Finite Element Based Structural Analysis

- Group No 7 -

Wishma Herath
Suneth Samarasinghe
Pubudu Premathilaka

Department of Computer Engineering
University of Peradeniya

CO328 : Project report
Semester 6

June 2020

Introduction

What is a structure?

A structure, as it relates to civil engineering, is a system of interconnected members used to support external loads. There are several types of civil engineering structures, including buildings, bridges, towers, arches, and cables. Members or components that make up a structure can have different forms or shapes depending on their functional requirements. Structural members can be classified as beams, columns and tension structures, frames, and trusses.

- Beams are structural members whose longitudinal dimensions are appreciably greater than their lateral dimensions.
- Columns are vertical structural members that are subjected to axial compression.
- Frames are structures composed of vertical and horizontal members
- Trusses are structural frameworks composed of straight members connected at the joints
- The cross-section of a beam and columns can be rectangular, circular, or triangular, or it can be of what is referred to as standard sections, such as channels, tees, angles, and I-sections

What is structural analysis?

Structural analysis is the prediction of the response of structures to specified arbitrary external loads. During the preliminary structural design stage, a structure's potential external load is estimated, and the size of the structure's interconnected members is determined based on the estimated loads. Structural analysis establishes the relationship between a structural member's expected external load and the structure's corresponding developed internal stresses and displacements that occur within the member when in service. This is necessary to ensure that the structural members satisfy the safety and the serviceability requirements of the local building code and specifications of the area where the structure is located.

What is the finite element method?

The finite element method (FEM) is the most widely used method for solving problems of engineering and mathematical models. Typical problem areas of interest include the traditional fields of structural analysis, heat transfer, fluid flow, mass transport, and electromagnetic potential. The FEM is a particular numerical method for solving partial differential equations in two or three space variables (i.e., some boundary value problems). To solve a problem, the FEM subdivides a large system into smaller, simpler parts that are called finite elements. This is achieved by a particular space discretization in the space dimensions, which is implemented by the construction of a mesh of the object: the numerical domain for the solution, which has a finite number of points.

The finite element method formulation of a boundary value problem finally results in a system of algebraic equations. The method approximates the unknown function over the domain. The simple equations that model these finite elements are then assembled into a larger system of equations that models the entire problem. The FEM then uses variational methods from the calculus of variations to approximate a solution by minimizing an associated error function.

Project Overview

For structural analysis in civil engineering, already there are tools developed and some of them are widely used in the field such as SAP 2000 and Opensees. SAP 2000 is used for linear analysis and Opensees facilitates Non-linear analysis as well. This project is on building a similar structural analysis and simulation software with some additional features in collaboration with the Department of Civil Engineering.

The Purpose of the Project

Currently, SAP2000 is the most widely used software for linear analysis of structures. It provides the capability of observing interactions between Axial force and Bending moment within the analysis. In addition to these two parameters, Shear force is another important measurement in the structural analysis which is not provided in SAP2000. One of the researches done by the department of civil engineering found a novel formulation to observe the behavior of Shear force in structures. The purpose of the project is to build a SAP2000 like software from scratch which facilitates the above-mentioned feature.

The Scope of the Work

The final version of the software with both front end and back end. The front end will take a look at a professional CAD software with a graphical user interface to input the structure by drawing on a canvas. The front end is developed by another party in the development team. In this phase of the project, the scope is to develop the software to be able to perform linear analysis of 3-dimensional structures. Basically, the necessary inputs (structure and material details) are given to the program as a JSON file. The results of the analysis are shown with a graphical representation of the structure.

Stakeholders

The Client: A research team from the department of civil engineering, University of Peradeniya. The Customer or Hands-on Users of the Product: The software is a general-purpose civil-engineering software ideal for anyone who does analysis and design of engineering constructions or other facilities.

Requirement Analysis

Actors

- User

Use cases

- Define material models
- Define cross-section types
- Draw the structure
- Apply loads and necessary constraints
- Start analysis

Use Case Diagram

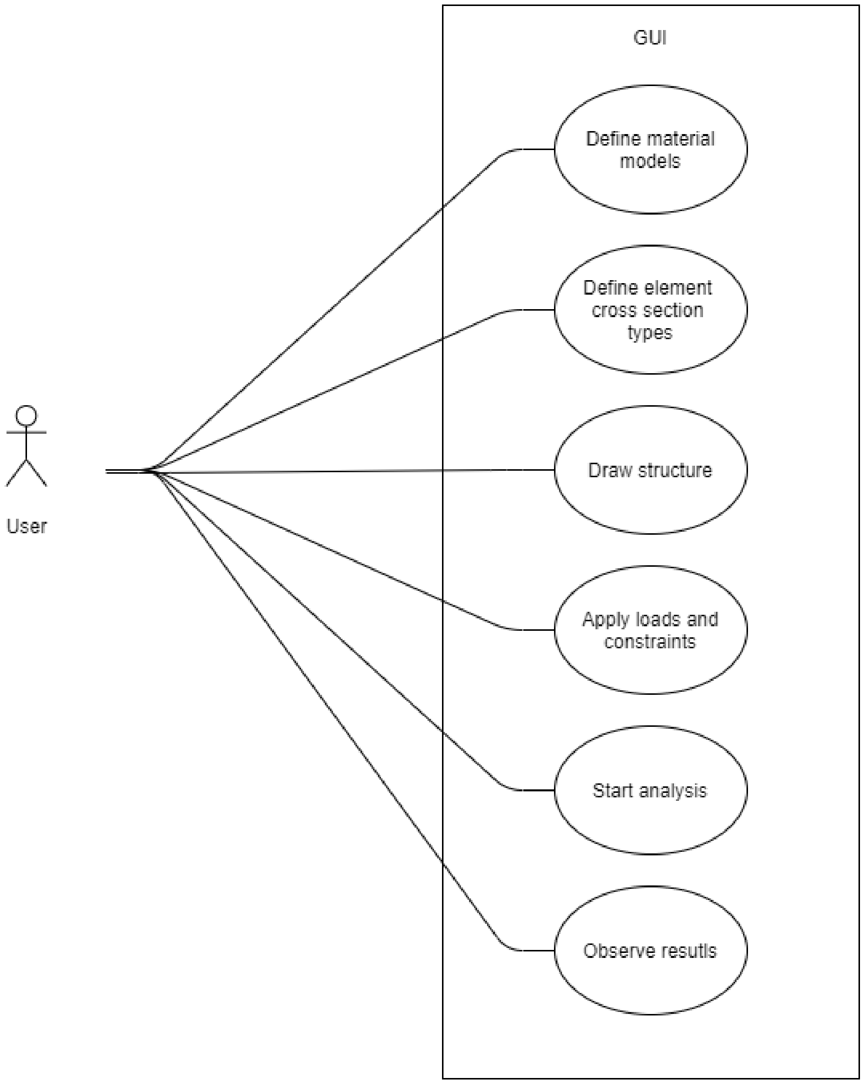


Fig. 1 Use Case Diagram

Activity Diagram

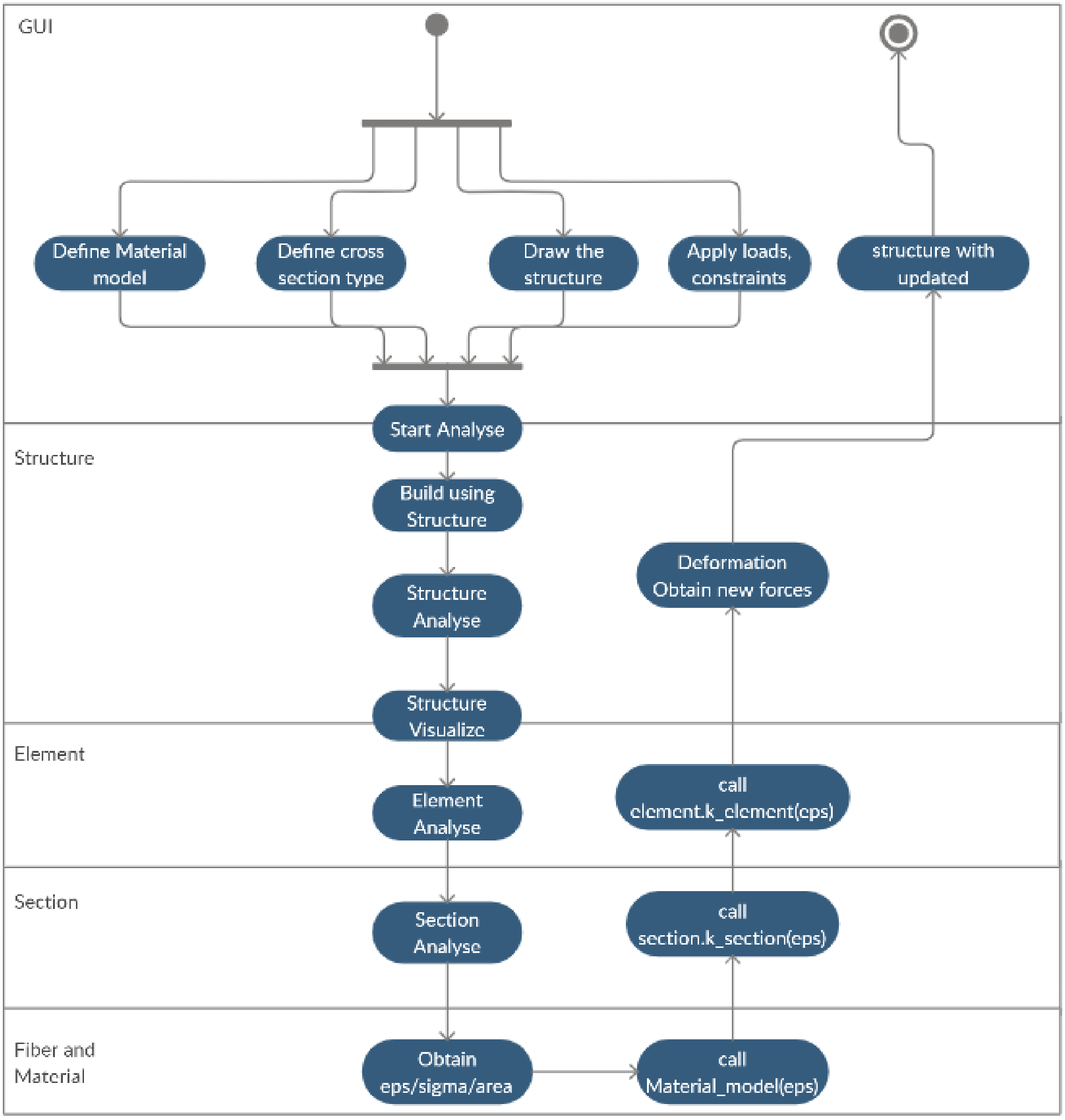


Fig. 2 Activity Diagram

Functional Requirement

Under the functional requirements different capabilities of the software is described.

- The user is facilitated by an interactive graphical user interface and this GUI allows the user to draw 2D, 3D structures.
- The analysis of the structure can be done either in Linear region or Non-linear region.
- The user has the ability to choose in which region he would like to perform the analysis
- Different material models can be choosed.
- If user didn't choose any material model, Steel model, and Concrete model are used by default
- User inputs can be categorized under the following sections.
  1. Element inputs
     - Cross section type : The user can choose the cross section type of the element
     - Start Node/ End Node: By choosing the start node and end node of each element, software backend will number each node accordingly such that these numbering order will be used in the algorithm as well
     - Angle: In the proposed GUI angle of each element is calculated and written into a json file and it is used in the calculation. For CLI version the user can be given the angle as input to the json file directly.
     - Length: In both GUI version and CLI version user has the specify the length of each element
  2. Section
     - Cross section type: Element cross section type can be selected
     - Fiber: Fiber type and number of fiber divisions per cross section can be choosed
  3. Cross section
     - Type: For the proposed implementation different cross sections are facilitated to the user. Also custom user defined cross section can be created by the user and it is also possible to use.

4. Fiber
   - Width and Height: User has to specify the width and the height of the fiber and the fiber area is calculated by the system and saved to the json file for algorithmic calculations
   - Fiber material type
5. Material
   - Type: User has facilitated a set of materials that can be used among them different materials can be used to do the analysis. Also for each of these materials data is taken from standard material models. Eg: (Young's modulus)

Non-Functional Requirements

SAP 2000 is an industry level similar software to this implementation in which the formulation is made using the axial force and bending moment interaction. The only request made by the client-side(Dept of Civil Engineering) is to build a similar software with similar performance, but with the novel formulation introduced by them.

1. Performance Requirements
   The performance requirement of this implementation should compete with SAP 2000. As of the client's requirements, the performance analysis can be done as follows.

2. Hardware Requirements
   Since this is an algorithmic implementation, the weight of the speed comes under the hardware platform specifications. The algorithmic implementation is divided into linear and non-linear analyses. Compared to linear analysis, non-linear analysis has immense calculations which include larger matrix multiplications, numerical iterations from fiber level to the structure level. Therefore, a general computer should have the following minimum requirements to run non-linear analysis of FEM. The problem size that can be solved and the speeds for solution response recovery increase considerably with more RAM.

3. Precision or Accuracy Requirements
   According to the client, for a given basic cantilever beam, software should output results within 0-3 % error for linear analysis. For nonlinear analysis it can go upto nearly 5%. This estimation is done by manual calculation and compare that result with software output.

Design

- Design goals
  - Design for understandability and maintenance
  - Design for change
  - Design for reuse
  - Design for robustness
- Design strategies to achieve design goals
  - Information hiding (hide likely changes)
  - Low coupling (reduce dependencies)
  - High cohesion (one purpose per class)
  - Low Representational Gap (align requirements and implementation)
- Trade-offs in designs with design goals and strategies
  Benefits of a modularized approach are

  1. **Ease of Use:**
     Allows simplicity, as rather than focusing on the entire lines of code in one go we can access it in the form of modules. This allows ease in debugging the code and prone to fewer errors.
  2. **Reusability:**
     Allows the user to reuse the functionality with a different interface without typing the whole program again.
  3. **Ease of Maintenance:**
     Helps in less collision at the time of working on modules
  4. **Collaboration:**
     Helps to work with proper collaboration while working on this project.
  5. **Scalability:**
     The ability of the software to adapt to increased demands, whether in terms of performance, maintenance, availability

If the modularization is not implemented, for some specific situations like smaller structures storing each element parameter in arrays and doing mathematical operations on them can be computationally more effective whereas, modularization can result in an overhead of generating the matrices needed beforehand the calculation. Adapting the proper use of both modularized and utilization of arrays to store the data whenever is necessary can improve the performance of the software.

How Strategies support the above goals

1. Modules reusability

   - Modular Composability, compose modules from different sources
   - Good modules contain well-defined tasks reusable in many contexts
     - Limited dependencies on other modules (low coupling)
     - Useful cohesive functionality that's worth reusing

2. Coupling
   Modules should depend on as few other modules as possible

   - Easier to understand (little context to understand)
   - Independent modules easier to modify without rippling effects (Design for Change)
   - Easier to reuse in different context

3. Cohesion
   All responsibilities of a module should be related and well defined – one purpose per class

   - A module with lots of unrelated functionality is unlikely to be reusable as it (or unnecessarily large)
   - A cohesive module is easier to understand

   As one of our design strategy we have chosen design alternatives with lower coupling and higher cohesion

4. Low Representational Gap
   Supports design for change: if problem structure and solution structure similar, problem changes should correspond to solution changes

Use of Design patterns

1. Creational Patterns

   - Singleton design pattern
     The Material Model class has only one instance in the software and a global point of access is provided. So that other object instances of other classes can get necessary details about the material models at any point of the calculations. This only one instance is created at the beginning of the software according to the material details provided by the user(Early instantiation)

2. Structural Patterns

   - Composite Pattern
     The real-life structure is recreated in the software virtually corresponding to the composition of objects in the structure. First an structure object is created, in that a list of Element objects are created, For each Element, list Section objects are created and Finally for each Section, list of Fiber objects are created.

Implementation details

Overview

Some terminology used this point onwards:

Structure Level: a system of interconnected element members used to support external loads

Element Level: elements are referred to as either a beam or a column.

Section Level: section is the part that an element is divided into for the simplicity of analysis

Fiber Level: fiber is the smallest unit to be analyzed and it is defined on the plane of the cross-section of an element. In a reinforced element they can be either concrete or steel.

Degrees of freedom(DOF): "possibility" to move in a defined direction.

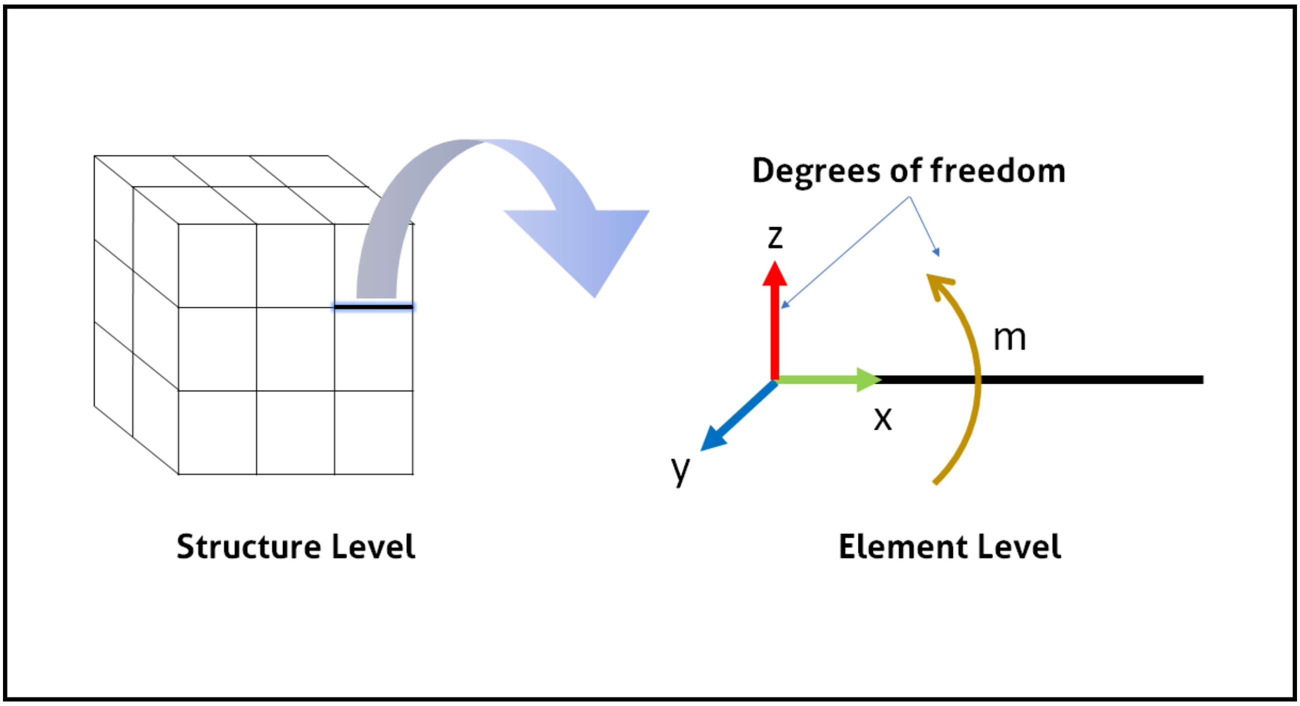


Fig. 3 Terminology

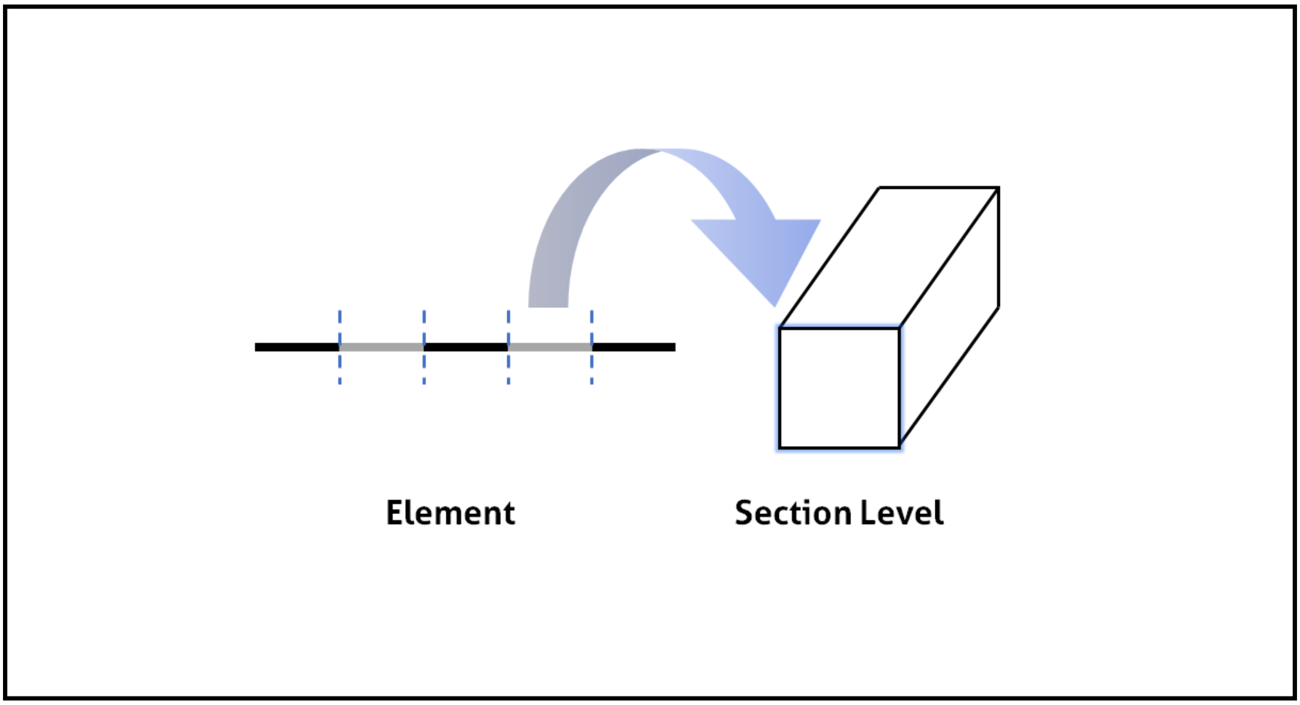


Fig. 4 Terminology

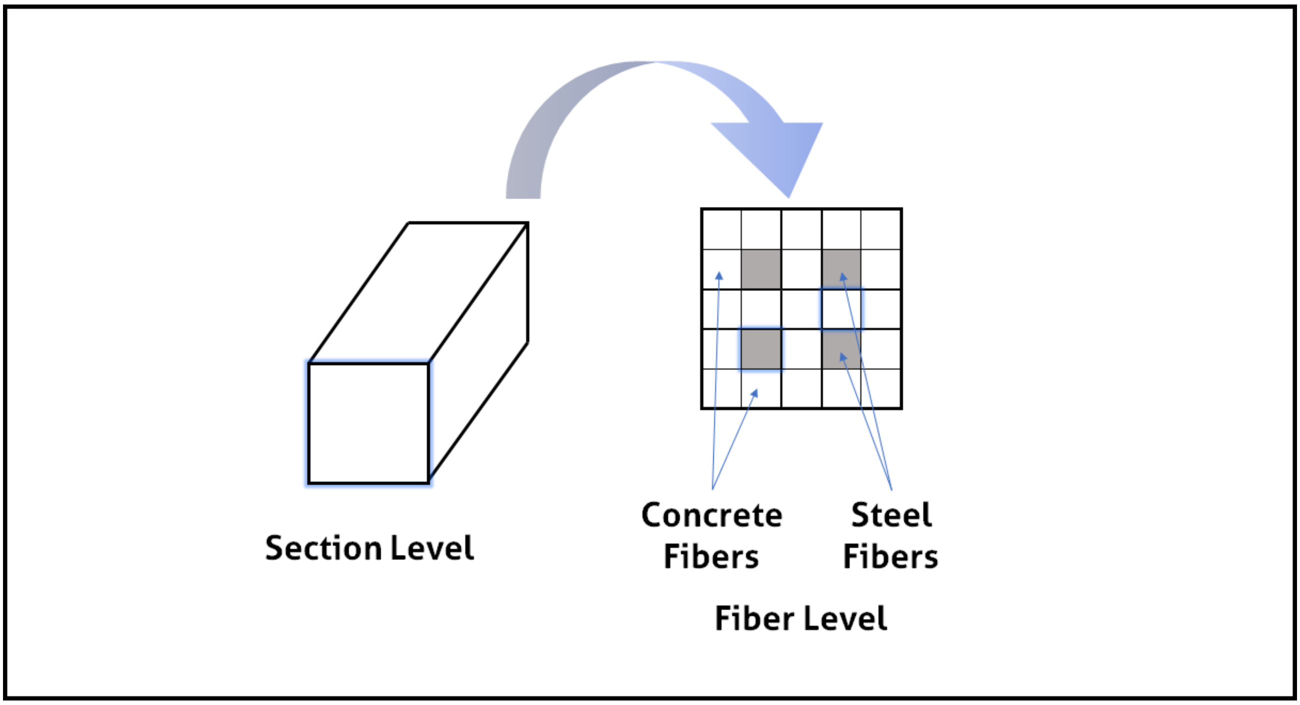


Fig. 5 Terminology

The arrangement of classes should be in a way that it is easy to load the structure from the given JSON file, and build it virtually and to manipulate to do the calculations required for the analysis in an optimally efficient way.

A beam with multiple members framing into it is created as a single object, just as it exists in the real world, and the meshing needed to ensure that connectivity exists with the other members is handled internally by the program.

Classes separately

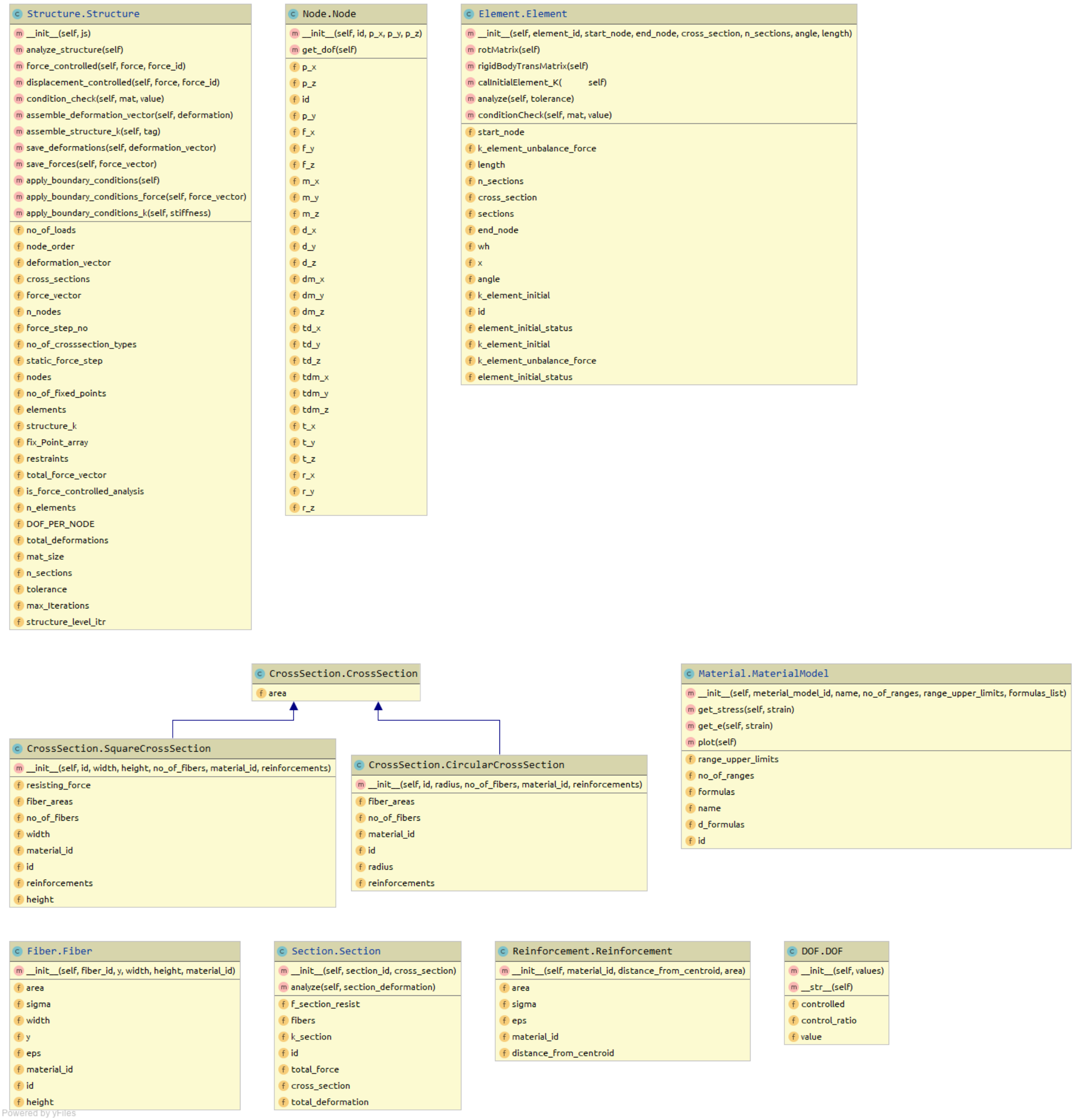

Structure.Structure
__init__(self, js)
analyze_structure(self)
force_controlled(self, force, force_id)
displacement_controlled(self, force, force_id)
condition_check(self, mat, value)
assemble_deformation_vector(self, deformation)
assemble_structure_k(self, tag)
save_deformations(self, deformation_vector)
save_forces(self, force_vector)
apply_boundary_conditions(self)
apply_boundary_conditions_force(self, force_vector)
apply_boundary_conditions_k(self, stiffness)
no_of_loads
node_order
deformation_vector
cross_sections
force_vector
n_nodes
force_step_no
no_of_crosssection_types
static_force_step
nodes
no_of_fixed_points
elements
structure_k
fix_Point_array
restraints
total_force_vector
is_force_controlled_analysis
n_elements
DOF_PER_NODE
total_deformations
mat_size
n_sections
tolerance
max_Iterations
structure_level_itr
Node.Node
__init__(self, id, p_x, p_y, p_z)
get_dof(self)
p_x
p_z
id
p_y
f_x
f_y
f_z
m_x
m_y
m_z
d_x
d_y
d_z
dm_x
dm_y
dm_z
td_x
td_y
td_z
tdm_x
tdm_y
tdm_z
t_x
t_y
t_z
r_x
r_y
r_z
Element.Element
__init__(self, element_id, start_node, end_node, cross_section, n_sections, angle, length)
rotMatrix(self)
rigidBodyTransMatrix(self)
calInitialElement_K(self)
analyze(self, tolerance)
conditionCheck(self, mat, value)
start_node
k_element_unbalance_force
length
n_sections
cross_section
sections
end_node
wh
x
angle
k_element_initial
id
element_initial_status
k_element_initial
k_element_unbalance_force
element_initial_status
CrossSection.CrossSection
area
CrossSection.SquareCrossSection
__init__(self, id, width, height, no_of_fibers, material_id, reinforcements)
resisting_force
fiber_areas
no_of_fibers
width
material_id
id
reinforcements
height
CrossSection.CircularCrossSection
__init__(self, id, radius, no_of_fibers, material_id, reinforcements)
fiber_areas
no_of_fibers
material_id
id
radius
reinforcements
Material.MaterialModel
__init__(self, meterial_model_id, name, no_of_ranges, range_upper_limits, formulas_list)
get_stress(self, strain)
get_e(self, strain)
plot(self)
range_upper_limits
no_of_ranges
formulas
name
d_formulas
id
Fiber.Fiber
__init__(self, fiber_id, y, width, height, material_id)
area
sigma
width
y
eps
material_id
id
height
Section.Section
__init__(self, section_id, cross_section)
analyze(self, section_deformation)
f_section_resist
fibers
k_section
id
total_force
cross_section
total_deformation
Reinforcement.Reinforcement
__init__(self, material_id, distance_from_centroid, area)
area
sigma
eps
material_id
distance_from_centroid
DOF.DOF
__init__(self, values)
__str__(self)
controlled
control_ratio
value
Powered by yFiles

Fig. 6 Classes

Classes Diagram

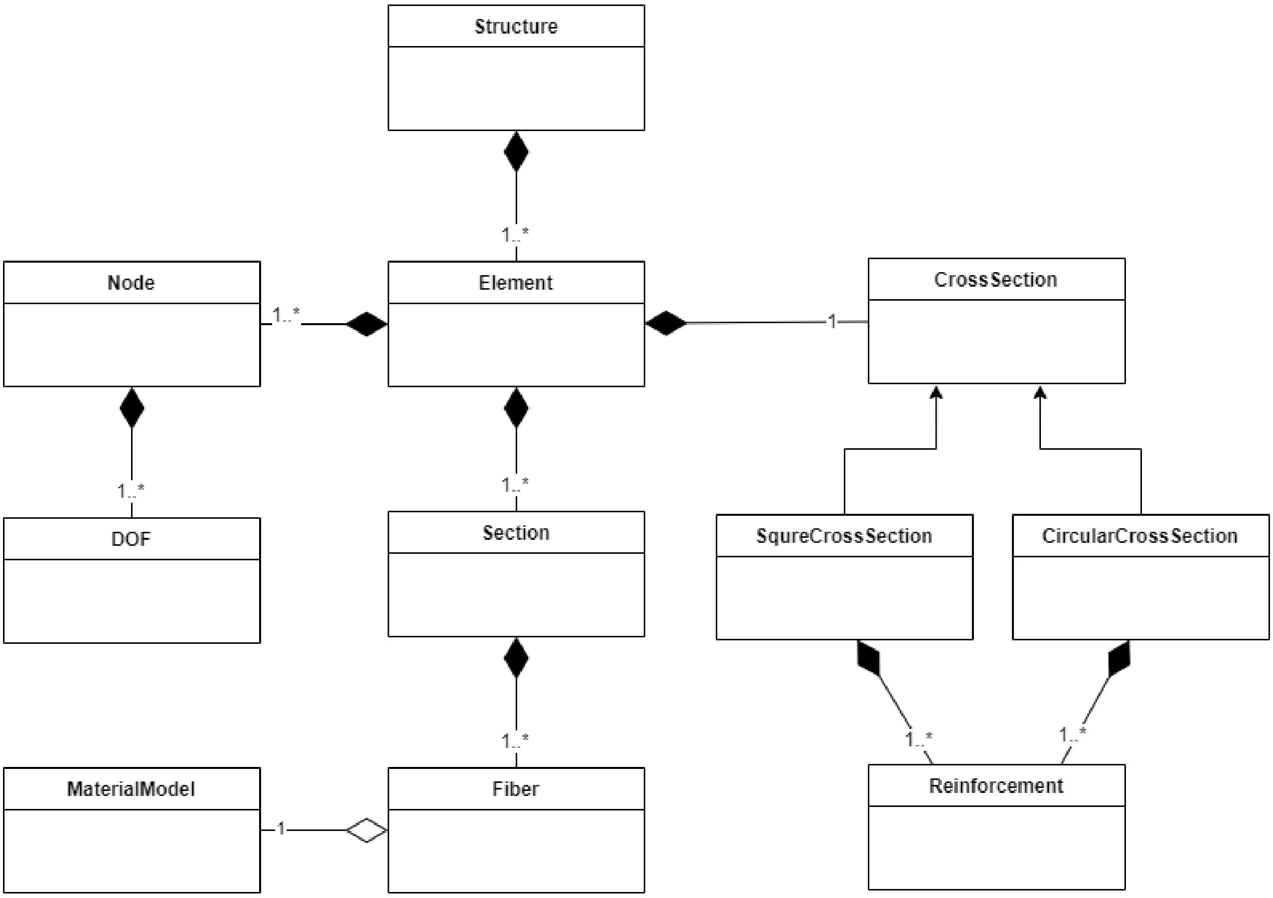


Fig. 7 Class Diagram

Sequence Diagram

Following sequence diagram is for a case from the user entering the inputs to the software using GUI to obtain the deformations and updated forces after the analysis.

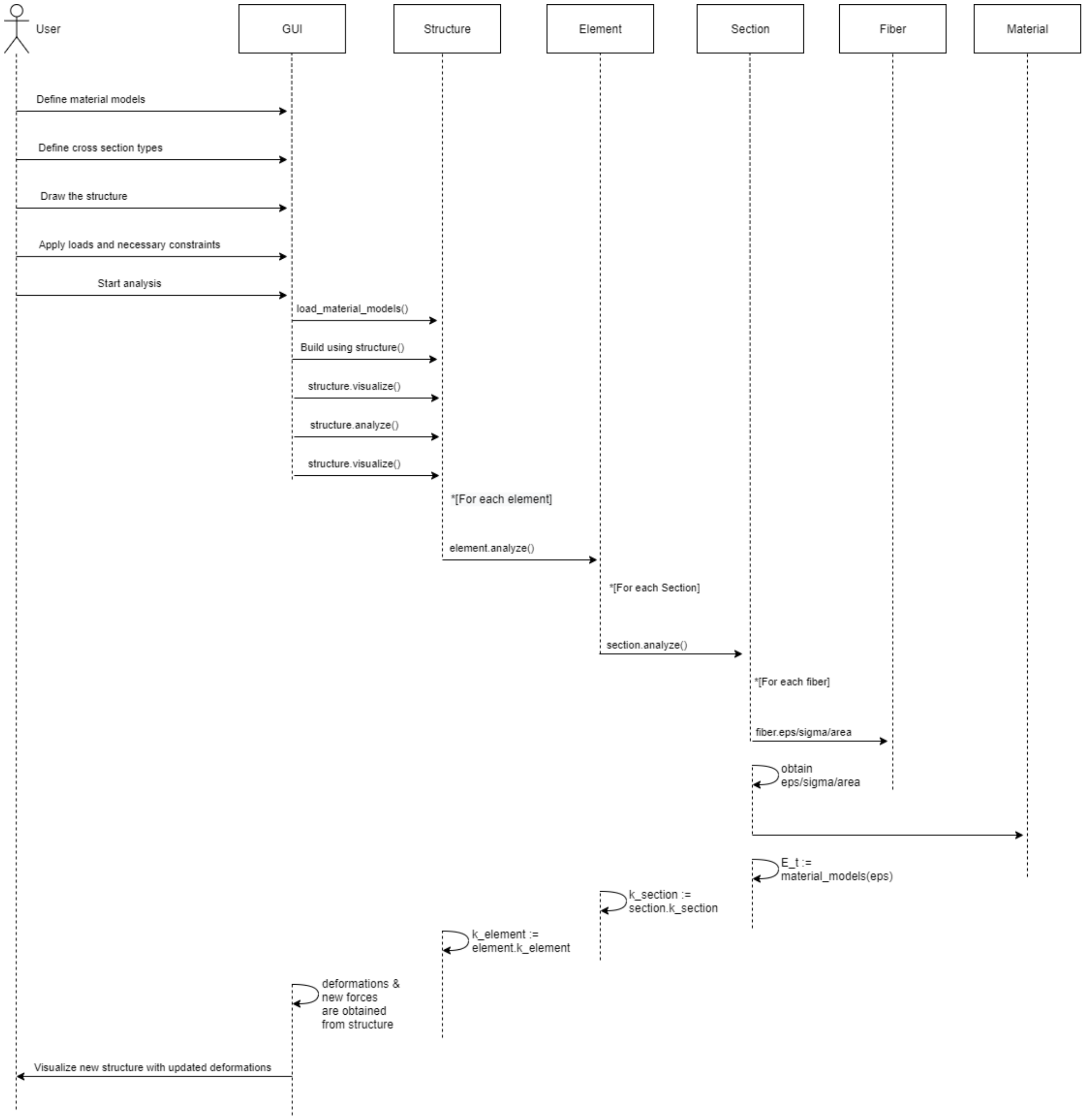


Fig. 8 Sequence Diagram

Collaboration diagram

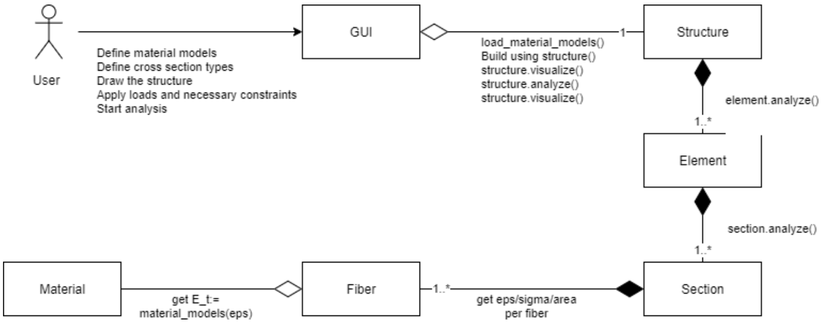
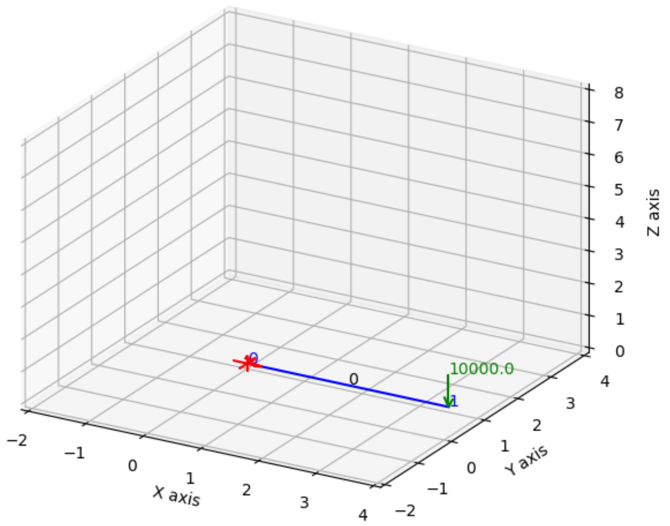


Fig. 9 Collaboration diagram

Unit Testing

Approach

Unit testing is performed using the simplest structure which consists of only one element fixed at one end restraining all translational degrees of freedom in X, Y, Z directions. It is subjected to a load of 10000N in -Z direction at the other end.



Features to be unit tested

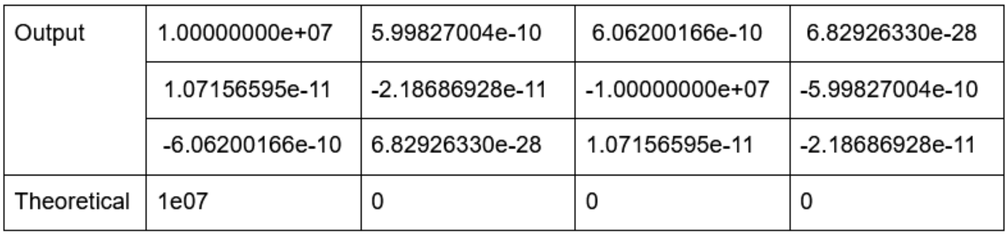
1. Structure Level
2. Element Level
3. Section Level

For each of the above levels there are iteration steps to pick components from the lower level and do the analysis at that level. For the simplicity of each step, only the outputs of each level of 0th iteration and hand calculated theoretical values are recorded.

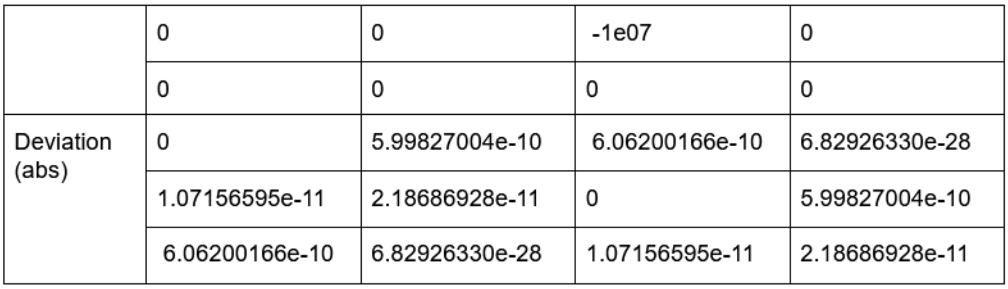
Results

Structure level iteration: 0

Element stiffness



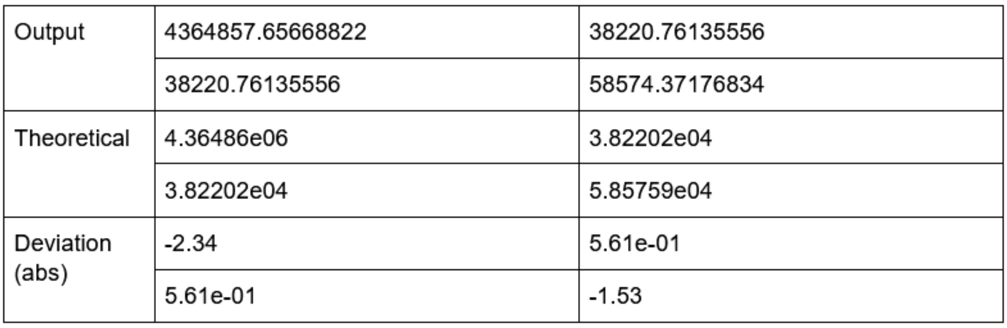
Output	1.00000000e+07	5.99827004e-10	6.06200166e-10	6.82926330e-28
	1.07156595e-11	-2.18686928e-11	-1.00000000e+07	-5.99827004e-10
	-6.06200166e-10	6.82926330e-28	1.07156595e-11	-2.18686928e-11
Theoretical	1e07	0	0	0



	0	0	-1e07	0
	0	0	0	0
Deviation (abs)	0	5.99827004e-10	6.06200166e-10	6.82926330e-28
	1.07156595e-11	2.18686928e-11	0	5.99827004e-10
	6.06200166e-10	6.82926330e-28	1.07156595e-11	2.18686928e-11

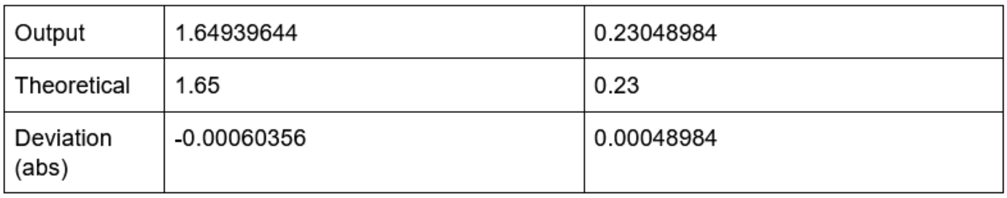
Element level iteration: 0

Section stiffness



Output	4364857.65668822	38220.76135556
	38220.76135556	58574.37176834
Theoretical	4.36486e06	3.82202e04
	3.82202e04	5.85759e04
Deviation (abs)	-2.34	5.61e-01
	5.61e-01	-1.53

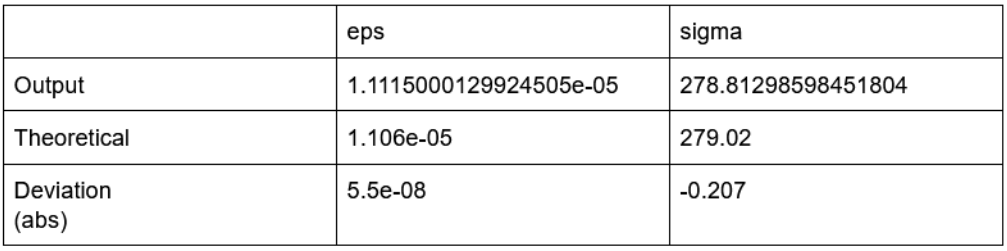
Unbalanced force



Output	1.64939644	0.23048984
Theoretical	1.65	0.23
Deviation (abs)	-0.00060356	0.00048984

Section level iteration: 0

Fiber details



	eps	sigma
Output	1.1115000129924505e-05	278.81298598451804
Theoretical	1.106e-05	279.02
Deviation (abs)	5.5e-08	-0.207