rvesimulator: An automated representative volume element simulator for data-driven material discovery

Jiaxiang Yi
Department of Material Science and Engineering
Delft University of Technology
Mekelweg 2,Delft, 2628 CD, the Netherlands
J.Yi@tudelft.nl

Miguel Bessa
School of Engineering
Brown University
184 Hope St.,Providence, RI 02912, USA
miguel_bessa@brown.edu

Abstract

The *rvesimulator* aims to provide a user-friendly, automated Python-based framework conducting Representative Volume Element (RVE) simulation via powerful Finite Element Method (FEM) software Abaqus. By utilizing this repository, large amount of reliable FEM data-set generation is possible with RVEs encompassing materials from elastic to plastic composites . *rvesimulator* provides: (1) a cross-platform function to run arbitrary Python-Abaqus script without graphical user interface (GUI), it offers users a convenience way to run their unique scripts; (2) Python-Abaqus scripts to simulate RVE with different design of experiments including various micro-structures, material laws, and loading; (3) benchmarks of running prevalent RVEs covering elastic, hyper-elastic, plastic materials are provided, which illustrates the general pipeline (preprocess, execution, and postprocess) of the developed framework; By sharing the developing framework, we are aiming to reduce the labor-intensive process of generating massive of simulations data for new materials and structure discovery. Therefore, it facilitates the application and development of machine learning method for new material discovery. More details about the *rvesimulator* can be referred to the GitHub repository: `https://github.com/bessagroup/rvesimulator.git`

1 Introduction

The pursuit of discovering new materials remains the ultimate goal for materials scientists. Wing kam Liu [1] reviewed the development of Finite Element Method (FEM) over the past eighty years, and claimed that Neural Networks (NNs) based on FEM drives to new avenues for discovering new material with unprecedented properties. Recently, advanced machine learning technologies such as Multi Layer Perceptrons (MLP)[2], Convolution Neural Network (CNN) [3], Graph Neural Network (GNN)[4], Recurrent Neural Network (RNN)[5], etc. are utilized for learning and finding better material properties. Large high-fidelity data-set is expected for training well performed aforementioned machine learning models, whereas the process of generating such data is notably absent from the existing literature. There are several open-source FEM packages, such as Fenics[6], JAX-FEM[7], however their capability for high nonlinear and complex material like composite is limited. Abaqus [8], renowned for its powerful nonlinear solver capable of handling intricate and large-scale simulations, is prevalent among material science researchers. However, GUI operation still is the common way to obtain data-set for researchers who are using Abaqus even though it is a repeatable process.

The *rvesimulator* is a flexible and user-friendly repository based on Python, focusing exclusively on RVE simulations while filtering out other functionalities of ABAQUS, such as contact and thermal analyses. With the developed framework, the repeatable processes such as geometry modeling,

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

periodical boundary condition, job submission, and postpocess are already coded. For the potential design variables which are mirco-structures, material models, and loading; the user can use design of experiments (DoE) methods to draw their DoE. Then, the DoE can be regard as the arguments to the *rvesimulator* to run simulations automatically. By introduce this ease to use repository, we hope to narrow of gap of machine learning and new material design.

2 General workflow

The schematic of the developing *rvesimulator* is shown in Fig.1, which comprises two important modules. The first module is encapsulated as functions and could be seamlessly embedded into the data-driven material design and optimization process. The other is Python-Abaqus scripts for executing RVE simulations. Two modules only communicate with *'AbaquSimulator'* object that handle the input and output data flow. In the **Master control** module, the essential configurations like micro-structure information, material model parameters, and loading are set and then pass to *'AbaquSimulator'*. Within this module, a Monte Carlo algorithm is implemented [9] to generate the above depicted micro-structures Fig.1 such that users can generate desired RVE micro-structure with certain volume fraction and particle distribution. Further details about will be illustrated in Section 3 along with concrete examples. Additional information of *'AbaquSimulator'* object and Python-Abaqus scripts will be presented in the following subsections.

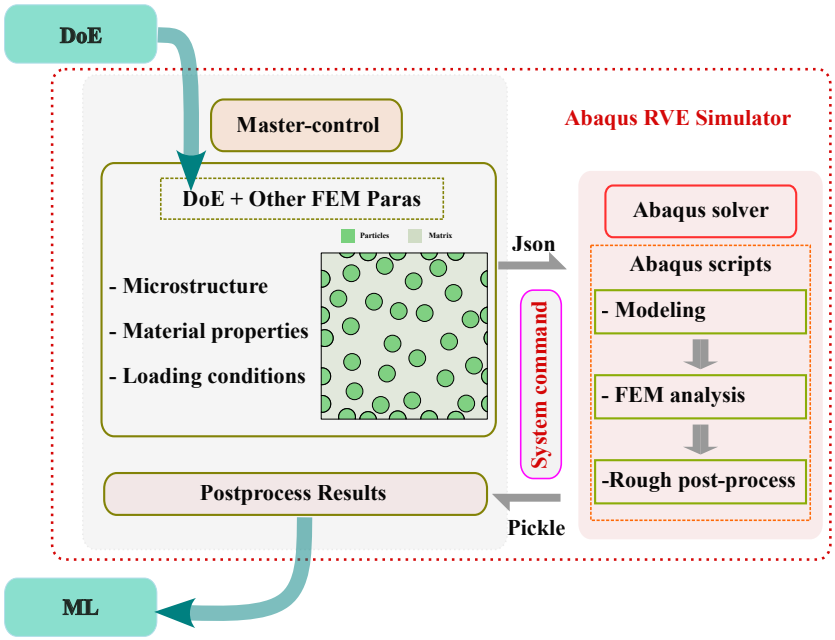

Figure 1: Workflow of the developing *rvesimulator*. DoE goes into the Master control module. The *rvesimulator* encodes DoE (including microstructure information, material properties, and loading conditons) and other FEM parameters into a JSON file. Subsequently, *rvesimulator* has a class object *'AbaqusSimulator'* to execute Abaqus solver, following a standard ABAQUS simulation process and postprocesss procedure. The resulting Pickle file from FEM simulation will be read back to Python and can be utilized for various machine learning methods.

2.1 Cross-platform function for NoGUI Abaqus job execution

While it's common for advanced Abaqus users to run simulations without opening the Abaqus interface, this typically involves terminal operations for parameter adjustments. Although it can accelerate the process, it doesn't achieve the goal embedding RVE simulation to the data-driven modeling and optimization process. Considering the commonalities of this repeatable process, we implemented a class object in *rvesimulator* for resolving this bottleneck such that we can communicate

Python and Abaqus can realize the goal of getting data silently. The usage of the cross-platform class object *AbaqusSimulator* is outlined as follows:

```
from rvesimulator.abaqus2py.abaqus_simulator import AbaqusSimulator
# folder info dict contains locations of ABAQUS benchmark scripts
folder_info = {}
# sim_info dict contains one raw of design of experiment numpy array
sim_info = {}
# intialize the simulator
simulator = AbaqusSImulator(sim_info=sim_info, folder_info=folder_info
    )
# run simulation
simulation.run()
# get results
results = simulator.read_back_results()
```

With this *AbaqusSimulator* object in place, it can handle any Abaqus simulation theoretically as long as the user slightly modifies their scripts according to our online guidance. Specifically for running RVEs simulations, we already abstracted basic Python-Abaqus scripts, and designated corresponding Class objects for different benchmarks inherited *AbaqusSimulator* object.

2.2 Python-Abaqus scripts

In our developing repository, we have abstracted the essential functionalities required for conducting RVE simulations. Following the common procedure of FEM, we have organized the Python-Abaqus scripts following the illustration in Fig.2. As shown in Fig.2, we can see that the benchmark scripts are assembled with seven different classes that handle different sub-modules of Abaqus simulation pipeline. Benchmark scripts are managed by *AbaqusSimulator* where the user can call it as conveniently. It's important to note that the repository already includes numerous benchmarks covering a wide range of cases, including RVEs with porous materials, composite material microstructures, elastic, hyper-elastic, and plastic material models, as well as regular and history-dependent loading curves. For standard usage, users are not required to interact the Python-Abaqus scripts directly, instead they can select the target cases from the *Master control* and dedicate on design the variables they want to study and pass it to corresponding class inherited *AbaqusSimulator*. If users have specific cases that beyond the scope benchmarks scripts, they can easily make their **Benchmark script** following the existed ones.

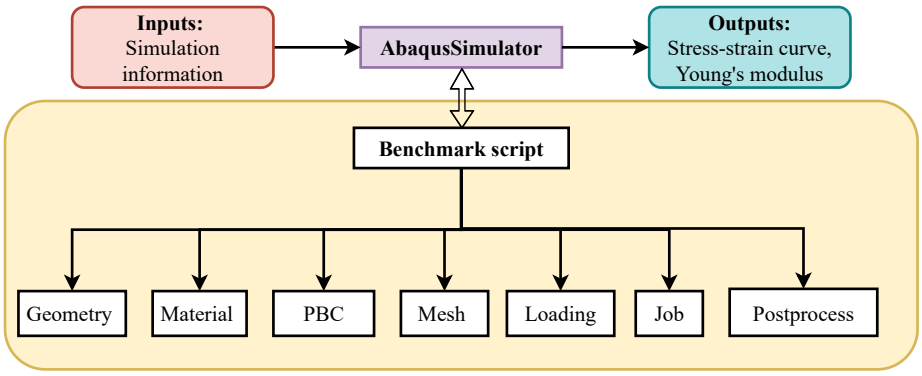

Figure 2: Abstraction of Abaqus script module. The script base is organized by different benchmark scripts which are assembled by geometry, material, periodical boundary condition(PBC), mesh, loading, job, and postprocess. For standard usage, users don't need to touch this module because it is already encapsulated to several common investigated RVEs categories.

3 Experiments

In the repository, we have provided several examples by replicating RVE simulations found in the literature[5, 10, 11, 12, 13]. Detailed instructions for each RVE can be found in the online

documentation. In this section, we would like to give one concrete example on how to generate data via our repository and how to make use of the data for training advanced machine learning models.

3.1 Data-generation for cooperative data-driven modeling

In context of Cooperative Data-Driven Modeling (CDDM) paper [13], the proposed continue learning is employed to learn the constitutive law of composite RVEs under history dependent loading. Overall, different tasks has some similarities and dissimilarities, where the size of RVEs, Young's modules of matrix materials, Poisson ratio of both matrix and paticles are same for all tasks; other parameters like volume fraction, particle distribution, and other material properties are different among different tasks. Different task can be realized by the same *CDDM_RVE* class object by updating different simulation parameters. It is noting that a *Microstructure* class object is utilized in the *CDDM_RVE* to generate corresponding micro-structures for different task, the arguments for the *Microstructure* are the size of RVE, particle distribution parameters. The history dependent loading is the design of experiment variable, 1000 different historical loadings are generated via the *AmplitudeGenerator* class object.

After generating the data-set, integrating it into the continuous learning algorithm is straightforward. Here, for illustrative purposes, we adopt standard RNN to learn constitutive law for a single task, where 800 data points for training and 200 data points for testing. Fig. 3 shows a random point from the test data set, it can observed that RNN learns the complex constitutive law excellently. The trained model can subsequently be used for material property optimization and discovery. More details of experiments can be found in Appendix A

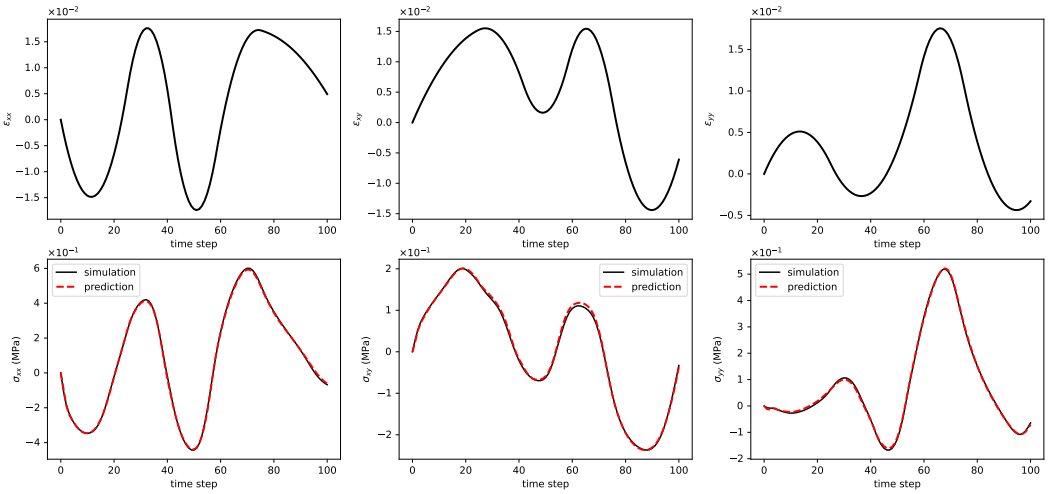

Figure 3: Composite constitutive law learning via RNN: (1) the top row shows the strain components of a random selected RVE of the test data set, which is served as the input of RNN; (2) The bottom row illustrates the simulated stress components of the RVE (solid line) alongside the learned stress components obtained via RNN (dashed line)

4 Conclusion

The shared *rvesimulator* is a flexible and user-friendly platform for generating high fidelity FEM simulation data for RVEs. It includes a variety of implemented benchmarks that cover a wide range of commonly investigated RVEs. Users also have the capability to generate their specific datasets by designing unique experiments tailored for different machine learning methods. For users seeking more advanced RVE simulations, the platform allows for the adaptation of their own Python-Abaqus scripts, enabling the execution of numerous simulations with ease. Additionally, the developing repository can easily adapt to parallelization tools, providing the potential for further acceleration of simulations.

References

[1] Wing Kam Liu, Shaofan Li, and Harold S. Park. Eighty years of the finite element method: Birth, evolution, and future. *Archives of Computational Methods in Engineering*, 29(6):4431–4453, 2022.

[2] Chun-Teh Chen and Grace X. Gu. Machine learning for composite materials. *MRS Communications*, 9(2):556–566, 2019.

[3] Zijiang Yang, Yuksel C. Yabansu, Reda Al-Bahrani, Wei keng Liao, Alok N. Choudhary, Surya R. Kalidindi, and Ankit Agrawal. Deep learning approaches for mining structure-property linkages in high contrast composites from simulation datasets. *Computational Materials Science*, 151:278–287, 2018.

[4] Nolan Black and Ahmad R. Najafi. Learning finite element convergence with the multi-fidelity graph neural network. *Computer Methods in Applied Mechanics and Engineering*, 397:115120, 2022.

[5] M. Mozaffar, R. Bostanabad, W. Chen, K. Ehmann, J. Cao, and M. A. Bessa. Deep learning predicts path-dependent plasticity. *Proceedings of the National Academy of Sciences*, 116(52):26414–26420, 2019.

[6] Matthew W. Scroggs, Jørgen S. Dokken, Chris N. Richardson, and Garth N. Wells. Construction of arbitrary order finite element degree-of-freedom maps on polygonal and polyhedral cell meshes. *ACM Transactions on Mathematical Software*, 48(2):1–23, may 2022.

[7] Tianju Xue, Shuheng Liao, Zhengtao Gan, Chanwook Park, Xiaoyu Xie, Wing Kam Liu, and Jian Cao. Jax-fem: A differentiable gpu-accelerated 3d finite element solver for automatic inverse design and mechanistic data science. *Computer Physics Communications*, 291:108802, 2023.

[8] Abaqus: Finite element analysis for mechanical engineering and civil engineering. `https://www.3ds.com/products-services/simulia/products/abaqus/`.

[9] A.R. Melro, P.P. Camanho, and S.T. Pinho. Generation of random distribution of fibres in long-fibre reinforced composites. *Composites Science and Technology*, 68(9):2092–2102, 2008.

[10] Zeliang Liu, M.A. Bessa, and Wing Kam Liu. Self-consistent clustering analysis: An efficient multi-scale scheme for inelastic heterogeneous materials. *Computer Methods in Applied Mechanics and Engineering*, 306:319–341, 2016.

[11] M.A. Bessa, R. Bostanabad, Z. Liu, A. Hu, Daniel W. Apley, C. Brinson, W. Chen, and Wing Kam Liu. A framework for data-driven analysis of materials under uncertainty: Countering the curse of dimensionality. *Computer Methods in Applied Mechanics and Engineering*, 320:633–667, 2017.

[12] Bernardo P. Ferreira, F. M. Andrade Pires, and Miguel A. Bessa. Adaptive clustering-based reduced-order modeling framework: Fast and accurate modeling of localized history-dependent phenomena. 2021.

[13] Aleksandr Dekhovich, O. Taylan Turan, Jiaxiang Yi, and Miguel A. Bessa. Cooperative data-driven modeling. *Computer Methods in Applied Mechanics and Engineering*, 417:116432, 2023.

A Descriptions of experiments set up

Four tasks have been generated in the CDDM paper [13] and the first task is adopted in this paper for illustration purpose.

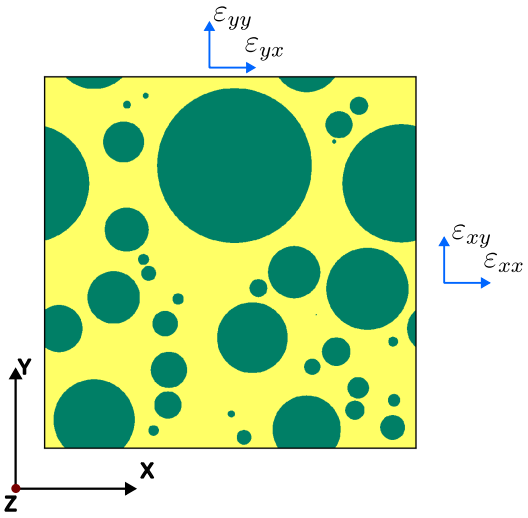

Figure 4: Micro-structure and loading configuration of the illustrated RVE simulation task. The orange area is represents the matrix material phase and green domain is the particle material phase

A.1 Problem description

RVE simulation serves as an effective computational approach to investigate the constitutive laws of composites. Generally, An RVE has two material Phases which are matrix phase and fiber phase. The micro-structure and loading configurations of the illustrated task is shown in Fig. 4.

According to Fig. 4, the size of the RVE is 0.048mm; and the particle is not uniformed distributed. Instead, they follow a normal distribution $\mathcal{N}(0.01, 0.003)$. As for the materials of both phases, an elastic material law with Young's modulus 10MPa and Poisson ratio 0.19 is utilized. The matrix phase employs a Von Mises plastic material law with Young's modulus 100MPa Poisson ratio 0.3, and hardening law $\sigma_y = 0.5 + 0.5\bar{\varepsilon}$, where $\bar{\varepsilon}$ is the accumulative plastic strain. Meanwhile, as shown in Fig. 4 complex loadings $[\varepsilon_{xx}, \varepsilon_{xy}, \varepsilon_{yy}]$ [1] will be enforced to the RVE simultaneously. These loadings are path dependent as shown on the top row of Fig. 3.

A.2 Implantation for getting quantity of interests

In terms of implementing the above-mentioned RVE configuration, obtaining 1000 data points sequentially is easily accomplished by submitting the following command to the Python terminal. No additional GUI operation is required after submitting the script, simulation results will be extracted automatically upon finishing all jobs, one of which is shown as the solid line on the bottom plots of Fig. 3. While parallelization is not implemented in the *rvesimulator*, users can leverage any third party package to achieve this functionality[2].

```
# import objects from rvesimulator
from rvesimulator.benchmarks.cddm_rve import CDDM_RVE
from rvesimulator.additions.ampitudesampler import AmplitudeGenerator
from rvesimulator.additions.hardening_law import LinearHardeningLaw

# number of path
num_paths = 1000
# initialize path sampler
path_sampler = AmplitudeGenerator(num_dim=3)
# get paths
paths = path_sampler.get_amplitude(
    num_amplitude=num_amplitude,
```

[1] ε_{xy} usually same as ε_{yx}, so it is omitted for simplifying notation

[2] For instance, another package named f3dasm is employed for parallelizing the data generation process, the python scripts for this can be found in the GitHub repository

```
    num_control=8,
    num_steps=100,
    arg_name="strain_amplitude",
)
# convert to dict
samples_dict = paths.to_dict("records")
# initialization for simulation task
task = CDDM_RVE()
# update simulation info
task.update_sim_info(mesh_partition=100,
                     vol_req=0.45,
                     radius_mu=0.01,
                     radius_std=0.003,
                     youngs_modulus_fiber=10,
                     youngs_modulus_matrix=100,
                     hardening_law=LinearHardeningLaw(a=0.5, b=0.5,
    yield_stress=0.5))
# empty dict to save results
results = {}
# calculate responses of simulation in sequential
for ii in range(len(samples_dict)):
    results[ii] = task.run_simulation(sample=samples_dict[ii],
                                      folder_index=ii)
```

A.3 Machine learning model for learning material law

While the main focus of this paper is not on the machine learning model, a standard RNN is employed, given its widespread use in learning composite constitutive laws [5, 13]. In principle, other machine learning models, such as MLP, also hold potential for this problem. Researchers are encouraged to explore the dataset and experiment with different models according to their preferences.