1. Computational Toolbox

In view of unavailability of an IFEM computational package, developing a comprehensive toolbox has been an inevitable part of this project. First, a general-purpose FEM toolbox has been developed in MATLAB and a functional Graphical User Interface (GUI) has been added to handle the process of finite element analysis. Afterwards, the described IFEM procedure has been implemented in MATLAB using the interval toolbox INTLAB to handle interval computations. This part of the toolbox is partly based on refactoring the code developed for analyzing inverse problems. Then, a GUI has been developed to facilitate the modeling and analysis procedure.

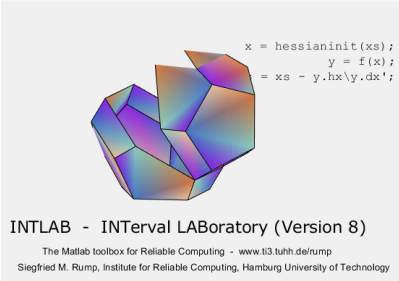

Figure 1. Running INTLAB toolbox

1.1 Implementation in MATLAB

In this project, MATLAB is chosen over other programming languages mainly because it is easier to follow and expand by other researchers and engineers working in this area. MATLAB has a great set of scientific computing libraries including numerical linear algebra which is extremely required in computational mechanics problems and useful for prototyping. On the other side, it has some drawbacks, such as having a single global namespace, difficulties in handling polymorphism, and limited GUI library, which some of them can be tackled through integrating with other programming languages like C++ and Java.

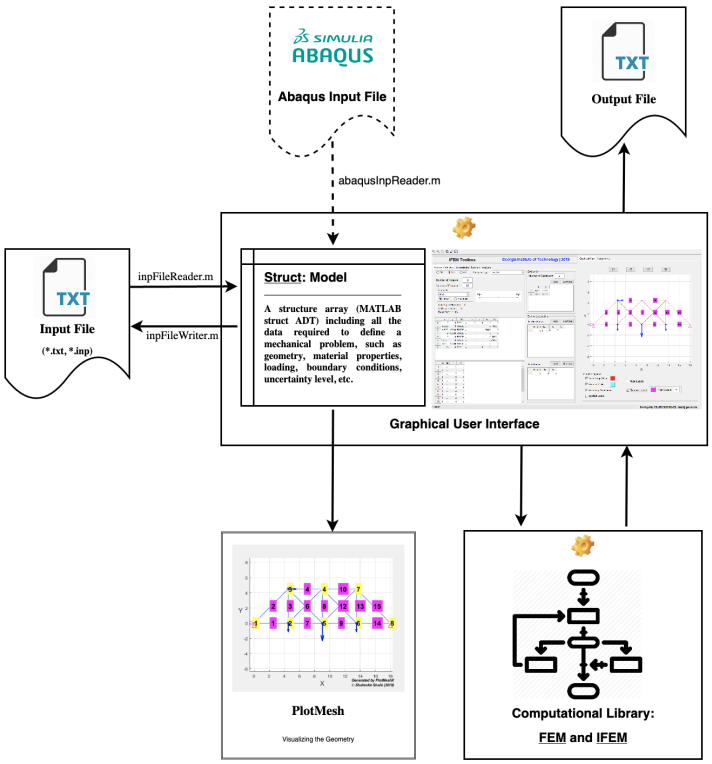

Figure 2. The main components of the toolbox

This computational software is comprised of various features. Figure (2) illustrates a diagram of main components and their interactions. The key part of this diagram is a structure array called Model. In general, a structure array is a data type that groups the related data using data containers known as fields, which is very similar to struct objects in C++. In this package, the struct Model is defined to encapsulate all the data required to define a mechanical problem including geometry such as node coordinates and element connectivity, material properties such as modulus of elasticity and cross-sectional areas, boundary conditions, loading, uncertainty levels, etc. Unifying the data definition in the program, such encapsulation in data is less prone to errors and can effectively be passed from one function to another.

A user can input data through either creating an input file by using specific keywords or employing the GUI to define the mechanical problem, interactively. It is also possible to employ a combination of these two approaches, i.e. inputting some data by introducing an input file and then adding or modifying some data within the GUI. The visualizing tools in the GUI helps to check the correctness of the problem definition.

Once the problem definition is completed, the proper function can be called by the interface to analyze the defined problem and displaying the results on the screen or storing them in an output file. The computational libraries include implementations of both FEM and IFEM formulations.

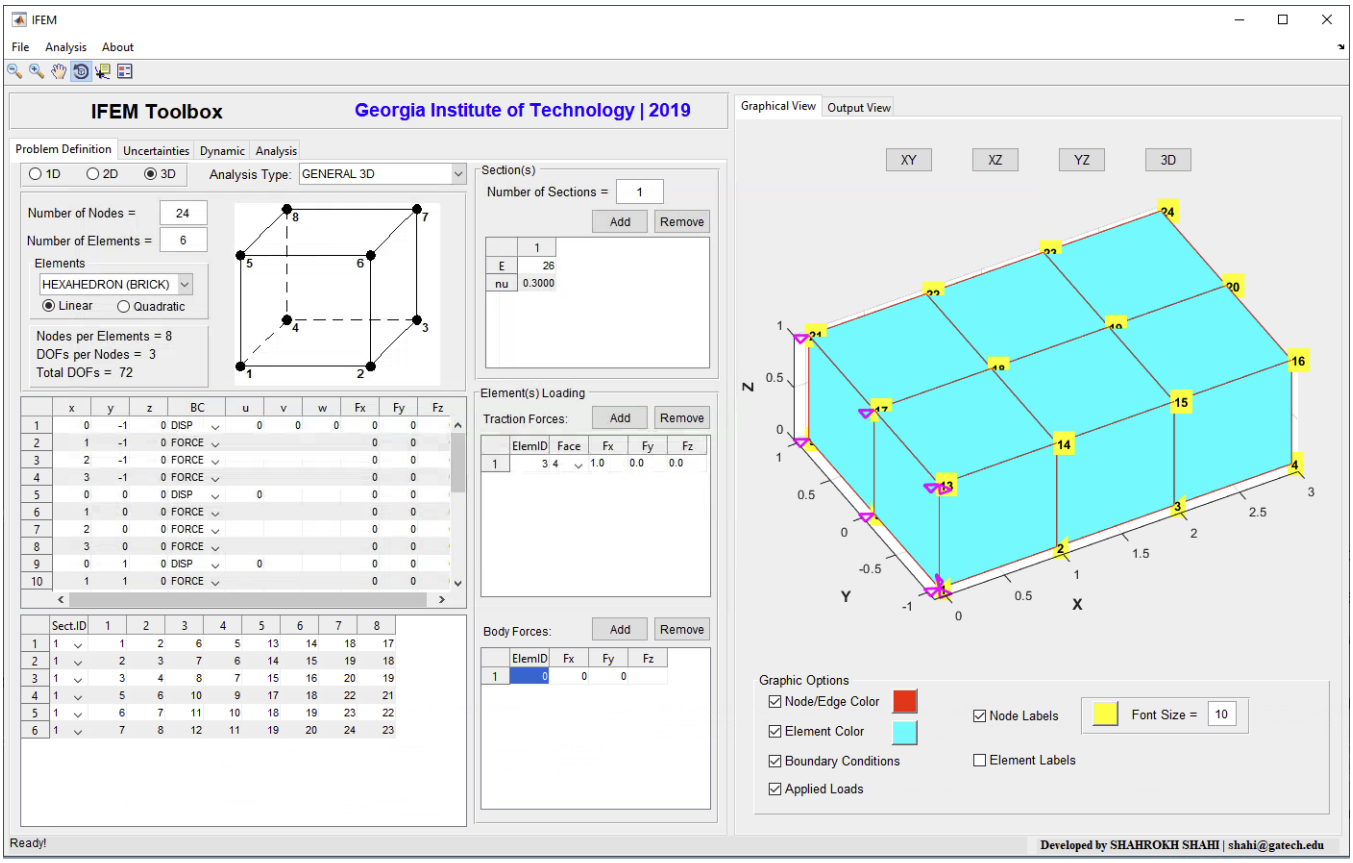

Figure 3. Graphical User Interface main menu

In practice, the source code is structured in a set of directories which their path and dependencies need to be defined in MATLAB. Moreover, this package includes an external toolbox INTLAB to handle the interval computations which requires a set of environment variables depending on the operating system. To facilitate this procedure, a driver function is developed to define all the dependencies and set all the required environmental variables in the first execution of the software (Figure (4)).

Another feature which is added to this package is reading input files generated by Abaqus, a commercial finite element analysis software. In Abaqus/CAE software, after pre-processing step, an input file will be generated including all the geometry i.e. mesh and boundary conditions, material properties, and loading information. In general, this file will be internally processed by Abaqus processor. We can take most advantage of the great pre-processing tools in Abaqus environment for creating a finite element model, if the internal input file can be parsed to extract model features. This is one of the very first steps to integrate our IFEM package to commercial packages like Abaqus and NASTRAN.

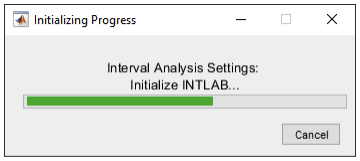

Figure 4. GUI initializing driver

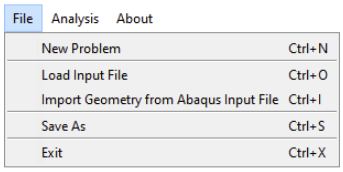

Figure 5. GUI menu; inserting geometry from Abaqus input files

1.2 Graphical User Interface

MATLAB supports developing applications with graphical user interface features. In this project, a MATLAB built-in feature, GUIDE, has been employed as the GUI development environment which is tightly integrated graph-plotting features. However, this GUI library is very limited. Even though MathWorks, the company that manufactured the MATLAB product, has introduced App Designer as an alternative to GUIDE, it still cannot provide a comprehensive library for designing a general-purpose software. This limitation can be overcome by integrating Java. In general, MATLAB programming environment uses Java for numerous tasks including graphical user interface and its internal Java classes can be accessed and used by users. Unlike MATLAB's interface with other programming language like C++, the internal Java classes and the MATLAB-Java interface were never fully documented by MathWorks. In this project, the original GUI controls are improved by adding or modifying corresponding Java codes, whenever it is required.

Designing an interface includes two primary tasks of application building – laying out the visual components (controls) and programming the behavior. In this software, the priority is keeping the appearance of the interface as simple and practical as possible to enable users to easily define the problem and obtain the analyzing results, in a step-by-step fashion. In the programming of controls behaviors, modular programming approach is employed to reduce the redundancy in the code and improve the debugging and maintenance procedures.

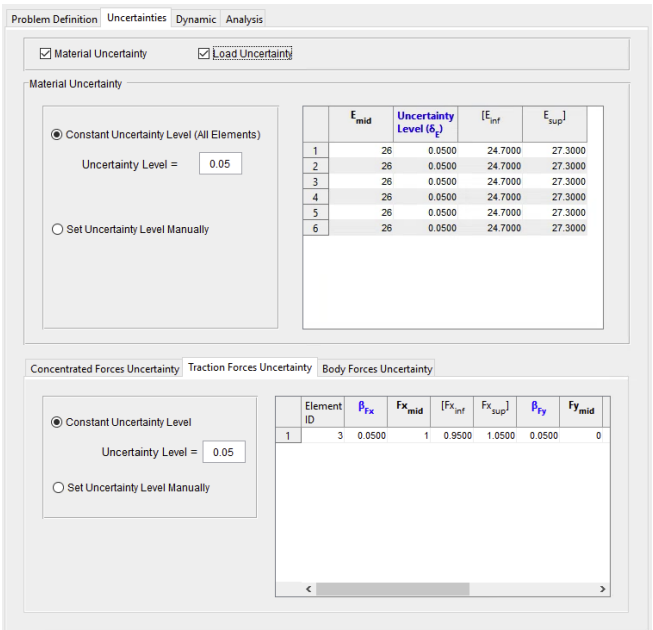

Figure 6. Defining uncertainties in material properties and applied loads