HOOP: a Python software for homogenization of multiphase composite materials using object-oriented architecture

Carlos André dos S. Lima[1], Rodrigo Mero S. da Silva[1], Leonardo de Melo Medeiros[2]

[1]Instituto Federal de Alagoas (IFAL), Campus Palmeira dos Índios
Av. Alagoas, S/N - Palmeira de Fora, Palmeira dos Índios - AL, 57608-180, Brazil
casl2@aluno.ifal.edu.br, rodrigo.mero@ifal.edu.br
[2]Instituto Federal de Alagoas (IFAL), Campus Maceió
R. Mizael Domingues, 530 - Centro, Maceió - AL, 57020-600
leonardo.medeiros@ifal.edu.br

Abstract. HOOP (Homogenization Object-Oriented Programming) is a Python software package designed for efficient and flexible mean-field micromechanical analysis. Utilizing an object-oriented architecture, HOOP enables researchers to investigate mechanical and thermal phenomena within multiphase composite materials. By leveraging established libraries like NumPy, Seaborn, and Matplotlib, HOOP offers robust numerical operations, data manipulation capabilities, and compelling visualizations. The strategic integration of various Python libraries within HOOP's architecture fosters inherent flexibility and interoperability. This enables tailored workflows and seamless integration with established tools. Users can leverage alternative libraries for specific tasks, expanding capabilities beyond core functionalities. Moreover, HOOP has been rigorously tested and validated, emerging as a powerful and cost-effective alternative to commercial software for multiphase composite analysis. This makes it an invaluable tool for academic researchers and engineers alike, democratizing access to advanced micromechanical analysis capabilities.

Keywords: composite materials, framework, object-oriented programming (OOP), python.

1 Introduction

In the industry, the combination of materials is a crucial strategy to achieve significant improvements in the essential properties of products, whether mechanical, thermal, magnetic, or others. This practice not only optimizes the performance of materials but also prolongs their service life. For example, in civil engineering, the combination of materials with concrete not only strengthens its mechanical properties but also improves its functionality, generating efficiency in construction [1].

However, a problem persists: the lack of products on the market that offer robust analytical capabilities to perform large-scale analyses without the need for field tests. In this work, we present a computational tool called HOOP, originally proposed by [1], a software developed in Python using Object-Oriented Programming (OOP), specifically designed to meet the demands of the industry. This software is capable of performing numerical and graphical analyses of a mechanical or conductivity nature. Although there is the possibility of implementing other types of analysis, given its flexibility, facilitating experiments outside laboratories or for independent research.

With this solution, we aim to increase the efficiency and competitiveness of the industry, promoting significant advances in the area of composite material analysis.

2 Methods

2.1 Modeling

To develop software that is scalable and capable of effectively and flexibly meeting the mentioned needs, the OOP paradigm is chosen. OOP is a programming paradigm that allows the representation of real-world entities in the computational context through classes and methods. This provides lean and scalable code, susceptible to modifications and updates without major issues, and, of course, is based on the idea of objects [2].

The project follows an architecture in which a "parent" class is defined, and other classes use its attributes. For example, the "Analysis" class, one of the "child" classes, contains important data about the nature of the problem. These classes serve as reference points for improvements in the software experience, as well as in the tools and features provided.

The entire process will be developed in Python, which, besides being open source (constantly evolving due to its large community), has flexibility and compatibility with other established technologies in the market, making Python an invaluable tool in modern software development.

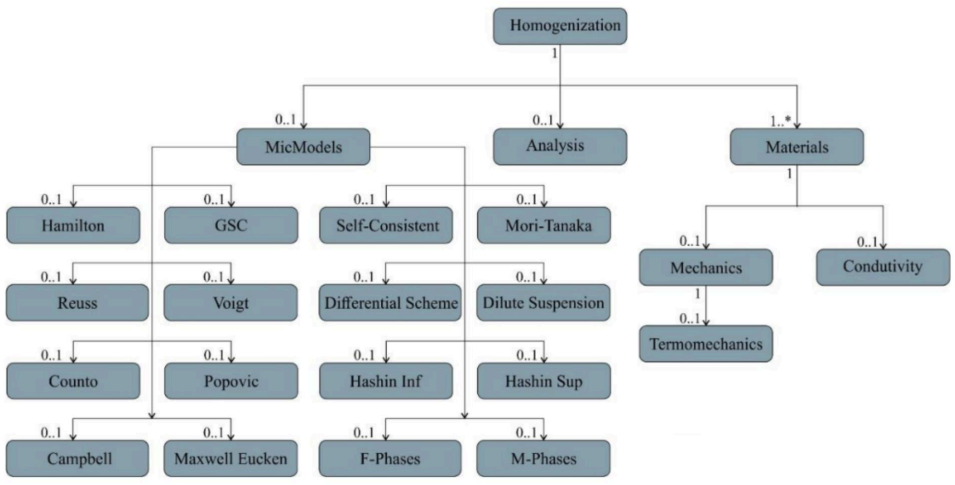

Figure 1: Class diagram for homogenization modulus

2.2 Design Patterns

To maintain its scalability and offer viable alternatives for other developers, HOOP adopts several software engineering guidelines. This ensures HOOP's integration into the open-source universe, providing the software with robustness and reach for future collaborators. Additionally, it follows PEP 8, a style guide widely accepted by the Python community, which includes aspects such as clarity in variable naming [3]. It embraces the DRY (Don't Repeat Yourself) principle, which advises avoiding repetitions whenever possible, especially in OOP code [4]. Moreover, detailed documentation and a user guide for the software will be developed, enabling understanding and adjustment. Tools like Ruff are used to automatically check for PEP 8 compliance, aiming to simplify the incorporation of new features in the future.

2.3 Implementation of new models

Integrating a new model into HOOP is as simple as adding a book to a categorized bookshelf. Just place the "new_model.py" file in the "micmodels" folder, and it becomes part of the software's capabilities. This allows for plotting and computation applicable to thermal or mechanical analysis, depending on the model's purpose. This interactive process, where we add a new file, import it, and plot it, is at the core of the software. It permeates and completely supports its architecture. It will be up to future researchers to decide whether to implement new models or make adjustments within the pre-existing models.

2.4 Validation

The validation of the implemented classes, whether for design, new models, or methods, is tested against previously published articles on the studied topic (homogenization), in addition to debugging, thus ensuring the software's reliability. Below is a comparison between a mechanical problem found in Miled et al. [6] Fig. 2 and the same problem performed in HOOP Fig. 3.

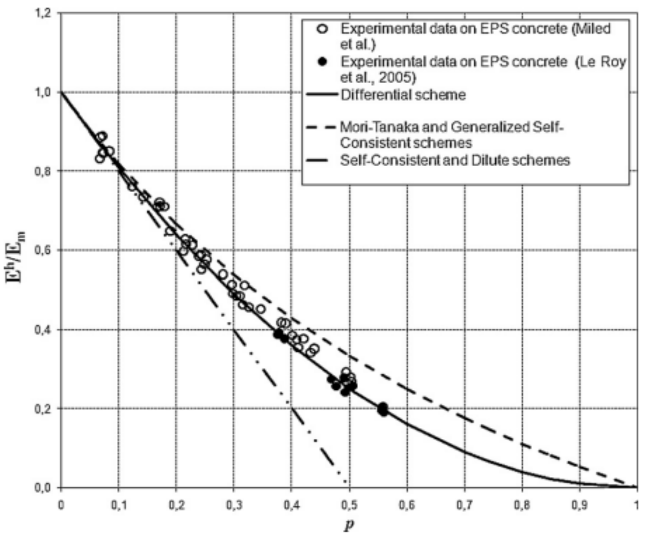

Figure 2. Variation of the EPS concrete Young's modulus according to porosity (p): homogenization schemes vs experimental data.

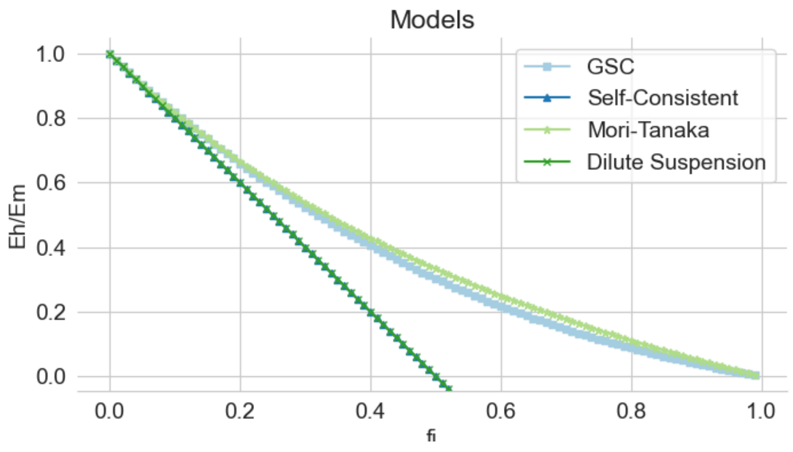

Figure3. The result generated by our software shows that the behavior of the models demonstrates the correct construction of the problem's mathematics.

2.5 Results

Although still in its early stages in terms of interface, HOOP has proven to be a promising tool compared to existing solutions for homogenization using computational tools, especially as opposed to closed-source tools. This is primarily due to its flexibility and scalability. Below, in Tab.1, is a real case with validated results generated by HOOP. To date, the tool includes dozens of models and two types of analysis, each with its specific design method: mechanical and conductivity.

Table 1. Coefficients in constitutive relations

Materials	E	G	v
Concret	28 Gpa	11.2 Gpa	0.20
Glass	70 Gpa	27 Gpa	0.22

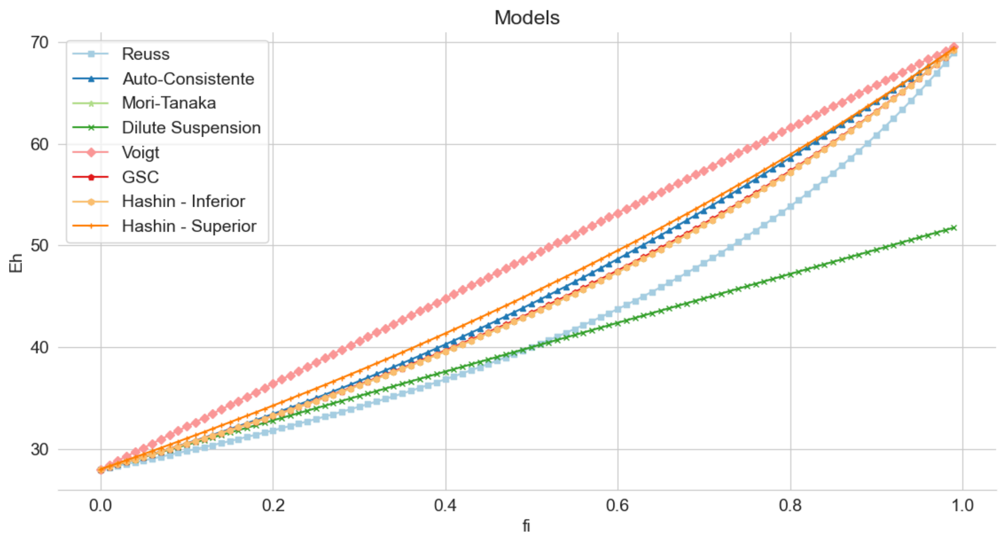

Figure 4. Results computed by HOOP, including eight models for Tab. 1

Where, In Tab.1 **E** denotes the Modulus of Elasticity, **G** represents the Shear Modulus, and **v** is Poisson's Ratio. Also, **GPa** stands for gigapascals.

3 Conclusions

The HOOP integrates software engineering approaches with the versatility of the Python language and its frameworks, all implemented in an object-oriented manner. In this work, we showed how we built and tested this for composite material analysis, demonstrating that it can be adapted and improved over time with minimal alterations to its OOP architecture.

The authors hereby confirm that they are the sole liable persons responsible for the authorship of this work, and that all material that has been herein included as part of the present paper is either the property (and authorship) of the authors, or has the permission of the owners to be included here.

References

[1] R. M. S. da Silva, "Coupling between geometric optimization models of particulate systems and micromechanics of mean fields for multiscale analysis of multiphase cementitious composites," 2022.
[2] A. Kay, "Dr. Alan Kay on the Meaning of 'Object-Oriented Programming'," 2003. [Online]. Available: https://example.com. [Accessed: Jun. 10, 2024].
[3] G. van Rossum, "PEP 8 – Style Guide for Python Code," *Python*. [Online]. Available: https://peps.python.org/pep-0008/. [Accessed: Jun. 10, 2024].
[4] A. Hunt and D. Thomas, *The Pragmatic Programmer: From Journeyman to Master*, 1st ed., US: Addison-Wesley, 1999, pp. 320. ISBN: 978-0201616224.
[5] R. M. S. da Silva and A. S. R. Barboza, "Concrete modeling using micromechanical multiphase models and multiscale analysis," *Revista IBRACON de Estruturas e Materiais*, vol. 16, 2023, Art. e16501.
[6] K. Miled, K. Sab, and R. Le Roy, "Effective elastic properties of porous materials: Homogenization schemes vs experimental data," *Mechanics Research Communications*, vol. 38, no. 2, pp. 131–135, 2011.