OpenBeam: Off-Line and On-Line Tools to Solve Static Analysis of Mechanical Structures

José Luis Blanco-Claraco[1](✉), Javier López-Martínez[2], Francisco Javier Garrido-Jiménez[2], Pedro Gómez-Calvache[2], and José Manuel García-Manrique-Ocaña[3]

[1] Engineering Department, Agrifood Campus of International Excellence (ceiA3), University of Almería, Almería, Spain
jlblanco@ual.es

[2] Engineering Department, University of Almería, Almería, Spain

[3] Civil Engineering, Materials, and Fabrication Department, University of Málaga, Málaga, Spain

Abstract. The direct stiffness matrix method for static calculation of structures represents one of the most precise and efficient paradigms to address the analysis of the structures most typically used in construction. The present work intends to fill a niche in open-source software in the field of computational mechanics in relation to said matrix methods, providing a C++ programming library and a set of associated tools that allow an easy approach to structural analysis. This new project, named *OpenBeam*, presents a design that emphasizes didactic applications with, among other features: an easy parameterization of structures, the presentation of diagram graphs, and the creation of animations of the deformed structures. In addition, an interactive version of the software is offered as a freely accessible online tool for use on any desktop or mobile device without the need for installations since it runs directly on the web browser. The application is accessible from https://openbeam.github.io/openbeam/.

Keywords: Stiffness matrix method · finite elements · educative innovation · web applications

1 Introduction

Matrix methods are widely known and used today for static calculation of structures typically used in construction [1, 2] and, through the finite element method, of arbitrary pieces in two or three dimensions [7]. With this work we intend to fill a niche in the open-source software of the field of computational mechanics in relation to said matrix methods, providing a C++ library and a set of associated tools to ease computational structural analysis. The software is design emphasizing a didactic aiming by means of, among other features, parameterization of structures, the presentation of force diagrams, and the creation of animations of the deformed structures.

A. Vizán Idoipe and J. C. García Prada (Eds.): IACME 2022, *Proceedings of the XV Ibero-American Congress of Mechanical Engineering*, pp. 57–63, 2023.
https://doi.org/10.1007/978-3-031-38563-6_9

The availability of versatile, open-source and universally free matrix calculation tools for structures would have multiple applications. Firstly, as a reusable block (programming language library) in the creation of static analysis tools for engineering projects. In the educational field, it would allow the automation of tasks such as the creation of graphics and animations of different structures, including the parametric generation of structures and the automatic calculation of static solutions (reactions, deformations, and stresses), thus facilitating the creation of teaching material, including randomized exercises and exams. Carrying all this out online, from a web browser, minimizes the access threshold to the tool.

At present there are numerous software for structural calculation using the matrix method and/or by the finite element method, but most are proprietary software. This work presents an open-source software that, in addition, in its offline form is compatible with all major operating systems (Windows, Mac, and GNU/Linux), and in its online form works from any computer or modern mobile device. Another of its noteworthy feature is its ability to define nodal coordinates not concordant with the global coordinate axis [10], allowing the definition of sliders with arbitrary orientations.

The presented software is written in C++ 17 [11] and makes use of the well-known Eigen3 library [3] for calculations with dense and sparse matrices. Documentation and online tools are available on the project website [4] and the Git repository[1].

The rest of this article is organized as follows. Section 2 exposes the materials and methods, Sect. 3 presents the results of the work, and finally the conclusions are summarized in Sect. 4.

2 Methods

2.1 Direct Stiffness Matrix Method

This matrix method is not limited by the type of structure as other classical calculation methods are, by means of organizing all the information about the structure in the form of matrices. In addition, a greater number of unknowns can be solved compared to classic resolution methods in Strength of Materials [5] and with the benefit of being able to automate it. To be able to apply it to a continuous structure, it is necessary to model it through a discrete and finite set of variables. There is some freedom on the part of the engineer when choosing: (i) the way in which continuous bodies are divided into a discrete series of elements, and (ii) how many degrees of freedom (dof) each of these elements will have at the joints (or nodes). It should be noted that the use of the stiffness method is motivated by the greater ease of automation, allowing to define a library of predefined elements with stiffness matrices with known expressions from an earlier theoretical analysis [1, 7, 10].

2.2 Distributed Loads

The study of discretized problems implies that: (i) results are obtained only for the state vector dof, and (ii) only point loads can be defined. Therefore, any distributed

[1] URL: https://github.com/open-beam/openbeam.

loads must be converted into equivalent concentrated loads, for which the well-known methods in matrix calculus [10] are used using the Strength of Materials equations [5, 8, 9]. Regarding the axial, shear, bending and torsion moments, in our work we have opted to mesh the structures in sufficiently small elements, so that, by calculating the stresses at the ends of each element in a rigorous manner, linear piecewise stress diagrams are obtained. The following distributed loads have been implemented: uniform, trapezoidal, temperature variation, and non-nodal point.

2.3 Meshing

Meshing is the step in which a continuous solid is divided into a multitude of finite elements [7]. In our work, only basic meshing of linear elements (rods or beams) into smaller, also linear elements has been necessary, so the connectivity between elements after meshing is trivial as it is purely linear. There are two noteworthy aspects: (i) An element of a particular type (see Fig. 1), when meshed, can be converted into several elements of different types according to the degrees of freedom defined at its ends (e.g. when meshing a bi-articulated bar, an articulated-rigid element, several rigid-rigid elements, and finally a rigid-articulated element will be obtained), and (ii) distributed loads along a beam or bar must also be "meshed" to be distributed among the finite elements, which are the ones that are finally calculated.

2.4 Eigen3 Library

Eigen (version 3) is one of the most widely used C++ libraries in multiple engineering fields to represent and manipulate matrices, vectors, and tensors [3]. It allows two representation modes for matrices: dense and sparse. The formers are used in our work to represent the stiffness submatrices of finite elements, as well as generalized coordinate vectors. Sparse matrices are suited for stiffness matrices of very large structures, especially after meshing, since each element is typically only connected to a few neighboring elements. Once sparse matrices are represented, the resolution of canonical linear systems $Ax = b$, as required in our work, demands specifically algorithms to exploit the matrix sparsity and solve them in typically almost linear time $O(N)$ instead of cubic $O(N^3)$, with N the number of degrees of freedom. Two algorithms have been implemented in the library to solve this linear system: (i) the Cholesky decomposition (LL^T) algorithm for the K_{ff}, converted to a dense matrix, and (ii) Cholesky for the pure sparse matrix case [3]. By default, the dense matrix version is used, since most of the structures analyzed will be of a size small enough for the sparse method to not be computationally advantageous.

2.5 MRPT-opengl Library

MRPT ("Mobile Robot Programming Toolkit") is an open-software project that offers C++ libraries with algorithms and tools for mobile robot programming [12]. Its mrpt-opengl module offers a library for the generation of 3D graphics in a modular way through assembly and composition of basic visual primitives (lines, points, cylinders, etc.), chosen to generate and dynamically update the visualizations of structures.

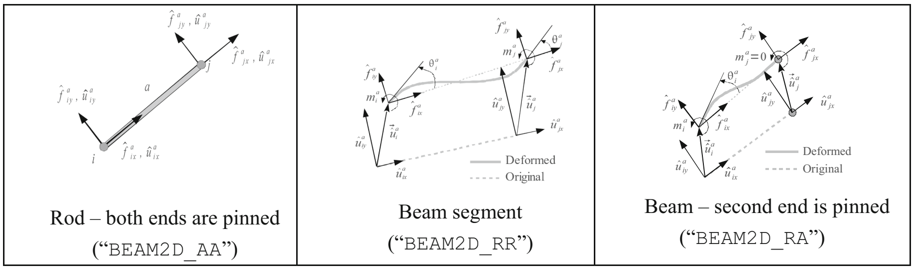


Fig. 1. A few of the available types of finite elements implemented in *libopenbeam*, together with their short names as specified in the structure definition files.

2.6 Emscripten

Emscripten [13] is a project, presented in 2011 and in active development, which provides a modified version of the clang compiler capable of cross-compiling from various languages (including C++) to JavaScript and WebAsm. By compiling MRPT, Eigen3, and OpenBeam all with Emscripten, web applications in Javascript have been developed that make use of all the high-level functions exposed in OpenBeam, such as analyzing a structure definition file, performing static analysis, or generating and updating its graphical representation in an html5 WebGL canvas [14].

3 Results

3.1 The Openbeam C++ Library

The main functionality developed in this work is integrated into a C++ library, which is used by the applications themselves, and which can be used by users interested in creating their own projects.

3.2 Implemented Finite Elements

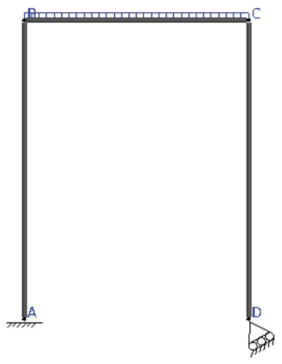
At present, implemented elements are eminently planar: rods, beams, and springs. Figure 1 illustrates a few of the implemented elements. As can be seen, they are all binary (connecting only two nodes) and each one makes use of a variable number of degrees of freedom at each of its two ends.

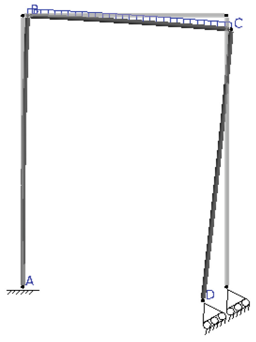
3.3 YAML Structure Definition Language

An example in Fig. 2 illustrates how users can describe the structure to analyze by means of our custom structure definition language in YAML [6], with all syntax details available online. Note that, whenever a numerical value is needed, mathematical expressions (algebraic operations, trigonometric functions, etc.) can be used at any point, as well as more complex constructions such as "if… Then… Else" if the user requires it (e.g. to define that a load only exists if a length meets certain criteria), allowing easy parameterization of structures.

```
parameters:
  G: 9.81
  L: 3.0
  H: 4.0
beam_sections:
- name: IPE200
  E: 2.1e11     # Young module
  A: 28.5e-4    # Area
  Iz: 1940e-8   # Second moment of area in z
nodes:
- {id: 0, coords: [0   , 0], label: A}
- {id: 1, coords: [0   , H], label: B}
- {id: 2, coords: [L   , H], label: C}
- {id: 3, coords: [L   , 0], label: D, rot_z: 30.0}
elements:
- {type: BEAM2D_RR, nodes: [0, 1], section: IPE200}
- {type: BEAM2D_RR, nodes: [1, 2], section: IPE200}
- {type: BEAM2D_RR, nodes: [2, 3], section: IPE200}
constraints:
- {node: 0, dof: DXDYRZ}
- {node: 3, dof: DY}
element_loads:
- {element: 1, type: DISTRIB_UNIFORM, q: 1000*G, DX: 0, DY: -1, DZ: 0}
```

(a)





(b) (c)

Fig. 2. (a) Example of the YAML structure definition language used in OpenBeam. Visual representation of the structure in (b) its original state, and (c) deformed under loads.

3.4 Applications

A command-line off-line program called "ob-solve" has been developed, with more than thirty arguments and flags to: load a structure from a YAML file, show the results of the static analysis to the console or to an HTML file, show the stiffness matrices, generate visualizations of the structure in its original or deformed states, etc. Examples of the results of this program are provided in an online repository[2]. The on-line version of this program is more interactive and is designed to be used by students and professors in the most intuitive way.

[2] URL: https://ingmec.ual.es/openbeam/fem/.

4 Conclusions

Several objectives have been achieved with this work: a new open-source library with a high-level API to define mechanical structures and solve static analysis of them. Two ready-to-use tools are also presented: (i) the command line program with potential for both students and professors, and (ii) the web page with the online tool, which makes it, to the best of the authors' knowledge, the first application capable of running on the web and mobile devices that allows arbitrarily complex structures to be defined and calculated, completely free of charge and with open-source code.

References

1. Rubinstein, M.F.: Matrix Computer Analysis of Structures. Prentice Hall (1966)
2. Ghali, A., Neville, A.M., Brown, T.G.: Structural Analysis: A Unified Classical and Matrix Approach, 6th edn. CRC Press (2017)
3. Guennebaud, G., Jacob, B., et al.: Eigen v3, https://eigen.tuxfamily.org/ (2010). Last accessed 1 Feb 2023
4. OpenBeam Homepage: https://open-beam.github.io/openbeam/. Last accessed 20 Mar 2022
5. Blanco-Claraco, J.L., Garrido Jiménez, F.J., López Martínez, J., Jiménez Alonso, J.F., Hernández Díaz, A.M.: Resistencia de materiales: Resumen de teoría y problemas resueltos, vol. 7. Editorial Universidad Almería (2016)
6. Ben-Kiki, O., Evans, C., Ingerson, B.: YAML Ain't Markup Language (YAML) Version 1.2 (2009)
7. Liu, G.R., Quek, S.S.: The Finite Element Method: A Practical Course. Butterworth-Heinemann (2013)
8. Ortiz-Berrocal, L.: Resistencia de materiales. McGraw-Hill (2007)
9. Vázquez, M.: Resistencia de Materiales, 4ª edn. Noela (2008)
10. Blanco-Claraco, J.L., González, A., García-Manrique-Ocaña, J.M.: Análisis estático de estructuras por el método matricial. Servicio de Publicaciones e Intercambio Científico de la Universidad de Málaga, (2012)
11. Smith, R.: Working Draft, Standard for Programming Language C++ N4659. Google Inc, 03–21 (2017)
12. Blanco-Claraco, J.L. et al.: Mobile robot programming toolkit (MRPT) (2022). https://www.mrpt.org. Last accessed 9 Nov 2022
13. Zakai, A.: Emscripten: an LLVM-to-JavaScript compiler. In: Proceedings of the ACM International Conference Companion on Object Oriented Programming Systems Languages and Applications Companion, pp. 301–312 (2011)
14. Parisi, T.: WebGL: Up and Running. O'Reilly Media, Inc. (2012)