International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Unleashing the Power of Open Sees: A Versatile Framework for SBFEM Analysis

Talkeshwar Ray[1*], Sukumar Baishya[2], Dipika Devi[3]

Submitted: 17/10/2023 **Revised**: 07/12/2023 **Accepted**: 17/12/2023

Abstract: The Open Sees software framework is a powerful tool for simulating the seismic behavior of both structural and geotechnical systems. This open source finite element analysis tool has been widely adopted within the earthquake engineering community for its ability to perform complex analysis through scripting languages. One of the key features of the Open Sees framework is its ability to generate both serial as well as parallel finite element computer applications as interpreters. This allows developers to take full advantage of the powerful capabilities of the software. Historically, the Tcl scripting language has been the primary tool used for building and analyzing models within Open Sees. In order to enhance the software's flexibility and offer users more choices for scripting languages, especially Python, the Open Sees interpreter interface underwent a redesign to include support for multiple interpreters. The outcome of this restructuring effort led to the development of Open Sees Py, a Python module enabling users to harness the distinctive capabilities offered by various scripting languages. By leveraging advanced finite element analysis models and algorithms, users can now develop applications that are tailored to their specific needs. This new multi-interpreter interface expands the capabilities of the software, making it even more powerful and versatile.

Key Words: - *Open Sees, Open Sees Py, geotechnical systems, finite element analysis*

1. Significance of work

Motivation for advanced simulation techniques in civil infrastructure is to provide more accurate and reliable predictions of the performance of structures and systems under natural hazards such as earthquakes and tsunamis. This allows for informed decisions about the design, construction, and retrofitting of infrastructure, ultimately leading to safer and more resilient communities.

Simulation provides a deeper understanding of the behaviour of structures and systems under various loading condition, helping to identify potential weaknesses and vulnerabilities, and inform decisions about risk mitigation and performance improvement. Additionally, simulation can help optimize design and retrofit options to make them more cost-effective, which can have a significant impact on infrastructure project costs.

The main objectives of civil infrastructure design and assessment range from ensuring immediate occupancy to achieving life safety standards. Simulation plays a crucial role in achieving these objectives in engineering practice, by allowing engineers to quantify the probabilities of exceeding limit states through the use of stochastic methods that take into account the diversity in system characteristics as well as outside forces.

To effectively develop novel simulation models within a versatile programming framework is required, where programmers can further develop using interchangeable modules already in place. One such software framework is Open Sees, which is an opensource and completely operational software for finite element analysis that was developed in the capacity of designated modeling environment utilized by the Pacific Earthquake Engineering Research (PEER) Center. Predominantly coded in C++, the software incorporates connections to certain FORTRAN libraries to address linear system equation solutions. The framework seamlessly integrates cutting-edge finite element representations and non-linear solving methods, thus offering sophisticated modeling and analysis capabilities suitable for a variety of applications.

2. Description of Software

The new multi-interpreter functionality in Open Sees allows for the integration of multiple programming languages for scripting, such as Python, Ruby, Julia, and R, in addition to the primary Tcl language. This empowers users to harness the distinct attributes of diverse languages, all the while utilizing the sophisticated models

[1]Research Scholar, Dept. of [1*]PhD Scholar, Civil Engineering Department, North Eastern Regional Institute of Science and Technology, Arunachal Pradesh, India*
[2]Professor, Civil Engineering Department, North Eastern Regional Institute of Science and Technology, Arunachal Pradesh, India
[3]Associate Professor, Civil Engineering Department, North Eastern Regional Institute of Science and Technology, Arunachal Pradesh, India
**Corresponding Author: Talkeshwar Ray*
CSE, VTU, BELAGAVI
[2]Associate Professor & Head, Dept. of CSE, VTUCPGS, Chikkaballapura
**Corresponding Author: V. Kiran Kumar*
**Research Scholar, Dept. of CSE, VTU, BELAGAVI*

and algorithms for finite element analysis that Open Sees offers. This also makes Open Sees more accessible to a wider range of users, as it removes the constraints inherent in using only one scripting language and provides more flexibility for specialized applications.

The DL_ Interpreter class, which is the core of the multi-interpreter functionality, establishes a set of eight fundamental functions for input and output for acquiring whole number, decimal, and text inputs. Every programming language for scripting has a dedicated interpreter class responsible for implementing these functions, allowing for language-specific operations such as initialization, printing, and environment setup. Additionally, within each interpreter class, a wrapper class is formulated to collectively manage the task of encapsulating Open Sees core APIs. This approach effectively isolates the core APIs from the interpreter, streamlining the process of introducing a new interpreter while minimizing the associated workload.

Overall, the multi-interpreter functionality of Open Sees provides a more versatile and user-friendly experience for engineers and researchers in the field of earthquake engineering. It allows for the integration of multiple scripting languages and provides more flexibility for specialized applications, while still leveraging the advanced modeling and analysis capabilities of the Open Sees software framework.

3. Architecture of Open Sees

Open Sees software design pattern is grounded in the Model-View-Controller (MVC) architecture, segregating the portrayal of data from the user's engagement with it. This modular design allows for easy modification and extension of the software, as well as better separation of concerns.

The Model component of Open Sees includes the Domain, Analysis, and Recorder objects, which represent the finite element model of the system, the calculations performed on the model, and the recording of the analysis results.

The View component of Open Sees includes the Graphical User Interface (GUI) extensions such as Building Tcl and Open Sees Navigator, which provide a visual representation of the model and the analysis results.

The Controller component of Open Sees includes the Input/Output (IO) functions, Interpreter classes, Wrapper class, and Core Open Sees API. These components handle the user's interaction with the software, including reading input, creating objects in the Domain, performing analyses, and recording results. The Interpreter classes additionally manage language-specific operations, encompassing tasks such as language initialization and the expansion of functions or commands. The Wrapper class is responsible for wrapping the Core Open Sees API and providing a consistent interface across different scripting languages.

In nutshell, Open Sees software architecture is designed to be modular and flexible, with distinct demarcation of responsibilities among the Model, View, and Controller components. This allows for easy modification and extension of the software, as well as better organization of the code, making it easier to maintain and develop.

The existing Open Sees framework would typically include the following main classes:

a) Domain: This class represents the system's finite element and includes information about the nodes, elements, loads, and constraints.

b) Model Builder: This class assumes the responsibility of populating the Domain containing essential element(s), node(s), load(s), as well as constraint objects, thereby constructing a representation of the system using finite element modeling.

d) Analysis: This class calculates the condition pertaining to the finite element model during a specific type of analysis, employing exchangeable solution techniques, solvers for linear equations, handlers for constraints, and methods for time integration.

e) Recorder: This class is responsible for recording the analysis results.

f) DL_ Interpreter: This class defines the fundamental input and output (IO) operations for acquiring whole numbers, decimal values, and text inputs.

g) Tcl Interpreter/Python Interpreter: These classes manage language-specific operations, including language initialization, information printing, configuration of environmental variables, loading startup modules and files, as well as the expansion of functions or commands.

h) Wrapper: Within an interpreter class, this specific class is established to collectively assume the role of encapsulating Open Sees core APIs.

i) GUI: This class embodies the Graphical User Interface (GUI), exemplified by applications like Building Tcl and Open Sees Navigator.

Note: - This is a general description of the class of the existing Open Sees framework and it might vary depending on the specific usage and implementation of the software.

It also enables developers to add new interpreters, or interfaces, to support different scripting languages and GUI platforms, providing a wider range of options for users. Additionally, the separation of concerns between the Model, View, and Controller components helps to keep the code organized and maintainable, making it easier for developers to add new features and functionality to the software.

The multi-interpreter interface for Open Sees would typically include the following main classes:

a) DL_ Interpreter: This class establishes fundamental input and output (IO) operations for acquiring whole numbers, decimal values, and text inputs. It serves as an interface for different interpreter classes.

b) Interpreter classes: These classes implement the DL_ Interpreter interface and handle operations associated with the language, such as language initialization, information printing, configuring environmental variables, loading initial modules and files, and expanding functions or commands. Examples of interpreter classes include Tcl Interpreter, Python Interpreter, Ruby Interpreter, Julia Interpreter, and R Interpreter.

c) Wrapper class: This class is well-defined within an interpreter class in order to share the responsibility of wrapping Open Sees core APIs. It is responsible for wrapping the Open Sees core APIs and abstracting the language-specific details.

d) Core Open Sees API: This class is responsible for pure Open Sees code for populating the model domain, running analyses, and manipulating internal objects.

e) Domain: This class represents the finite element model of the system and includes information about the nodes, elements, loads, and constraints.

f) Model Builder: This class holds the responsibility of populating the Domain with the essential node, element, load, and constraint objects required to construct a finite element model of the system.

g) Analysis: This class calculates the condition of the finite element model during a specific analysis type, utilizing interchangeable solution algorithms, linear equation solvers, constraint handlers, and time integration methods.

h) Recorder: This class is responsible for recording the analysis results.

i) GUI: This class represents the Graphical User Interface (GUI) such as Building Tcl as well as Open Sees Navigator.

Note: - This is a general description of the multi-interpreter interface for Open Sees and it might vary depending on the specific usage and implementation of the software.

In essence, the multi-interpreter capability of Open Sees is strategically developed to grant users the freedom to employ their favored scripting language, all the while harnessing the sophisticated finite element analysis models and algorithms within the Open Sees software framework. The DL_ Interpreter interface coupled with the wrapper class facilitates the isolation of the core Open Sees APIs from the interpreter, simplifying the addition of new interpreters and ensuring the software remains current with emerging scripting languages from the computational community. This methodology guarantees that Open Sees maintains its significance within its extensive user community and remains aligned with progress in both scripting languages and nonlinear finite element analysis.

4. Software functionalities

The Open Sees software offers an extensive array of capabilities to simulate the seismic behavior of both structural and geotechnical systems. Among the pivotal functionalities of the software are:

Finite element analysis: Open Sees presents an extensive selection of finite element models and solution algorithms tailored for nonlinear dynamic analysis of both structural and geotechnical systems. These encompass both linear and nonlinear solvers, alongside an assortment of time integration schemes.

Advanced simulations: Open Sees has undergone expansion to encompass advanced functionalities, which encompass parameter updating and sensitivity analysis, fire simulation, and the simulation of fluid-structure interaction.

Multi-interpreter support: Open Sees has introduced a fresh interface to its framework, known as the DL_ Interpreter class, which facilitates the integration of multiple interpreters. This enhancement broadens its accessibility to a wider spectrum of users.

Modularity: Open Sees has a modular design, which allows for easy modification and extension of the software.

Graphical User Interface (GUI): Open Sees also has several extensions, such as Building Tcl and Open Sees Navigator, which provide a GUI for creating and analyzing models.

ecorder: The software also allows to record the analysis results, which can be later used for post-processing

Scripting Interface: Open Sees also has a scripting interface, which allows users to create and analyze models using a scripting language such as Tcl,

Illustrative examples

Here is a simplified process, how to add S Element of SBFEM principle in Open Sees or Open Sees Py for static or seismic analysis:

a) Download Open Sees, i.e. git clone

https://github.com/OpenSees/OpenSees.git

b) In Open Sees/SRC/element folder create a new folder S Element.

c) Into this folder create this C++ files using SBFEM Principle as S Element. h, SElement.cpp, C Make Lists S Element.txt and OPS_SElement.cpp.

d) Open the C Make Lists.txt file in Open Sees/ SRC/ element and add a line as per instruction manual to add the S Element into the directory.

e) Open the file SRC/element /Tcl Element Commands. cpp and add the OPS_S Element function as per instruction of instruction manual.

f) Compile Open Sees or Open Sees py from the build directory. It should compile and Open Sees exe or Open Sees Py will be in build/bin folder

Numerical examples for static loading

SBFEM approach is used to describe a rectangular body, which is represented by three S Elements, as illustrated in figure 1. The dimensions of the rectangular element body are indicated in the same figure, where b is equal to 1 meter. The material properties of the body are described by Young's modulus E, which is equal to 10 GPa, and Poisson's ratio μ, which is equal to 0.25. The node numbers and S Element numbers are shown in figure 1(a) and (b) respectively. The input data of the S Elements is shown in figure 1(b), where the boundary of each S Element is defined using line elements, with element numbers indicated inside the circles and are locally numbered within each S Element. The connectivity of the line elements is defined by the connectivity of the nodes. When specifying the line elements, it's crucial to adhere to a counterclockwise orientation relative to the scaling center of the S Element. Each row corresponds to the nodal numbers of a single line element. The compatibility between the S Elements is automatically satisfied, and this is locally modeled by Element 1 in S Element 1 and Element 3 in S Element 2, where the two line elements share identical nodes, shape functions, and comparable displacements. The x and y coordinates of the scaling centers of the S Elements are typically chosen as the geometrical center of each S Element. The analysis of this rectangular body is performed using Open Sees Py.

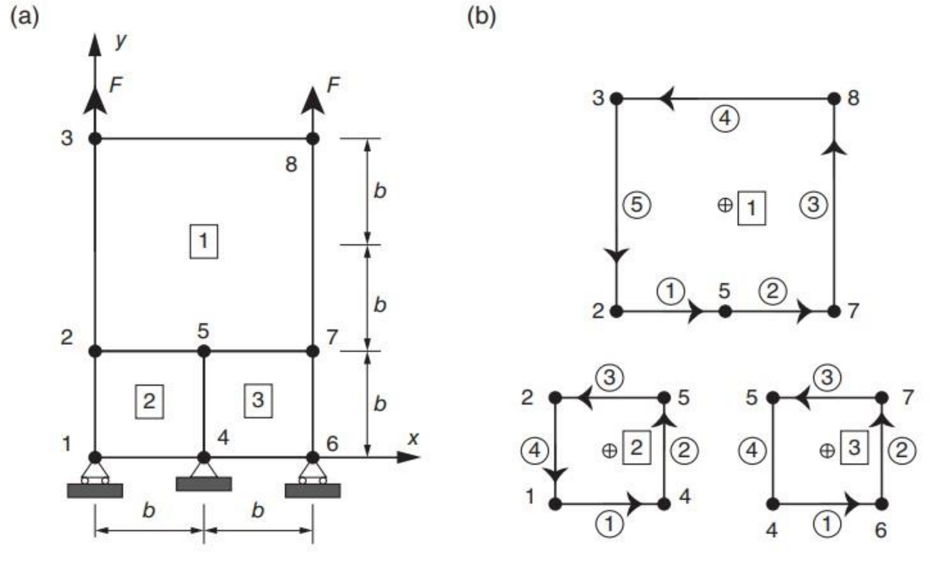

Fig 1: Modeling of a rectangular body using three S-Elements. This is shown with (a) Mesh (b) Isolated S-elements that have line elements to define the boundary.

Results Obtained

DOFs	OpenSeesPy		MATLAB	
	Nodal Displacement	Nodal Force	Nodal Displacement	Nodal Force
1	2.5000011E-05	-2.8421709E-14	2.5000000E-05	0.0000000E+00

2	0.0000000E+00	-5.0000007E+02	0.0000000E+00	-5.0000000E+02
3	2.5000020E-05	0.0000000E+00	2.5000000E-05	0.0000000E+00
4	1.0000001E-04	1.7053026E-13	1.0000000E-04	1.1368684E-13
5	2.5000030E-05	2.8421709E-14	2.5000000E-05	-8.5265128E-14
6	3.0000004E-04	1.0000000E+03	3.0000000E-04	1.0000000E+03
7	0.0000000E+00	-8.7891371E-07	0.0000000E+00	1.4210855E-13
8	0.0000000E+00	-1.0000000E+03	0.0000000E+00	-1.0000000E+03
9	2.0098165E-11	-8.5265128E-14	7.1174264E-20	-2.8421709E-14
10	1.0000002E-04	-3.4106051E-13	1.0000000E-04	0.0000000E+00
11	-2.5000006E-05	7.1054274E-15	-2.5000000E-05	0.0000000E+00
12	0.0000000E+00	-4.9999992E+02	0.0000000E+00	-5.0000000E+02
13	-2.4999978E-05	2.6645353E-15	-2.5000000E-05	9.7699626E-15
14	9.9999993E-05	1.7053026E-13	1.0000000E-04	5.6843419E-14
15	-2.4999974E-05	-5.6843419E-14	-2.5000000E-05	-5.6843419E-14
16	2.9999999E-04	1.0000000E+03	3.0000000E-04	1.0000000E+03

Table 1 Result of Open Sees Py and MATLB

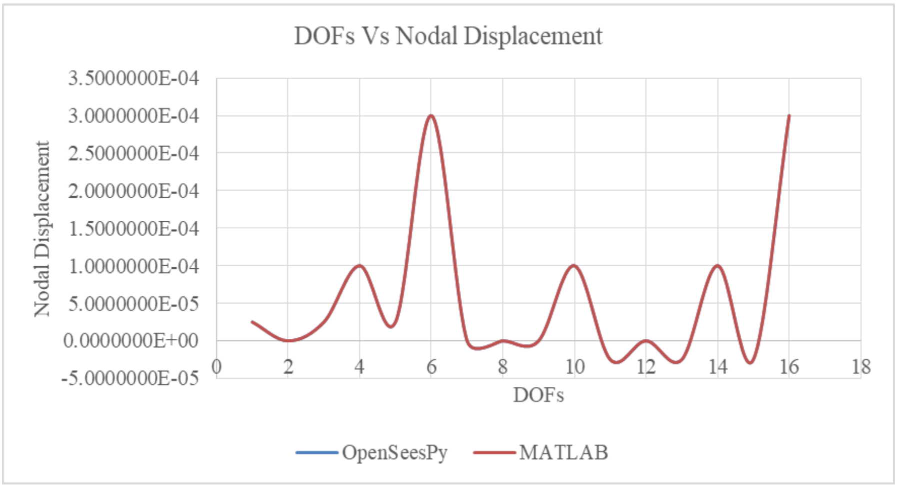

Fig 2:- A Plot Between Nodal Displacement and DOFs

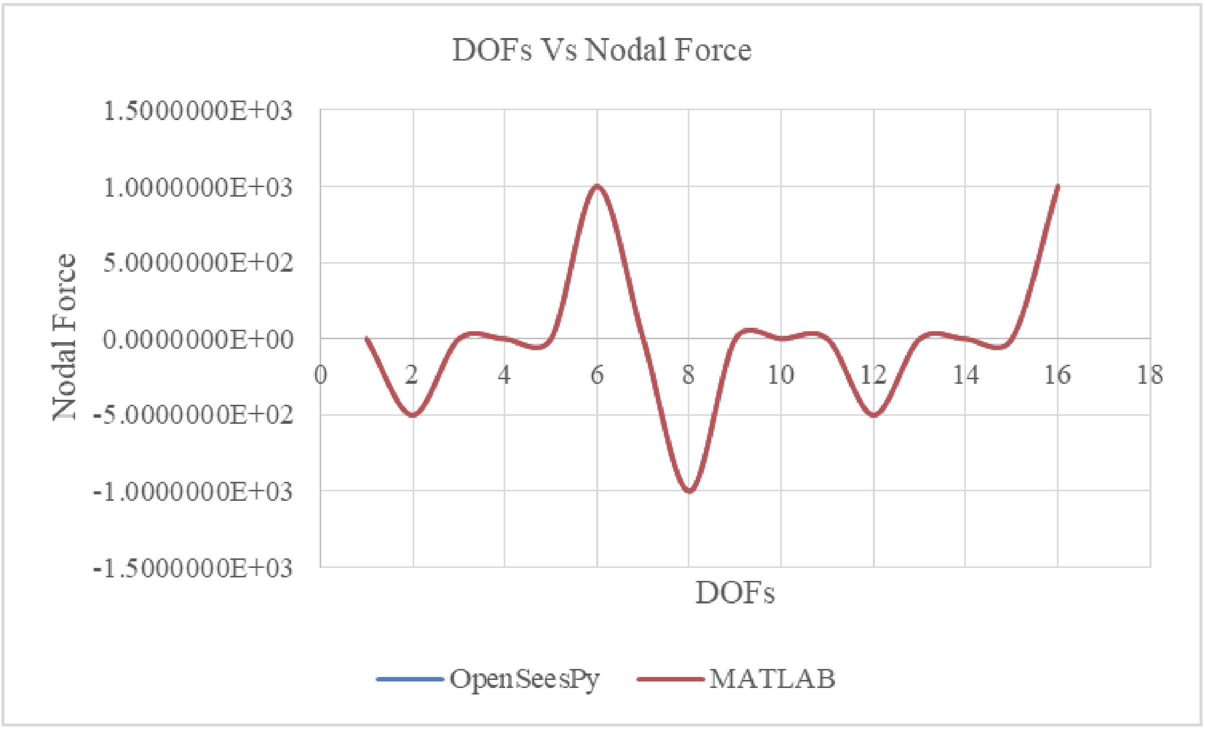

Fig 3:- A Plot Between Nodal Force and DOFs

5. Conclusions

As we all know that Open Sees is a powerful and flexible open-source software framework for simulating static as well as seismic response of infrastructure related to civil engineering. It provides an extensive selection of cutting-edge finite element representations and algorithms for solving nonlinear problems, coupled with advanced features such as parameter updating, sensitivity analysis, fire simulation, and fluid-structure interaction. The software's multi interpreter support allows users to take benefit of different scripting languages and libraries, making it more accessible to a broader range of users. The open-source nature of the software also allows for contributions and improvements from the community, keeping it up-to-date with the latest advances in both scripting languages and nonlinear finite element analysis. Implementation of the scaled boundary finite element method (SBFEM) in the form of S Element in Open Sees Py has shown promising results. The nodal displacement and nodal force results obtained in Open Sees Py using S Element are similar to those obtained using MATLAB, indicating the accuracy and reliability of the S Element implementation. The SBFEM has been proven to be an effective and efficient method for analyzing complex structures, and the implementation of S Element in Open Sees Py has made this method more accessible to researchers and practitioners in the field. The use of S Element in Open Sees Py can lead to significant improvements in the efficiency of structural analysis and design, which can have far-reaching impacts in the field of civil and structural engineering. Overall, the implementation of SBFEM in the form of S Element in Open Sees Py provides a valuable contribution to the field of structural analysis, and its potential for further applications and improvements makes it an exciting area for future research.

References

[1] Chen, X., Birk C. and Song, C. (2013). "*Modelling of wave propagation in unbounded domains using the scaled boundary finite element method".* APCOM & ISCM, 11-14, Singapore

[2] H. D. Rathod (1988). "*Some analytical integration formulae for a four node isoparametric element*", Comput. Struct. 30, 1101-1109.

[3] I. M. Smith and D. V. Griffiths (1988), *"Programming the Finite Element Method",* 2nd edn, Wiley, Chichester, New York.

[4] O. C. Zienkiewicz and R. L. Taylor (1989), "*The Finite Element Method*", Vol. 1, 4th edition, Mc-Graw Hill, London, New York.

[5] Open System for Earthquake Engineering Simulation (Open Sees):

https://openseesberkeley.edu/.)

[6] Song, C., and Wolf, J. P. (2000). "The scaled boundary finite-element method – A primer: Solution procedures." Comput. Struct., 78(1–3), 211–225

[7] Song C., (2018) "The Scaled Boundary Finite Element Method: Introduction to Theory and Implementation", 1st Edition, John Wiley & Sons, New Jersey.

[8] Talkeshwar Ray, Sukumar Baishya, Dipika Devi, "*A State-of-the-Art Review on Soil Structure Interaction*", International Journal of Scientific Research in Science and Technology (IJSRST), Online ISSN: 2395-602X, Print ISSN: 2395-6011, Volume 9 Issue 2, pp. 33-34, January-February 2022. Available at

doi : https://doi.org/10.32628/IJSRST22911

[9] Wolf, J. P. (2003). "The scaled boundary finite element method", Wiley, Chichester, U.K