1 Quick use

- Open commander_example.py in PyCharm;
- Edit the working directory and run it;
- Get the results in your working directory.

2 Structure of the programme and details of each *.py file

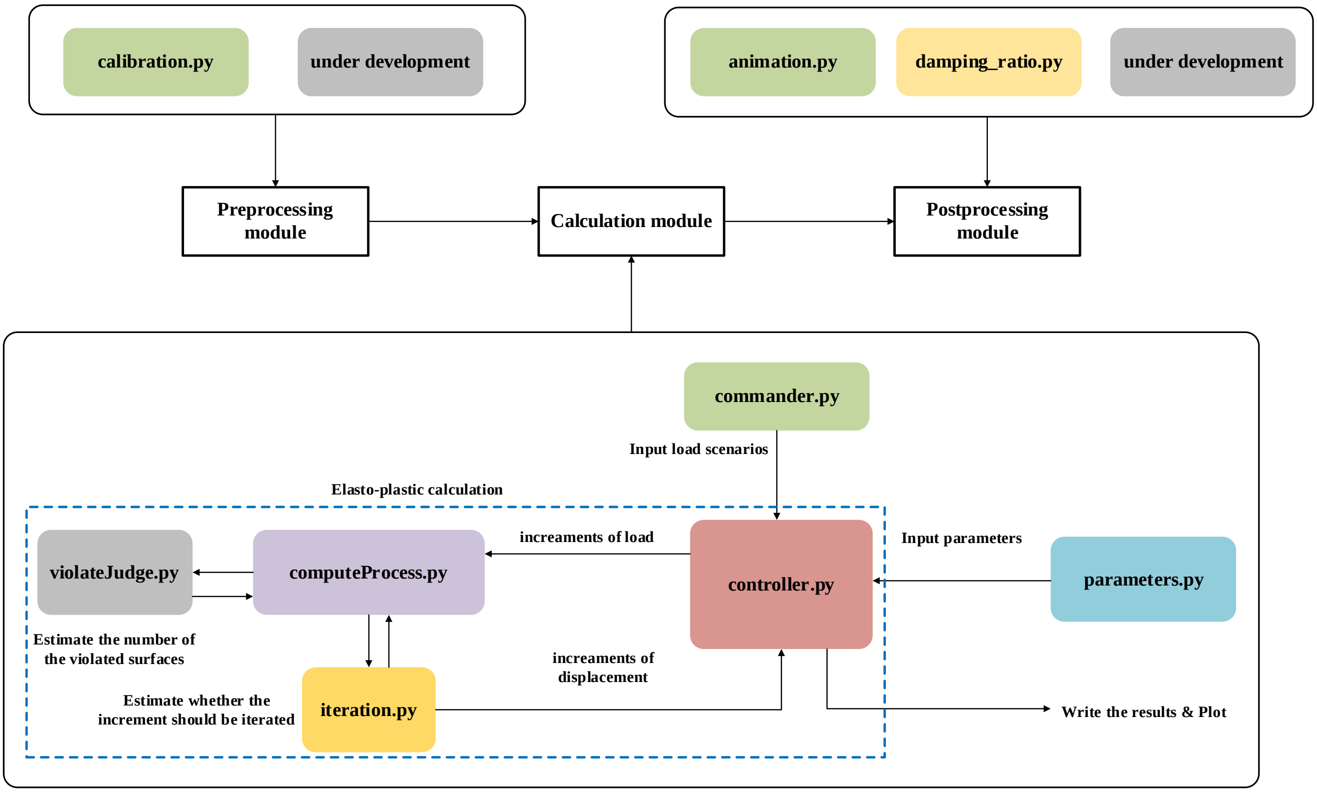

Fig. 1 Workflow of the programme

This project includes the preprocessing module, postprocessing module and the calculating module, as shown in Fig. 1. The preprocessing and postprocessing modules include the programmes of parameters calibration, animation generation, and the processes of the results (for example, damping ratios), more features may be developed in the future. The main body of the project is the calculating module, which consists of 6 *.py files: “commander.py”; “controller.py”; “computeProcess.py”; “iteration.py”; “parameters.py” and “violateJudge.py”. These 6 files work together through exchanging variables during the calculation. The core of the calculation module is “controller.py”, which conducts the calculation process, records the results, and plots the figures. The load scenarios are input by “commander.py”, and the parameters of the model are input by “parameters.py”. The elasto-plastic calculation is conducted by “controller.py” where the formulations of the derivations are provided by “computeProcess.py”. The “violateJudge.py” and “iteration.py” is used in the elasto-plastic calculation, estimating how many surfaces violated and whether the increment is converged, respectively.

2.1 commander.py

“commander.py” is where users input the load scenario and the working directory. Users can edit the variables “Fcy_V_Fa”, “F_average” and “F_cyclic” to change the average and cyclic loading components. The number of cycles of cyclic loading is also defined in this file. Details can be found in the annotation of the *.py files.

2.2 computeProcess.py

“computeProcess.py” is the formulation of a specific model, including the elastic and plastic stiffness matrixes, yield surfaces, derivations of yield surfaces, etc. Users can define new model by changing the formulations of necessary items.

2.3 controller.py

“controller.py” is the core of the calculation process. A calculation process includes:

(i) Execute the “ini” function to create the basic variables and obtain the parameters;

(ii) Execute the “load_Fcontrol_Uclose” function, where the input load is divided into a series of increments and elasto-plastic calculations is conducted using the formulations in “computeProcess.py”. The displacement increment is obtained by the elasto-plastic calculation**.**

(iii) For cyclic calculation, parameters are updated before the next load cycle, by executing “update_with_N” function, where the yield surfaces are contracted to simulate the soil softening.

(iv) During the calculation, the load-displacement relationships are recorded, and the movements of the yield surfaces are plotted.

2.4 iteration.py

This file is to estimate whether the last increment is converged, by the distance between the current load point and the yield surfaces. Users can test the threshold value by observe the divergence between the load point and the yield surfaces, and a value for indistinguishable divergence is expected, to balance the computational cost and accuracy.

2.5 parameters.py

The parameters of the yield surfaces are input in this file by a series of lists. The lists are generated by the calibration feature in the preprocessing module. The list is in order of the number of cycles, detailed the annotation of the *.py files.

2.6 violateJudge.py

This file is to estimate how many yield surfaces are violated by the load point, by substitute the load point into the formulations of yield surfaces. The yield surface is deemed as violated as the value is in a small range inside the surface, and the default value is set as 0.1 and it can be modified by users.

3 What should I do to establish a custom model?

To build a new model, the formulations of plasticity may be deducted by the user, i.e., the formulations in “computeProcess.py”. And the parameters in “parameters.py” should be re-calibrated. Some small changes may need in other *.py files to accommodate the new model established. For more details of the rationale of the model, see the papers listed in the references.

4 References

Houlsby, G. T., Abadie, C. N., Beuckelaers, W. J. A. P., & Byrne, B. W. (2017). A model for nonlinear hysteretic and ratcheting behaviour. International Journal of Solids and Structures, 120, 67-80.

Iwan, W. D. (1967). On a class of models for the yielding behavior of continuous and composite systems. J. Appl. Mech. 34, 612-617.

Page, A. M., Grimstad, G., Eiksund, G. R., Jostad, H. P. (2018). A macro-element pile foundation model for integrated analyses of monopile-based offshore wind turbines. Ocean Engineering, 167:23–35.

Page, A. M., Skau, K. S., Jostad, H. P., & Eiksund, G. R. (2017). A new foundation model for integrated analyses of monopile-based offshore wind turbines. Energy Procedia, 137, 100-107.

Skau, K. S., Grimstad, G., Page, A. M., Eiksund, G. R., & Jostad, H. P. (2018). A macro-element for integrated time domain analyses representing bucket foundations for offshore wind turbines. Marine Structures, 59, 158-178.

Zhang, C., Wang, D., & Zheng, J. (2023). A cyclic-softening macro element model for mono-bucket foundations supporting offshore wind turbines in clay. Computers and Geotechnics, 161, 105603.