GitHub repository: https://github.com/acse-2020/acse2020-acse9-finalreport-acse-tz3520

Imperial College London

Department of Earth Science and Engineering

MSc in Applied Computational Science and Engineering

Independent Research Project

Final Report

Computational Modelling of Pellet Strength and Prediction with Machine Learning

by

Tong Zheng

tz3520@imperial.ac.uk

GitHub login: acse-tz3520

Supervisors:

Dr. Ado Farsi

August 2021

GitHub repository: https://github.com/acse-2020/acse2020-acse9-finalreport-acse-tz3520

Abstract

The Finite Element Discrete Method is a widely used method in continuum mechanical simulations but requires advanced knowledge for the user to apply. This project would provide a user-friendly package to perform symmetric circular mesh generations and compression tests (to determine the maximum load) based on the Python3 Libraries Firedrake and GMSH. The results have been verified by the visualisation of the pellet simulation at critical breaking points. The dataset generated was used to train five models, including three linear models and two tree-based models. The R2 score has been used to measure the performance of the model, and the random forest was selected to be the final model that could predict maximum and minimum loading force under different conditions, given the pellet radius and the surface area defined. The model could be improved with more general data, such as having asymmetrical pellets in dataset generated.

Keywords:” Simulation”,” Pellet Strength Prediction”, “Finite Element Discrete Method”, “Random Forest”

Acknowledgement

First and foremost, I would like to thank my supervisor, Dr Ado Farsi, for his patience and willingness to help me throughout the project, from the detailed explanations of the strategies to such much advice on the project development. Thank you to the ACSE-9 teaching team for organising and supporting the project. Lastly, I would like to thank all my families and friends, whose support provided me an excellent environment to implement the project.

GitHub repository: https://github.com/acse-2020/acse2020-acse9-finalreport-acse-tz3520

Contents

GitHub repository: https://github.com/acse-2020/acse2020-acse9-finalreport-acse-tz3520

1 Introduction

1.1 Background

Catalysts are of great importance in manufacturing industries. They help save energy, make production quicker, easier, and more efficient, with less waste produced during manufacturing (Shafiq et al., 2020).

The catalyst support would be exposed to various conditions. Before being introduced into the reactors, the interactions such as collisions with the vehicle walls or the neighbouring pellet support could happen. During the reactions, the temperature and the pressure (therefore high energy) are also the possible factors leading to the support's breakage (Farsi et al., 2019). In order to reduce the potential cost in manufacture, the study on the strength of the catalyst support pellet is necessary.

Another essential factor of the catalyst is the surface area. The effect of catalyst (or the reaction rate) would be affected by the available contact area (Sun et al., 2018). To maximise the reaction rate and the heat transfer, the shapes of the pellet support are carefully inspected and purposefully designed, for example, to be cylindrical or spherical.

The previous study on the pellet strength has been carried out with 2D Finite Discrete Element Method (FDEM) (Munjiaza, A., 2004), the most widely used technique to model continuum mechanical problems. The key idea is to approximate the extensive particle system with plenty of infinitesimal elements, and the interaction of the elements could be investigated. The numerical simulations were only presented on two shapes, a solid one and one with four holes in previous work. The results of the simulation were remarkably agreed with the experimental data collected (Farsi et al., 2019).

The strength of the pellet is determined by the compression simulation. The circular pellet will be placed between two loading plates, one will be stationary, and the other will compress the pellet towards the other plate. Since the pellet is circular(cylindrical), the pellet will be expected to experience mode I failure, which is also known as the opening mode. This is the case when the tensile stress is normal to the plane of the crack (Irwin, G., 1957). Mode I failure parameter is the ratio between the first principal stress to the tensile stress. When the mode I failure parameter is greater than 1, the breakage of the material will be expected. This value will be determined further investigated in the simulation.

To determine the stress component during the compression test, the isotropic equation of linear elasticity can be used. The governing equation has the form:

$$\int \sigma \nabla v \, \mathrm{d}x - \int (\sigma \cdot \mathrm{n}) \cdot v \, \mathrm{d}s = \int f \cdot v \, \mathrm{d}x$$

with

$$\sigma = \lambda T_r(\varepsilon) + 2\mu\varepsilon$$

$$\varepsilon = \frac{1}{2}(\nabla u + (\nabla u)^T)$$

Where σ is the stress tensor, ε is the strain rate tensor, μ and λ are the Lame parameters, u is the unknown vector displacement field, and v is the finite elemene space(Slaughter, S., 2002). By solving this equation, the stress component could be determined, and the principal stress could then be calculated, which would then be used to compare with the tensile stress and check if the material will break with the conditions provided.

GitHub repository: https://github.com/acse-2020/acse2020-acse9-finalreport-acse-tz3520

1.2 Motivation

There are existing packages/libraries for FDE analysis. However, they do not have the function to simulate compression tests on particles or predict particle strength with given geometric feature. In addition, a certain level of relevant knowledge, including the mathematical and computational concepts, is required for the operation of these packages. The first aim of the project is to produce an easily accessible package for the mesh generation and the compression test simulation.

Even though the combination of the packages above could be used for numerical calculations and simulations, the potential time consumed to generate the results would significantly rely on the performance of the platforms/devices. Therefore, the second part of the project would concentrate on the model construction to make predictions as the reference for the geometric design. With the application of machine learning techniques and algorithms, predictions could be obtained more quickly.

2 Software Description

This package provides the users with a tool to create the simulation 2D meshes of the pellet, quantify the pellet's strength, and predict the strength with the input of the geometric parameters of the support.

2.1 Ecosystem & Usage Method

The two main libraries involved in the simulation modules are GMSH and Firedrake. GMSH is a finite element mesh generator with a built-in CAD engine, providing APIs for C++, Python and Julia (Geuzaine & Remacle, 2009). As for Firedrake, it provides an automated system to solve the partial differential equations using the finite element method (Rathgeber et al.,2016). Both packages could be operated in a Python environment. As stated in Section 1.2, they are very basic packages, a certain level of knowledge in geometric design and calculus is required for their operations. The software thus wraps these two packages, which provide the user with a simpler interface for the mesh generation and maximum loading calculations.

Depending on the input, the main executable would call different functions for different purposes, where the user could choose to use the whole package or some parts of it, including the maximum loading calculation and strength prediction.

The output of the mesh generation routine would be some msh files, which record the geometric features of the pellet, the physical group (for which some arcs would be assigned and the boundary conditions could be applied accordingly). The output files' names would be labelled with the geometric features (number of holes, for instance).

For the maximum and minimum loading calculation module, the mesh generated above would be the input. The output will be a csv file, which contains the table of material parameters. The plotting function is also enclosed in the maximum principal stress function. The contour for the maximum principal stress could be plotted, providing the supplementary method for verification and visualisation.

GitHub repository: https://github.com/acse-2020/acse2020-acse9-finalreport-acse-tz3520

The prediction module would require the input of the surface area and the pellet radius. All the possible combinations would be tested, and the one with the strongest structure would be returned and displayed.

2.2 Design of the Software

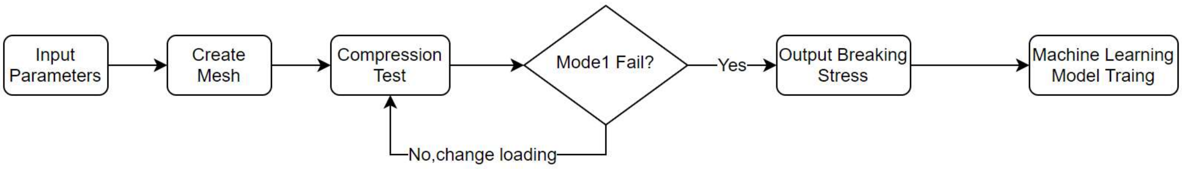

Figure 1: Simplified workflow of this project.

The simplified main workflow of the software is shown in Figure 1. Start by entering the values of the geometric features, the numbers would be passed to *create_mesh* function to generate a set of meshes with the given parameters. The generated mesh files will then be the input of the function compression_test. The corresponding maximum/minimum load and the mode one failure value would be determined and written in a table.

As the simulation completes, the resultant table would be loaded by the machine learning training module, and the data pre-processing and the model training will be carried on subsequently. Eventually, the model would be produced for the prediction of the strength of the pellet with the geometric parameters given.

2.3 Algorithm applied in the machine learning Module

Since the size of the dataset obtained is relatively small (only 4 features/parameters and less than 500 combinations of them), the complicated models were not considered during the algorithm selection, as their performance with too few features could be very sensitive to the model parameters setting (Fong et al., 2020). Therefore, three linear models: ordinary linear regression, lasso regression and ridge regression have been applied, with decision tree and random forest as the two tree-based models.

Linear Regression would fit the linear model with coefficients to minimise the sum of the squared residuals between the predicted values generated by the model and the observed value in the dataset, which is also known as the ordinary least square (OLS) method.

However, due to the limited size of the dataset prepared, the OLS method would have a greater chance for overfitting issues, where the trained model could not fit the new data well. The regularisation is therefore introduced to reduce the overfitting. For the Ridge Regression, a penalty term on the square of the coefficients would be imposed(L2-norm), and the goal of the model would then be minimising the penalised residual sum (Hilt & Donald,1997). Similarly, Lasso Regression would penalise the model with the sum of the coefficients, which is known as the L1-norm (Robert 1996).

As for the tree models, since they are nonlinear, they tend to be more prone to overfitting than the simple linear model, and this could also be controlled by tunings the parameters of the model, such as the maximum depth of the tree(s).

GitHub repository: https://github.com/acse-2020/acse2020-acse9-finalreport-acse-tz3520

2.4 Code Metadata

This package is developed in Python3, requiring the libraries GMSH, Firedrake, pandas, matplotlib and scikit-learn. The module was developed on both Ubuntu 18.04 and Windows 10 environments with a Windows Subsystem for Linux with Ubuntu 18.04 installation. The GMSH library would require the installation of libgLU1, libXrender1, libXcursor1, libXft2 and libXinerama1.

For the firedrake library, it has the following dependencies:

- A C and C++ compiler (for example gcc/g++ or clang), GNU make
- A Fortran compiler (for PETSc)
- Blas and Lapack
- Git, Mercurial
- Python version 3.8.x-3.6.x
- The Python headers
- autoconf, automake, libtool
- CMake
- zlib
- flex, bison
-

More information could be found on https://www.firedrakeproject.org/download.html

Name: Pellet Simulation and Strength Prediction

Version:1.2

GitHub & Documentation: https://github.com/acse-2020/acse2020-acse9-finalreport-acse-tz3520

3 Code Description and Implementation

The detailed implementation of code and the results produced would be explained and discussed in this section.

3.1 Simulation Module

The first function in the simulation module is the *create_mesh* function. This function would try to maximise the number of possible geometric arrangements for a given hole radius. The pellet radius is set to 10.

distance_hp stands for the distance between hole-centre and pellet-centre, *distance_hh* for the distance between adjacent two hole-centres, and the algorithm for the function could be represented by the pseudo-code below:

```
while True:
    while 1.5 * hole radius + distance_hp <= pellet radius:        (1)
            if 2.5 * hole radius <= distance_hh:                   (2)
                    create the mesh set                            (3)
            distance_hp += hole radius                             (4)
    hole number+= 1                                                (5)
    distance_hp = hole radius * 2                                  (6)
    while 2.5 * hole radius>= distance_hh:                         (7)
            distance_hp +=hole radius                              (8)
    if 1.5 * hole radius+ distance_hp >= pellet radius:            (9)
            break
```

The initial value of the distance between the hole centre and the pellet centre would be twice the hole radius to prevent the merge of the holes due to insufficient spacing between holes. As the codes run, there would be three checking points to validate the feasibility of the geometric arrangement. At the first checkpoint (1), the sum of the hole radius and the hole-pellet centre spacing would be compared with the pellet radius to ensure the holes are within the pellet. At the second checkpoint (2), the code would check if the spacing between two holes is sufficient, and then the set of the meshes will be created (3). The reason for a set rather than a single file is that the next function *compression* would simulate the process of the compression test in different directions. To reduce the calculation for the decomposition of the stress, the geometric orientation rather than the loading itself would rotate, in which way the loading would be downwards all the time. Running the simulation on meshes with the same geometric parameters but different orientation would be equivalent to the rotation of the pellet.

After completing the mesh set generations, the distance between the pellet and the hole centre would be increased by the value of the hole radius (4), which make the holes “move outwards”. The code will then return to checkpoint 1 to verify if the holes are still in the pellet. Once the holes could not move out anymore, the moving process will be stopped. The hole number would increase by 1(5), and the hole -pellet-centre spacing would be reset to twice the radius (6). If the spacing between the holes is insufficient, the holes will move out in the unit of the hole radius until the separation is enough (8). If the resultant sum of hole -pellet-centre spacing and the hole radius is greater than the pellet radius (9), this means the pellet could not have such number of holes, and therefore the function would stop.

GitHub repository: https://github.com/acse-2020/acse2020-acse9-finalreport-acse-tz3520

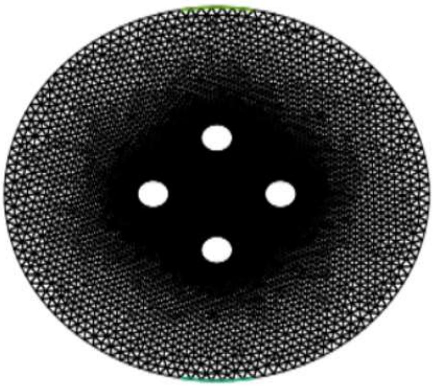

Figure 2: Demonstration on the physical group assigned. The plate will compress down on the green arc(top), and the blue arc will keep stationary(bottom). Noted this figure is not the actual scale as the contacting arcs should be extremely small (to approximate the contacting points).

In the second function *compression,* the material properties set for the pellet are $E_s = 40.5\text{GPa}, \nu_s = 0.17, \rho_s = 2310kg/m^2$, where E_s is the Young's modulus, ν_s is the Poisson's ratio and the ρ_s is the density(Farsi et al., 2017), and the corresponding shear modulus and strain could therefore be calculated. The sector contacting the loading plate is set to be 1 degree, as the compression should be acted on a point (the loading plate would be the tangent of the pellet). Two different boundary conditions are applied. For the bottom of the pellet, a Dirichlet boundary condition with both the horizontal and vertical element are set to be 0, to simulate the stationary lower plate in reality. While for the point at the top, a Neumann boundary condition would be introduced vertically downwards and no effect in the horizontal direction, simulating the upper loading plate compressing down(Alexander 1941). After solving the governing equation of the linear elasticity, the stress would be the output for this function.

The next function is *extract_stress_component*. With the application of principal stress equations, the maximum principal stress could be obtained with the horizontal and vertical components of the stress. Plotting is an optional method for this function, which would enable the plotting for the maximum stress contour and labelling the name automatically. At the end of the plotting, *plt.close("all")* would be used to clear the output preventing the program crashes due to lack of memory.

The maximum stress would then be the input of the next function *compute_failure_mode*, which would check if the pellet breaks. The pellet would be expected to experience mode one failure, where the tensile stress is normal to the plane of the crack. The tensile stress is set to be 5.07GPa(Farsi et al., 2019), and the mode 1 failure parameter would be calculated. If the pellet cracks, at least one of the elements would have a failure parameter greater than 1. The next function *checking_load_failure* will search throughout the whole mesh and will record the maximum failure parameters.

The main function of this module is *find_py*, which summing up the functions in this module. The mesh file would be the input, and the initial value for loading stress is set to be 10Pa by default. With this value, the function *compression, extract_stress_component* and *compute_failure_mode* would be called in order, and the mode 1 failure parameter would be checked. If the pellet does not crash with this loading stress, the stress would be multiplied by 10, and the checking process would be rerun. Eventually, the stress value would be obtained, which would be the upper bound for the actual critical stress value.

Search for the critical stress would be carried on with the binary search algorithm (also known as half-interval search algorithm). The initial lower bound is set to be 0. The midpoint between the two bounds would be taken for the checking process. If the pellet breaks with this test value, it indicates that the actual value should be somewhere between the lower bound and the midpoint, and the upper bound should then be updated to the midpoint value, and vice versa. The range of the search would be halved every time, and the searching would terminate if the difference between mode one failure parameter and 1 is smaller than the set tolerance (0.0001 by default), providing the failure parameter is greater than 1(condition for the breakage). The actual stress would be converted into force by calculating the product of the stress and the length of the arc contacted with the loading plate afterwards.

This module would finally generate a table with the hole numbers, the hole radius, the distance between the centre of the hole and the centre of the pellet, the loading force, the angle rotated and the failure parameters obtained in the test. This will then be aggregated, the geometric parameters, the maximum and minimum loading, together with their corresponding failure parameters would be generated as the final output with the pivotable function in the imported package *pandas*. Finally the module would outputs the table in a csv file.

3.2 Machine Learning

In this module, the data generated from the previous module would be the input, which would be pre-processed and be explored, followed by the performance tests on the five models stated. The package Scikit-learn is applied for data analysis and model training.

The perimeters (including the sum of perimeters for all the holes) and the ratio of pellet size to hole size would be calculated. These two parameters would be the features involved in the model training. The reason for defining the perimeter as “surface area” is, the simulation is operating in the 2D plane, while in 3D, the surface area would be the product of the length of the pellet and the perimeter, the surface area is therefore directly proportional to the perimeter.

The correlations between the maximum/minimum load and the other parameters would then be computed, which would help in the feature selections. Even though the surface area and the pellet radius would be the only input of the model, the importance of the other features could still be determined.

Standardisation on the variable would increase the speed of the gradient descent and rescale the variables on the same scale, preventing certain variables (with large variance) dominating the model. Therefore standardisation would be processed following the feature selections. Noted that the tree-based models would not be significantly affected by this, as the positions of the node in the tree are determined by the entropy rather than the scale or the gradient descent (Witten et al 2011).

Since both the Lasso and Ridge Regression have the penalty parameter alpha, adjusting alpha could optimise the performance. R2 score is applied to measure the performance for the model fitting the distribution of the observed data. The closer the R2 score to 1, the better the model fitting the data. Therefore, the function *LassoCV* and *RidgeCV* would apply the cross-validation for searching alphas over a range of values, and the ones with the highest R2 score would be selected. The advantage of cross-validation is that every data point would get to be in the validation set exactly once, which would reduce the bias.

GitHub repository: https://github.com/acse-2020/acse2020-acse9-finalreport-acse-tz3520

For the tree-based models, the method applied for hyperparameters tuning (thus optimising the model) is the grid search method. By providing a range of the parameters, the function *GridSearchCV* will try all the combinations of the hyperparameters and the cross-validation would be applied again. Even though this would take a longer time (compared to the other method, *RandomizedSearchCV,* for instance), it would provide the best combination of the hyperparameters and therefore the best model.

At the end of the module, the R2 scores for different models would be compared, and the best one would be selected. The model would then be saved as the final model for the prediction.

As the model trained could only predict the value of a single feature, two models would be trained for maximum and minimum load respectively.

For the prediction functionality, the input would be the surface area and the pellet radius. Similar to the algorithm in Section 3.1, all the possible combinations of the geometric features would be evaluated, and the one with the highest score would then be outputted.

3.3 Example workflow

The code could be run as below:

```
$ python3 simulation.py

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

Please select the functionality:

1.  Mesh Generation

2.  Mesh Generation and Maximum Loading Calculation

3.  Maximum and Minimum Loading Prediction

4.  Quit
```

In the above example, the user would be able to choose the specific functionality by entering the number. If the “Mesh Generation” is chosen, the hole radius and the pellet radius will then be required to be entered to start the mesh generation. The output directory would be the same folder of the python scripts. For option 2, the function in option 1 would be combined with the calculation modules, which would simulate the compression tests and calculate the strength. Option 3 will provide the user with the predicted maximum and minimum load where the surface area and the radius of the support are given by the user if the input is applicable. Option 4 will provide the user a way to terminate the program.

3.4 Results

Series of msh files, as well as the plotting for the critical point of the pellet breakage were produced. Boundary conditions and stress could be observed in Figure 3.

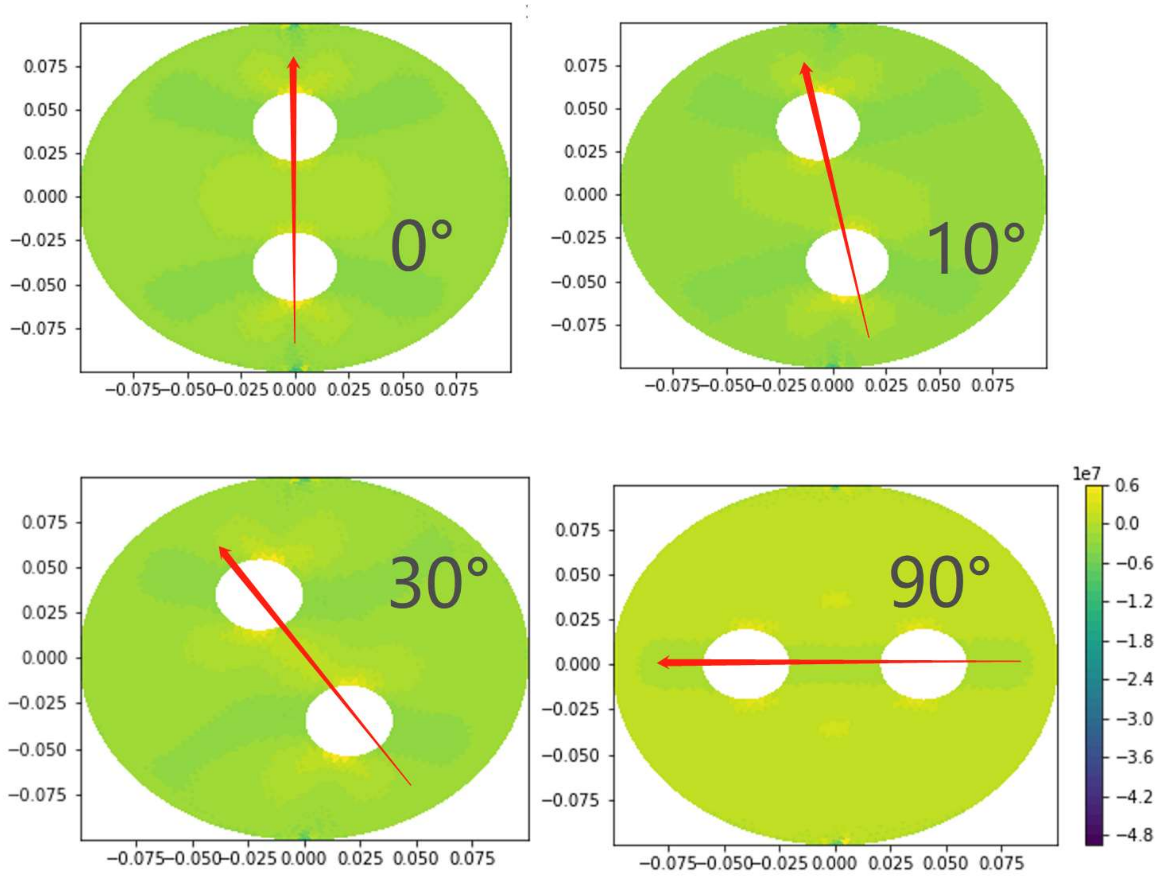

Figure 3: Meshes with the same geometric features but different orientations were generated to simulate the pellet's rotations for the compression test in different directions. The angles of rotation are shown labelled on corresponding plots.

The highest stress (blue points) could be observed on the two vertical ends, which verify that the stress generated is the maximum principal stress. The failure parameters were all found to be smaller than 1.0001, proving the algorithm to control the failure parameters within the tolerance is valid. A table for all the possible parameter combinations was produced as a csv file. The stress appears to be higher when the orientation of the two holes is 90 degrees, compared to 0 degrees, for instance (greener). Therefore the latter will have a stronger resistance to the compression, and therefore the maximum loading would be greater than the former.

GitHub repository: https://github.com/acse-2020/acse2020-acse9-finalreport-acse-tz3520

	Maximum Loading	Minimum Loading
Linear Regression	0.4264	0.7622
Ridge Regression	0.4588	0.7684
Lasso Regression	0.4264	0.7623
Decision Tree	0.7369	0.8499
Random Forest	0.7948	0.9260

Table 1: R2 scores for maximum and minimum loading for different algorithms tuned for the best performance.

The R2 score results for maximum loading obtained by the 5 model predictions are shown in Table 1. The parameters used in training are all five parameters: hole radius, hole number, pellet hole ratio, surface area and spacing between the hole and pellet centre. The tree-based model performs better than linear models, while Random Forest performs better than the others as it prevents the overfitting by “random”. Therefore, the two random forest models (one for maximum and one for minimum loading) would be saved as the final models for predictions.

3.5 Documentation and Testing

The documentation would be found in the GitHub repository, with the detailed information of the pre-requisites and the installation instructions, and the user guide for the different purposes/functions.

Simple unit tests for the simulation module could also be found in the repository, where the testing for the machine learning model is applicable, as the R2 score has been applied to evaluate the model in section 3.4.

GitHub repository: https://github.com/acse-2020/acse2020-acse9-finalreport-acse-tz3520

4 Discussion

4.1 Evaluation

The first part of the simulation successfully achieved the aim of generating all the possible meshes with the given geometric parameters, and simplifying the maximum loading force calculation, which appreciably reduces the user's effect on understanding the mathematical and computational work. The models trained in the second part show an R2 score of 0.7948 for the maximum load and 0.9260 for the minimum load.

The most difficult parts of the module development would be studying the libraries and the Ubuntu system. Since the libraries are not commonly used compared to the popular libraries, the information that could be found was relatively limited. As a result, the HPC support team could only provide insufficient aid, which takes extra time to run the deployment and code.

4.2 Limitation & Future Work

The accuracy of the prediction made by the model could still be improved by either adding more parameters or generating a larger dataset. Therefore, more complicated models could be trained and tested.

Meshes generated in the package were all symmetric. The module could be extended to generate random shapes of the pellet. The function calculating maximum principal stress would still be applicable for the change. Thus, a more general dataset could be obtained, which would also help improve the model if trained with the additional data.

The package still requires two uncommon libraries, whose requirements for installation are relatively strict. Particularly, the Firedrake requires the PETSc and h5py built-up, whose installations easily crashed due to the network or the platform infrastructure.

5 Conclusion

This software was developed to simplify the calculation processes and provide references for the geometric design of the catalyst support. By wrapping up several functions and libraries, the software could perform mesh generations, maximum loading calculations and predictions of the maximum and minimum load for given geometric parameters. The example output of the functions, including the msh file, the visualisation for critical stress on the pellet and the output csv file recording the compression test results could be found in the GitHub repository. The predictions for the maximum and minimum load have R2 scores of 0.7948 and 0.9260, proving the models could fit the observations reasonably.

Multiple improvements could be made to the package in the future. For example, the program could be extended to generate asymmetric meshes, which would provide a more general dataset, thus improving the model trained. Additionally, the simulation and the mesh generation modules reply on two uncommon libraries, which would increase the uncertainty in the deployment. Substitution on the libraries relied on could be attempted in the future.

GitHub repository: https://github.com/acse-2020/acse2020-acse9-finalreport-acse-tz3520

6 Reference

Farsi, A., Xiang, J., Carlsson, M., Stitt, E., Marigo, M., 2019. Strength and fragmentation behaviour of complex-shaped catalyst pellets: A numerical and experimental study, pp. 1-2.

Fong, S., Li,G., Dey,N., Crespo, R., Herrera-Viedma, E., 2020, Finding an Accurate Early Forecasting Model from Small Dataset: A Case of 2019-nCoV Novel Coronavirus Outbreak, pp. 2.

Steel, G., Torrie, H.,1960, Pinciples and Procedures of Statistics with Special Reference to Biological Sciences,pp. 172-173.

Chicco, D., Warrens, J., Jurman, G., 2021, The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation, pp. 1–24.

Oñate, E., Rojek, J., Combination of discrete element and finite element methods for dynamic analysis of geomechanics problems, 2004, Comput. Methods Appl. Mech. Engrg., Vol 193, 3087-3128.

Munjiaza, A., 2004, The Combined Finite-Discrete Element Method, Wiley, 2004.

Barbosa, R., Ghaboussi, J., 1992, Discrete Finite Element Method, Engineering Computations, Vol. 9 No. 2, pp. 253-266.

Sun, S., Li,H., Xu, Z., Impact of Surface Area in Evaluation of Catalyst Activity, Joule 2, pp. 1019-1027.

Geuzaine, C., Remacle, J., 2009, GMSH: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities, International Journal for Numerical Methods in Engineering 79(11), pp. 1309-1331.

Rathgeber, F., Ham, D., Mitchell, L., Lange, M., Luporini, F., Mcrae, A., Bercea, G., Markall, G., Kelly, P.,2016, Firedrake:automating the finite element method by composing abstractions, ACM Trans. Math. Softw., 34(3):24:1-24:27.

Hilt, D., Seegrist, D., 1997, 'Ridge, a computer program for calculating ridge regression estimates', pp. 1-4.

Robert, T, 1996, Regression Shrinkage and Selection via the Lasso, Journal of the Royal Statistical Society, Series B(methodological). Wiley.58(1):267-88.

Witten, I., Frank E., Hall, A., 2011, Data Mining:practical machine learning tools and techniques, pp. 102-103.

Harris, R., Millan, J., van der Walt, J., et al., 2020, Array programming with NumPy. Nature 585, 357-362.

The pandas development team, 2020, pandas-dev/pandas: Pandas, Zenodo, https://pandas.pydata.org/

GitHub repository: https://github.com/acse-2020/acse2020-acse9-finalreport-acse-tz3520

Hunter, D., 2007, Matplotlib: A 2D graphics environment, Computing in Science \& Engineering, pp. 90-95.

Pedregosa, F., Varoquaux, G., et al., 2011, Scikit-learn: Machine Learning in Python, Journal of Machine Learning Research,2825-2830.

Waskom, M.,2021, seaborn: statistical data visualization, Journal of Open Source Software, pp. 3021.

Irwin, R., 1957, Analysis of Stresses and Strains Near the End of a Crack Traversing a Plate, Journal of Applied Mechanics, Vol. 24, pp. 361-364.

Slaughter, S., 2002, The linearized theory of elasticity, Birkhauser.