Figure 3.11: Strain for ribs in d3part file on META

To use LUNAR, all data should be transformed into a specific format in CSV files. One feature of the required format is that all curves should have the same length.

Lengths of original data from Ls-Dyna are different because of different durations of simulations. Curve interpolation is used to satisfy this requirement.

Cubic spline interpolation method was used to transform all the simulation data. This interpolation step was done in a Python script (included in the Appendix 3) was used to read all the curves to find out the one with longest length. Then this length was stored as a private variable in the class which was defined in the Python script. Using the package from Scipy (an open-source Python algorithm library and math toolkit.), all curves left were interpolated to the same length as the longest curve, shown in figure 3.13 and 3.14.

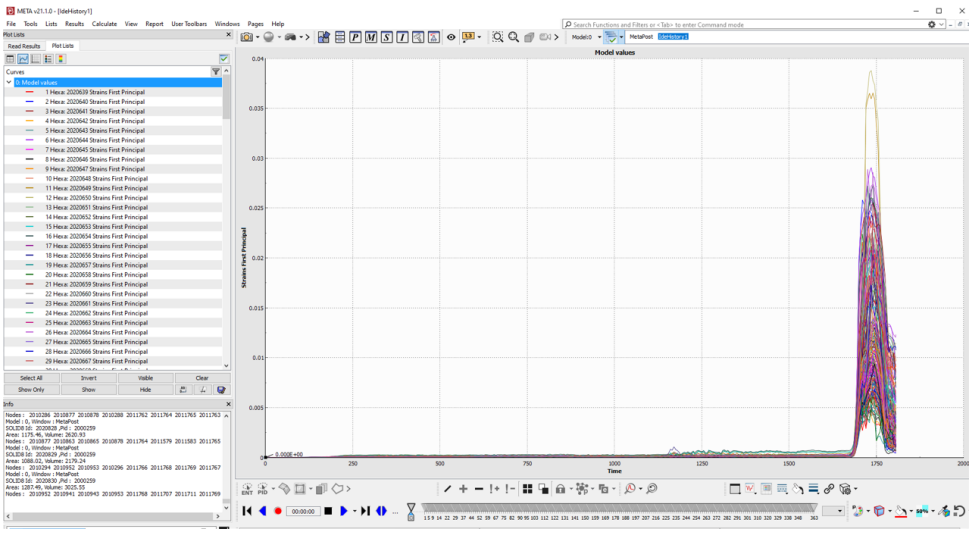


Figure 3.12: One example of strain curves in META

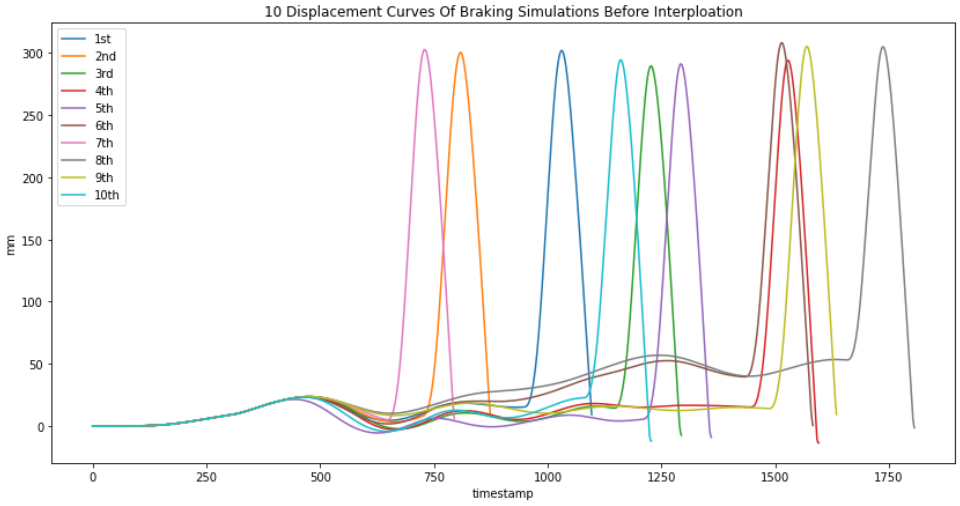


Figure 3.13: 10 original head node's displacement curves from braking simulations

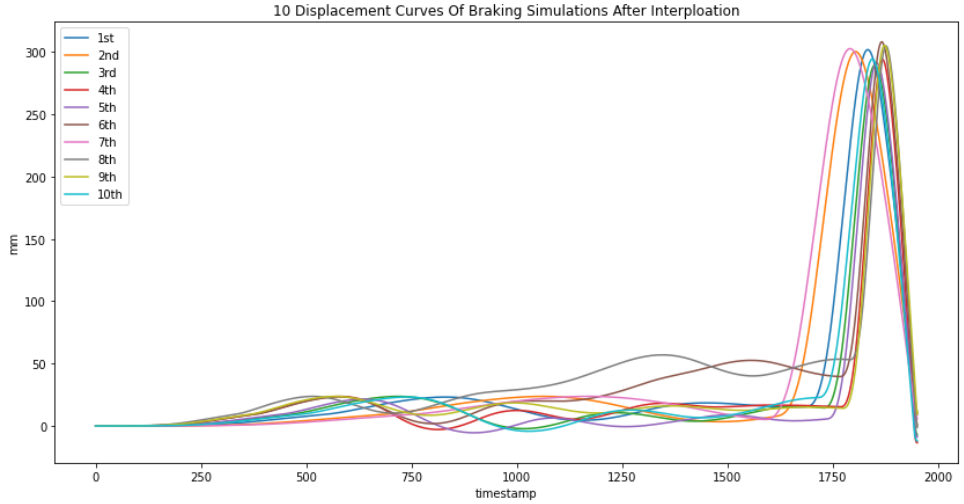


Figure 3.14: 10 interpolated head node's displacement curves of braking simulations

3.4.2 Training and selecting parameters

```
-------------------------------------------------------------------------------------------------
For the dataset #1 named Dataset_1, the best method for all validation cases
with a CRITERION CoefficientOfDetermination_R2 (18) is pod_rbf_gaussian_r=0.25_Modes=40 (66)
with a AVERAGE score of 5.67326
-------------------------------------------------------------------------------------------------

pod_rbf_gaussian_r=0.25_Modes=40 for validation case #1 : 8.40351
pod_rbf_gaussian_r=0.25_Modes=40 for validation case #2 : -1.18831
pod_rbf_gaussian_r=0.25_Modes=40 for validation case #3 : 5.88097
pod_rbf_gaussian_r=0.25_Modes=40 for validation case #4 : -37.1213
pod_rbf_gaussian_r=0.25_Modes=40 for validation case #5 : 43.9421
pod_rbf_gaussian_r=0.25_Modes=40 for validation case #6 : 7.2132
pod_rbf_gaussian_r=0.25_Modes=40 for validation case #7 : 61.9342
pod_rbf_gaussian_r=0.25_Modes=40 for validation case #8 : 36.3048
pod_rbf_gaussian_r=0.25_Modes=40 for validation case #9 : 67.6769
pod_rbf_gaussian_r=0.25_Modes=40 for validation case #10 : 9.59459
pod_rbf_gaussian_r=0.25_Modes=40 for validation case #11 : 9.78178
pod_rbf_gaussian_r=0.25_Modes=40 for validation case #12 : 11.5789
pod_rbf_gaussian_r=0.25_Modes=40 for validation case #13 : -25.7874
pod_rbf_gaussian_r=0.25_Modes=40 for validation case #14 : 33.5976
pod_rbf_gaussian_r=0.25_Modes=40 for validation case #15 : -247.964
pod_rbf_gaussian_r=0.25_Modes=40 for validation case #16 : 25.5217
pod_rbf_gaussian_r=0.25_Modes=40 for validation case #17 : -20.7279
pod_rbf_gaussian_r=0.25_Modes=40 for validation case #18 : 57.1049
pod_rbf_gaussian_r=0.25_Modes=40 for validation case #19 : -2.98336
pod_rbf_gaussian_r=0.25_Modes=40 for validation case #20 : 70.7019

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 The Compare ROM methods script finished!
 All files are saved in the directory named UserScript_Compare_ROM_methods in the working directory
 See <LUNAR_UserScript_Compare_ROM_methods_Dataset_1.csv>
-------------------------------------------------------------------------------------------------
```

Figure 3.15: Results of running LUNAR's script

A Quasar script provided by Lunar was used to compare ML algorithms. The script automatically trained different machine learning models with training data (60 curves) and loaded the x-validation points to make predictions. With the validation Y data (20 curves) corresponding to these x-validation data provided by FE-simulations, the script evaluated the performance of each algorithm automatically, and printed the best algorithm and the best parameters into a text file afterwards. Figure 3.15 shown above is an example of script's results showing that the best method for the particular dataset was POD with RBF as interpolation method.

For each dataset, this script is carried out to investigate which algorithm is the best option. Selecting results are shown as table 3.1 below:

Table 3.2: Methods used in LUNAR for each dataset

Name	Simulation		Solver
Head node acceleration	Braking		POD Rbf
Head node acceleration	Steering	Total	Clustering Kriging
Head node acceleration	Steering	X axis	POD Krigging
Head node acceleration	Steering	Y axis	POD Rbf
Head node displacement	Braking		POD Krigging
Head node displacement	Steering	Total	POD Rbf
Head node displacement	Steering	X axis	POD Krigging
Head node displacement	Steering	Y axis	POD Krigging
Seatbelt force	Braking		POD Krigging
Seatbelt force	Steering		POD Krigging
Rib strain	Braking		POD Rbf
Rib strain	Steering		POD Krigging

The next step is to use the trained model to predict curves according to training DOE. This step was carried out in Lunar's 'Sensitivity' part: Import the testing DOE into Lunar and Click button 'Run' to make Lunar predict curves (YLUNAR). See below figure 3.16.

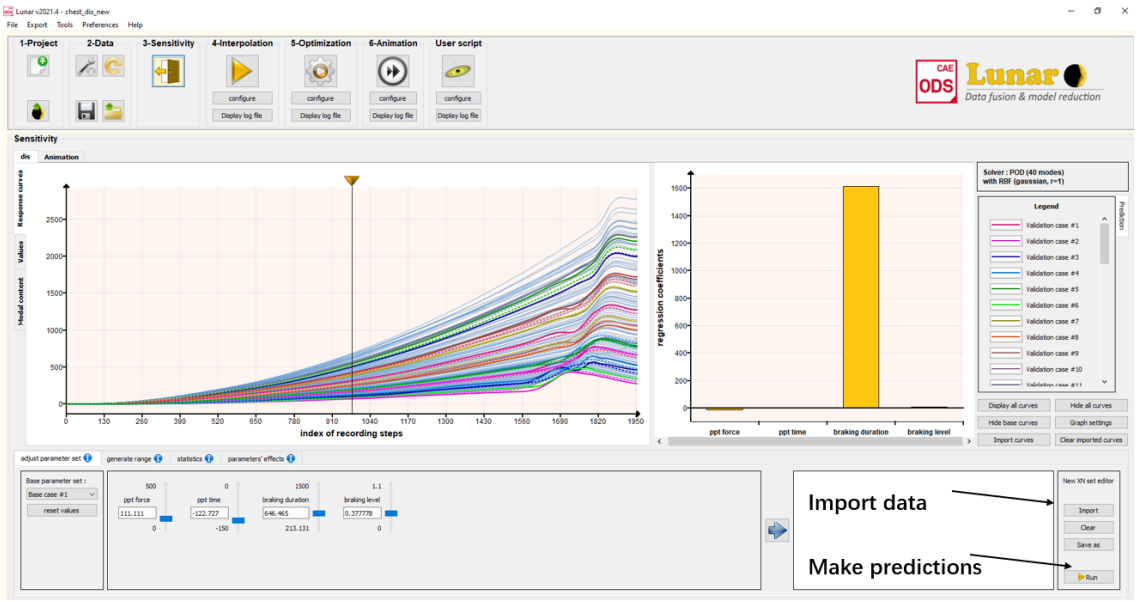


Figure 3.16: Predicting new cases using LUNAR

After this, predicted curves are exported into one CSV file for later analysis.

3.4.3 Reversing prediction curves

All Lunar predicted curves (YLUNAR) were as long as the longest curve in the whole dataset. However, the test dataset Y which comes from FE-simulations (YFE) have different lengths. It's necessary to reverse all Lunar predicted curves back to as long as the original length. This step was also carried out by one function of Scipy. One example is shown as below figure 3.17.

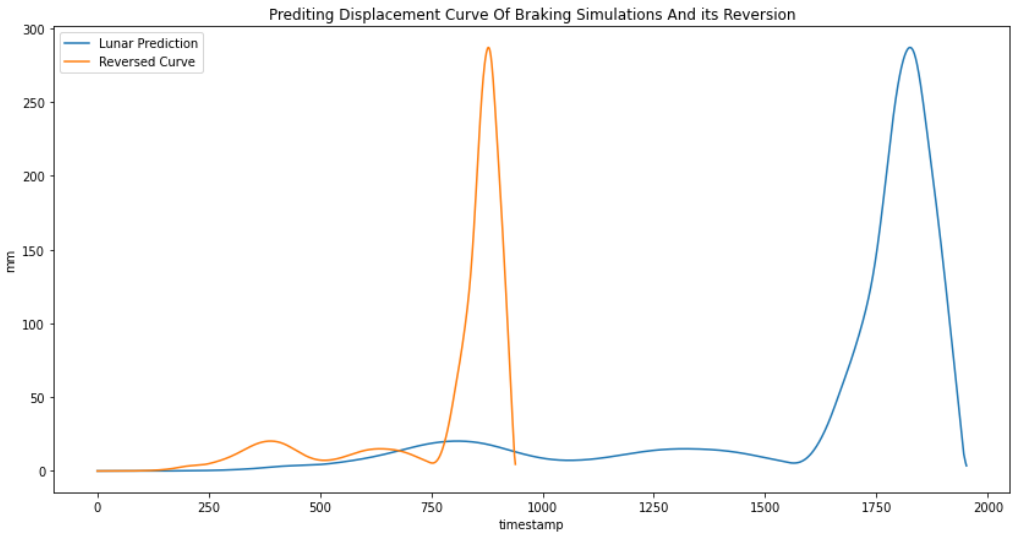


Figure 3.17: Reversed and original curves of head node's displacement of one braking simulation

3.5 Evaluation

With predicted test YLUNAR and YFE, it's possible to evaluate model's performance. For each case in testing DOE, there is one predicted curve and one original curve from simulation. Mean Square Error (MSE), Root-Mean-Square Error (RMSE), Mean Absolute Error (MAE) and Mean Absolute Percentage Error (MAPE) are calculated based on each two curves. Functions from Scikit-learn were used to calculate these four metrics. Then these metrics will be summed and divided by the number of test samples.
Considering most features are close to remain constant or very little before the crash, there is a need to separate the simulation into two parts for evaluation. For each test sample, there are three values of each metric corresponding to three parts of the curves are calculated: pre-crash phases, in-crash phases and the overall curves. For each dataset, four final metrics are used for performance evaluation: MSEavg, RMSEavg, MAEavg and MAPEavg.

4

Results

4.1 Prediction from machine learning

In this section, head node acceleration, head node displacement, belt force and rib strains datasets extracted from FE simulations and corresponding ML predictions are all shown as curves in below figures. Comparation of curves and errors are also included in below figures and tables. For each dataset, two of the good comparison are shown separately.

4.1.1 Head node acceleration

4.1.1.1 Head node acceleration for braking simulations

Head node acceleration of braking simulations datasets are in figure 4.1:

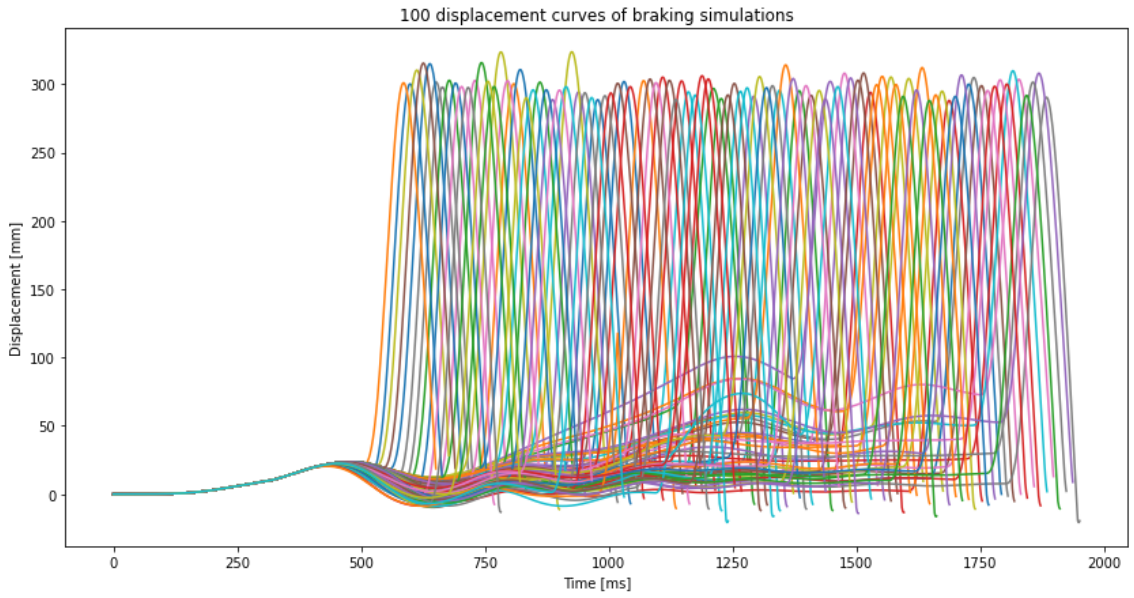


Figure 4.1: Total head node's acceleration curves of steering simulations

Two normal example test cases comparisons between FE-simulation and Lunar prediction of test cases of braking simulation's head node acceleration are shown in appendix D figure D.1 and D.2.
Head node acceleration of braking simulations evaluation results, MSE, RMSE, MAE and MAPE are listed in table 4.1.

Head node's max acceleration values of braking simulations are also predicted and evaluated. The evaluation results, MSE, RMSE, MAE and MAPE are listed in table 4.1.
Separately, errors for pre-crash phase and in-crash phase are listed in table 4.2 and table 4.3.

Table 4.1: Errors of head node acceleration prediction for braking simulations.

	MSE $[(mm/ms^2)^2]$	RMSE $[mm/ms^2]$	MAE $[mm/ms^2]$	MAPE
Head node acceleration for braking simulations	0.0009	0.0270	0.0101	
Peak value of head node acceleration for braking simulations	0.0052	0.0725	0.0605	18.74%

Table 4.2: Errors of head node acceleration prediction for braking simulations in pre-crash scenarios.

	MSE $[(mm/ms^2)^2]$	RMSE $[mm/ms^2]$	MAE $[mm/ms^2]$
Head node acceleration for braking simulations (pre-crash)	0.0001	0.0096	0.0034

Table 4.3: Errors of head node acceleration prediction for braking simulations in in-crash scenarios.

	MSE $[(mm/ms^2)^2]$	RMSE $[mm/ms^2]$	MAE $[mm/ms^2]$
Head node acceleration for braking simulations (in-crash)	0.0069	0.0699	0.0544

4.1.1.2 Head node acceleration for steering simulations

Total head node acceleration curves of steering simulations datasets are in figure 4.2.

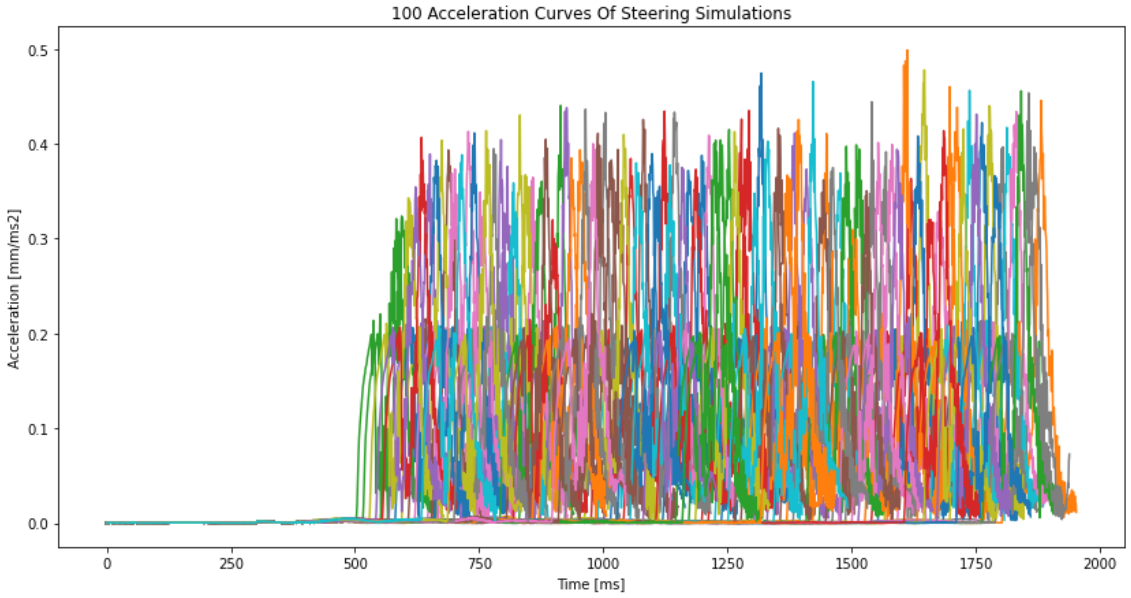


Figure 4.2: Total head node's acceleration curves of steering simulations

Two normal example test cases comparisons between FE-simulation and Lunar prediction of test cases of overall steering simulation's head node acceleration are in appendix D figure D.3 and D.4.
Head node total acceleration of steering simulations evaluation results, MSE, RMSE, MAE and MAPE are listed in table 4.4. Head node's overall max acceleration values of steering simulations are predicted and evaluated. The evaluation results, MSE, RMSE, MAE and MAPE are listed in table 4.4.
Curves of head node acceleration on X axis of steering simulations datasets are in figure 4.3.

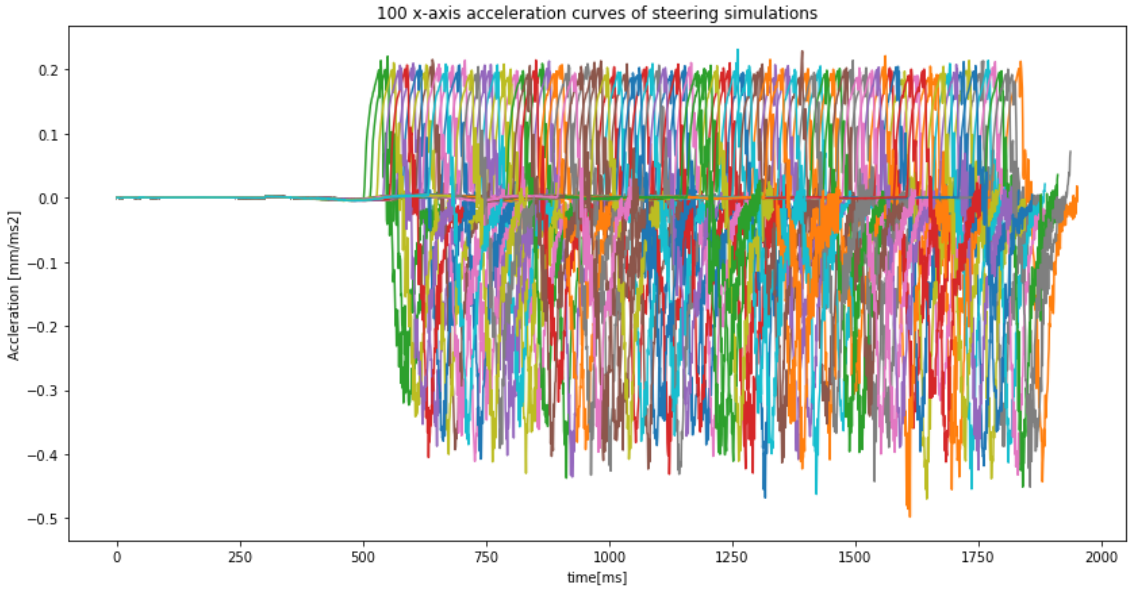


Figure 4.3: Curves of head node acceleration on X axis for steering simulations

Two normal example test cases comparisons between FE-simulation and Lunar prediction of test cases of steering simulation's x-axis head node acceleration are in appendix D figure D.5 and D.6.

Head node acceleration on X axis of steering simulations evaluation results, MSE, RMSE, MAE and MAPE are listed in table 4.4. Head node's max acceleration values on x-axis of steering simulations are predicted and evaluated. The evaluation results, MSE, RMSE, MAE and MAPE are listed in table 4.4.

Curves of head node acceleration on Y axis for steering simulations datasets are in figure 4.4.

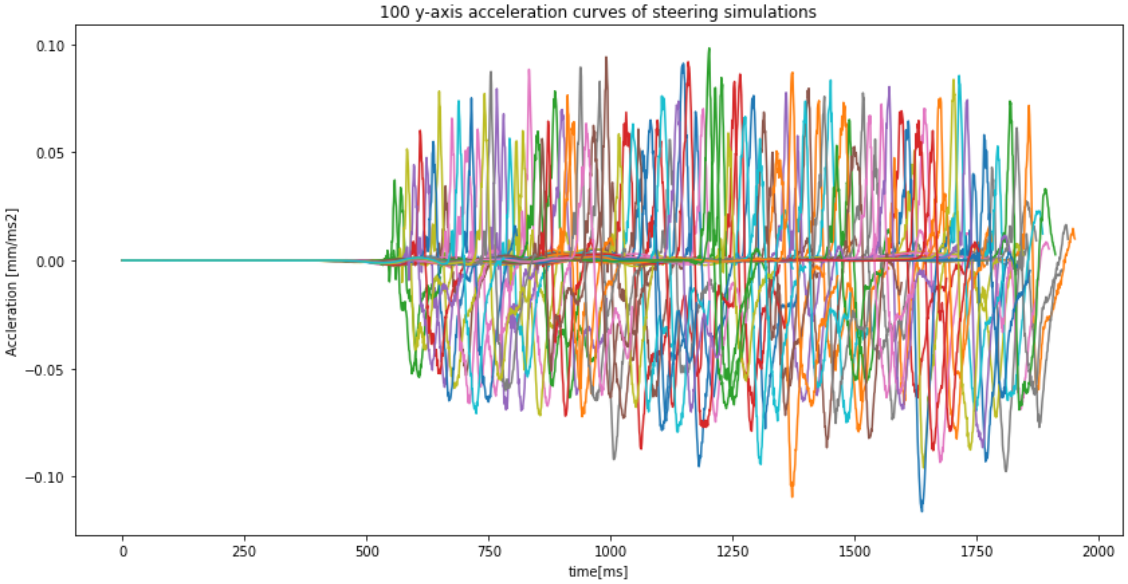


Figure 4.4: head node acceleration on Y axis for steering simulations

Two normal example test cases comparisons between FE-simulation and Lunar prediction of test cases of steering simulation's head node y-axis acceleration are in appendix D figure D.7 and D.8.

Head node acceleration on Y axis of steering simulations evaluation results, MSE, RMSE, MAE and MAPE are listed in table 4.4. Head node's max acceleration values on y-axis of steering simulations are predicted and evaluated. The evaluation results, MSE, RMSE, MAE and MAPE are listed in table 4.4.

Separately, errors for pre-crash phase and in-crash phase are listed in table 4.5 and table 4.6.

Table 4.4: Errors of head node acceleration prediction for steering simulations.

		MSE $[(mm/ms^2)^2]$	RMSE $[mm/ms^2]$	MAE $[mm/ms^2]$	MAPE
Head node acceleration for steering simulations	Total	0.0009	0.0303	0.0113	
	X axis	0.0009	0.0281	0.0108	
	Y axis	7.548e-5	0.0084	0.0026	
Peak value of head node acceleration for steering simulations	Total	0.0156	0.1249	0.1184	29.87%
	X axis	0.0077	0.0877	0.0782	19.70%
	Y axis	0.0003	0.0184	0.0142	18.39%

Table 4.5: Errors of head node acceleration prediction for steering simulations in pre-crash scenarios

Head node acceleration for steering simulations (pre-crash)	MSE $[(mm/ms^2)^2]$	RMSE $[mm/ms^2]$	MAE $[mm/ms^2]$
Total	5.334e-5	0.0063	0.0021
X axis	5.750e-5	0.0069	0.0026
Y axis	2.016e-7	0.0004	0.0003

Table 4.6: Errors of head node acceleration prediction for steering simulations in in-crash scenarios

Head node acceleration for steering simulations (in-crash)	MSE $[(mm/ms^2)^2]$	RMSE $[mm/ms^2]$	MAE $[mm/ms^2]$
Total	0.0056	0.0736	0.0586
X axis	0.0047	0.0670	0.0026
Y axis	0.0004	0.0211	0.0150

4.1.2 Head node displacement

4.1.2.1 Head node displacement for braking simulations

Head node displacement of braking simulations datasets are shown in figure 4.5 below.

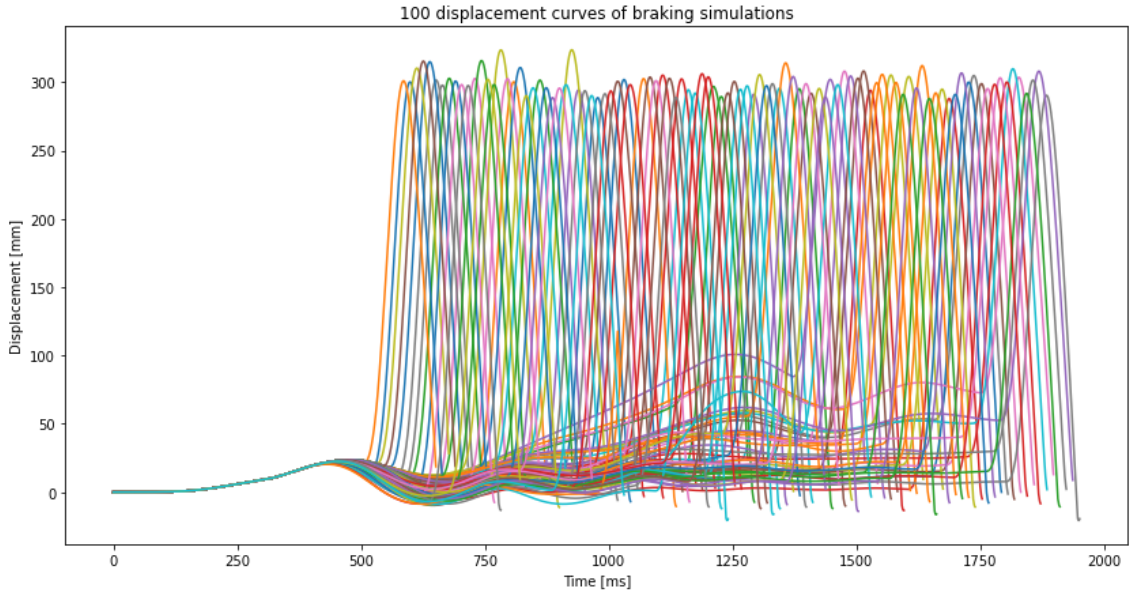


Figure 4.5: Head node's displacement curves of braking simulations

Two normal example test cases comparisons between FE-simulation and Lunar prediction of test cases of braking simulation's head node displacement are in appendix D figure D.9 and D.10.

Head node displacement of braking simulations evaluation results, MSE, RMSE, MAE and MAPE are listed in table 4.7.

Maximum value of head node displacement of braking simulations evaluation results, MSE, RMSE, MAE and MAPE are listed in table 4.7.

Separately, errors for pre-crash phase and in-crash phase are listed in table 4.8 and table 4.9.

Table 4.7: Errors of head node displacement prediction for braking simulations.

	MSE $[mm^2]$	RMSE $[mm]$	MAE $[mm]$	MAPE
Head node displacement for braking simulations	291.771	11.423	6.5847	
Peak value of head node displacement for braking simulations	1619.20	40.239	17.835	10.43%

Table 4.8: Errors of head node displacement prediction for braking simulations in pre-crash scenarios

	MSE $[mm^2]$	RMSE $[mm]$	MAE $[mm]$
Head node displacement for braking simulations (pre-crash)	176.2206	8.1997	4.8640

Table 4.9: Errors of head node displacement prediction for braking simulations in in-crash scenarios

	MSE $[mm^2]$	RMSE $[mm]$	MAE $[mm]$
Head node displacement for braking simulations (in-crash)	1718.7528	23.8487	19.6297

4.1.2.2 Head node displacement for steering simulations

Total head node displacement curves of steering simulations datasets are in figure 4.6 as below.

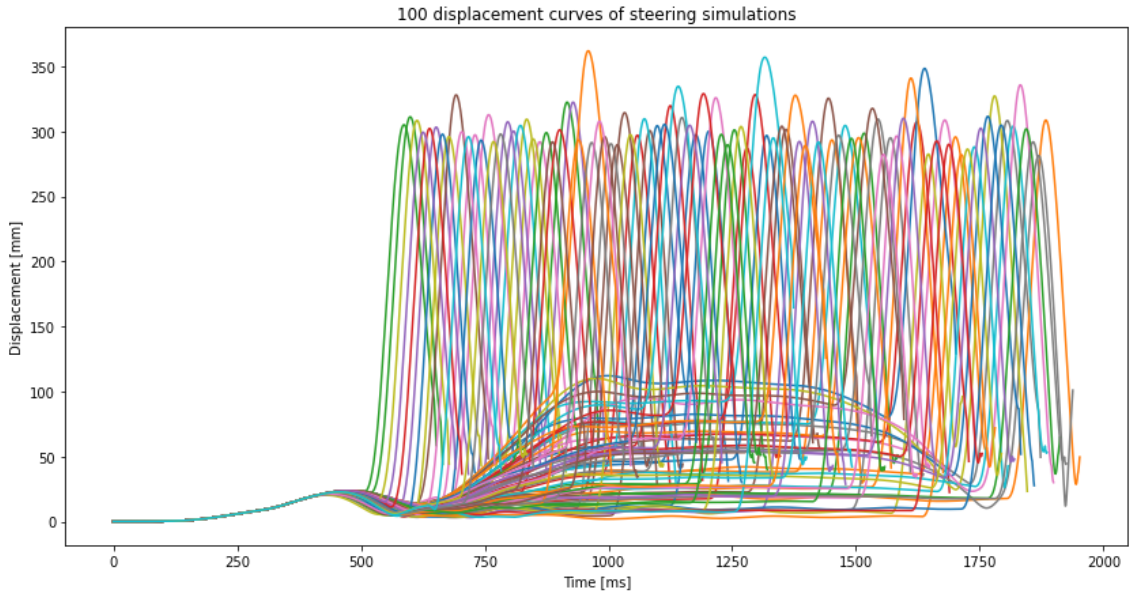


Figure 4.6: Total head node's displacement curves of steering simulations

Two normal example comparisons between FE-simulation and Lunar prediction of test cases of steering simulation's head node displacement are in appendix D figure D.11 and D.12.

Total head node displacement of steering simulations evaluation results, MSE, RMSE, MAE and MAPE are listed in table 4.10. Maximum value of total head node displacement of steering simulations results, MSE, RMSE, MAE and MAPE are listed in table 4.10.

Curves of head node displacement on X axis for steering simulations datasets are in figure 4.7 as below.

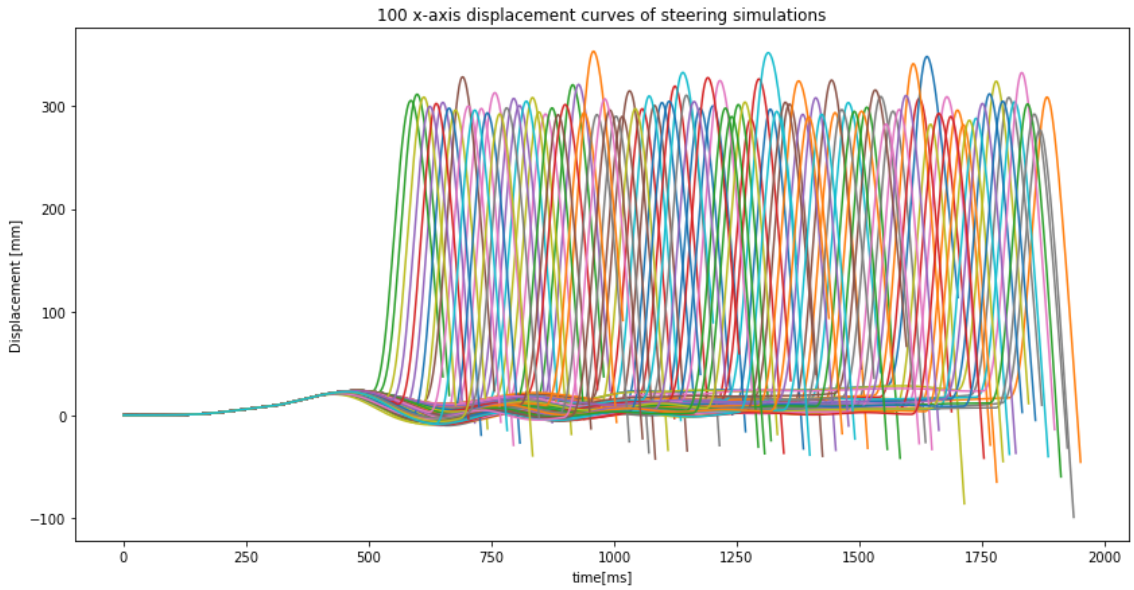


Figure 4.7: Curves of head node displacement on X axis for steering simulations

Two normal example comparisons between FE-simulation and Lunar prediction of test cases of steering simulation's head node displacement are in appendix D figure D.13 and D.14.

Head node displacement on X axis for steering simulations evaluation results, MSE, RMSE, MAE and MAPE are listed in table 4.10. Maximum value of head node displacement on X axis for steering simulations evaluation results, MSE, RMSE, MAE and MAPE are listed in table 4.10.

Curves of head node displacement on Y axis for steering simulations datasets are in figure 4.8 as below.

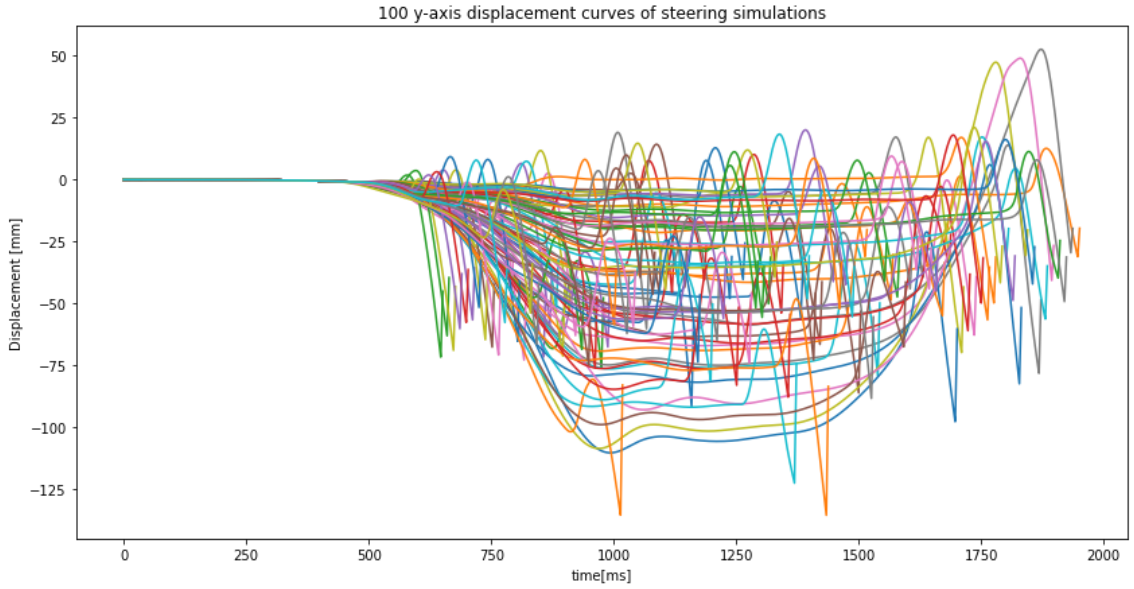


Figure 4.8: Curves of head node displacement on Y axis for steering simulations

Two comparisons between FE-simulation and Lunar prediction of test cases of steering simulation's head node displacement are in appendix D figure D.15.

Head node displacement on Y axis for steering simulations evaluation results, MSE, RMSE, MAE and MAPE are listed in table 4.10. Maximum value of head node displacement on Y axis of steering simulations evaluation results, MSE, RMSE, MAE and MAPE are listed in table 4.10. Separately, errors for pre-crash phase and in-crash phase are listed in table 4.11 and table 4.12.

Table 4.10: Errors of head node displacement prediction for steering simulations.

		MSE $[mm^2]$	RMSE $[mm]$	MAE $[mm]$	MAPE
Head node displacement for steering simulations	Total	128.331	10.016	5.5308	
	X axis	87.6225	8.6758	5.2925	
	Y axis	14.8550	3.6583	2.1424	
Peak value of head node displacement for steering simulations	Total	169.804	13.030	10.493	3.524%
	X axis	241.304	15.534	11.510	3.842%
	Y axis	83.689	9.1484	7.9200	13.74%

Table 4.11: Errors of head node displacement prediction for steering simulations in pre-crash scenarios

Head node displacement for steering simulations (pre-crash)	MSE $[mm^2]$	RMSE $[mm]$	MAE $[mm]$
Total	21.1567	4.4151	2.9985
X axis	26.4578	5.0058	3.4154
Y axis	6.1348	2.2569	1.4134

Table 4.12: Errors of head node displacement prediction for steering simulations in in-crash scenarios

Head node displacement for steering simulations (in-crash)	MSE $[mm^2]$	RMSE $[mm]$	MAE $[mm]$
Total	584.8812	21.7126	17.4170
X axis	356.0453	17.2618	14.1289
Y axis	58.7281	7.2002	5.9414

4.1.3 Belt force

4.1.3.1 Belt force for braking simulations

Belt force of braking simulations datasets are in figure 4.9 as below.

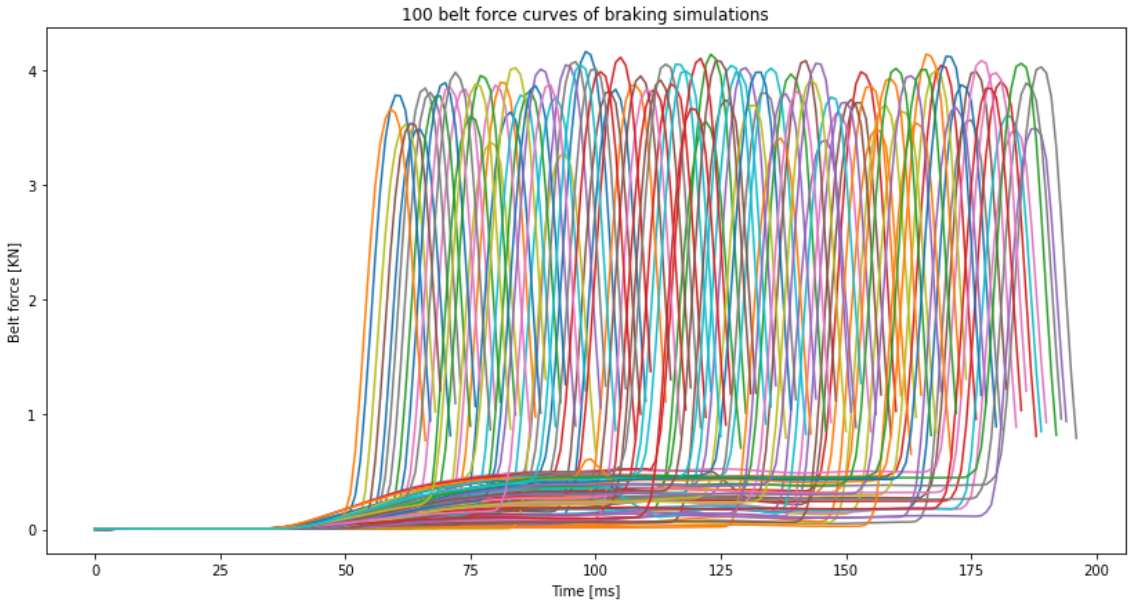


Figure 4.9: Belt force curves of braking simulations

Two comparisons between FE-simulation and Lunar prediction of test cases of braking simulation's belt force are in appendix D figure D.16 and D.17.

Belt force of braking simulations results, MSE, RMSE, MAE and MAPE are listed in table 4.5.

4.1.3.2 Belt force for steering simulations

Belt force of steering simulations datasets are in figure 4.10 as below.

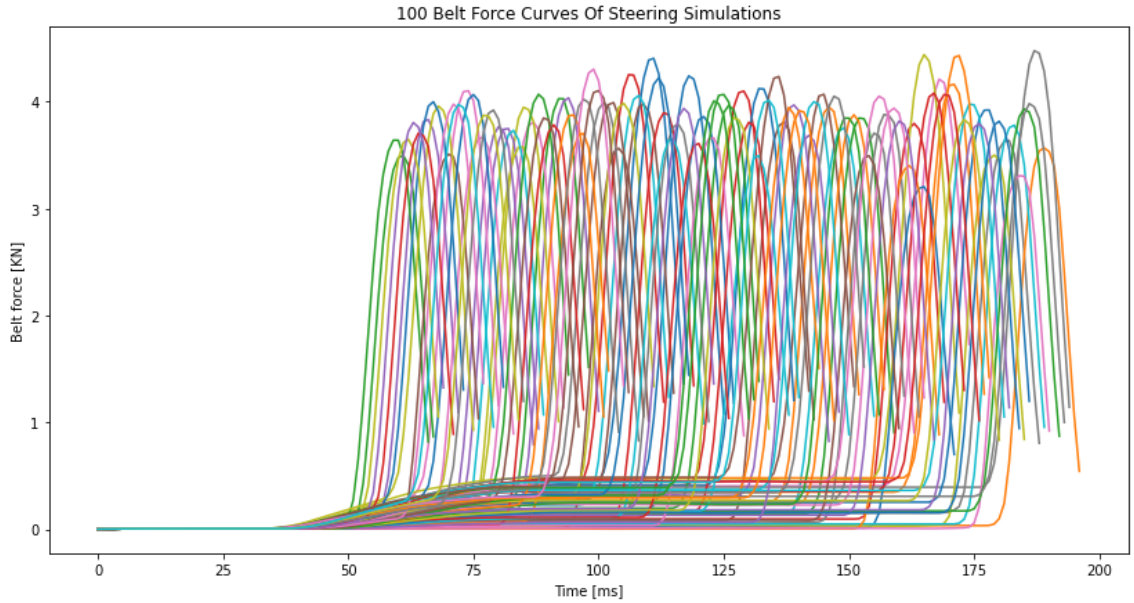


Figure 4.10: Belt force curves of steering simulations

Two comparisons between FE-simulation and Lunar prediction of test cases of steering simulation's belt force are in appendix D figure D.18 and D.19.

Belt force of steering simulations results, MSE, RMSE, MAE and MAPE are listed in table 4.13. Separately, errors for pre-crash phase and in-crash phase are listed in table 4.14 and table 4.15.

Table 4.13: Errors of seatbelt force prediction for braking and steering simulations.

	MSE $[kN^2]$	RMSE $[kN]$	MAE $[kN]$
Braking simulations	0.0698	0.2060	0.0872
Steering simulations	0.0231	0.1461	0.0661

Table 4.14: Errors of seatbelt force prediction for braking and steering simulations in pre-crash scenarios

Seatbelt force (pre-crash)	MSE $[kN^2]$	RMSE $[kN]$	MAE $[kN]$
Braking simulations	0.0130	0.0603	0.0281
Steering simulations	0.0012	0.0310	0.0188

Table 4.15: Errors of seatbelt force prediction for braking and steering simulations in in-crash scenarios

Seatbelt force (in-crash)	MSE $[kN^2]$	RMSE $[kN]$	MAE $[kN]$
Braking simulations	0.5890	0.5475	0.4805
Steering simulations	0.1259	0.3441	0.2892

4.1.4 Rib strains

4.1.4.1 Rib strains for braking simulations

Rib strains of braking simulations datasets are in figure 4.11 as below,

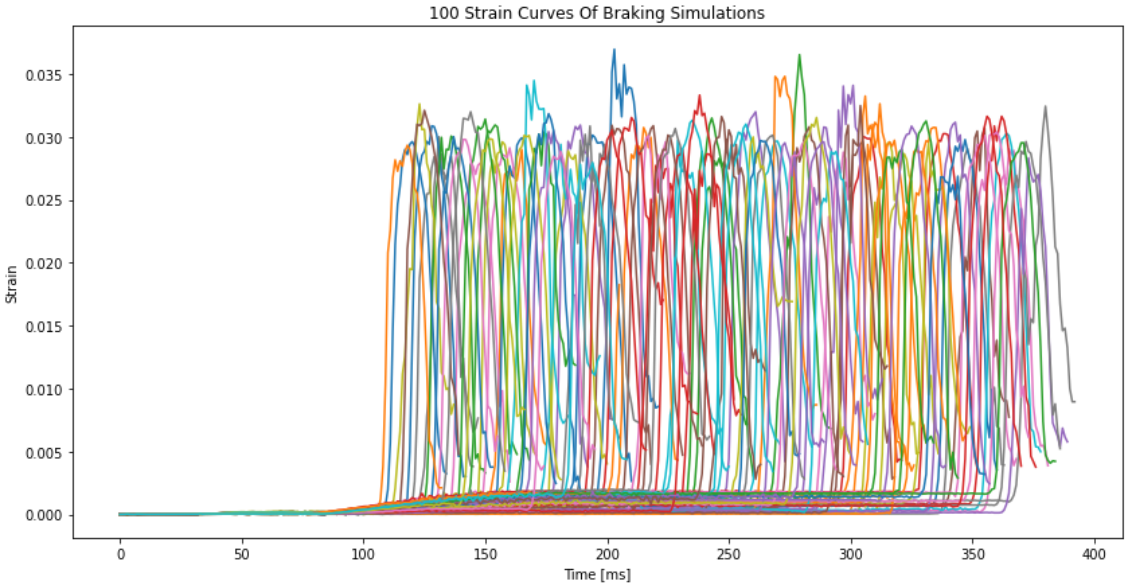


Figure 4.11: Rib strain curves of braking simulations

Two comparisons between FE-simulation and Lunar prediction of test cases of braking simulation's Strains are in appendix D figure D.20 and D.21.

Strains of braking simulations results, MSE, RMSE, MAE and MAPE are listed in table 4.6.

4.1.4.2 Rib strains for steering simulations

Rib strains of steering simulations datasets are shown in figure 4.12,

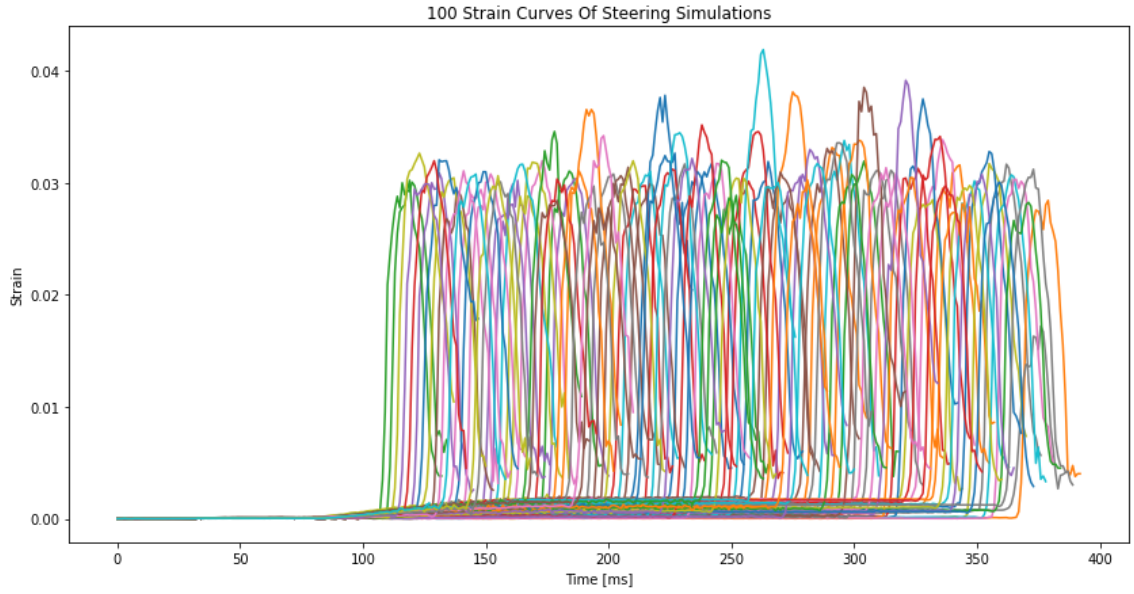


Figure 4.12: Rib strain curves of steering simulations

Two comparisons between FE-simulation and Lunar prediction of test cases of steering simulation's Strains are in appendix D figure D.22 and D.23.

Strains of steering simulations results, MSE, RMSE, MAE and MAPE are listed in table 4.16. Separately, errors for pre-crash phase and in-crash phase are listed in table 4.17 and table 4.18.

Table 4.16: Errors of rib strain prediction for braking and steering simulations.

	MSE	RMSE	MAE
Braking simulations	6.598e-6	0.0020	0.0008
Steering simulations	3.446e-6	0.0017	0.0006

Table 4.17: Errors of rib strain prediction for braking and steering simulations in pre-crash scenarios

Rib strain (pre-crash)	MSE	RMSE	MAE
Braking simulations	1.1314e-6	0.0005	0.0002
Steering simulations	5.0629e-8	0.0002	9.069e-5

Table 4.18: Errors of rib strain prediction for braking and steering simulations in in-crash scenarios

Rib strain (in-crash)	MSE	RMSE	MAE
Braking simulations	4.8647e-5	0.0060	0.0049
Steering simulations	1.8525e-5	0.0041	0.0033

4.2 Time cost reduction

In traditional finite element method, a human body model for crash simulation may cost one day or even more days. In this project, the Hybrid III FAST model is a simpler model than HBMs. One pre-crash and crash simulation takes mostly between 25 and 60 minutes. 200 simulations are done and 199 of them finished successfully, so the average time cost for successful simulations could be calculated. When several machine learning algorithms in Lunar software are used to generate models and predict specific data, the process becomes very fast. And based on many experiences, the time is about 5 seconds.
FE simulations run in high-performance computers (HPC) provided by Autoliv computing clusters remotely.
While machine learning predictions are made on Virtual Desktop Infrastructure (VDI). Its processor is Intel(R) Xeon(R) Gold 6150 CPU @ 2.70 GHz 2.69 GHz (2 processors) and installed RAM is 24.0 GB.
In valid 99 simulations for braking maneuvers and 100 simulations for steering maneuvers, the former takes an average of 2196.53 seconds, and the steering simulation takes an average of 2070.55 seconds. If the time of 199 simulations is calculated together, and the average time is 2133.22 seconds. the percentage of reduction in time consumption can be calculated by following formula. The time reduction is calculated in below table 4.19.

$$TRPS = \frac{Time_{OneFE} - Time_{OneML}}{Time_{OneFE}} \times 100\% \tag{4.1}$$

TRPS is time reduction per simulation. $Time_{OneFE}$ is the average time cost of one FE simulation, and $Time_{OneML}$ is the average time cost for one ML prediction. Table 4.19 shows the time reduction per simulation in percentage.

Table 4.19: Time cost per simulation FE vs. ML

	FE	ML	Time reduction
One braking simulations	2196.53s	5s	99.77%
One steering simulations	2070.55s	5s	99.76%
All	2133.54s	5s	99.77%

In this project, for every 100 braking and steering simulations, total time reduction could be calculated as below,

$$TTR = \frac{Time_{100FE} - Time_{ML}}{Time_{100FE}} \times 100\% \tag{4.2}$$

$$Time_{ML} = Time_{80FE} + Time_{MLValidation} + Time_{MLTrain} + Time_{MLPredict} \tag{4.3}$$

TTR is total time reduction. $Time_{100FE}$ is the time cost of all 100 FE simulations. $Time_{80FE}$ is the time cost of 80 FE simulations which will be used for ML training and validation. $Time_{MLValidation}$ is the time cost of validation used for selecting best algorithm for each situation, and here it is about 10 seconds. $Time_{MLTrain}$ is the time cost of ML training, and here it's around 30 seconds. $Time_{MLPredict}$ is the time cost of remaining 20 ML predictions and it should be $5\text{s} \times 20 = 100s$.
Table 4.20 shows the total time reduction.

Table 4.20: Time cost per simulation FE vs. ML

	FE	ML	Time reduction
Braking simulations	219653s	175862.4s	19.94%
Steering simulations	207055s	165784s	19.93%
All	213354s	170823.2s	19.93%

5

Discussion

In this project, Hybrid III FAST model is used in combined pre-crash and crash simulations with 6 varying parameters to obtain datasets for ML prediction. 199 valid simulations are finally obtained by FE method and corresponding prediction curves are gotten by ML with different algorithms provided by LUNAR. The two methods, FE and ML, can be compared in terms of accuracy and time consumption to determine if it is applicable and efficient to use ML method for occupant pre-crash kinematics and in-crash injury risks.

5.1 Accuracy

In terms of accuracy, ML prediction curves and FE simulation curves can be compared visually and intuitively in plots. Visually, some curves like acceleration, seat belt force and rib strain prediction curves, shown in appendix D, don't match the FE simulation curves well in certain parts, especially the short part before crash happens which is going down or up while the FE simulation curve is almost flat. It may be caused by the interpolation method we used to deal with original FE simulation curves, which will rescale the time axis. The other curves predicted by LUNAR are relatively consistent with the curves simulated by FE, the basic shapes, the peaks and slopes are similar. Mathematically, MSE, RMSE, MAE and MAPE can reflect the error quantitatively. Because the whole curve contains a lot of values around zero, the numerator in equation of MAPE in table 2.1 could be about zero. It means the value of MAPE may be extremely high so that the evaluation is meaningless. For this reason, MAPE is only used for evaluation for peak value predictions and it is especially useful. So, for pre-crash phases, MSE, RMSE and MAE can better represent the performance of prediction. As for in-crash phases, MAPE could represent it directly.

5.1.1 Head node acceleration

For the whole acceleration curve predictions, both in pre-crash and crash scenarios, in above table 4.1 and table 4.4, except acceleration on Y axis for steering simulations, MSE, RMSE, MAE are all at a low level. MSEs are all around 0.0009 $(mm/(ms)^2)^2$. RMSEs are around 0.03 $mm/(ms)^2$ and MAEs are around 0.01 $mm/(ms)^2$. Compared to the FE simulation curves which have a maximum value about 0.4 $mm/(ms)^2$, shown in figure 4.1, 4.2 and 4.3, these errors are low and acceptable. As for acceleration prediction on Y axis, it has lower errors than above.

But in this whole curve, it combines two scenarios, pre-crash and crash. And pre-crash takes a longer period than crash. Separately, evaluation errors for the pre-crash and in-crash phases are quite different. From table 4.2, 4.3, 4.5 and 4.6, compared with the whole curves' errors, MSE, RMSE and MAE for pre-crash phase are all much lower and very close to zero while the errors for in-crash phase are all much higher. RMSE and MAE for total acceleration in in-crash phase go to about 0.07 $mm/(ms)^2$, which is relatively high compared with FE curves' values. And it takes about 17% of maximum value, it is obviously that the low errors for pre-crash reduces the overall error level. To research on the crash part, in above table 4.4, the errors of peak value prediction for head node are higher than whole curves' errors. It can be seen from MAPE. MAPE of the maximum value prediction for head node acceleration is as high as 29.87%. It means that the average error of the prediction is almost 30% of the FE simulation value. MAPEs of the maximum value prediction for head node in other situations are also around 20%. These are non-negligible errors and it can be said that the accuracy is much lower than whole curve predictions.
From the observable plots that obtained, for example in appendix D figure D.3, one possible reason is that the interpolating method rescaled the curve so that the phase of the curve was also changed. And this may affect the machine learning training process and cause the amplitude of the predicted curves to be quite different from the FE simulation curves.

5.1.2 Head node displacement

In table 4.7 and table 4.10, for the whole displacement curve predictions in braking simulations, total displacement and displacement on X axis in steering simulations, RMSEs are all around 10 mm and MAEs are all around 5.8 mm. Compared to the FE simulation curves in corresponding situations which have a maximum value about 320 mm, shown in figure 4.5, figure 4.6 and figure 4.7, the errors are very low. The same to prediction of Y axis displacement curve in steering simulations. RMSE is 3.6583 mm and MAE is 2.1424 mm. While in FE simulations, the maximum value of displacement on Y axis is between 50 to 130 mm, shown in figure 4.8. These errors are also low and acceptable. For pre-crash and in-crash scenarios separately, similar with head acceleration, compared with the whole curves' errors, MSE, RMSE and MAE for pre-crash phase are all lower while the errors for in-crash phase are all higher. But there are less gap and difference between prediction curves and FE simulation curves, as shown in appendix D example figure D.10 and D.11. It also can be seen from the peak value prediction that MAPE shows the errors clearly. In steering simulations, when considering the total displacement and displacement on X axis, MAPEs are both lower than 4%. It means the average error of the prediction is as low as about 4% of the FE simulation value. It shows a very high accuracy. While for displacement on Y axis in steering simulations, MAPE reaches to 13.74%. It is still at a low level but higher than 4% as above. One possible reason can be found in figure 4.8. In steering simulations, there is only steering pulse on Y axis so the curves change a lot according to varying parameters like steering level. So, the Y axis displacement curves look irregular, which reduces the accuracy of the

prediction. In braking simulations, MAPE is 10.43%. It also means low error and high accuracy.
Totally, performance of head node displacement prediction seems better than head node acceleration prediction. One possible reason is that there is much more variability in the acceleration signals while the shape of the displacement curve is simpler and more monotonic than the acceleration curve, which makes the ML model been trained better.

5.1.3 Belt force

In above table 4.13, for the whole belt force prediction curves in braking and steering simulations, the average RMSE is 0.17 kN and MAE is 0.077 kN. From figure 4.9 and figure 4.10, the maximum belt force in FE simulations is between 3 to 4 kN. So, the errors take about 2%-6% of the maximum value. For the seatbelt force, the prediction curves fit well. If separate them to pre-crash and in-crash phases, in table 4.14 and table 4.15, same with previous cases, errors for pre-crash phase are lower than overall curve while errors for in-crash phase are higher. In crash scenarios, RMSE and MAE are around 0.5 kN in braking simulation which take about 15% of the maximum value of FE simulations. And RMSE and MAE are around 0.3 kN in steering simulations which take about 9% of the maximum value.

5.1.4 Rib strain

In above table 4.16, for the whole rib strains curves in braking and steering simulations, MSE are both close to zero. The average RMSE is 0.00185 and MAE is around 0.0007. In figure 4.11 and figure 4.12, the maximum rib strain is between 0.025 and 0.04. RMSE and MAE take 2%-7% of the maximum value. So, the rib strain prediction curves fit also very well. If separate them to pre-crash and in-crash phases, in table 4.17 and table 4.18, same with previous cases, errors for pre-crash phase are lower than overall curve while errors for in-crash phase are higher. In crash scenarios, RMSE and MAE are around 0.0055 in braking simulation which take about 15% of the maximum value of FE simulations. And RMSE and MAE are around 0.0035 in steering simulations which take about 10% of the maximum value.
In all above curve predictions, there are some points that are easier to predict, such as a curve near zero at the beginning of the simulation which is called pre-crash phases, and their low errors will lower the overall errors.

5.2 Time consumption

In terms of time reduction, in table 4.19, by using FE method, the average time spent per simulation is around 2100 seconds while the average time for each prediction is about only 5 seconds. The percentage of average time reduction per simulation reached 99.77%. While ML prediction needs preparation datasets for training and validation, which should also be considered as parts of time cost of ML. So, the time cost of ML prediction should be the sum of time cost of FE simulations required

for training and validation, time cost of ML training, validation and prediction. As shown in table 4.20, the total time reduction by using ML is about 19.93%. Because in ML method, there are still 80 FE simulations required for training and validation which took most time. If reduce the amount of FE simulations required in ML method, the time reduction will be much higher, but the accuracy may decrease accordingly.

6
Conclusion

In summary, for whole workflow, ML can reduce a lot of FE simulation computation time, which reaches to about 19.93
When predicting curves, pre-crash phases always have much higher accuracy than in-crash phases and the very low errors for pre-crash phases reduced the overall error levels. In all pre-crash scenarios for every case, the performance of prediction is good. It could be seen both from figures in appendix xx visually and from the low RMSE and MAE levels in tables in section 4.1. The only problem is that the short period before crash which may be caused by interpolating method.
As for in-crash phases, if we define that if RMSE and MAE take lower than 15% of the FE simulations' maximum value or MAPE is lower than 15%, the accuracy is high. If they are between 15% and 25%, then they have lower accuracy. And if they are higher than 25%, it could be considered as a very low accuracy. For head node displacement, rib strain and belt force prediction in both braking and steering simulations, ML has a high accuracy. The accuracy will be lower if use ML to predict head node acceleration in braking simulations. The accuracy is very low when predicting head node acceleration in steering simulations, and the mean average percentage error can be as high as 30
In short, ML can be a time-efficient method for accurate analysis of occupant pre-crash kinematics and in-crash injury risks in most situations.

7
Future work

LUNAR software also provides the animation predictions for pre-crash and crash models. In theory, it can directly predict the animation of the simulation, so it can intuitively evaluate the predicted result visually. It could be done in future work.
In this thesis, only Hybrid III FAST model is used. As the more common and complex models, AHBM and HBM, could be tested in the similar FE and ML methods for the same objective. Additionally, AHBM and HBM can provide more datasets about kinematics and human injury risks such as lumbar force and brain strains.
Besides, in discussion part, we noticed that the interpolating method we used for original FE simulation curves in this project may have a bad influence on the accuracy of the prediction curves. So, another interpolating method may mitigate or even eliminate this effect. For instance, padding zeros in the beginning part of curves. By using this method, it may also introduce other disadvantages, such as prediction accuracy issues in pre-crash scenarios. It's worth trying in future work.

Bibliography

[1] Peden, Margie, Scurfiled, R., Sleet, D., Mohan, D., Hyder, A., Jarawan, E. (2004). World report on road traffic injury prevention. WHO.

[2] Bachani, A. M. , Peden, M. , Gururaj, G. , Norton, R. , Hyder, A. A. . (2017). Road Traffic Injuries.

[3] Manby F. (2009). Clunk, click–an invention that's saved lives for 50 years. Yorkshire Post.

[4] Foulois, B. D. , Glines, C. V. . (1968). From the wright brothers to the astronauts: the memoirs of benjamin d. foulois.

[5] Volvo Celebrates 20 Years of Airbag Production. (2007). Motor Equipment News, 24.

[6] Joshua-Philip Okeafor. (2019, September 6). The 3-point car seat belt: 60 years old and still going strong. https://naijauto.com/car-events/3-point-car-seat-belt-4749.

[7] B. Pipkorn and L. Jaber. PPMI Benefits. (2015). Autoliv Development AB.

[8] Cummings, P. (2002). Association of seat belt use with death: a comparison of estimates based on data from police and estimates based on data from trained crash investigators. Inj Prev, 8(4), 338-341.

[9] Ibanez. (2014, March 19). Tipos de airbags en los coches. https://www.circulaseguro.com/tipos-de-airbags-en-los-coches/.

[10] Lim, H. S. M., Taeihagh, A. (2019). Algorithmic decision-making in AVs: Understanding ethical and technical concerns for smart cities. https://doi.org/10.3390/su11205791

[11] Harper, C. D. , Hendrickson, C. T. , Samaras, C. . (2016). Cost and benefit estimates of partially-automated vehicle collision avoidance technologies. Accident Analysis Prevention.

[12] Sydney Robinson. (2016, December 13). For Decades, Crash Tests Only Used Male Dummies: The Result? Women Were Injured and Killed More in Car Accidents. https://trofire.com/2016/12/13/decades-crash-tests-used-male-dummies-result-women-injured-killed-car-accidents/.

[13] Niranjan Poojary Y, Srinivas A. (2021). Finite Element Human Body Model-based Injury Prediction using Machine Learning: Development of simulation-based surrogate models for injury metrics. [Master's thesis, Chalmers University of Technology]. https://odr.chalmers.se/handle/20.500.12380/303774

[14] Gurram S, Reddy V S K. (2021). Vehicle Occupant Kinematics prediction using Machine Learning: A study to understand applications of machine learning in prediction of vehicle occupant kinematics during

crash and pre-crash. [Master's thesis, Chalmers University of Technology]. https://odr.chalmers.se/handle/20.500.12380/304192

[15] José Antonio Fernández Gascón. (2020). Development of a time-efficient method for evaluation of vehicle occupant pre-crash and crash kinematics and crash injury risk. [Master's thesis, Chalmers University of Technology].

[16] Zienkiewicz, O. C., Taylor, R. L., Zhu, J. Z. (2005). The finite element method: its basis and fundamentals (6. ed.). Elsevier Butterworth-Heinemann.

[17] Hastings, J. K., Juds, M. A., Brauer, J. R. . (1985). Accuracy and Economy of Finite Element Magnetic Analysis, 33rd Annual National Relay Conference.

[18] M. D. Gilchrist. (2005). IUTAM Symposium on Impact Biomechanics: From Fundamental Insights to Applications. Springer.

[19] Toyota Motor Corporation. (2020, June 16). Toyota Offers Free Access to THUMS Virtual Human Body Model Software. https://global.toyota/en/newsroom/corporate/32665896.html

[20] Larsson, E., et al. (2019). Active human body model predictions compared to volunteer response in experiments with braking, lane change, and combined manoeuvres. Proceedings of the International IRCOBI Conference.

[21] TOYOTA. (2022). About THUMS. https://www.toyota.co.jp/thums/about/.

[22] Humanetics. (2022). Hybrid III 50th Male FE. https://www.humaneticsgroup.com/products/virtual-models/frontal-impact-atd-virtual-models/hybrid-iii-50th-male-fe/hybrid-iii-50th-3: :text=The%20Hybrid%20III%2050th%20Percentile%20Male%20Crash%20Test, automotive%20safety%20restraint%20systems%20in%20frontal%20crash%20testing.

[23] Jordan, M. I., Mitchell, T. M. (2015). Machine learning: trends, perspectives, and prospects. American Association for the Advancement of Science. Science, 349(6245), 255.

[24] Ghahramani, Z. (2015). Probabilistic machine learning and artificial intelligence. Nature, 521(7553), 452. https://doi.org/10.1038/nature14541

[25] Shinde, P. P. , Shah, S. . (2019). A Review of Machine Learning and Deep Learning Applications. ICCUBEA 2018. IEEE.

[26] Mahesh, B. . (2019). Machine Learning Algorithms -A Review.

[27] Akinsola, J. . (2017). Supervised Machine Learning Algorithms: Classification and Comparison.

[28] Alloghani, M. , Al-Jumeily, D. , Mustafina, J. , Hussain, A. , Aljaaf, A. J. . (2020). A systematic review on supervised and unsupervised machine learning algorithms for data science.

[29] Likas, A. , Vlassis, N. , Verbeek, J. J. . (2003). The global k-means clustering algorithm. Pattern Recognition, 36.

[30] Jian, Y. , David, Z. , Frangi, A. F. , Jing-Yu, Y. . (2004). Two-dimensional pca: a new approach to appearance-based face representation and recognition. IEEE transactions on pattern analysis and machine intelligence, 26.1 (2004): 131-137.

[31] Mohiuddin Ahmed, Raihan Seraj, Syed Mohammed Shamsul Islam. (2020). The k-means Algorithm: A Comprehensive Survey and Performance Evaluation. Electronics, 9(1295), 1295. https://doi.org/10.3390/electronics9081295

[32] T. Chai, R. R. Draxler. (2014). Root mean square error (RMSE) or mean absolute error (MAE). Geoscientific Model Development Discussions, 7.1 (2014): 1525-1534

[33] M. I. o. Technology. (2017). Optimal Latin Hypercube Technique. https://abaqus-docs.mit.edu/2017/English/IhrComponentMap/ihr-c-Reference-OptimalLatin.htm.

[34] Johnson, M. E. (1), Moore, L. M. (2), Ylvisaker, D. (3). (1990). Minimax and maximin distance designs. Journal of Statistical Planning and Inference, 26(2), 131-148–148. https://doi.org/10.1016/0378-3758(90)90122-B

A

Appendix 1

200 DOE matrix for braking simulations and steering simulations:

100 Braking simulations:

Table A.1: 100 DOE for braking simulations.

Pre-pretensioner force [N]	Activation time of pre-pretensioner force [ms]	Braking duration [ms]	Braking level [g]
111.111	-122.727	646.465	0.377778
166.667	-63.6364	423.232	0.355556
343.434	-39.3939	843.434	0.388889
333.333	-57.5758	1145.45	0.533333
363.636	-131.818	909.091	0.155556
191.919	-93.9394	1132.32	0.922222
196.97	-27.2727	344.444	0.322222
45.4545	-65.1515	1355.56	0.566667
55.5556	-74.2424	1184.85	0.0222222
348.485	-89.3939	777.778	0.744444
464.646	-19.697	593.939	0.833333
217.172	-150	685.859	0.0111111
282.828	-10.6061	1001.01	0.266667
474.747	-7.57576	1303.03	0.0888889
378.788	-112.121	554.545	1.07778
257.576	-136.364	1368.69	0.5
90.9091	-46.9697	1092.93	0.577778
404.04	-42.4242	278.788	0.0666667
121.212	-15.1515	1224.24	0.822222
409.091	-4.54546	961.616	0.911111
176.768	-33.3333	1342.42	0.122222
40.404	-77.2727	1171.72	0.844444
101.01	-3.03031	830.303	1.06667
247.475	-116.667	764.646	0.888889
207.071	-86.3636	265.657	0.333333

Table A.2: 100 DOE for braking simulations.(continued)

Pre-pretensioner force [N]	Activation time of pre-pretensioner force [ms]	Braking duration [ms]	Braking level [g]
5.0505	-31.8182	541.414	0.7
136.364	-56.0606	1434.34	0.866667
75.7576	-30.303	252.525	0.255556
272.727	-148.485	1158.59	1.04444
318.182	-12.1212	291.919	0.644444
237.374	-60.6061	1421.21	0.588889
126.263	-119.697	1106.06	0.688889
500	-87.8788	1027.27	0.733333
484.848	-134.848	1381.82	0.933333
393.939	-36.3636	1500	0.188889
10.101	-22.7273	396.97	1.02222
30.303	-34.8485	1066.67	1.08889
0	-78.7879	633.333	0.655556
85.8586	-24.2424	1447.47	1.1
156.566	-54.5455	528.283	1.01111
35.3535	-139.394	436.364	0.877778
15.1515	-125.758	974.747	0.455556
151.515	-6.06061	475.758	0.944444
141.414	-96.9697	725.253	0.611111
70.7071	-103.03	1486.87	0.755556
60.6061	-28.7879	869.697	0
171.717	-25.7576	410.101	0.633333
262.626	-16.6667	948.485	0.8
313.131	-9.09091	1289.9	0.544444
232.323	-104.545	1473.74	0.722222
308.081	-133.333	895.96	0.422222
338.384	-113.636	213.131	0.9
292.929	-51.5152	1197.98	0.988889
20.202	-72.7273	357.576	0.311111
489.899	-80.303	817.172	1.03333
252.525	-146.97	383.838	0.977778
297.98	-69.697	659.596	1
439.394	-18.1818	502.02	0.46666
65.6566	-115.152	1119.19	0.133333

Table A.3: 100 DOE for braking simulations.(continued)

Pre-pretensioner force [N]	Activation time of pre-pretensioner force [ms]	Braking duration [ms]	Braking level [g]
267.677	-110.606	1408.08	1.05556
358.586	-145.455	567.677	0.766667
212.121	-62.1212	922.222	0.811111
353.535	-142.424	1079.8	0.711111
429.293	-121.212	935.354	0.955556
383.838	-101.515	1276.77	0.855556
449.495	0	1460.61	0.622222
186.869	-124.242	1394.95	0.1
131.313	-13.6364	987.879	0.277778
181.818	-48.4848	738.384	0.3
328.283	-68.1818	318.182	0.6
368.687	-21.2121	607.071	0.166667
116.162	-59.0909	226.263	0.788889
146.465	-107.576	462.626	0.966667
202.02	-95.4545	305.05	0.677778
25.2525	-84.8485	1250.51	0.366667
479.798	-98.4848	449.495	0.666667
434.343	-50	882.828	0.555556
287.879	-66.6667	488.889	0.144444
424.242	-75.7576	200	0.344444
373.737	-106.061	1211.11	0.511111
469.697	-137.879	620.202	0.522222
303.03	-100	1237.37	0.2
50.5051	-118.182	239.394	0.488889
227.273	-71.2121	1014.14	0.477778
494.949	-43.9394	751.515	0.111111
95.9596	-128.788	1329.29	0.411111
222.222	-1.51516	698.99	0.4
80.8081	-83.3333	712.121	0.211111
242.424	-53.0303	1040.4	0.0444444
459.596	-90.9091	790.909	0.0555556
414.141	-37.8788	1316.16	0.777778
444.444	-143.939	672.727	0.0777778
454.545	-40.9091	1263.64	0.444444
106.061	-140.909	804.04	0.222222
388.889	-45.4545	1053.54	0.177778
323.232	-81.8182	856.566	0.244444
161.616	-130.303	515.152	0.433333
419.192	-92.4242	331.313	0.0333333
277.778	-127.273	370.707	0.233333
398.99	-109.091	580.808	0.288889

100 steering simulations:

Table A.4: 100 DOE for steering simulations.

Pre-pretensioner force [N]	Activation time of pre-pretensioner force [ms]	Steering duration [ms]	Steering level [g]
373.737	-124.242	790.909	0.466667
262.626	-60.6061	1066.67	0.360606
121.212	-118.182	528.283	0.572727
383.838	-81.8182	672.727	0.388889
126.263	-19.697	541.414	0.487879
80.8081	-78.7879	974.747	0.190909
217.172	-89.3939	475.758	0.30404
151.515	-56.0606	1421.21	0.516162
282.828	-128.788	462.626	0.0565657
136.364	-101.515	1092.93	0.523232
434.343	-1.51516	712.121	0.530303
348.485	-15.1515	1316.16	0.332323
141.414	-107.576	1132.32	0.169697
414.141	-48.4848	895.96	0.162626
116.162	-18.1818	423.232	0.20505
449.495	-95.4545	1053.54	0.657576
404.04	-40.9091	1289.9	0.615152
257.576	-139.394	1184.85	0.0494949
60.6061	-142.424	449.495	0.431313
101.01	-93.9394	436.364	0.120202
409.091	-115.152	935.354	0.459596
45.4545	-131.818	1500	0.127273
196.97	-63.6364	869.697	0.282828
489.899	-87.8788	344.444	0.692929
393.939	-143.939	1486.87	0.558586
474.747	-127.273	1263.64	0.0989899
227.273	-150	685.859	0.410101
15.1515	-3.03031	1250.51	0.60101
237.374	-140.909	1119.19	0.593939
156.566	-45.4545	213.131	0.0848485
85.8586	-122.727	738.384	0.353535
222.222	-113.636	265.657	0.219192
20.202	-24.2424	646.465	0.0919192
55.5556	-83.3333	370.707	0.494949
500	-69.697	620.202	0.0707071
30.303	-46.9697	1394.95	0.643434

Table A.5: 100 DOE for steering simulations.(continued)

Pre-pretensioner force [N]	Activation time of pre-pretensioner force [ms]	Steering duration [ms]	Steering level [g]
50.5051	-53.0303	1434.34	0.240404
252.525	-54.5455	1381.82	0.664646
5.0505	-96.9697	1224.24	0.176768
424.242	-50	200	0.685859
353.535	-104.545	1276.77	0.551515
328.283	-66.6667	1001.01	0
25.2525	-27.2727	305.05	0.544444
10.101	-68.1818	1447.47	0.445455
70.7071	-125.758	1158.59	0.325253
464.646	-145.455	291.919	0.622222
363.636	-136.364	1355.56	0.254545
186.869	-121.212	817.172	0.0777778
479.798	-22.7273	1329.29	0.0636364
171.717	-109.091	1460.61	0.212121
166.667	-25.7576	909.091	0.537374
181.818	-130.303	1368.69	0.438384
95.9596	-10.6061	1237.37	0.40303
202.02	-4.54546	410.101	0.367677
277.778	0	698.99	0.00707071
454.545	-6.06061	633.333	0.233333
232.323	-77.2727	1197.98	0.183838
146.465	-7.57576	764.646	0.289899
287.879	-90.9091	659.596	0.0212121
0	-43.9394	922.222	0.678788
494.949	-57.5758	357.576	0.268687
75.7576	-31.8182	567.677	0.7
333.333	-13.6364	488.889	0.509091
40.404	-59.0909	515.152	0.226263
131.313	-72.7273	1211.11	0.381818
90.9091	-148.485	1145.45	0.671717
35.3535	-39.3939	830.303	0.339394
308.081	-16.6667	1473.74	0.155556
267.677	-92.4242	1342.42	0.374747
106.061	-65.1515	751.515	0.579798
161.616	-21.2121	1408.08	0.0282828
297.98	-51.5152	987.879	0.636364
247.475	-9.09091	1106.06	0.106061
111.111	-134.848	804.04	0.565657
65.6566	-33.3333	1027.27	0.148485
484.848	-30.303	961.616	0.50202

Table A.6: 100 DOE for steering simulations.(continued)

Pre-pretensioner force [N]	Activation time of pre-pretensioner force [ms]	Steering duration [ms]	Steering level [g]
-28.7879	1171.72	0.261616	444.444
343.434	-36.3636	1079.8	0.424242
176.768	-42.4242	882.828	0.0141414
368.687	-80.303	1040.4	0.346465
419.192	-112.121	278.788	0.134343
242.424	-37.8788	554.545	0.113131
318.182	-12.1212	843.434	0.247475
459.596	-75.7576	1303.03	0.0424242
191.919	-98.4848	777.778	0.473737
313.131	-34.8485	502.02	0.311111
207.071	-74.2424	318.182	0.452525
378.788	-84.8485	580.808	0.19798
212.121	-106.061	226.263	0.629293
439.394	-100	948.485	0.0353535
323.232	-86.3636	725.253	0.586869
292.929	-119.697	1014.14	0.275758
388.889	-137.879	856.566	0.141414
469.697	-62.1212	252.525	0.480808
338.384	-71.2121	239.394	0.29697
398.99	-133.333	607.071	0.318182
303.03	-146.97	593.939	0.650505
272.727	-116.667	396.97	0.417172
358.586	-110.606	383.838	0.608081
429.293	-103.03	331.313	0.39596

B

Appendix 2

A Python script for updating parameters in key files of 200 FE simulation models for braking and steering automatically:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import math
#from alive_progress import alive_bar
import time
# create functions between value in DOE and value in key file.
X -> number in DOE, Y -> number in key file.
def ppt_force(X):
X = float(X)
Y = round(X / 170, 4)
return ['R PPTFRC',Y]

        def ppt_time(X):
            X = float(X)
            Y = round(500 + X, 4) # X is smaller than 0
            return ['R PRET_T',Y]

        def barking_level(X):
            X = float(X)
            Y = round(X / 1.1, 4)
            return ['R BREAKG',Y]

        def steering_level(X):
            X = float(X)
            Y = round(X / 1.1, 4)
            return ['R STEERG',Y]

        def barke_steering_duration(X):
            X = float(X)
            Y = round(X, 4)
            return ['R CRASHINI',Y]

# create a fuction to change parameters in key file.
```

```
def change_para(file_address,target_parameter,target_value):
            target_value = list(str(target_value))
            length = len(target_parameter)

            f = open(file_address,'r+')
            find_or_not = 0
            for line in f.readlines():
                if line[0:length] == target_parameter:
                    target_line = line
                    find_or_not = 1
            if find_or_not == 0:
                return('this parameter key is not found,
                please check the name')

            replace_line = list(target_line)
            for i in range(len(target_value)):
                replace_line[-2-i] = target_value[-1-i]
            replace_line = ''.join(replace_line)
            f.close()

            ff  = open(file_address,'r+')
            content_before = ff.read()
            content_after = content_before.replace(target_
            line,replace_line)
            ff.seek(0,0)
            ff.truncate()
            ff.write(content_after)
            ff.close()
            return 0
# one function which will change everything
        def fix_everything(file_address,doe):
            num_file = len(file_address)
            for key in doe.keys():
                if len( doe[key]) != num_file:
                    return('size of files and DOE cannot
                    match')
            # with alive_bar(num_file) as bar:
            for i in range(num_file):
                target_file = file_address[i]

                input_para = ppt_force(doe['ppt_force'][i])
                change_para(target_file,input_para[0],
                input_para[1])

                input_para = ppt_time(doe['ppt_time'][i])
                change_para(target_file,input_para[0],
```

```
            input_para[1])

            if 'braking_level' in doe.keys():
                input_para = barking_level(doe['braking_
                level'][i])
                change_para(target_file,input_para[0],
                input_para[1])

                input_para = barke_steering_duration(doe
                ['braking_duration'][i])
                change_para(target_file,input_para[0],
                input_para[1])
            else:

                input_para = steering_level(doe
                ['steering_level'][i])
                change_para(target_file,input_para[0],
                input_para[1])

                input_para = barke_steering_duration(doe
                ['steering_duration'][i])
                change_para(target_file,input_para[0],
                input_para[1])
        return('success')
```

C

Appendix 3

Python for interpolating datasets to make all the curves' length to fit with longest length:

```
class Curves:


    def interploat_all(self):
        for curve in self.data:
            if len(curve) >= self.max_length:
                pass
            else:
                curve = self.interploat(curve)
            if type(curve) == 'numpy.ndarray':
                curve = curve.tolist()
            self.full_length_curve.append(np.array(curve))
        return 'Inerploation has done,
        use function check_inter to see the performace. '
```

D

Appendix 4

Some examples of comparations between FE simulation and prediction curves in different situations:

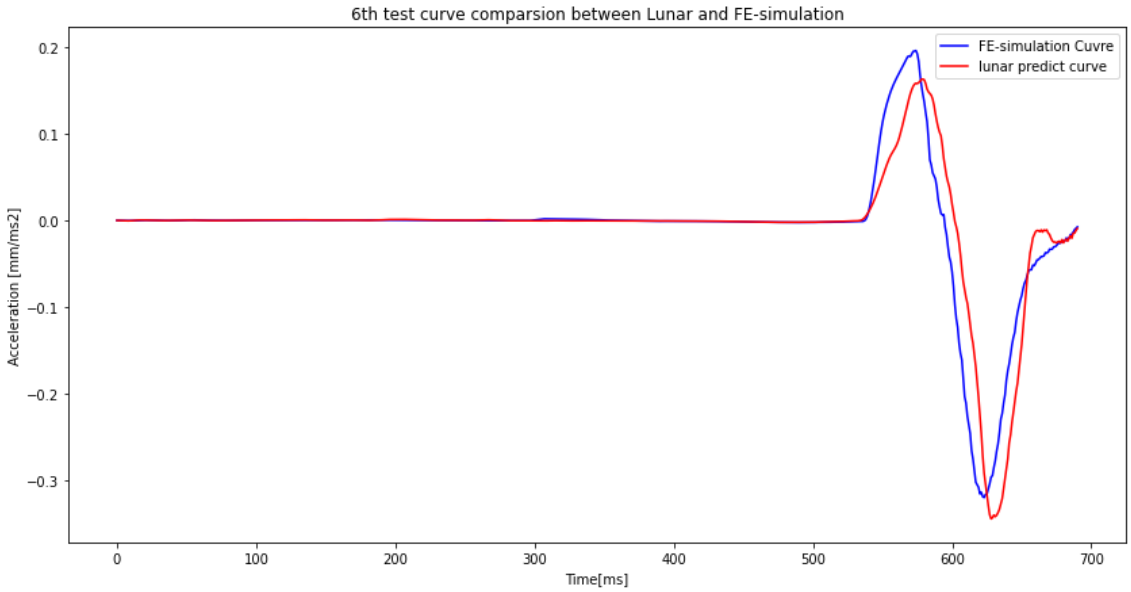


Figure D.1: 6th test case of head node's acceleration between FE-simulation and LUNAR prediction from braking simulations

Table D.1: Error evaluation of 6th test case of head node's acceleration of braking simulation

	MSE $[(mm/ms^2)^2]$	RMSE $[mm/ms^2]$	MAE $[mm/ms^2]$
Overall	0.0009	0.0313	0.0124
Pre-crash	4.0837e-7	0.0006	0.0004
In-crash	0.0045	0.0669	0.0551

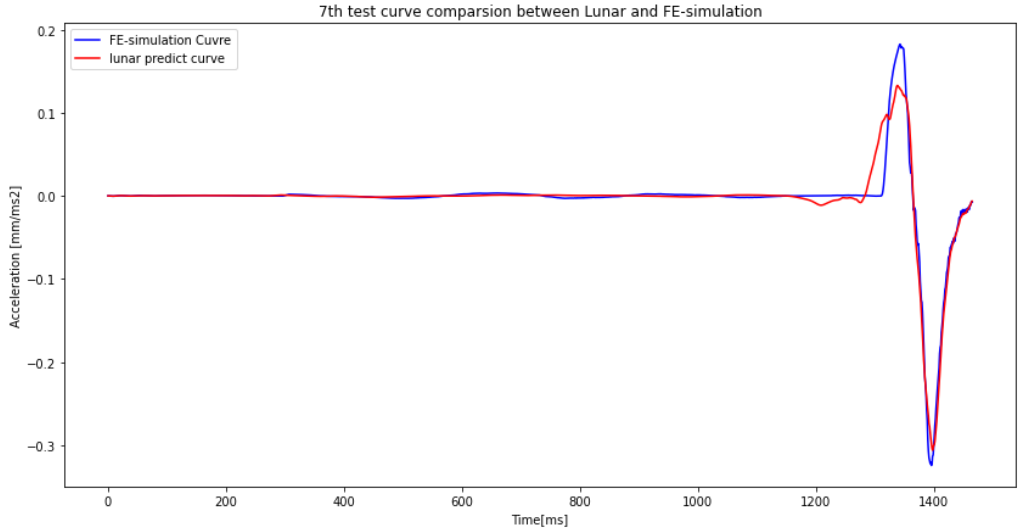


Figure D.2: 7th test case of head node's acceleration between FE-simulation and LUNAR prediction from braking simulations

Table D.2: Error evaluation of 7th test case of head node's acceleration of braking simulation

	MSE $[(mm/ms^2)^2]$	RMSE $[mm/ms^2]$	MAE $[mm/ms^2]$
Overall	0.0001	0.0117	0.0045
Pre-crash	6.4054e-5	0.0080	0.0026
In-crash	0.0007	0.0277	0.0207

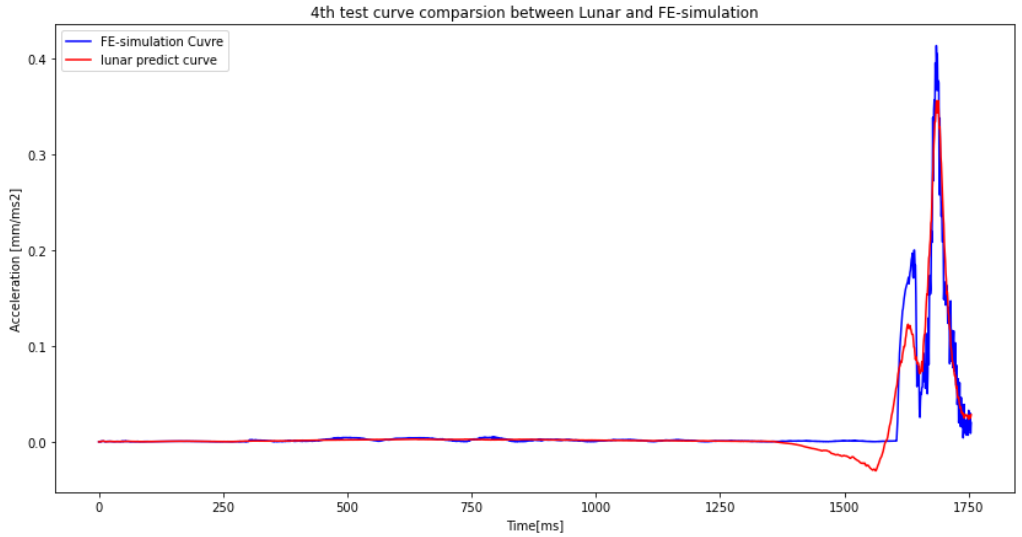


Figure D.3: 4th test case of head node's overall acceleration between FE-simulation and LUNAR prediction from steering simulations

Table D.3: Error evaluation of 4th test case of head node's acceleration of steering simulation

	MSE $[(mm/ms^2)^2]$	RMSE $[mm/ms^2]$	MAE $[mm/ms^2]$
Overall	0.0002	0.0138	0.0531
Pre-crash	3.9349e-5	0.0063	0.0025
In-crash	0.0018	0.0423	0.0344

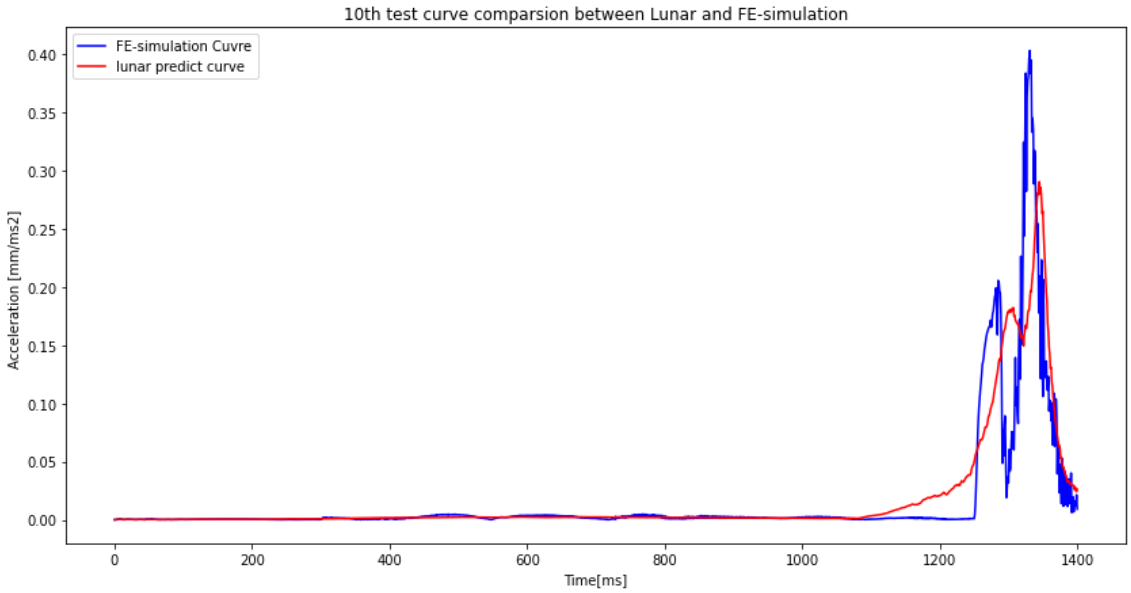


Figure D.4: 10th test case of head node's overall acceleration between FE-simulation and LUNAR prediction from steering simulations

Table D.4: Error evaluation of 10th test case of head node's acceleration of steering simulation

	MSE $[(mm/ms^2)^2]$	RMSE $[mm/ms^2]$	MAE $[mm/ms^2]$
Overall	0.0008	0.0286	0.0095
Pre-crash	4.4481e-5	0.0067	0.0025
In-crash	0.0071	0.0845	0.0663

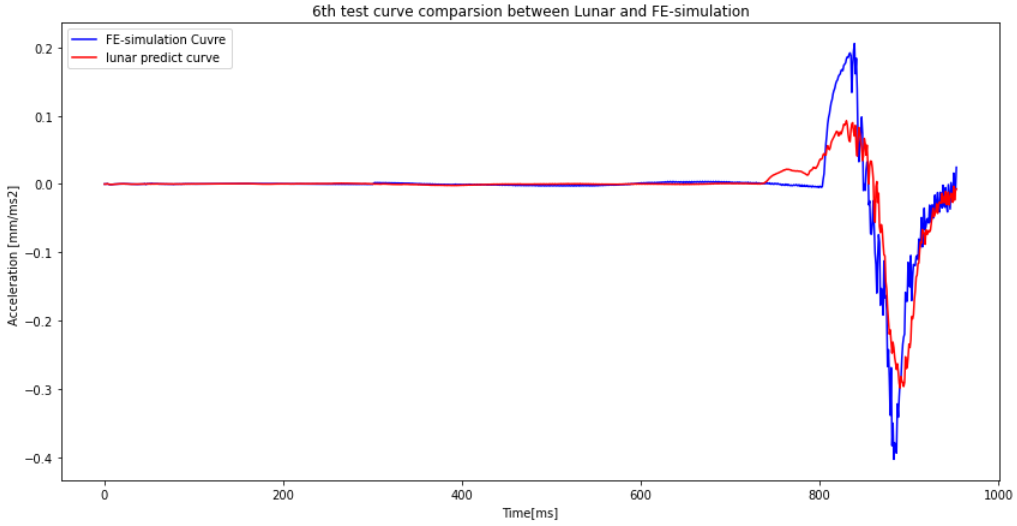


Figure D.5: Error evaluation of 6th test case of head node's x-axis acceleration of steering simulation

Table D.5: Error evaluation of 6th test case of head node's x-axis acceleration of steering simulation

	MSE $[(mm/ms^2)^2]$	RMSE $[mm/ms^2]$	MAE $[mm/ms^2]$
Overall	0.0008	0.0274	0.0107
Pre-crash	3.6797e-5	0.0061	0.0026
In-crash	0.0045	0.0674	0.0536

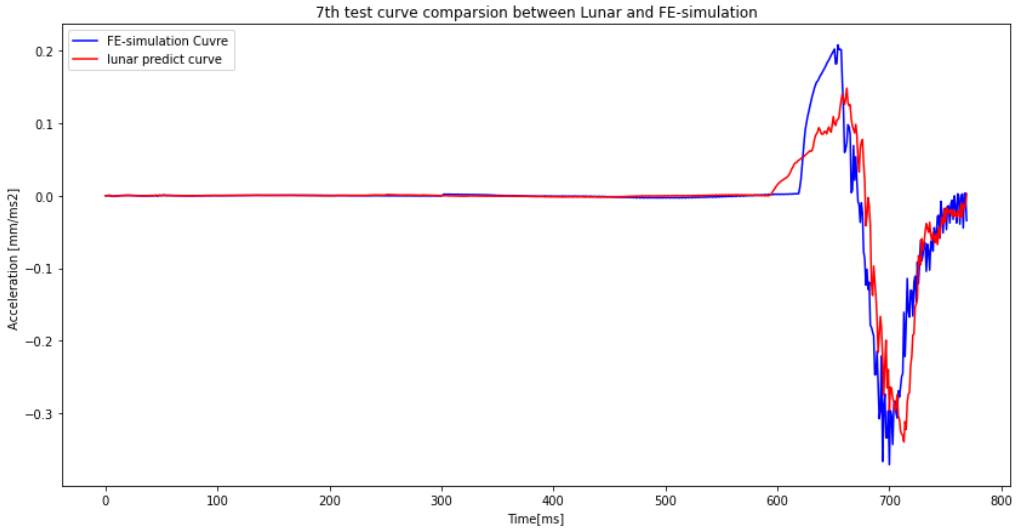


Figure D.6: 7th test case of head node's x-axis acceleration between FE-simulation and LUNAR prediction from steering simulations

Table D.6: Error evaluation of 7th test case of head node's x-axis acceleration of steering simulation

	MSE $[(mm/ms^2)^2]$	RMSE $[mm/ms^2]$	MAE $[mm/ms^2]$
Overall	0.0010	0.0309	0.0124
Pre-crash	2.4726e-5	0.0050	0.0016
In-crash	0.0047	0.0688	0.0563

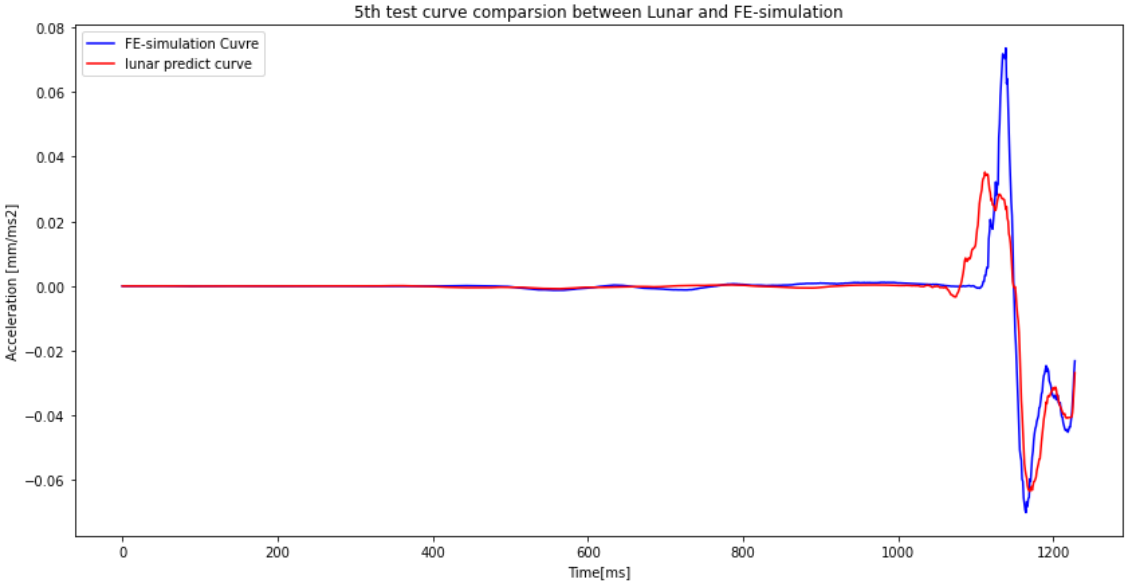


Figure D.7: 5th test case of head node's y-axis acceleration between FE-simulation and LUNAR prediction from steering simulations

Table D.7: Error evaluation of 5th test case of head node's y-axis acceleration of steering simulation

	MSE $[(mm/ms^2)^2]$	RMSE $[mm/ms^2]$	MAE $[mm/ms^2]$
Overall	3.8890e-5	0.0062	0.0020
Pre-crash	3.4898e-7	0.0006	0.004
In-crash	0.0003	0.0177	0.0131

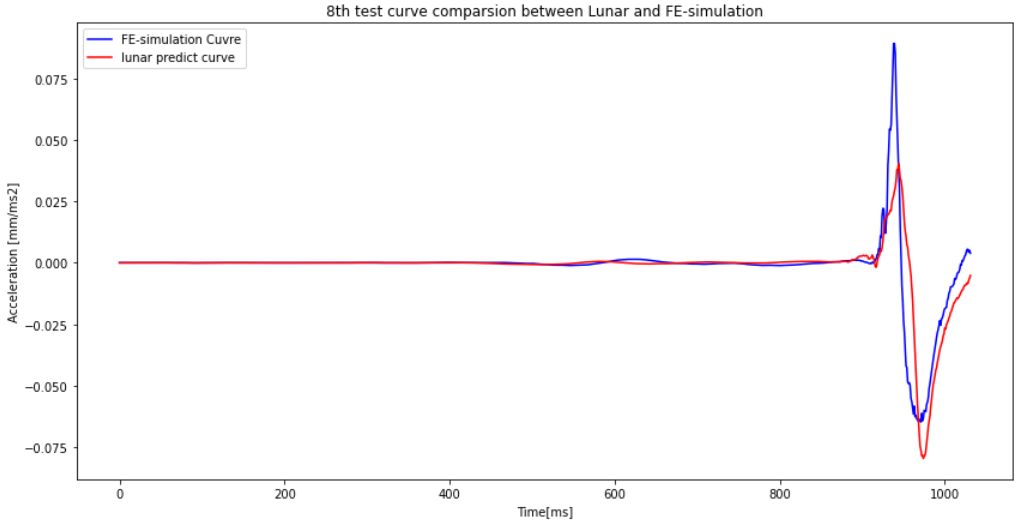


Figure D.8: 8th test case of head node's y-axis acceleration between FE-simulation and LUNAR prediction from steering simulations

Table D.8: Error evaluation of 8th test case of head node's y-axis acceleration of steering simulation

	MSE $[(mm/ms^2)^2]$	RMSE $[mm/ms^2]$	MAE $[mm/ms^2]$
Overall	6.5229e-5	0.0080	0.0024
Pre-crash	3.3187e-7	0.0006	0.004
In-crash	0.0004	0.0210	0.0143

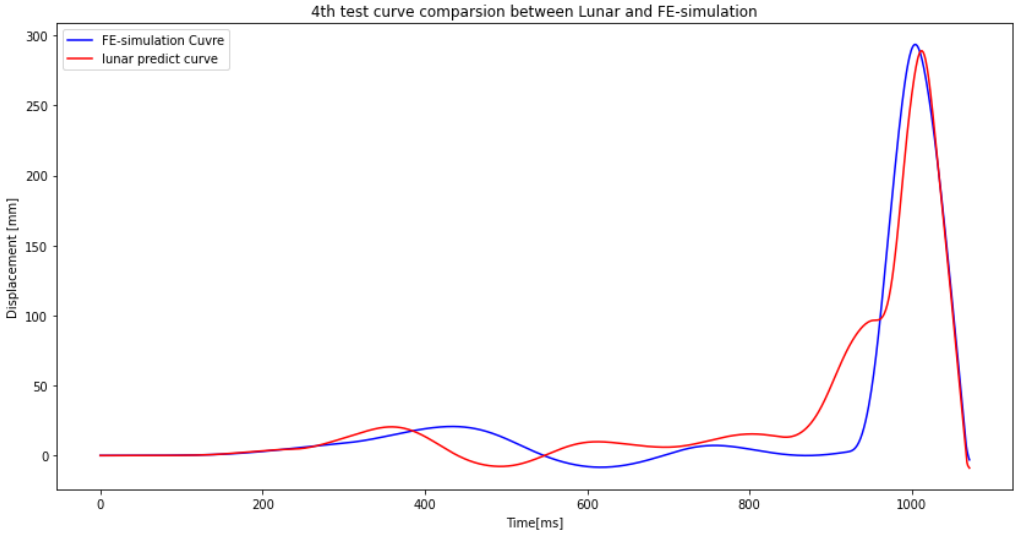


Figure D.9: 4th test case of head node's displacement between FE-simulation and LUNAR prediction from braking simulations

Table D.9: Error evaluation of 4th test case of head node's displacement of braking simulation

	MSE $[mm^2]$	RMSE $[mm]$	MAE $[mm]$
Overall	441.5603	21.0133	12.4170
Pre-crash	213.1241	14.5988	9.1679
In-crash	1824.2007	42.7107	32.0823

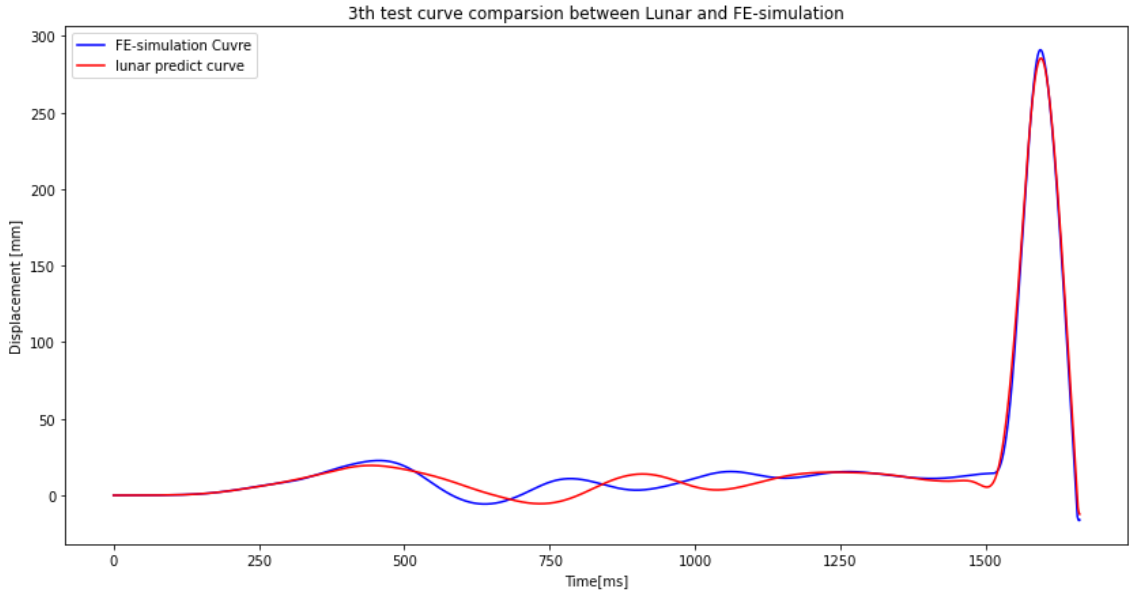


Figure D.10: 3rd test case of head node's displacement between FE-simulation and LUNAR prediction from braking simulations

Table D.10: Error evaluation of 3rd test case of head node's displacement of braking simulation

	MSE $[mm^2]$	RMSE $[mm]$	MAE $[mm]$
Overall	30.9684	5.5649	3.8025
Pre-crash	29.4382	5.4257	3.6005
In-crash	46.1803	6.7956	5.8111

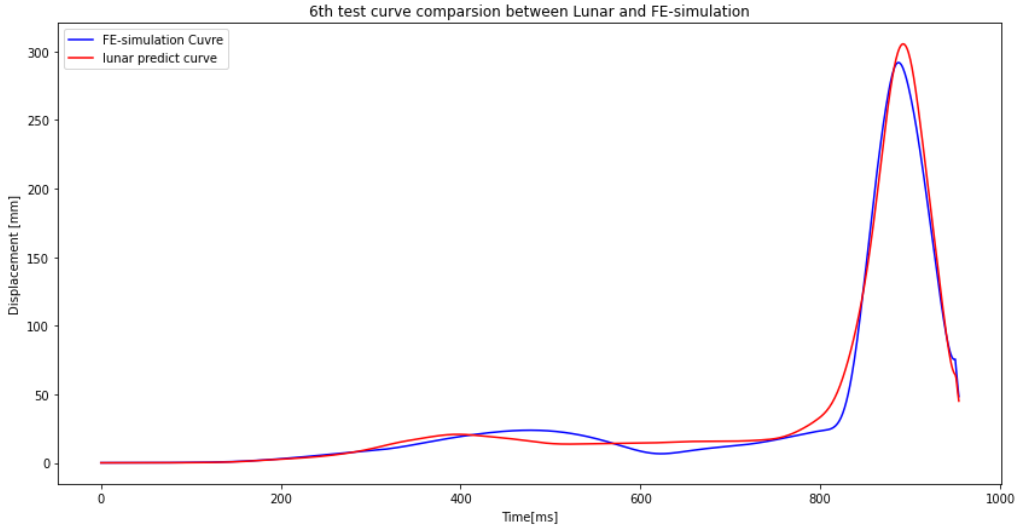


Figure D.11: 6th test case of head node's overall displacement between FE-simulation and LUNAR prediction from steering simulations

Table D.11: Error evaluation of 6th test case of head node's displacement of steering simulation

	MSE $[mm^2]$	RMSE $[mm]$	MAE $[mm]$
Overall	54.6362	7.3916	4.6712
Pre-crash	16.6075	4.0752	2.8840
In-crash	253.9764	15.9366	14.0392

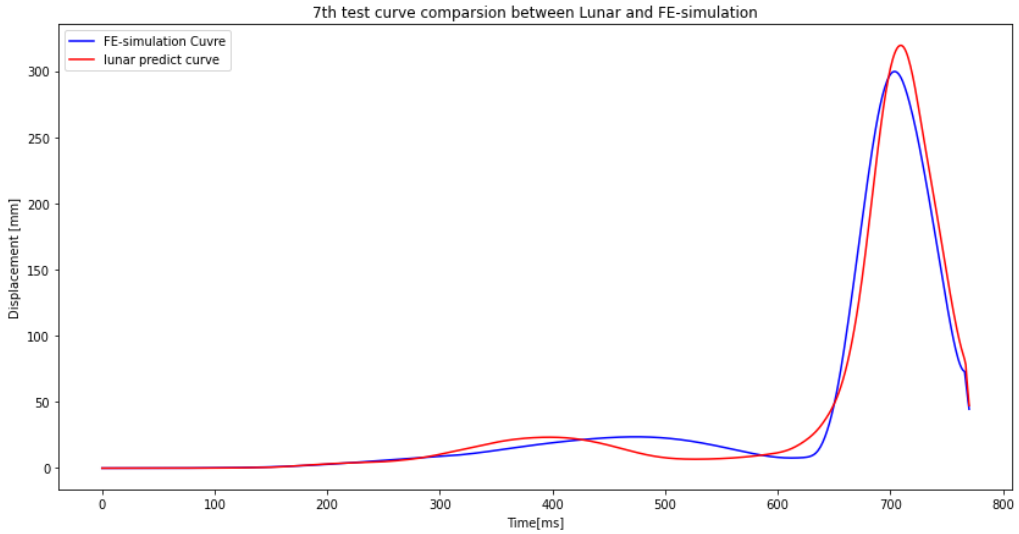


Figure D.12: 7th test case of head node's overall displacement between FE-simulation and LUNAR prediction from steering simulations

Table D.12: Error evaluation of 7th test case of head node's displacement of steering simulation

	MSE $[mm^2]$	RMSE $[mm]$	MAE $[mm]$
Overall	134.6177	11.6024	6.9867
Pre-crash	32.5226	5.7029	3.5620
In-crash	547.0018	23.3881	20.8201

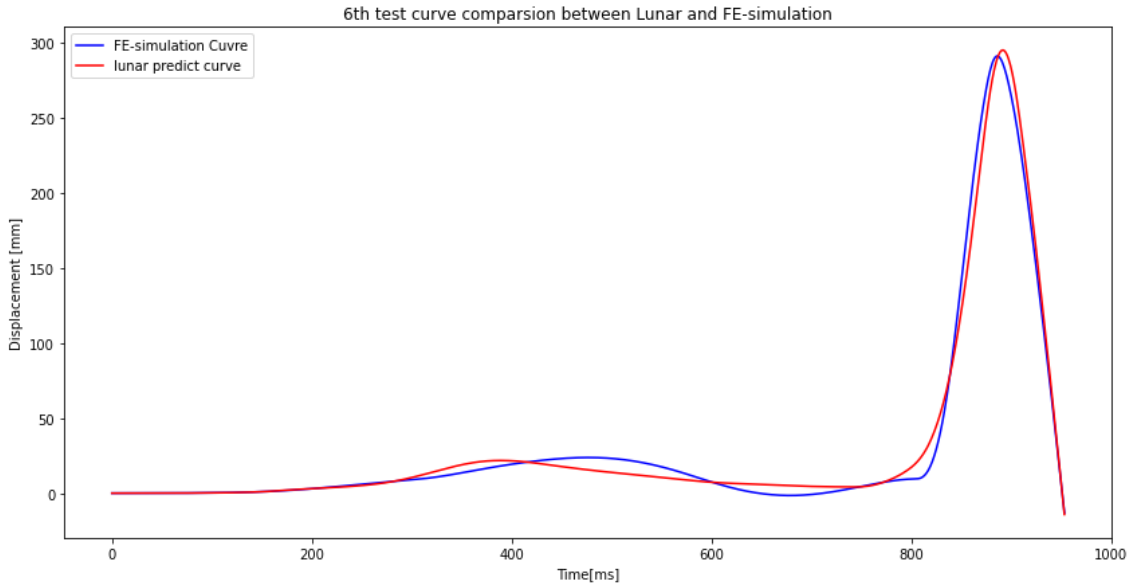


Figure D.13: 6th test case of head node's x-axis displacement between FE-simulation and LUNAR prediction from steering simulations

Table D.13: Error evaluation of 6th test case of head node's x-axis displacement of steering simulation

	MSE $[mm^2]$	RMSE $[mm]$	MAE $[mm]$
Overall	52.7161	7.2605	4.5831
Pre-crash	17.3869	4.1698	2.9244
In-crash	239.1243	15.4636	13.3351

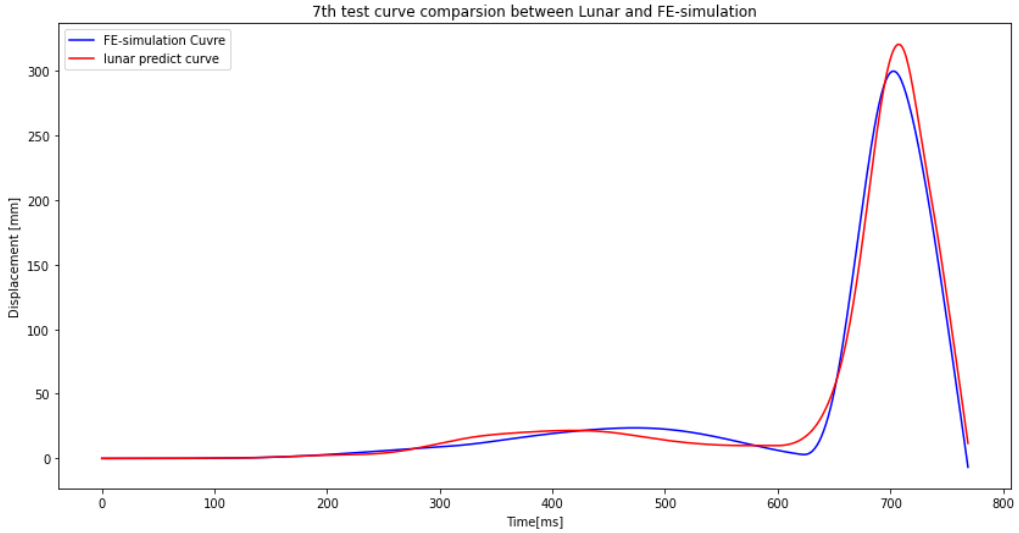


Figure D.14: 7th test case of head node's x-axis displacement between FE-simulation and LUNAR prediction from steering simulations

Table D.14: Error evaluation of 7th test case of head node's x-axis displacement of steering simulation

	MSE $[mm^2]$	RMSE $[mm]$	MAE $[mm]$
Overall	75.6581	8.6981	5.3099
Pre-crash	13.3997	3.6606	2.4752
In-crash	328.7877	18.1325	16.8357

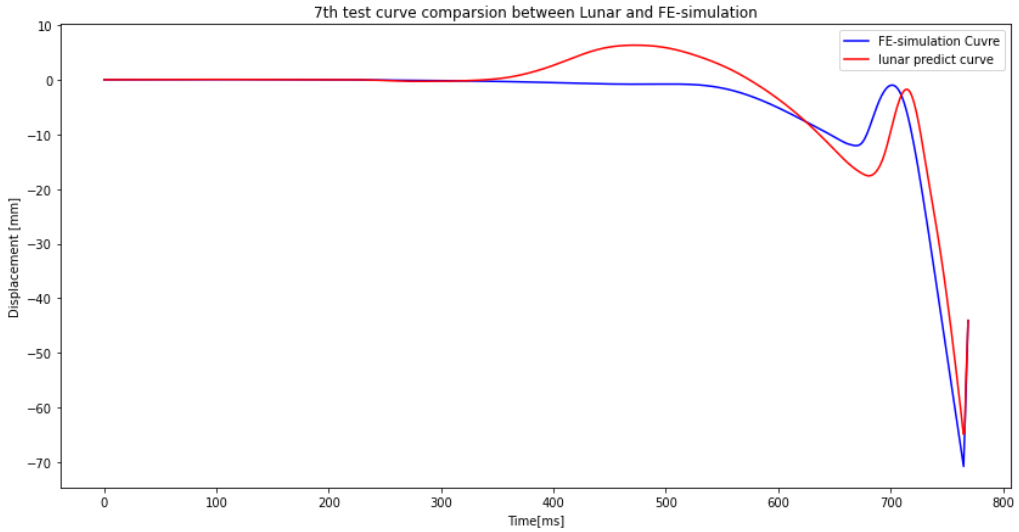


Figure D.15: 7th test case of head node's y-axis displacement between FE-simulation and LUNAR prediction from steering simulations

Table D.15: Error evaluation of 7th test case of head node's y-axis displacement of steering simulation

	MSE $[mm^2]$	RMSE $[mm]$	MAE $[mm]$
Overall	17.5028	4.1836	2.6285
Pre-crash	9.6674	3.1092	1.8434
In-crash	49.3601	7.0256	5.8205

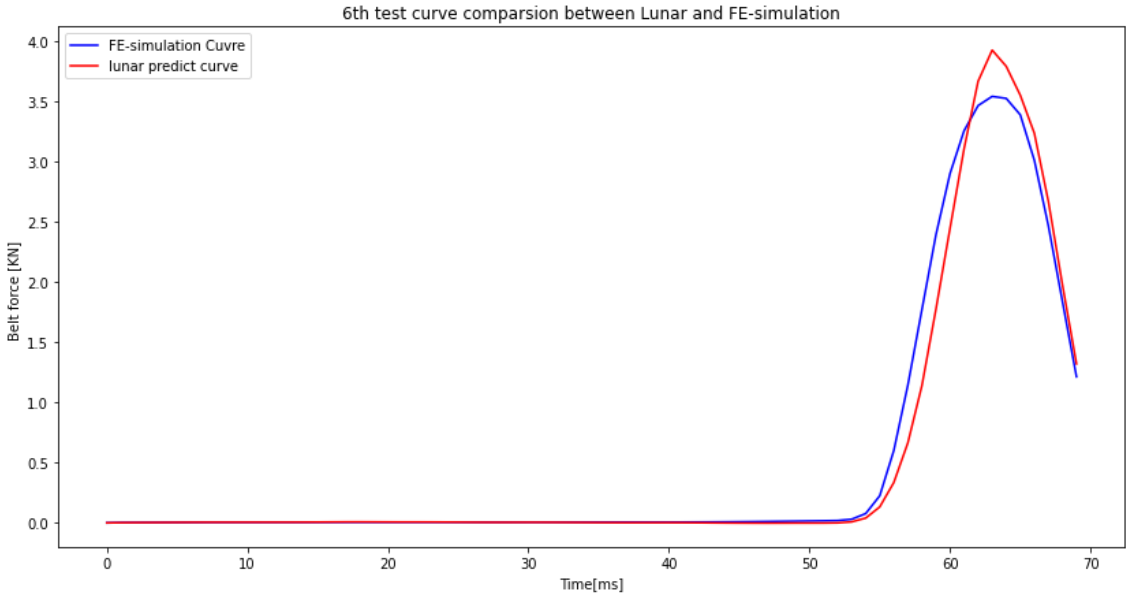


Figure D.16: 6th test case of belt force between FE-simulation and LUNAR prediction of braking simulations

Table D.16: Error evaluation of 6th test case of belt force of braking simulation

	MSE $[kN^2]$	RMSE $[kN]$	MAE $[kN]$
Overall	0.0247	0.1572	0.0663
Pre-crash	4.3038e-5	0.0066	0.0038
In-crash	0.1017	0.3189	0.2612

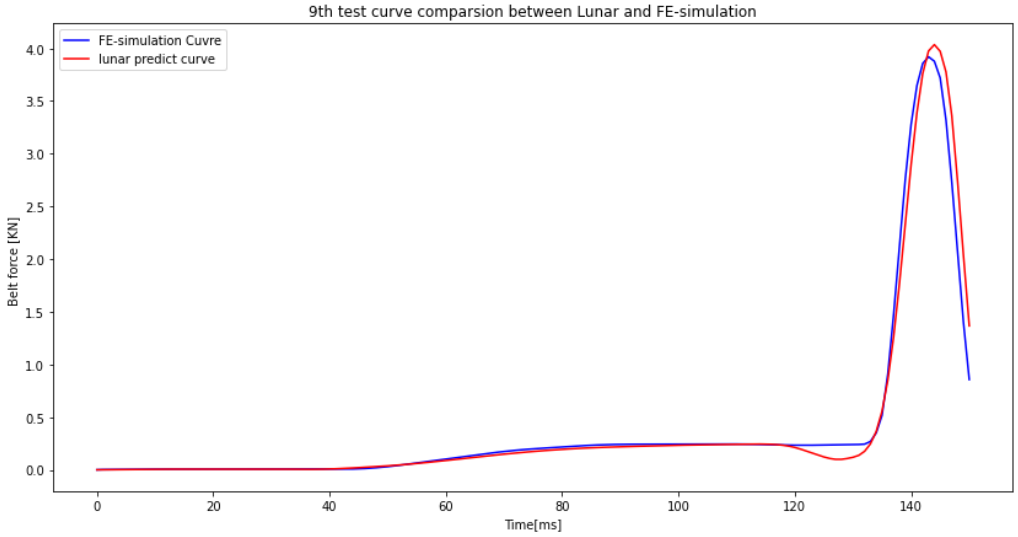


Figure D.17: 9th test case of belt force between FE-simulation and LUNAR prediction of braking simulations

Table D.17: Error evaluation of 9th test case of belt force of braking simulation

	MSE $[kN^2]$	RMSE $[kN]$	MAE $[kN]$
Overall	0.0169	0.1301	0.0501
Pre-crash	0.0012	0.3404	0.0176
In-crash	0.1413	0.3760	0.3071

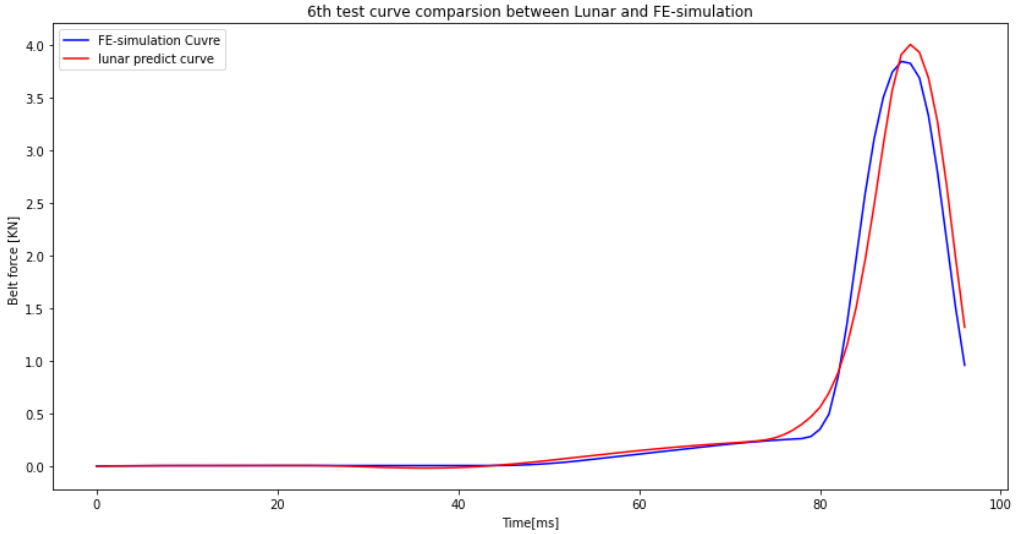


Figure D.18: 6th test case of belt force between FE-simulation and LUNAR prediction of steering simulations

Table D.18: Error evaluation of 6th test case of belt force of steering simulation

	MSE $[kN^2]$	RMSE $[kN]$	MAE $[kN]$
Overall	0.0262	0.1620	0.0737
Pre-crash	0.0011	0.0332	0.0183
In-crash	0.1446	0.3802	0.3350

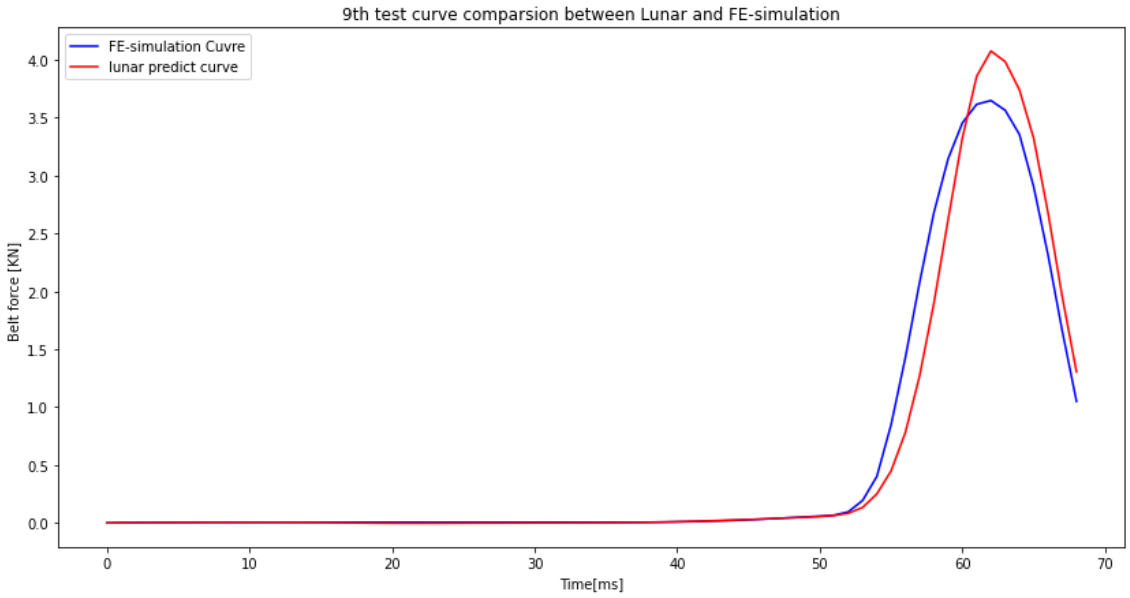


Figure D.19: 9th test case of belt force between FE-simulation and LUNAR prediction of steering simulations

Table D.19: Error evaluation of 9th test case of belt force of steering simulation

	MSE $[kN^2]$	RMSE $[kN]$	MAE $[kN]$
Overall	0.0462	0.2150	0.0939
Pre-crash	1.7395e-5	0.0042	0.0031
In-crash	0.1876	0.4331	0.3721

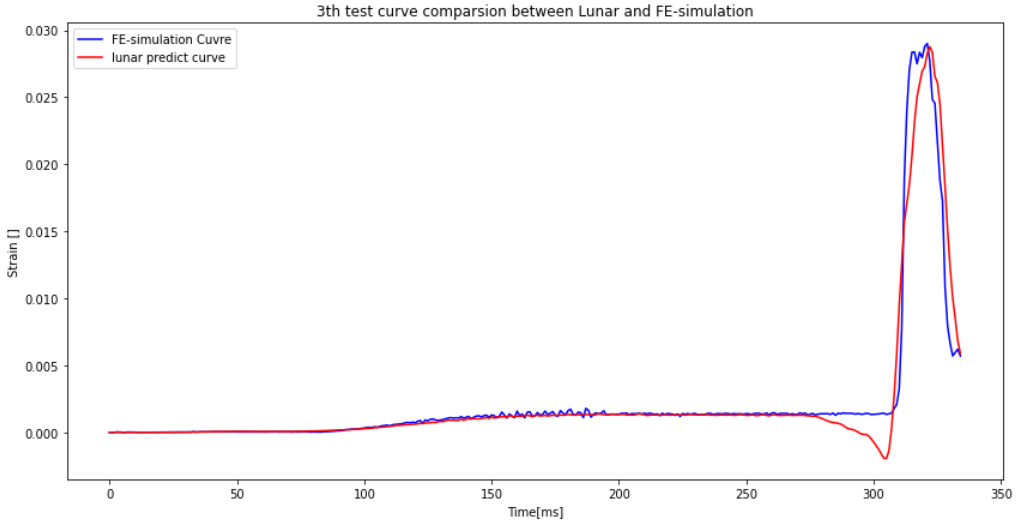


Figure D.20: 3rd test case of rib strain between FE-simulation and LUNAR prediction of steering simulations

Table D.20: Error evaluation of 3rd test case of rib strain of braking simulation

	MSE	RMSE	MAE
Overall	1.9403e-6	0.0014	0.0005
Pre-crash	1.3013e-7	0.0004	0.0002
In-crash	1.8506e-5	0.0043	0.0036

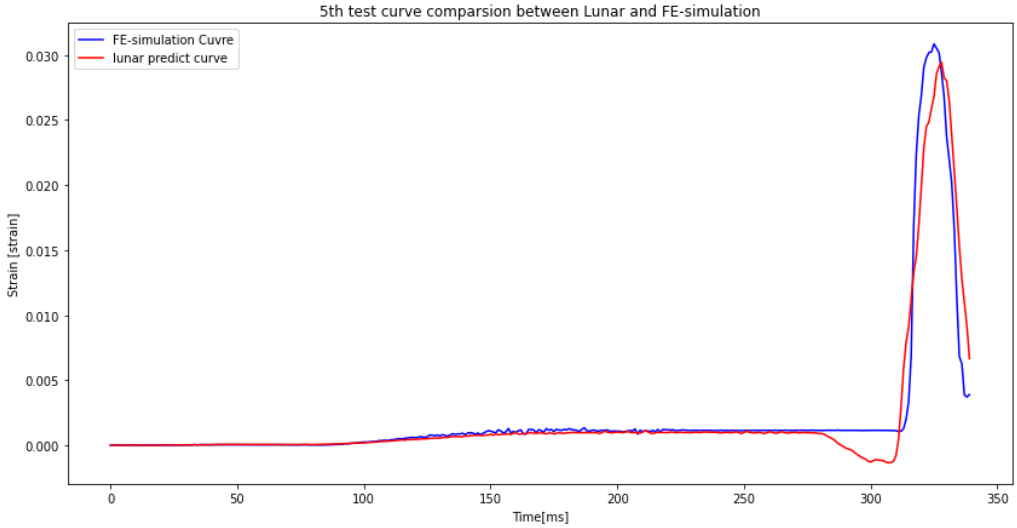


Figure D.21: 5th test case of rib strains between FE-simulation and LUNAR prediction of steering simulations

Table D.21: Error evaluation of 5th test case of rib strain of braking simulation

	MSE	RMSE	MAE
Overall	2.5669e-6	0.0016	0.0006
Pre-crash	2.7266e-7	0.0005	0.0002
In-crash	2.3911e-5	0.0049	0.0044

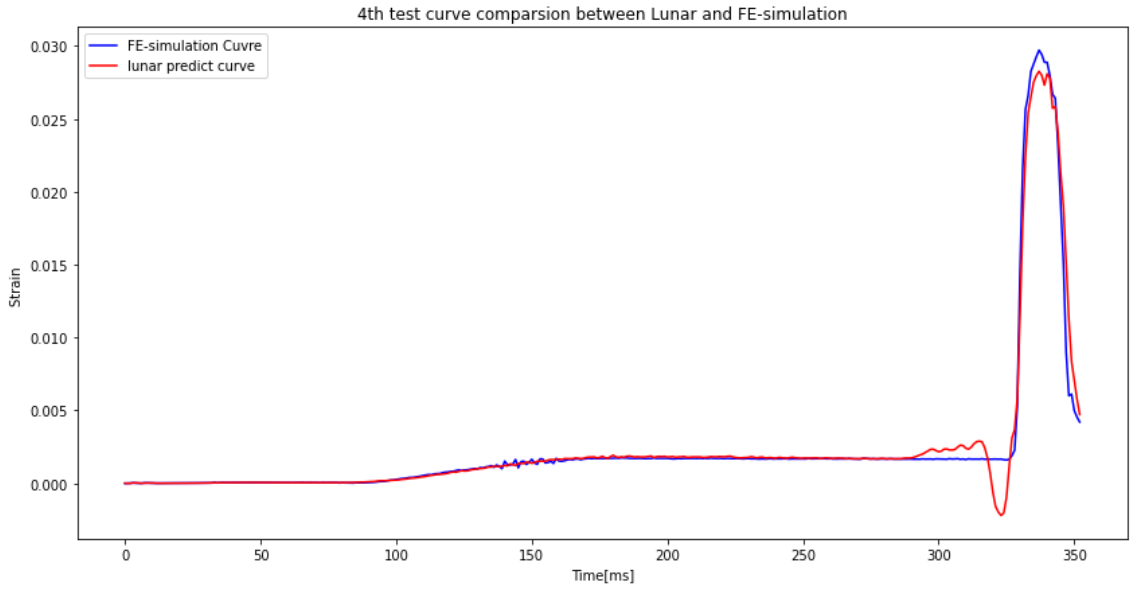


Figure D.22: 4th test case of rib strain between FE-simulation and LUNAR prediction of steering simulations

Table D.22: Error evaluation of 5th test case of rib strain of steering simulation

	MSE	RMSE	MAE
Overall	6.530e-07	0.0008	0.0003
Pre-crash	5.4987e-8	0.0002	0.0001
In-crash	6.4523e-6	0.0025	0.0021

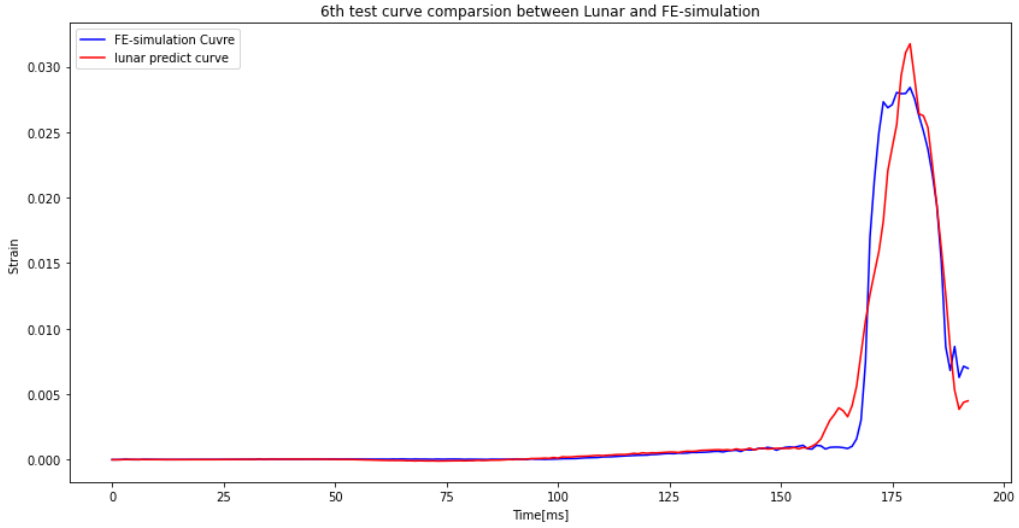


Figure D.23: 6th test case of strain between FE-simulation and LUNAR prediction of steering simulations

Table D.23: Error evaluation of 6th test case of rib strain of steering simulation

	MSE	RMSE	MAE
Overall	2.350e-06	0.0015	0.0006
Pre-crash	9.3768e-9	9.6834e-5	6.9520e-5
In-crash	1.3698e-5	0.0037	0.0030

DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY