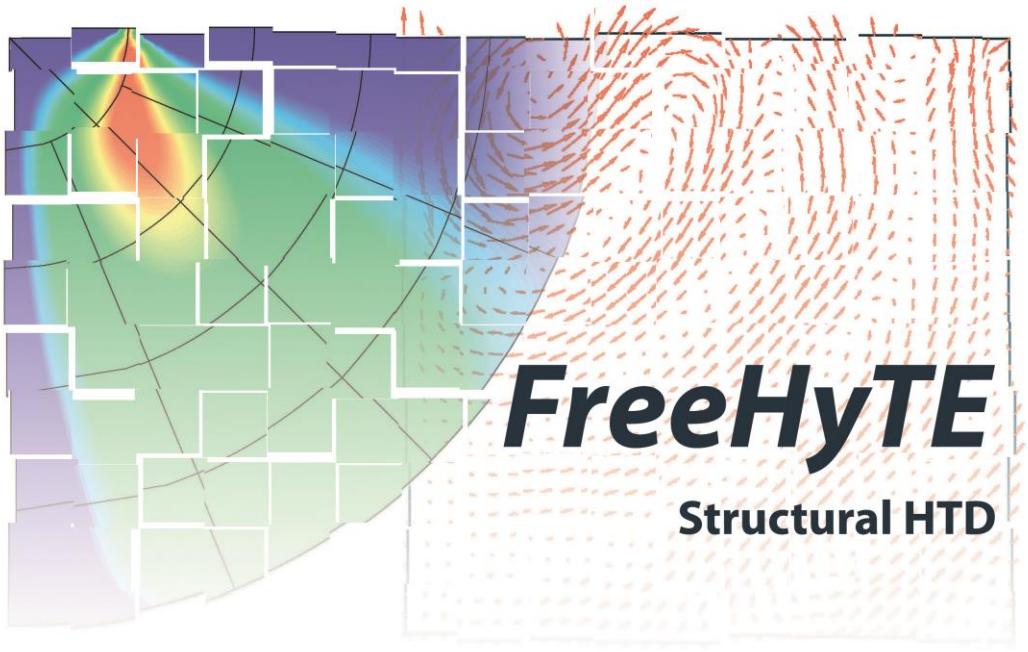
CERIS : Civil Engineering Research and Innovation for Sustainability



USER'S MANUAL

v 1.2

TÉCNICO LISBOA | DECIVIL
DEPARTAMENTO DE ENGENHARIA CIVIL, ARQUITECTURA E GEORRECURSOS

DISCLAIMER

FreeHyTE – Structural HTD

Hybrid-Trefftz Displacement Finite Elements for Structural Plane Elasticity

CERIS : Civil Engineering Research and Innovation for Sustainability

TÉCNICO LISBOA

ACKNOWLEDGEMENTS

FreeHyTE – Structural HTD

Hybrid-Trefftz Displacement Finite Elements for Structural Plane Elasticity

The following people (listed alphabetically) contributed, in one way or another, to the development of this software:

Ana Coutinho, Antero Sequeira, Belen Palácios, Carol Correia, Corneliu Cismasiu, Cristina Silva, Daniel Pereira, Ildi Cismasiu, Ionut Moldovan[1], João Freitas, Lucian Radu, Michael Klanner, Rita Geraldes, Vasco Silva.

The development of this software was partially funded by Fundação para a Ciência e a Tecnologia of Portugal through Grant SFRH/BPD/87317/2012.

[1] Correspondence to dragos.moldovan@tecnico.ulisboa.pt

TABLE OF CONTENTS

1. INTRODUCTION

1.1. FreeHyTE

FreeHyTE is a collection of finite element solvers for elliptic, parabolic and hyperbolic initial boundary value problems using hybrid and hybrid-Trefftz finite elements.

The **FreeHyTE** computational platform is developed at the CERIS Research Centre, Instituto Superior Técnico, University of Lisbon. The development structure relies heavily on the constant engagement of Senior Year Master of Science students to perform the bulk of the coding duties for the various modules of **FreeHyTE**. Consequently, each module of the platform is developed and deployed separately, although they all share the same workflow, data structures, computational procedures and I/O sequences.

Each module of **FreeHyTE** is released under the GNU Public Licence and is a free software. You are welcome to use it, improve it, and expand it, provided you only release improvements and/or expansions under the GNU Public Licence.

FreeHyTE stemmed from the gradual realization that vast amounts of code developed in higher education are simply lost when the main developer moves out, rendering most of the subjacent research virtually unreproducible; and that a change in the research paradigm was required to preserve and disseminate the results of our work. **FreeHyTE** is an attempt at the standardization of data structures, processing routines and numerical procedures that are pervasive in hybrid finite element formulations, aiming to secure a platform for the diverse developments to be plugged in with minimal coding necessities.

To the best of our knowledge, **FreeHyTE** is, as of 2016, the only user-friendly and publicly available software employing hybrid and hybrid-Trefftz finite elements. We take pride in placing the (considerable) advantages of these formulations at the fingertips of users and researchers around the world.

Unfortunately, however, research is conducted nowadays under a 'Publish or Perish' vision that verges, at times, on insanity. We are no exception. Consequently, **FreeHyTE** is more focused on providing a simple testing ground for new ideas rather than the best possible experience for the user (though we still think it's fairly simple to use). While we try not to be exceedingly sloppy as we code, we cannot afford to invest heavily into the ultimate optimization of the execution times, memory allocation or bug-proofing. You are welcome to help us improve the code and we'll give you credit if you do.

1.2. TREFFTZ FINITE ELEMENTS IN FreeHyTE – STRUCTURAL HTD

FreeHyTE – STRUCTURAL HTD uses the displacement model of the hybrid-Trefftz finite elements for the solution of plane elasticity problems (i.e. under plane stress or plane strain conditions).

Unlike conforming displacement (conventional) finite elements implemented in the vast majority of commercial codes, hybrid-Trefftz displacement (HTD) finite elements use independent approximations of the displacements in the domain of the elements and of the tractions (forces) on the Dirichlet and interior boundaries (hence the *hybrid* label of the elements). Moreover, the domain displacement bases are constructed using approximation (shape) functions that satisfy exactly all domain equations (hence the *Trefftz* label of the elements). Such functions are thus tailored for each particular problem that is being solved and embody relevant physical information about the model.

As a direct consequence, hybrid-Trefftz models endorse the use of *extremely large* finite elements as compared to the conventional models and are able to handle problems involving stress concentrations, high solution gradients and discontinuous solution fields without cumbersome local mesh refinements. Nearly-incompressible media, awkward topologies, shear locking and gross mesh distortions are also efficiently handled by hybrid-Trefftz elements.

Users acquainted to conventional finite elements must be familiar with their node-wise approximation functions, typically linear or bi-linear in shape, with having the nodal displacements as degrees of freedom, and with the mesh refinement as a mean to improve the finite element solution. Conversely, hybrid-Trefftz finite elements move away from the nodes as the pivotal finite element concept to offer user more flexibility in choosing arbitrary, high-order approximations for the unknown fields. This means that, while mesh refinement remains a valid option for improving the quality of the Trefftz finite element solution, the same effect can now be obtained by simply incrementing the order of the approximation functions instead. Definition of distinct orders of refinement for each finite element and essential (i.e. exterior Dirichlet and interior) boundary is also affordable. A systematic analysis of the differences between conventional and hybrid-Trefftz finite elements from a user's standpoint is presented in Section 3.1.

FreeHyTE – STRUCTURAL HTD offers you all these advantages and flexibility, and features a Graphical User Interface (GUI) to endorse a seamless structure definition, with minimal effort. We hope you enjoy it and contribute to it in the near future.

1.3. ABOUT THIS MANUAL

This manual is aimed at presenting when and how **FreeHyTE – STRUCTURAL HTD** module should be used. Consequently, the manual follows a strictly a need-to-know, user-oriented perspective and should provide enough information for the needs of beginner and advanced users alike.

Chapter 2 focuses on *when* should you use **FreeHyTE – STRUCTURAL HTD**. It presents the main simplifying hypotheses under which the program operates and helps identifying the practical situations where it should be used.

Chapter 3 explains just as much about the hybrid-Trefftz displacement (HTD) elements as a user should know in order to safely run the module. As most finite element users are acquainted to conventional finite elements, the text stresses the similarities and differences between conventional and hybrid-Trefftz formulations.

Chapter 4 introduces the steps a user should take for a complete structural definition using HTD elements and explains the conventional positive directions of the involved referentials.

Chapter 5 focuses on *how* **FreeHyTE – STRUCTURAL HTD** should be used. It presents the Graphical User Interface (GUI) and employs the practical example of a cantilever with (and without) a circular hole to illustrate its use.

Chapter 6 is dedicated to more advanced structural definition options which require localized changes in the `InputProc.m` file of the code itself.

This manual complements the **FreeHyTE – STRUCTURAL HTD Installation Manual**, which introduces the various module deployment options and the respective installation steps.

Please note that the module was tested in Matlab versions 2012b, 2013a and 2015a. Kindly let us know if you experience difficulties in using it under other versions of Matlab.

2. STRUCTURES AND MODELS

2.1. COMPUTATIONAL MODELS

The modelling process works in two steps. The first step deals with the definition of the mathematical model, where the phenomenon is described using some set of algebraic and differential equations. In the second step, approximate solutions of these equations are found using some numerical method implemented in a software. Both steps induce errors to the solution and it is very important to understand that *just because you get a result, it doesn't mean that it's the right resul*t. It is the task of the computational analyst to assess the magnitude of the error and to decide whether the results are good enough for his/her practical purposes.

2.1.1. Mathematical model

The mechanical behaviour of structures is a highly complex phenomenon which cannot be fully captured by any mathematical mean. Therefore, simplifying assumptions must be made in order to reduce its complexity to a manageable level.

The following simplifying assumptions were considered for the derivation of the mathematical models implemented in **FreeHyTE – STRUCTURAL HTD**:

- the material behaviour is **linear-elastic** (physical linearity), meaning that the stresses and strains occurring under the applied constraints are related by a linear relationship. As a consequence, no yielding is considered to occur and the structure is expected to return to its undeformed shape as the constraints are withdrawn;
- the material is **piecewise homogeneous and uniform**, meaning that its mechanical properties are identical in all points of some arbitrary regions of the structure. Consequently, insertions of one material into another can be modelled, but smooth transitions of the material's properties, taking place over some transition regions, cannot;
- the material is **isotropic**, meaning that its mechanical characteristics are the same in all directions;

- structural **strains and displacements are small** as compared to the geometrical dimensions of the structure (geometrical linearity). This means that the equations governing the response of the structure can be written on the undeformed shape of the structure.

Based on these assumptions, the governing mathematical model expresses the *equilibrium* of a differential element in the domain of the structure under the stresses it is subjected to, the *compatibility* between the displacement and strain fields and the *constitutive equations* relating the stress and strain fields through a matrix of constants. For the full definition of the problem, the displacements and forces applied to the boundaries of the structure must be specified by the user.

Since the objective of the structural modelling is to predict the real behaviour of the structure, working with an inadequate mathematical model will compromise the results no matter how refined the computational model is. Therefore, the results obtained with **FreeHyTE – STRUCTURAL HTD** are only as good as the mathematical model allows them to be.

2.1.2. Approximate solutions

A sufficiently accurate mathematical model is only half way to practically usable results since the equations governing the structural response are only analytically solvable in a few, very simple cases. For the vast majority of practical situations, approximate solutions must be found instead.

In this context, **FreeHyTE – STRUCTURAL HTD** is a hybrid-Trefftz finite element solver of the differential equations describing the structural response. Upon completion of the analysis, **FreeHyTE – STRUCTURAL HTD** should be able to predict the displacement, strain and stress fields that occur at any arbitrary point of the structure.

Since it *approximates* the solution of the governing equations, **FreeHyTE – STRUCTURAL HTD** is a source of errors that adds to those caused by the simplifying assumptions of the mathematical model. The errors induced by the computational model can and should be controlled by the analyst through a process known as the *refinement* of the model, as discussed in Section 3.1.4. However, it is important to note that a high quality finite element solution only goes as far as the mathematical model allows it to go. If the latter does not meaningfully reproduce the reality, the solution is unusable no matter how refined the finite element model is.

Besides the simplifying assumptions presented in Section 2.1.1, the following limitations specific to the **FreeHyTE – STRUCTURAL HTD** module are in place:

- the structure has a *plane state* behaviour (either plane stress or plane strain). As described in Section 2.2, this does not necessarily mean that the structure must be plane, but that its behaviour can be fully captured by knowing the states of displacement, strain and stress in a single plane. Fully three-dimensional hybrid-Trefftz finite elements must be formulated anew since Trefftz bases are problem-dependent, thus the extension is not trivial. Provided enough help is secured, the extension will be attempted in the future, but it's not on our short-term agenda;
- no volume/area forces can be applied to the finite elements in the current version. The force and displacement application is confined to the exterior boundaries of the structure. Applying constant volume forces (e.g. own weight) is relatively simple to implement and should be achieved in the near future. Applying arbitrary forces require more adaptation and is less relevant from a practical standpoint, so the extension is not on our short-term agenda;
- no Robin type (i.e. spring) boundary conditions can be applied in the current version of **FreeHyTE – STRUCTURAL HTD**. Only pure Dirichlet (i.e. static) and Neumann (i.e. kinematic) boundary conditions can be defined. Adding Robin boundary conditions is, however, in our short-term plans.

2.2. PLANE STATES

Plane elasticity exploits the simplifications that occur when a three-dimensional structure is expected to present a null principal stress or strain, to reduce it to a two-dimensional computational model, for faster and more efficient analysis.

Two such situations can be modelled in **FreeHyTE – STRUCTURAL HTD**: the plane stress state and the plane strain state.

A *plane stress state* occurs when stresses are constant along one dimension of the structure and the principal stress is null in the same direction. Plane stress occurs in structures subjected to excitations in a single plane and free to deform in the direction orthogonal to it, meaning that stresses will develop in that plane alone.

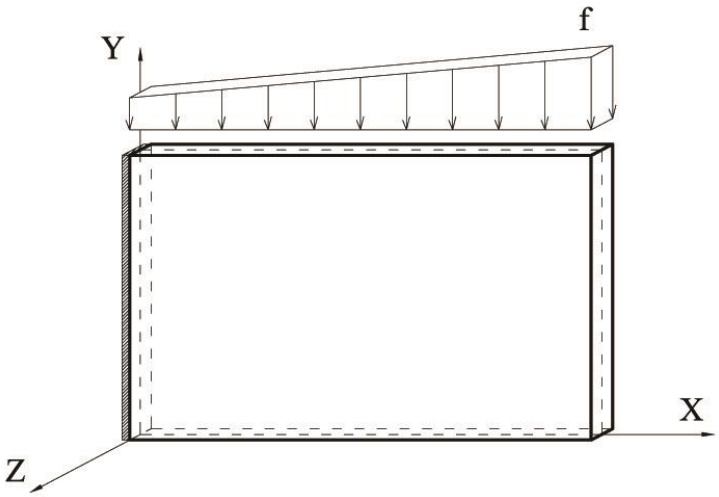


Figure 1. Structure under plane stress conditions

As illustrated in Figure 1, this is typically the case of structures where one dimension is significantly inferior to the other two, as for instance, in shear walls. Plane frames, however, are also cases of plane stress states.

A *plane strain state* occurs when strains are constant along one dimension of the structure and the principal strain is null in the same direction. Plane strain typically occurs in structures unable to deform in the direction orthogonal to a plane, meaning that strains will develop in that plane alone.

As illustrated in Figure 2, plane strain is typical to structures where one dimension is very large compared to the other two (e.g. geotechnical structures, tunnels, gravity and

embankment dams). In this case, the principal strain in the direction of the longest dimension is constrained and can be assumed to be zero.

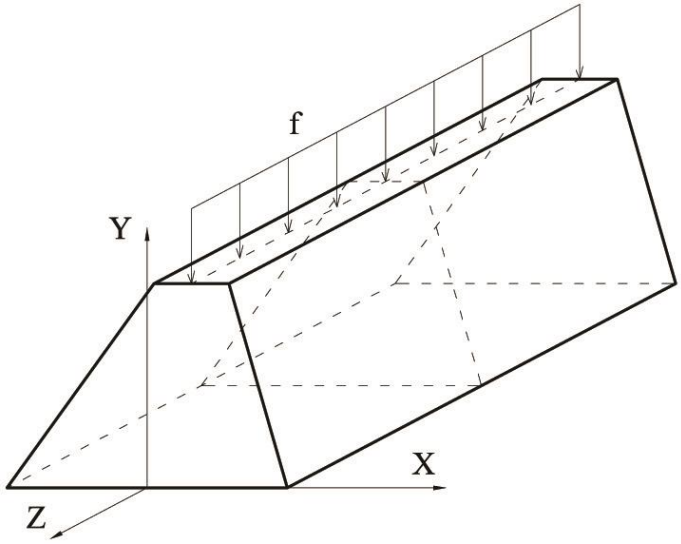


Figure 2. Structure under plane strain conditions

3. HYBRID-TREFFTZ FINITE ELEMENTS

A brief perspective over some key features of hybrid-Trefftz finite elements is presented in this chapter. The objective is to introduce the reader to the main differences between handling conventional (conforming displacement) and hybrid-Trefftz finite elements from a user's standpoint. It is assumed that the reader is acquainted with the basics of the former.

The presentation is structured in two parts. A comparison between conventional and hybrid-Trefftz finite elements is made in Section 3.1. Based on the features described there, the advantages and drawbacks of using hybrid-Trefftz finite elements to solve plane elastostatic problems are described in Section 3.2.

3.1. CONVENTIONAL vs. HYBRID-TREFFTZ DISPLACEMENT FINITE ELEMENTS

The comparison between conventional and hybrid-Trefftz displacement finite elements focuses the way these formulations enforce the domain and boundary equations, the features of the approximation functions, the nature of the solving system, and the handling of the mesh and basis refinements. It is important to note that many (and indeed diverse) hybrid-Trefftz finite element formulations exist, so the features described next refer strictly to the formulation implemented in **FreeHyTE – STRUCTURAL HTD**.

3.1.1. Enforcement of the governing equations

To recover the solution of plane elasticity problems, three domain and two boundary equations must be solved, namely the equilibrium, compatibility and constitutive equations in the domain of each element, and the kinematic (Dirichlet) and static (Neumann) equations on their exterior boundaries. As exact solutions cannot be found, in general, some of these equations are enforced in an approximate (average, or weak) form, and some in an exact (strong) form. The enforcement procedure strongly influences the behaviour of the finite element formulation so it is important to understand how it is handled.

Conventional finite elements. Conventional finite elements are strictly compatible. This means that they satisfy exactly the *compatibility* equation in the domain of the element and the *kinematic* boundary conditions on the exterior Dirichlet and interior boundaries. The domain equilibrium equation is enforced weakly and the static boundary conditions are typically enforced at the nodes of the mesh. As a consequence, the displacement field is typically approximated with considerably superior precision as compared to the stress field.

Hybrid-Trefftz displacement finite elements. Hybrid-Trefftz displacement elements satisfy strongly *all domain* equations. The *static* and *kinematic* boundary conditions are enforced weakly on the Neumann and Dirichlet boundaries of the structure. The kinematic boundary conditions are also enforced weakly on the interior boundaries of the mesh. The static boundary conditions are not enforced on the interior boundaries. Since static/equilibrium and kinematic/compatibility boundary conditions are enforced in the same way (except for the interior boundaries), the predictions of the displacement and stress fields are much more balanced in terms of quality than in the case of conventional elements, although some slight bias towards the displacement field is still present.

3.1.2. Approximation functions

Conventional finite elements. Nodes are the pivotal concept in the definition of the approximation functions in conventional finite elements. Nodal displacements are the main unknowns (degrees of freedom) of the elements and all fields in all points of the structure are completely determined by their values. The number and localization of the nodes also determine the expressions of the (polynomial) approximation functions for the displacement field in the domain of the element. Consequently, the redefinition of the nodes requires the recalculation of all approximation functions. In order to ensure the strong compliance with the kinematic boundary conditions, the nodes of adjacent elements need to be connected (i.e. the mesh must be conforming), so the insertion of additional nodes in one element may require the redefinition of all elements of the mesh. The approximation functions are intrinsically able to recover exactly a constant displacement field, through the partition of unity property, meaning that the displacement field will converge to the exact solution as the elements grow smaller (i.e. the mesh gets more refined).

Hybrid-Trefftz displacement finite elements. Hybrid-Trefftz displacement elements approximate not only the displacement field in the domain of the element, but also the force field on its essential (i.e. exterior Dirichlet and interior) boundaries. Both approximations abandon the concept of nodes altogether. The nodal displacements are no longer the main unknowns of the problem and the redefinition of the nodes does not call for any redefinition of the approximation functions. Since nodes lose their significance, hybrid-Trefftz meshes need not be conforming. All approximation bases are *hierarchical*, meaning that the addition of a new approximation function does not require the redefinition of those previously present in the basis. Bounded to satisfy all domain equations, the displacement approximation functions embody relevant physical information on the phenomenon they model. They are tailored for each problem being solved and account for most super-convergent features of the hybrid-Trefftz elements, as discussed in Section 3.2. However, they are generally not polynomials and may be more difficult to integrate than their conventional element counterparts, especially when their order is high. Still on the drawback side, the unknowns of the discretized problem (i.e. the weights of the approximation bases) do not have any physical meaning, being generalized displacements and forces instead. The approximation functions are intrinsically able to recover exactly a constant displacement field, through the inclusion of the explicit unit monomial in the domain approximation basis, meaning that the displacement field will converge to the exact solution as the elements grow smaller (i.e. the mesh gets more refined).

3.1.3. Finite element solving system

Conventional finite elements. The assemblage of the solving system is based on the enforcement of the nodal equilibrium condition. As a consequence, the solving system is sparse and symmetric, but the nodal displacement variables are shared by all finite elements that converge in the respective node. Summation of stiffness matrices from neighbouring elements occurs on the main diagonal of the system. The system is always *kinematically indetermined* (meaning that all nodal displacements cannot be recovered using compatibility equations and kinematic boundary conditions alone), provided there is at least a single node with a free displacement. The system is not *singular* provided enough displacements are enforced to limit the rigid body displacements of the structure and generally not *ill-conditioned*.

Hybrid-Trefftz displacement finite elements. The solving system of hybrid-Trefftz finite elements is assembled with no summation of stiffness matrices. The generalized displacement variables are thus element-specific and not shared between neighbouring elements. While this trait enables localized refinements of the approximation bases (see Section 3.1.4), it also means that the solving systems are larger than those of the conventional elements for the same number of finite elements and the same order of the domain displacement bases. On the other hand, since very large Trefftz elements are affordable, the solving systems generally result smaller, in practice, than the conventional ones (see Sections 5.6.1 and 5.6.2). Sparseness and symmetry properties of the conventional systems are preserved.

The solving system of hybrid-Trefftz finite elements is *not* always *kinematically indetermined*. Instead, the user must secure the kinematic indeterminacy through an adequate choice of the orders of approximation bases in the finite elements and on their essential (Dirichlet and interior) boundaries, as discussed in Section 4.5.2. The system is not *singular* provided enough displacements are enforced to limit the rigid body displacements of the structure, but can be more *ill-conditioned* than its conventional counterpart. Ill-conditioning is controlled in **FreeHyTE – STRUCTURAL HTD** by using pre-conditioners. Should the system still be ill-conditioned after the pre-conditioner is applied, a special-purpose, pseudo-inverse solver is used to obtain the solution (and a warning is issued), but the results may be affected.

3.1.4. Improvement of solutions

Finite element solutions can be improved in two ways: by reducing the size of the finite elements (h-refinement) and by increasing the order of the approximation bases (p-refinement). Combinations of the two may also be used. Both strategies are available for conventional and hybrid-Trefftz elements, as explained below.

Conventional finite elements. Mesh refinement is the main mean of improving the solution in conventional finite elements (and indeed the *only* mean in some commercial software). Mesh refinement can be localized (e.g. in zones where larger solution gradients are expected), as long as it secures the conformity of the mesh, meaning that the nodes of adjacent elements need to be connected. Since the domain bases are not hierarchical, the p-refinement of conventional elements requires the insertion of new nodes and the re-computation of all approximation functions. Moreover, the addition of new nodes and the mesh conformity requirement renders localized p-refinement virtually impossible – refining a single element requires all elements to be refined as well. Such drawbacks hinder the adoption of the p-refinement as the main solution improvement strategy and limits, in practice, the domain bases to low-order polynomials.

Hybrid-Trefftz displacement finite elements. Basis refinement is the main mean of improving the solution in hybrid-Trefftz finite elements. Due to the physical information built in the domain bases, *the elements are super-convergent under p-refinement*. The hierarchical approximation bases mean that the addition of new functions does not call for the re-calculation of the previous ones. The removal of the node as the key concept in hybrid-Trefftz finite elements and the uncoupling of the elemental stiffness matrices in the solving system enable distinct definitions for the orders of approximation on each finite element and essential boundary of the mesh (i.e. localized p-refinement is affordable). For these reasons, hybrid-Trefftz bases are typically of higher order as compared to the conventional elements. Exceedingly high orders of the finite element bases may, however, cause numerical instability due to the integration and system solution errors, a situation that should be avoided (see Section 5.6.2). The alternative mesh refinement may be slightly less convergent than for conventional finite elements, but can be performed without securing the conformity of the mesh since the nodes of adjacent element do not need to match.

3.2. ADVANTAGES AND DRAWBACKS OF HYBRID-TREFFTZ FINITE ELEMENTS

The advantages and drawbacks of hybrid-Trefftz finite elements as compared with conventional finite elements are listed below, following a user's perspective. Algorithmic advantages (like the fact that no domain integration is needed to compute the stiffness matrices) are not treated.

A practical guide on how to exploit the advantages while avoiding the pitfalls is presented in Chapter 4.

3.2.1. Advantages of hybrid-Trefftz finite elements

- better balance between the quality of the displacement and stress results;
- shape functions that embed the physics of the problem;
- very large finite elements are affordable. Leading dimensions can be 10-30 times larger than those of conventional elements;
- very high convergence under p-refinement. High-order approximation bases are mainstream;
- total flexibility in choosing the orders of the approximation bases in each element and on each essential (Dirichlet and interior) boundary. Localized p-refinement is affordable;
- results are insensitive to high solution gradients, near incompressibility, gross mesh distortions and free from locking;
- mesh conformity is irrelevant.

3.2.2. Drawbacks of hybrid-Trefftz finite elements

- slightly lower convergence under h-refinement;
- the flexibility in choosing different orders of approximation in the elements and on the essential boundaries is a mixed blessing as it can be tough to handle for inexperienced users;
- users need to ensure the kinematic indeterminacy of the solving system;
- solving system is prone to ill-conditioning, especially if the orders of approximation bases are too high.

4. STRUCTURE DEFINITION IN FreeHyTE – STRUCTURAL HTD

4.1. INTRODUCTION

This chapter is a hands-on guide to the structure definition in **FreeHyTE – STRUCTURAL HTD**. Its main focus is to guide the reader through the decision making process, from the choice of the most appropriate mesh generator for the given problem, the mesh definition, the refinement of the approximation bases (with special emphasis on securing the kinematic indeterminacy of the governing system), and the definition of the boundary conditions.

FreeHyTE – STRUCTURAL HTD features a **regular mesh generator**, recommended for **rectangular structures** (generates **rectangular elements**) and a **non-regular mesh generator**, recommended for **non-rectangular structures**, or **structures with holes, wedges and/or point forces** (generates **triangular elements**).

These concepts are applied to the solution of two beam problems, one solid and the other with a circular hole at its geometrical centre. The first beam is analysed using two regular meshes with rectangular elements, while for the second beam the two meshes feature triangular elements and are obtained with an automatic generator built into Matlab. For both cases, one mesh is coarse and the other fine. The examples are chosen to illustrate the balance between the orders of h- and p-refinements and to give a quantitative perspective over the acceptable sizes of the elements and levels of basis refinement.

After the models' definitions are completed, the results of the analysis with **FreeHyTE – STRUCTURAL HTD** are presented and compared, along with the respective runtimes (see Section 5.6). The comparison illustrates the effect of the refinement choices on the points that matter most for the user: the quality of the solutions and the runtime.

To keep the presentation as light as possible, the description of the Graphical User Interface used for the definition of the models in **FreeHyTE – STRUCTURAL HTD** is saved for Chapter 5.

4.2. SYSTEMS OF REFERENCE

Before moving on to the presentation of the two examples, it is important to understand the definitions and positive directions of the referentials used for the descriptions of the structure's geometry and boundary conditions.

Structure's geometry is defined in a **global referential**. The **boundary conditions** are defined in the **boundary normal – tangential referential**. Both referentials are Cartesian.

To illustrate this principle, Figure 3 presents a structural domain with nine edges. The geometry of the domain must be specified in the global referential (XY). The origin of the global referential is arbitrary when the non-regular mesh generator is used for the structural definition and corresponds to the lower-left corner of the rectangular structure when using the regular mesh generator.

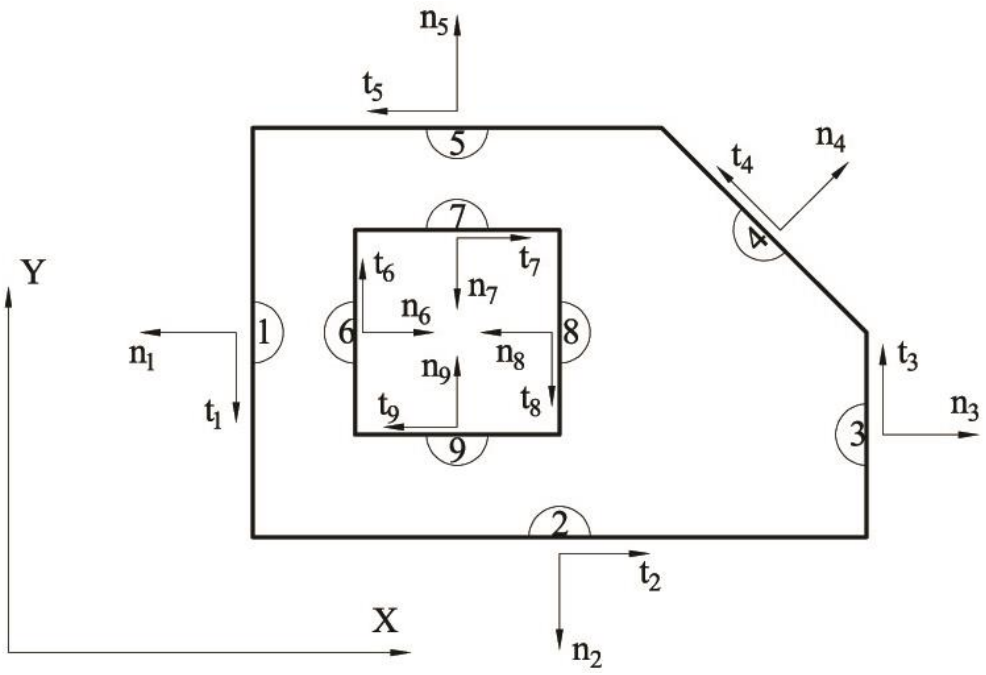


Figure 3. Positive directions of global and side referentials

All boundary conditions must be defined in the normal-tangential (nt) referential of the boundary they are applied to. Their sign must be tuned to the conventional positive directions of the (nt) referential, which are as follows:

- the **boundary normal axis** (n) is oriented **outwards from the element**;
- the **boundary tangential axes** (t) are oriented such as to **encircle the structure in a counter clockwise direction on the outside and in a clockwise direction on the interior openings**.

Please note, therefore, that the definition of the boundary conditions is **not** performed in the global Cartesian referential.

4.3. DESCRIPTION OF THE SAMPLE STRUCTURES

As previously mentioned, the structure definition process is presented here using two practical examples, namely two rectangular beams without and with a circular hole in their geometric centre. The two structures are presented in this section.

4.3.1. The solid beam structure

Consider the rectangular beam presented in Figure 4, made of an homogeneous and isotropic elastic material with a Young's modulus $E = 1000$ and a Poisson's coefficient of $\nu = 0.2$. The beam is fully fixed on its left $(X = 0)$ boundary and partially fixed $(u_x = 0)$ on its right $(X = 6)$ boundary. On the upper $(Y = 3)$ boundary of the beam, a linearly distributed load is applied in the negative normal direction (see Section 4.2), with the maximum intensity of 60 at $(X, Y) = (6, 3)$.

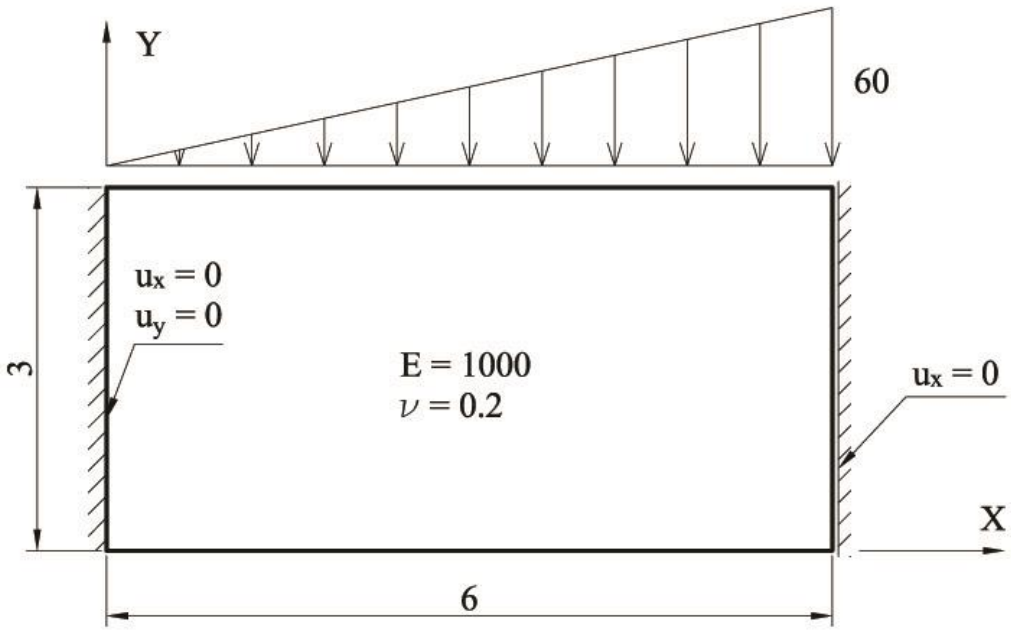


Figure 4. Solid beam structure

The objective of the analysis is to determine the displacement and stress fields for the above structure using **FreeHyTE – STRUCTURAL HTD**.

4.3.2. The hollow beam structure

The second sample structure, presented in Figure 5, is identical to the solid beam structure presented in Section 4.3.1, except for an embedded circular hole with a diameter $\phi = 1$, with the centre at $(X,Y) = (3,1.5)$.

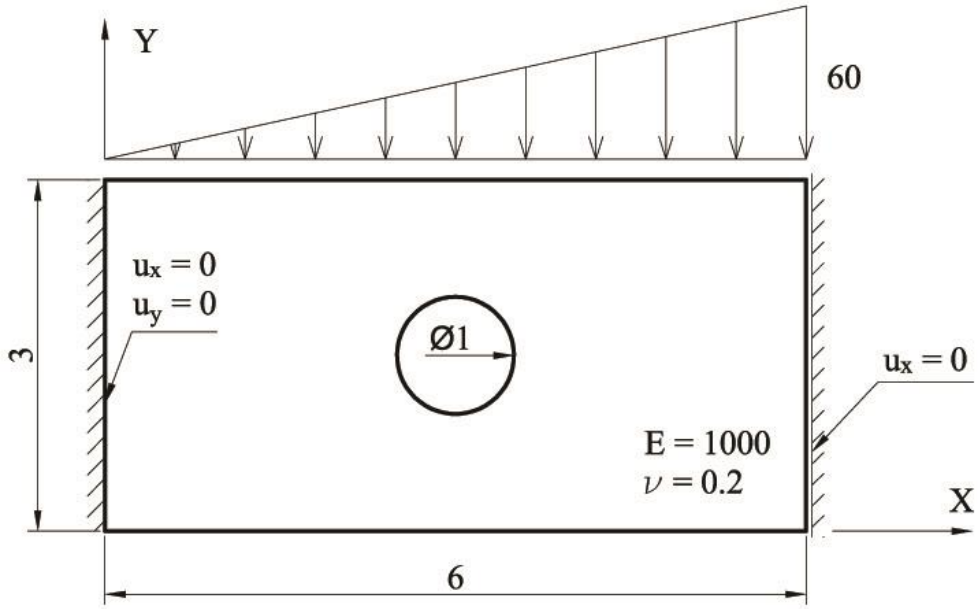


Figure 5. Hollow beam structure

The objective of the analysis performed with **FreeHyTE – STRUCTURAL HTD** is also the same.

4.4. GEOMETRY AND MESHING

The definitions of the geometry and finite element mesh depend on the regularity of the structure and the mesh generator used for its discretization. Two mesh generators are currently implemented in **FreeHyTE – STRUCTURAL HTD**, namely a regular mesh generator and a non-regular mesh generator.

- the **regular mesh generator** can be used **only** for solid, **rectangular structures**;
- the **non-regular mesh generator** can be used for **any structural geometry**.

While the non-regular mesh generator can be used for rectangular structures, adopting the regular generator ensures a faster and simpler structural definition, with no risk of mesh distortion.

- the **regular mesh generator** produces **rectangular finite elements of uniform size**;
- the **non-regular mesh generator** produces **triangular finite elements**.

The usage of the two mesh generators for the definition of the beams presented in Sections 4.3.1 and 4.3.2 is described next.

4.4.1. Regular mesh generator

The regular mesh generator is used to divide a rectangular structure in an array of uniformly sized, rectangular finite elements. The geometry of the structure is defined by simply specifying its dimensions, D_x and D_y, in X and Y directions (Figure 6). The **origin of the global referential** is taken by default **in the lower left corner of the beam** and cannot be changed.

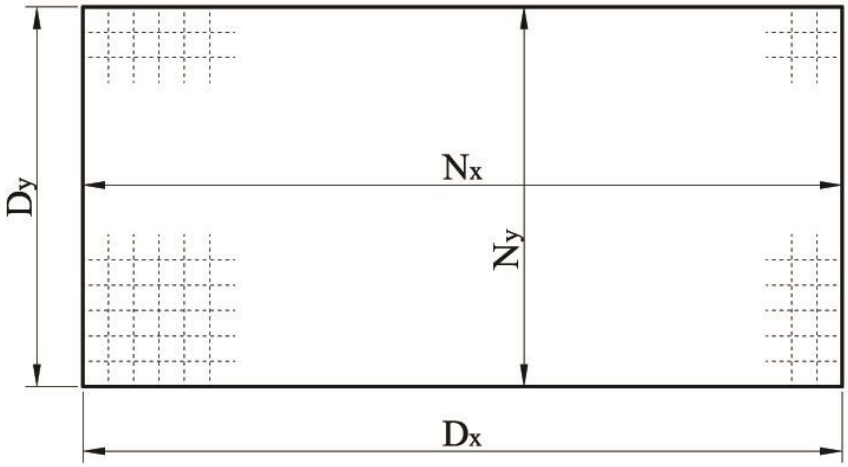


Figure 6. Regular mesh definition of a rectangular structure

Likewise, for the definition of the mesh it suffices to specify the number of finite elements in *X* and *Y* directions, N_x and N_y.

As a general rule, the **leading dimension of the hybrid-Trefftz elements** can be taken **10 to 30 times larger than** that of **conventional elements** without compromising the quality of the results.

The solid beam described in Section 4.3.1 is discretized using a fine mesh with 18 finite elements (Figure 7a) and a coarse mesh with a single finite element (Figure 7b).

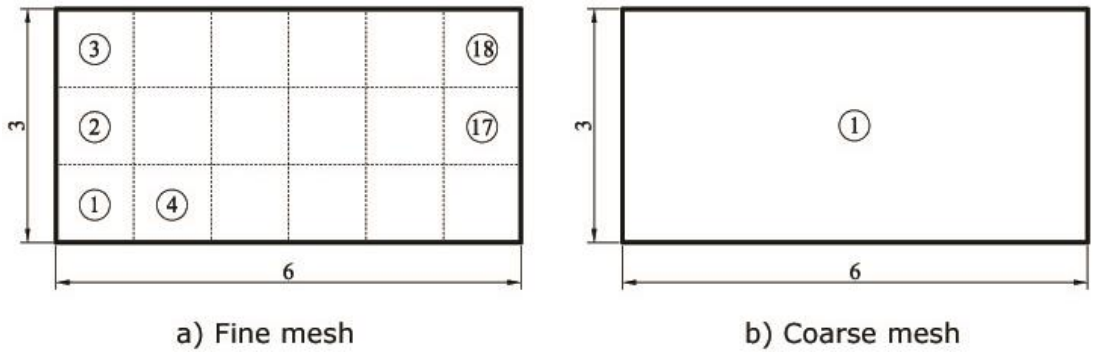


Figure 7. Meshes used for the solid beam structure

4.4.2. Non-regular mesh generator

The non-regular mesh generation protocol in **FreeHyTE – STRUCTURAL HTD** is based on geometry definition and meshing procedures built in Matlab. The option to use native Matlab functions is justified by their efficiency, user-friendliness and increased odds of being compatible with upcoming versions.

There are essentially two ways to create a geometry in Matlab: **programmatically** (code-based) or using the Graphical User Interface (GUI) **`pdetool`**.

Code-based geometry definition enables the creation of virtually any shape, but you do not see the geometry as you create it and need to code a geometry function. The procedures are covered, among other places, here, but their description falls outside of the scope of this manual.

The GUI `pdetool` is arguably the best way to create the geometry at the starting level. It features a simple click-and-drag interface and you get to see the geometry as you create it. Basic shapes as rectangles, ellipses and polygons are used as building blocks and you cannot create geometries that are not a combination of such shapes. However, keep in mind that the mesh generator is limited to triangular elements, so creating an elaborate, curved shape geometry just to triangulate it for meshing should probably not be on your agenda.

A manual describing the use of `pdetool` for the geometry creation and meshing is available at this link. Here, we shall only cover the basic steps needed to generate and mesh the hollow beam presented in Section 4.3.2.

Geometry definition using `pdetool`. The `pdetool` command is **automatically invoked** during the execution of **FreeHyTE – STRUCTURAL HTD** (see Section 5.2.3). The command opens the GUI dialog presented in Figure 8. The geometry definition consists of the following steps:

- *draw the simple shapes that compose the geometry*: Rectangular, elliptical and polyline shapes are accessible from the buttons indicated in the upper ribbon (Figure 8). For our particular case, these shapes are a rectangle, constituting the solid part of the beam, and a circle for the inner opening. Each shape receives an automatic label (*E1* and *R1*), as visible in Figure 9;
- *correction of the constituent shapes*: At this stage, it is unlikely that the shapes have the correct sizes and locations. Double click on each shape and specify its exact dimensions and positioning (Figure 10). Change the size of the window from *Options->Axes Limits* to fully visualize your structure;

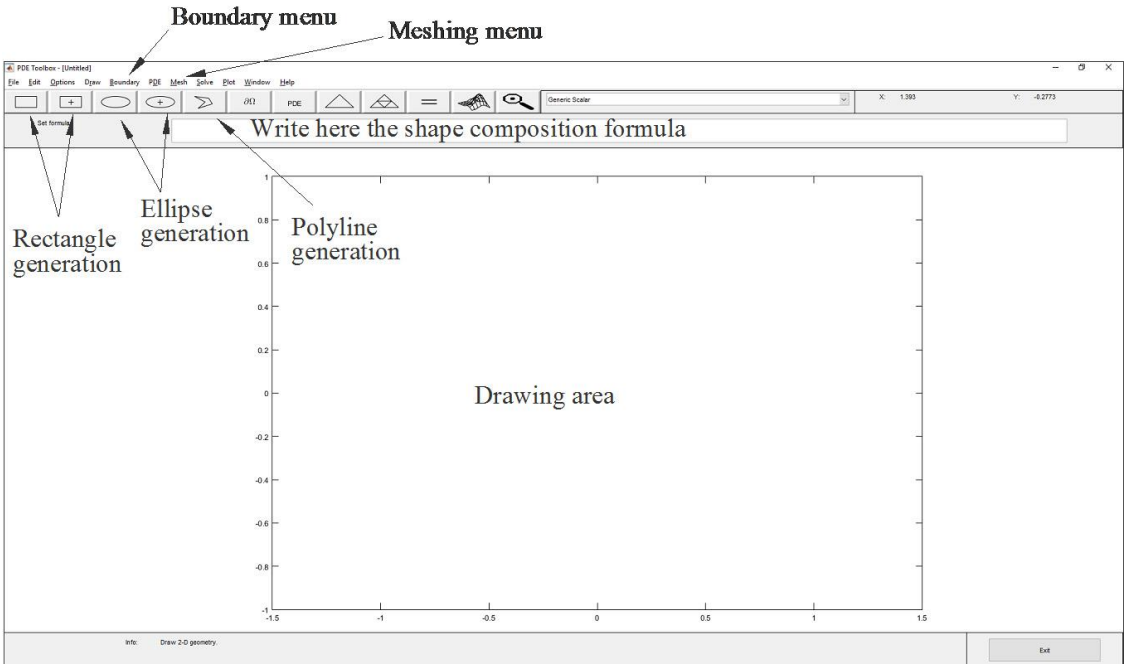


Figure 8. Initial `pdetool` screen

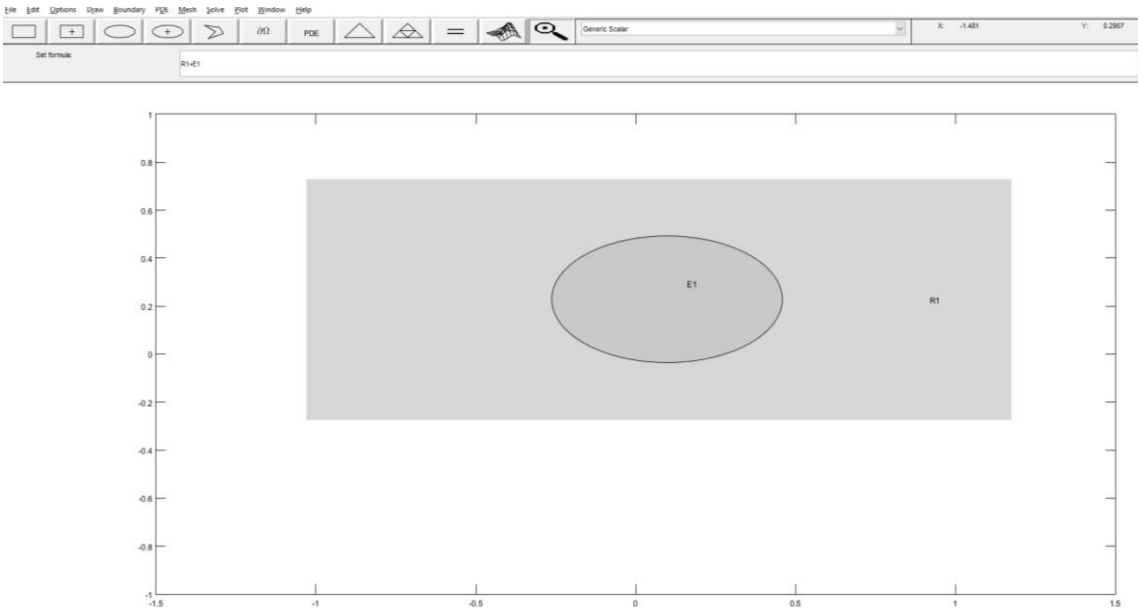


Figure 9. Initial sketch of the geometry

- *define the composition formula in the 'Set formula' zone*: Composition formulae let you add and/or subtract the simple shapes to get the final geometry of the structure. For our case, you will want to create the opening *E1* in the rectangle *R1*, so the correct formula should read *R1-E1*. After setting up the composition formula, click om *Boundary->Boundary Mode* to apply it to your structure (Figure 11).

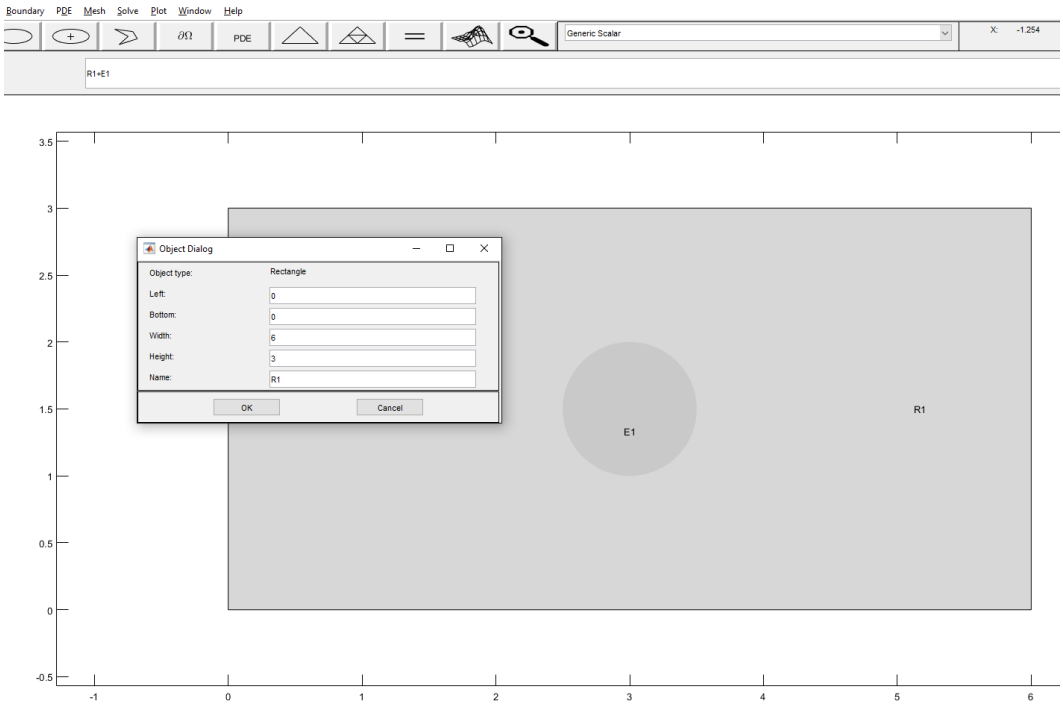


Figure 10. Object definition dialog. Constituent shapes in final form

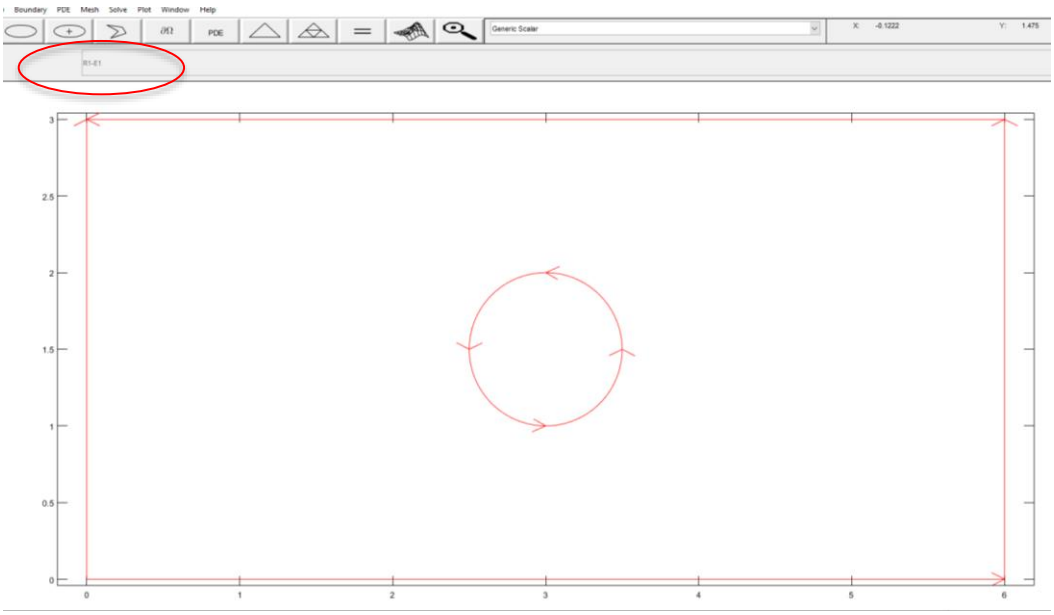


Figure 11. Shape composition formula. Final geometry concluded

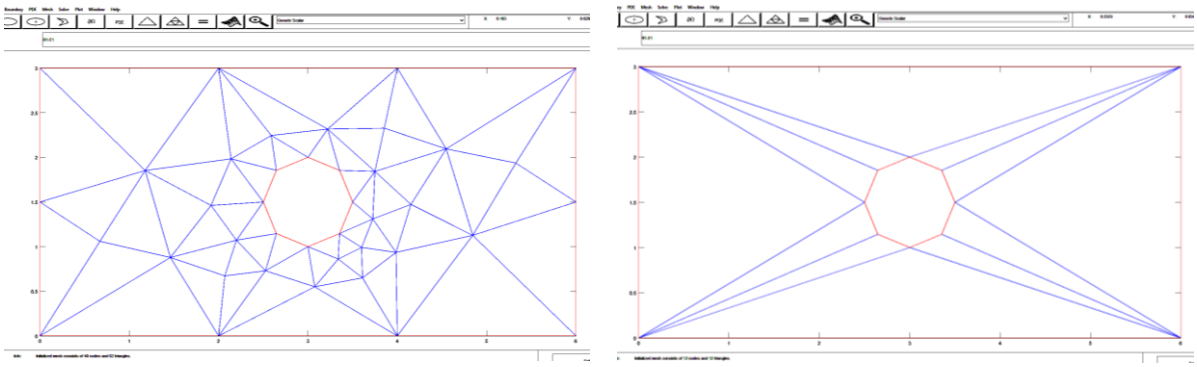
Mesh definition using `pdetool`. Once the geometry definition is concluded, it is time to generate the finite element mesh. The procedure involves the following steps:

- *set up the meshing parameters*: Use the *Parameters* option in the *Mesh* menu of the `pdetool` interface. The available meshing options are explained here. The most

critical options, which we shall tune to define the meshes for our hollow beam structure, are:

- *maximum edge size*: Defines the largest triangle edge length. Two refinement levels are used for the hollow beam, corresponding to the values of 2 and `inf` (from *infinity*, that is, as large as it can possibly be);
- *mesh growth rate*: Defines the rate at which the mesh size increases away from the small parts of the geometry. The value must be between 1 (all geometry is meshed according to the finest region) and 2 (for some reason, `pdetool` does not go beyond 2). Since hybrid-Trefftz elements can be large, we use the value of 1.99 for both refinement levels;

- *generate the mesh*: Use the *Initialize Mesh* option in the *Mesh* menu of the `pdetool` interface to generate the mesh. The meshes obtained for the two refinement levels are presented in Figure 12.



a) Fine mesh b) Coarse mesh

Figure 12. Meshes used for the hollow beam structure

- *export the mesh*: The mesh information must be exported to the `base` workspace in order to be read by **FreeHyTE – STRUCTURAL HTD**. To do so, go to the *Export Mesh* option in the *Mesh* menu and just click *Ok* in the *Export Mesh* dialog. Do **not** change the mesh variable names from the default `p`, `e` and `t` since doing so will render the mesh information illegible to **FreeHyTE – STRUCTURAL HTD**;
- *save the mesh as an *.m file* (*File->Save As*) if you wish to be able to modify it later using `pdetool`.

From Figure 12, it is clear that the main factor conditioning the mesh generation is the presence of the circular hole. Since the native Matlab mesh generator uses only triangles to discretize the structure, the circular hole is modelled as a polygon (a regular octagon in our case). This may substantially over-estimate the stress fields in the vortices of the polygon, but reasonable stress predictions should be recovered close by.

In the case of the fine mesh, the modelling of the circular hole requires relatively small finite elements in its vicinity, but larger finite elements are used in the next layer, as indicated by the *Mesh growth rate* parameter. For the coarse mesh, setting the *Maximum edge size* parameter to `inf` caused extremely large and/or distorted finite elements to emerge. This is obviously a poor quality mesh and is not recommended, but is useful in the context of this manual to illustrate the behaviour of the hybrid-Trefftz finite elements under more extreme conditions.

You can call `pdetool` from the Matlab command window before executing **FreeHyTE – STRUCTURAL HTD**, or you can execute **FreeHyTE – STRUCTURAL HTD** and let it launch `pdetool` for you at the right time during the execution. **The latter option is recommended**.

A new description of the mesh (the `FEMesh` object) was introduced in version 2015a. For backward compatibility, **FreeHyTE – STRUCTURAL HTD** uses the legacy `Pet` mesh format. If you need to convert `FEMesh` to `Pet`, please use the `meshToPet` command. However, **you should not need any conversions if you use `pdetool`** as described above.

4.5. BASIS REFINEMENT

4.5.1. Strategy for basis refinement

The possibility of controlling the basis (p-)refinement of each finite element and essential (i.e. Dirichlet and interior) boundary is one of the most important traits of the hybrid-Trefftz finite elements implemented in **FreeHyTE – STRUCTURAL HTD**, but also one of the trickiest obstacles to the use of these elements by inexperienced users. Three reasons may cause an inexperienced user to hold back from embracing this additional flexibility (as compared to the conventional finite elements):

- first, the independent force approximation and the localized p-refinement are not common concepts in conventional finite elements, so having to deal with them may be puzzling;
- second, while the correct calibration of the bases refinements enables one to obtain excellent results with very coarse meshes and very few degrees of freedom, the incorrect calibration of the p-refinements risks to swiftly compromise the results altogether, so some experience is needed to take advantage of the former while avoiding the pitfalls of the latter; and,
- third, the orders of p-refinements must be chosen such as to guarantee that the finite element solving system is **kinematically indeterminate**, which is a trivial requirement in conventional elements, but not in hybrid-Trefftz elements.

To respond to such concerns, **FreeHyTE – STRUCTURAL HTD favours simplicity at the expense of flexibility**, so that less experienced users can benefit straightaway from the main advantages of hybrid-Trefftz elements without having to worry about the localized p-refinement. More advanced users, however, are still able to take advantage of the additional flexibility the calibration of each basis offers, as explained in Section 6.2.

The p-refinements of all finite elements and the p-refinements of all essential boundaries are uniform when **FreeHyTE – STRUCTURAL HTD** is ran using the Graphical User Interface.

Moreover, the difficulties associated with securing a kinematically indeterminate solving system are eliminated by choosing the finite element and essential boundary refinements according to the simple rules presented in the next section.

4.5.2. Orders of basis refinement

The *order of basis refinement* is conceptually similar to the *order of the polynomial basis* used in conventional finite elements. Approximation bases of the conventional finite elements for plane problems are built by combining monomials from the Pascal's triangle. The order of the basis is commonly defined as the degree of the last *complete* row in the Pascal's triangle. Consequently, three and four nodes elements are linear, six nodes triangular and eight nodes rectangular elements are quadratic and so on. Hybrid-Trefftz finite elements implemented in **FreeHyTE – STRUCTURAL HTD** do not use polynomial bases in the domains of the elements (they do, however, on the essential boundaries), but the concept is similar and can be intuitively applied in the same way.

The (uniform) orders of p-refinement in the finite elements (denoted here by n_D) and on the essential boundaries (denoted here by n_Γ) must be chosen to ensure that, for each finite element, **the total number of kinematic** (domain) **degrees of freedom is larger than the total number of static** (boundary) **degrees of freedom**. This restriction is imposed in order to secure the kinematic indeterminacy of the solving system.

While, *in the limit*, observation of the kinematic indeterminacy criterion is not trivial, for its *loose* observation it is enough to ensure that,

$$\begin{cases} n_D \geq 2n_\Gamma + 3, \text{ for the rectangular element meshes, and} \\ n_D \geq 1.5n_\Gamma + 2, \text{ for the triangular element meshes} \end{cases}$$

Besides the observation of the above inequalities, it is recommended that $\boxed{1 \leq n_\Gamma \leq 7}$ in order to avoid the ill-conditioning of the solving system. The orders of refinement respecting these criteria and thus recommended for a numerically stable analysis with **FreeHyTE – STRUCTURAL HTD** are listed in Table 1, along with the recommended number of Gauss-Legendre quadrature points for the computation of the system's coefficients.

It is noted that the values listed in Table 1 should serve as a guidance, not as a guarantee for correct or stable results. They should, however, provide a good starting point for less experienced users.

n_Γ	n_D (triangular)	n_D (rectangular)	Nº of Gauss points
1	4	5	10
2	5	7	10
3	7	9	10
4	8	11	15
5	10	13	15
6	11	15	20
7	13	17	20

Table 1. Recommended orders of refinement for domains and essential boundaries

p-refinement for the solid beam structure. For the solid beam structure presented in Section 4.3.1, two levels of p-refinement are used. They must be chosen in accordance to the levels of h-refinement described in Section 4.4.1, being lower for the finer mesh (Figure 7a) and higher for the coarser (Figure 7b). Accordingly, $n_\Gamma = 6$ and $n_D = 15$ is the choice for the finer mesh and $n_\Gamma = 13$ and $n_D = 29$ is the choice for the single-element mesh.

p-refinement for the hollow beam structure. Two levels of p-refinement are also used for the hollow beam structure presented in Section 4.3.2. The finer h-refinement (Figure 12a) corresponds to orders of boundary and domain bases of $n_\Gamma = 6$ and $n_D = 11$, while for the coarser mesh (Figure 12b), $n_\Gamma = 13$ and $n_D = 22$ was the choice.

For both structures, 20 Gauss-Legendre quadrature points were used for the less p-refined cases and 30 for the higher order models.

4.6. BOUNDARY CONDITIONS

4.6.1. General definition

Two types of boundary conditions can be defined in **FreeHyTE – STRUCTURAL HTD**:

- **Dirichlet** (or kinematic) **boundary conditions**, where the **boundary displacements** are specified; and,
- **Neumann** (or static) **boundary conditions**, where the **boundary forces** are specified.

No Robin (or elastic) boundary conditions are currently implemented in **FreeHyTE – STRUCTURAL HTD.**

All **boundary conditions are applied to the exterior edges** of the structure. They are **defined in the normal and tangential directions** to the boundary **by polynomials of arbitrary degrees**.

The positive directions of the normal and tangential boundary conditions are in accordance with the side referentials defined in Figure 3.

The definition of a boundary condition is made by **specifying its values in as many equally spaced points along the boundary as needed** to define its polynomial variation. The first point of the sequence must correspond to the beginning of the side (according to its orientation specified in Figure 3) and the last point must correspond to the final point of the side.

Consider, for example, that on edge 4 of the structure presented in Figure 3 a force is applied in the normal direction, as shown in Figure 13. The force is defined by a polynomial of degree N. In **FreeHyTE – STRUCTURAL HTD**, such force is input as a vector containing the values of the force in $N+1$ equally spaced points along side 4, including the first and the last, for instance,

$$f(t_4) \equiv \begin{pmatrix} f_1 & f_2 & f_3 & \cdots & f_N & f_{N+1} \end{pmatrix}$$

The values are given in the order prescribed by the orientation t_4 of the side and the signs are in accordance with the orientation of the normal axis, n_4 (meaning, for instance, that f_1 to f_3 are positive and f_N and f_{N+1} are negative).

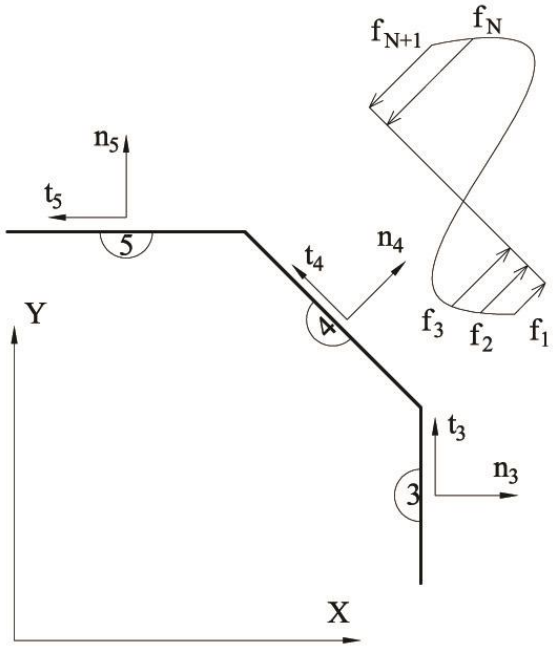


Figure 13. Normal polynomial force applied to side 4

Since the applied force has no component in the boundary tangential direction, the respective forcing term must be defined as zero,

$$g(t_4) \equiv 0$$

Since the tangential force is constant, it is, of course, sufficient to specify its value in a single point.

4.6.2. Special cases

Point loads. Point loads cannot be directly defined in **FreeHyTE – STRUCTURAL HTD** (but neither can they in reality!). On the other hand, point loads can be easily simulated by defining a very small edge where a *distributed* load is applied, such that its resultant is equal to the concentrated load. The main difficulty when such strategy is applied to conventional finite elements is related to the mesh distortions such small edge may cause and/or the very high mesh refinement required in the vicinity of the point load. Hybrid-Trefftz finite elements are well suited to circumvent such difficulties, however, due to their excellent robustness to mesh distortion and ability to accommodate supersized elements.

To illustrate the results that can be obtained using **FreeHyTE – STRUCTURAL HTD** to simulate point loads, Figure 14 presents the finite element mesh used to model a solid beam clamped on its lateral sides and subjected to a concentrated load $P = 10$, applied at the middle of its lower horizontal edge.

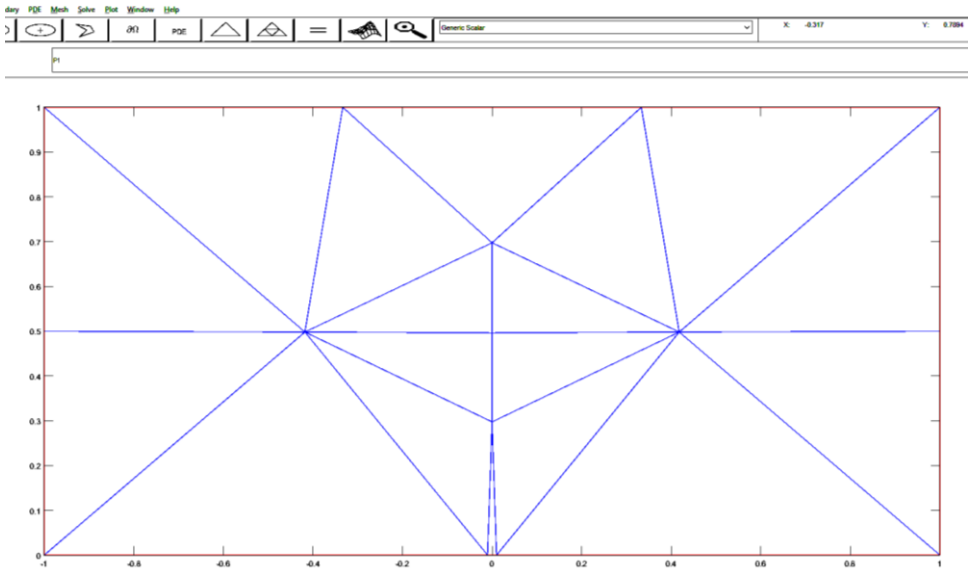
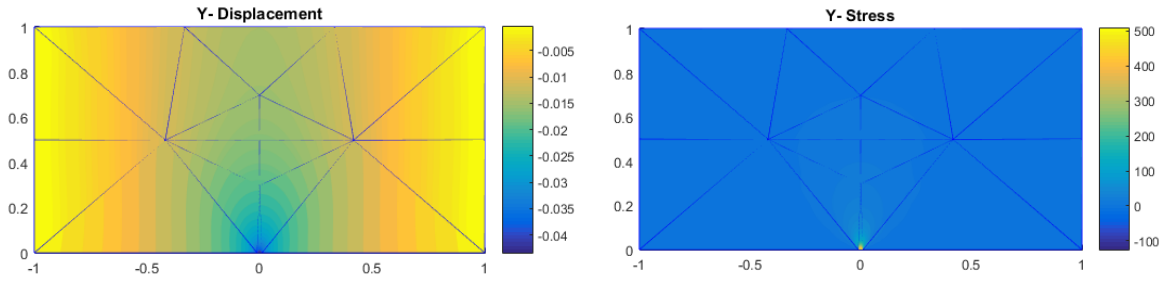


Figure 14. Hybrid-Trefftz finite element mesh for point load modelling

The point load is modelled as a uniformly distributed load of $f = 500$ applied on a Neumann boundary with a length of 0.02, that is, $1/100$ of the length of the beam. The distortion factor of the triangular finite element where the load is applied (defined as the ratio between its maximum side and its minimum side) is superior to 15, which would be completely unacceptable for a conventional element analysis. However, despite the coarseness and distortion of the mesh, hybrid-Trefftz elements recover well the enforced boundary conditions, both in terms of displacements and of stresses, as illustrated in Figure 15, where the vertical displacement and normal stress fields are shown.



a) Vertical displacement field b) Vertical normal stress field

Figure 15. Point load modelling results

Mixed boundary conditions. In this context, mixed boundary conditions designate the situations where **displacements and forces are prescribed in different directions** on a boundary. A very **typical case** of mixed boundary condition is the **roller support** (a.k.a. sliding support), where the normal displacement and tangential force are both enforced null. This situation is illustrated in Figure 16, where a roller-type boundary condition is enforced on edge 4 of the structure presented in Figure 3.

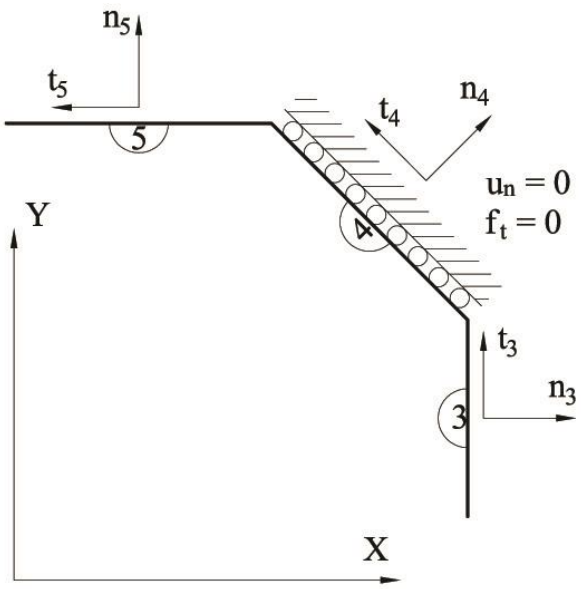


Figure 16. Example of a roller support

Mixed boundary conditions can be defined in **FreeHyTE – STRUCTURAL HTD** with **displacements and forces enforced in either normal or tangential directions**. The enforced displacements are defined by polynomial expressions, as explained in Section 4.6.1. A current limitation of **FreeHyTE – STRUCTURAL HTD** is that **the forces associated to a mixed boundary condition are implicitly zero**. As this is indeed the case in the vast majority of practical situations involving mixed supports, this limitation is not considered important enough to be on our 'to do list' for the near future.

A boundary to which a **mixed condition** is applied **must be defined as Dirichlet** in **FreeHyTE – STRUCTURAL HTD**. **The enforced displacement in the direction where a null force is needed must be defined as NaN**.

NaN is the Matlab's slang for *Not a Number*. In **FreeHyTE – STRUCTURAL HTD** Graphical User Interface, all boundary conditions are predefined as NaN, so it is generally enough to leave the respective displacement field untouched (see Section 5.4).

4.6.3. Boundary conditions for the solid and hollow beams

Following the recipes given in Sections 4.6.1 and 4.6.2, the boundary conditions for the beam problems defined in Sections 4.3.1 (Figure 4) and 4.3.2 (Figure 5) are listed in Table 2.

Location	Boundary type	Direction	Solid beam	Hollow beam
Left boundary, $X=0$	Dirichlet	Normal	0	0
		Tangential	0	0
Left boundary, $X=6$	Dirichlet	Normal	0	0
		Tangential	NaN	NaN
Bottom boundary, $Y=0$	Neumann	Normal	0	0
		Tangential	0	0
Top boundary, $Y=3$	Neumann	Normal	-60 to 0 (linear)	-60 to 0 (linear)
		Tangential	0	0
All edges of the opening	Neumann	Normal	-	0
		Tangential	-	0

Table 2. Boundary conditions for the solid and hollow beam structures

5. GRAPHICAL USER INTERFACE

5.1. INTRODUCTION

This chapter describes the Graphical User Interface (GUI) implemented in **FreeHyTE – STRUCTURAL HTD**, using the beam examples presented in Chapter 4 to illustrate its behaviour.

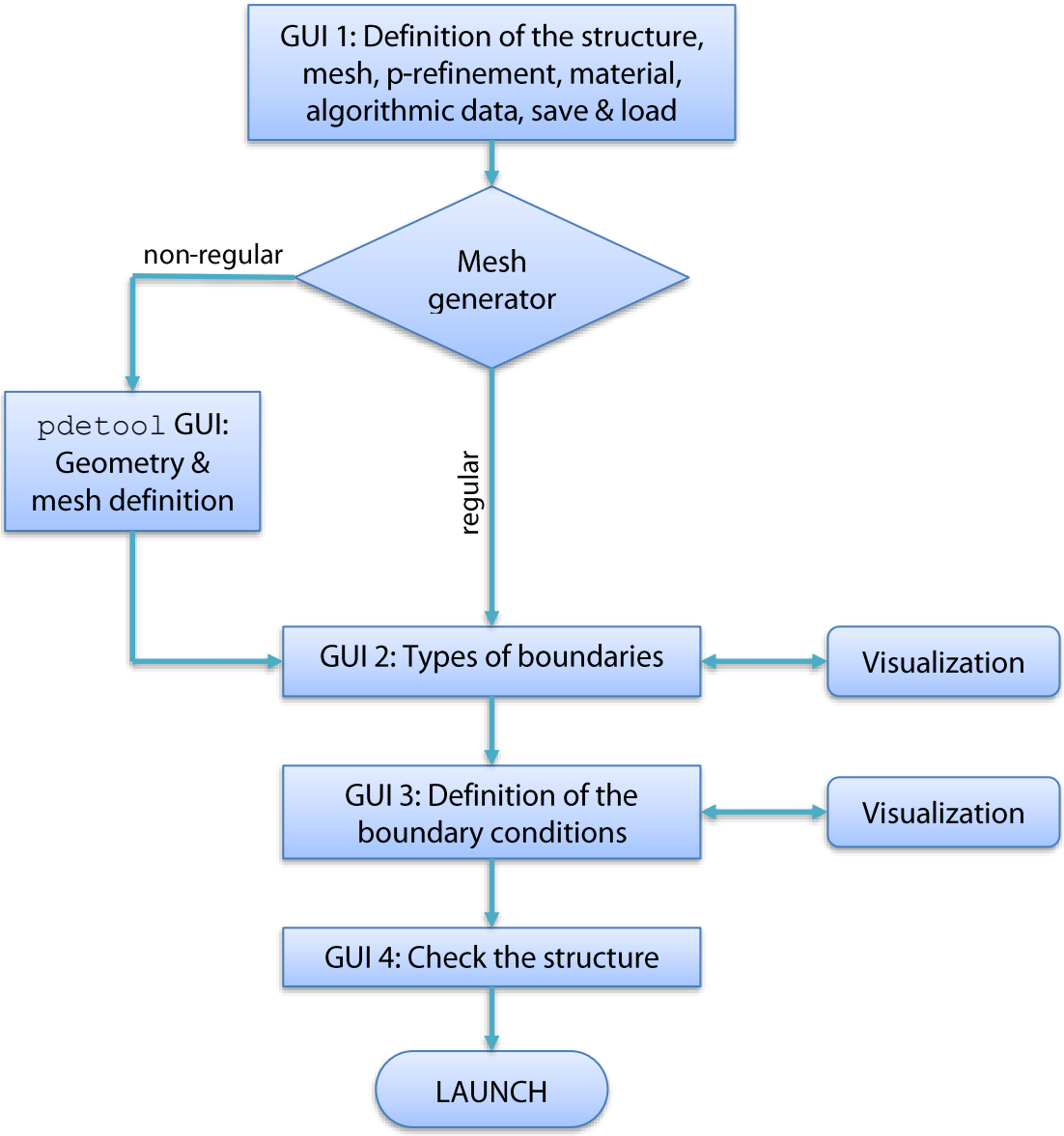


Figure 17. GUI flowchart

The general structure of the GUI is presented in Figure 17, in a flowchart format. It consists of four main GUI, complemented by the `pdetool` GUI for the definition of non-regular structures and meshes, and an optional visualization interface to assist with the definition of boundary conditions.

Free sequential navigation is supported between interfaces, in both directions. Upon exiting an interface and moving to the next, **FreeHyTE – STRUCTURAL HTD** writes the data acquired from the ended interface to a **.mat* file. As a consequence,

- you must run **FreeHyTE – STRUCTURAL HTD** from a folder where you have writing rights; and,
- when **FreeHyTE – STRUCTURAL HTD** is ran from the source (*.m) files, it is possible to start its execution from *any point* of the flowchart in Figure 17, provided you want to reuse all data from the interfaces you skipped.

The latter capability is not available when you run **FreeHyTE – STRUCTURAL HTD** as a Matlab APP, but since data from the previous run is always loaded at the launching of an interface, reusing it without changes should be possible by just clicking the Next button (note that in this chapter the buttons are designated by their label in a small button frame).

The four main GUI and the visualization interface are described in the next sections. The `pdetool` GUI was described in Section 4.4.2 and is not revisited here.

5.2. GUI 1: STRUCTURAL AND ALGORITHMIC DEFINITIONS

The first GUI is used to define the structure's geometry, h- and p-refinements, material characteristics, algorithmic options, to save the current model and to load previously saved models.

A typical configuration of GUI 1 as **FreeHyTE – STRUCTURAL HTD** is initiated is presented in Figure 18. The main data zones of the interface are identified with red frames. Each of these zones is briefly described below.

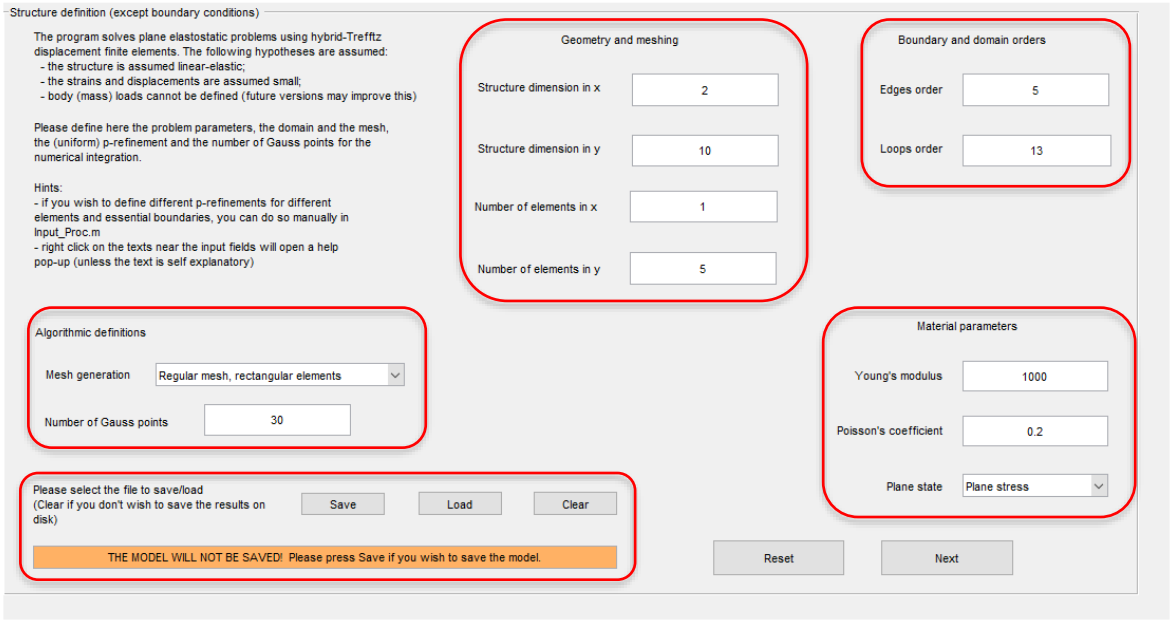


Figure 18. Layout of GUI 1

5.2.1. General features of the interface

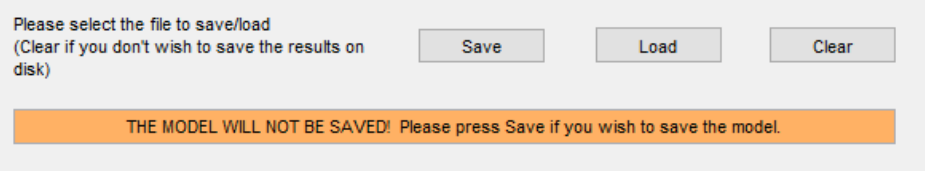
- GUI 1 is **pre-filled with the data from the last run** before it is made visible, if such data is available in the working folder. The Reset button deletes pre-filled data;
- hovering the mouse over the text accompanying the edit boxes opens up **context help** explaining in more detail what input is expected from the user;
- edit boxes are protected from incorrect input. Inserting data of incorrect type will open an error window explaining what data is acceptable for that field and prompting the user to correct it. However, you should be minimally careful when feeding in the data, since bullet-proofing the data checks was not on our plans.

5.2.2. Saving and loading

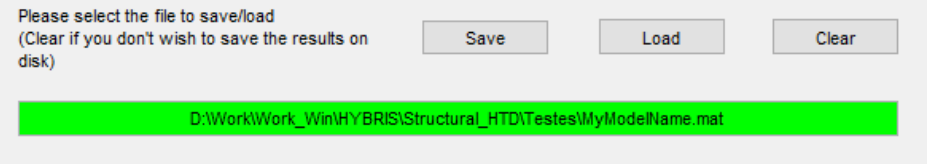
FreeHyTE – STRUCTURAL HTD offers **two levels of save & load**, namely an implicit level and an explicit level.

Implicit save & load. Implicit saving and loading is an automatic procedure performed without user's intervention. Each time the Next button in clicked in a GUI, the data input by the user is stored in some default **.mat* files which remain in the working folder. As you load GUI 1, the data from the corresponding **.mat* file is implicitly loaded.

Explicit save & load. Only the data from the last run is implicitly saved. This means that if you define another model, the data from the previous model is overwritten. If you wish to store the data from a model, you need to use the Save button in GUI 1. **FreeHyTE – STRUCTURAL HTD** prompts you to specify the saving folder and the file name and changes the message from the text box in the lower left side of GUI 1 from,



to,



using the path and the name specified for your model. If you save the model, the data from each GUI is stored not only in the default files, for the implicit loading procedure, but also in the **.mat* file specified by the user. Use the Clear button if you wish to cancel the saving of the model.

Explicitly saving the model also saves a file with the values of the displacement and stress fields in the Gauss points of each element, named *MyModelName_ND**xx**_NB**yy**.dat*, where **xx** is the domain basis refinement and **yy** is the boundary basis refinement.

The output file is formatted for direct loading in the post-processing software Tecplot, but can be used in other visualization software as well since what it contains is just a list of points and displacement/stress values. **The results are not saved in an output file if you choose not to save your model, but plots of displacements and stresses are always shown as the analysis is completed**.

Loading a model overwrites the default **.mat* files of each GUI with the values from the save file.

Modifying the data from a saved model in a GUI and running it with the new data **does not overwrite the old save file**. If you wish to store the changes, you must use the Save button once again.

The mesh data is saved with both explicit and implicit processes, **enabling the reuse of the same mesh in subsequent runs**. **However, the mesh data is not sufficient for modifying the mesh in `pdetool`**. If you wish to store the mesh information such as to be possible to operate on it using `pdetool`, please do so from the `pdetool` GUI, as explained in Section 4.4.2.

The Save and Load buttons are only available in GUI 1.

5.2.3. Data input in GUI 1

Algorithmic definitions. Choose between the regular and non-regular geometry and mesh generators and specify the number of Gauss-Legendre quadrature points to perform the integrations. Less Gauss-Legendre points decrease the duration of the analysis, but increase the numerical integration errors. For some guidance on choosing the number of Gauss-Legendre points, please see Section 4.5.2.

Geometry and meshing. This area is **only editable if the regular mesh generator is used** to define the structure. Its fields correspond to the geometrical and mesh data D_x, D_y, N_x and N_y detailed in Section 4.4.1. **If the non-regular mesh generator was chosen in the algorithmic definitions, `pdetool` is launched automatically** when you click the Next button in GUI 1 (see Section 4.4.2 for details on its usage).

If you selected the non-regular mesh generator and wish to **reuse the mesh data from the previous run** (or from a loaded file) rather than defining a new mesh, **just close the `pdetool` GUI immediately after it opens and proceed to the next GUI**.

Boundary and domain orders. Insert the (uniform) orders of p-refinement on the essential boundaries and in the domain of the elements. For guidance on how to choose these orders, please consult Section 4.5 of this manual.

Material parameters. Choose the type of plane state that you wish to analyse and specify the values of the Young's modulus and Poisson's coefficient.

5.2.4. GUI 1 data for the solid and hollow beams

The GUI 1 data for the beam problems defined in Sections 4.3.1 (Figure 4) and 4.3.2 (Figure 5) are listed in Figure 19 and Figure *20*, respectively. Both sets of data refer to the finer meshes shown in Figure 7a andFigure *12*a, respectively.

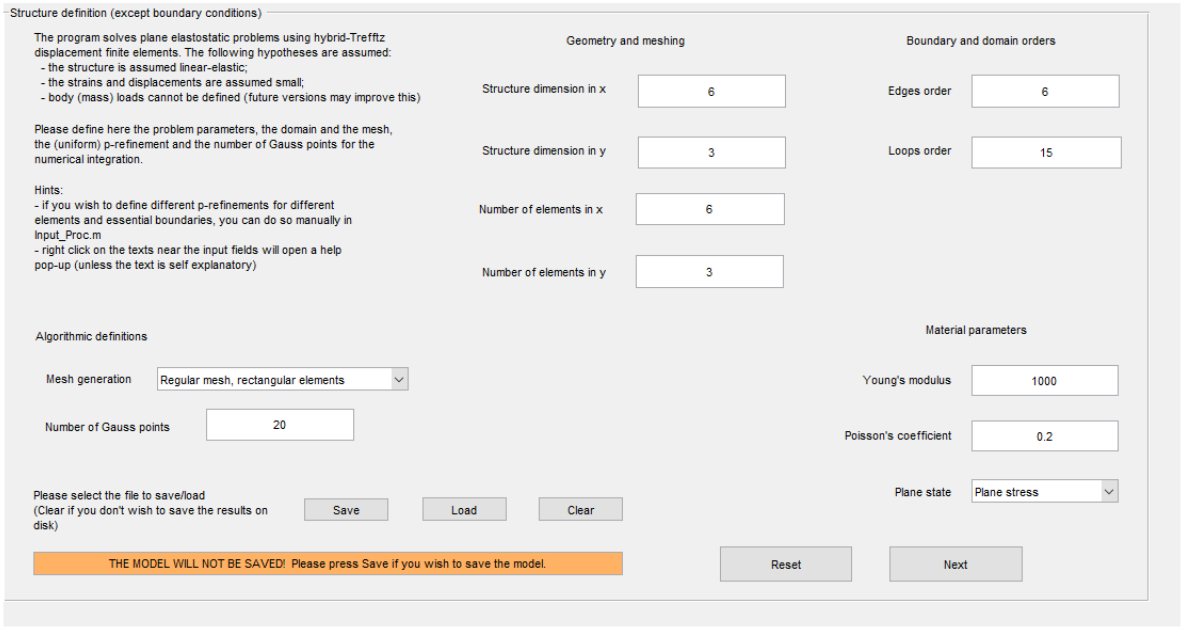


Figure 19. GUI 1 data for the solid beam structure (finer mesh)

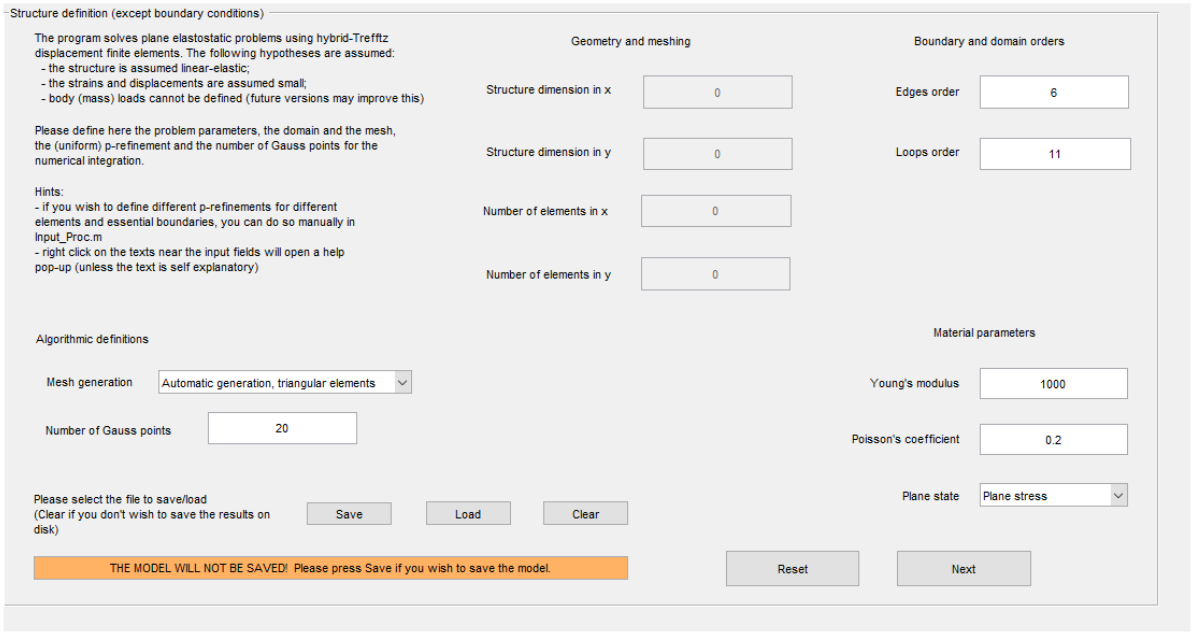


Figure 20. GUI 1 data for the hollow beam structure (finer mesh)

5.3. GUI 2: DEFINITION OF THE BOUNDARY TYPES

The second GUI is used to define the type (Dirichlet or Neumann) for each exterior side of the structure.

The configuration of GUI 2 for the finer mesh of the solid beam problem is presented in Figure 21. The main data zones of the interface are identified with red frames. Each of these zones is briefly described below.

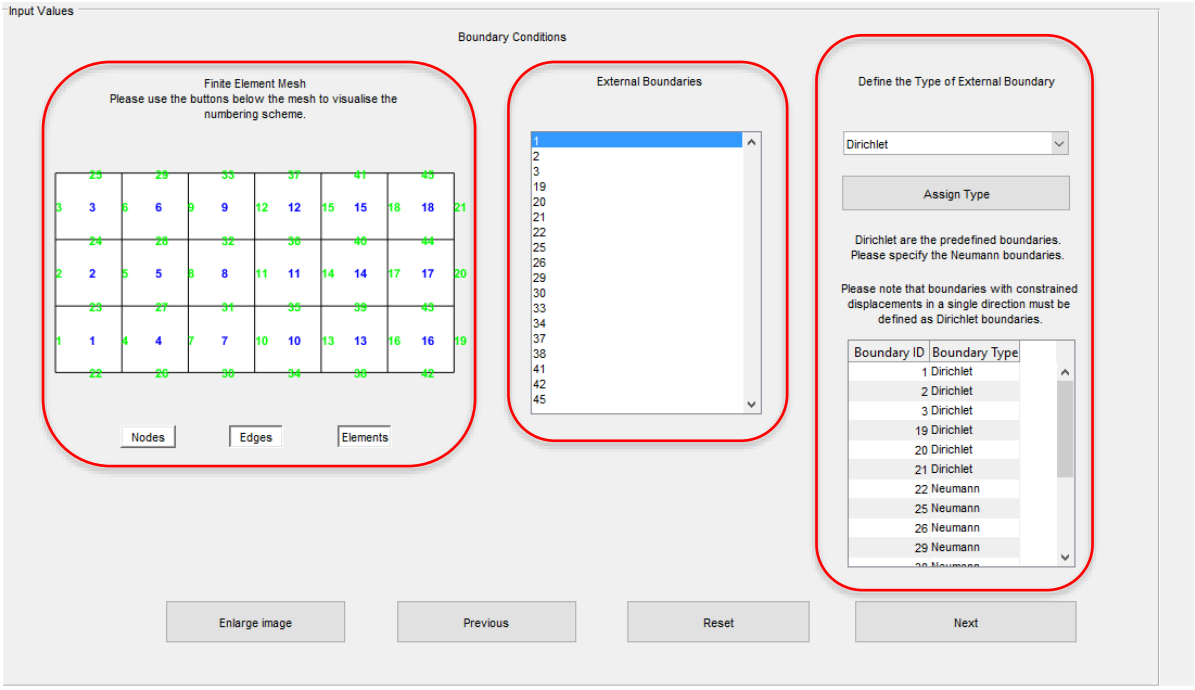


Figure 21. Layout of GUI 2

5.3.1. Structure visualization zone

The structure visualization zone is located on the left side of GUI 2. It consists of an interactive plot of the structure with three buttons controlling what information should be displayed in the plot. In the case presented in Figure 21, this information consists of the edge and element numbers, as the respective buttons are pressed. Un-pressing a button deactivates the information associated to it.

In some cases, the visualization area in GUI 2 may be too small to support a clear read of the structural data. If this is the case, the Enlarge image button can be used to open a

separate, visualization-only interface, with zoom, pan and data cursor capabilities, where the structure can be visualized with any degree of detail (Figure 22).

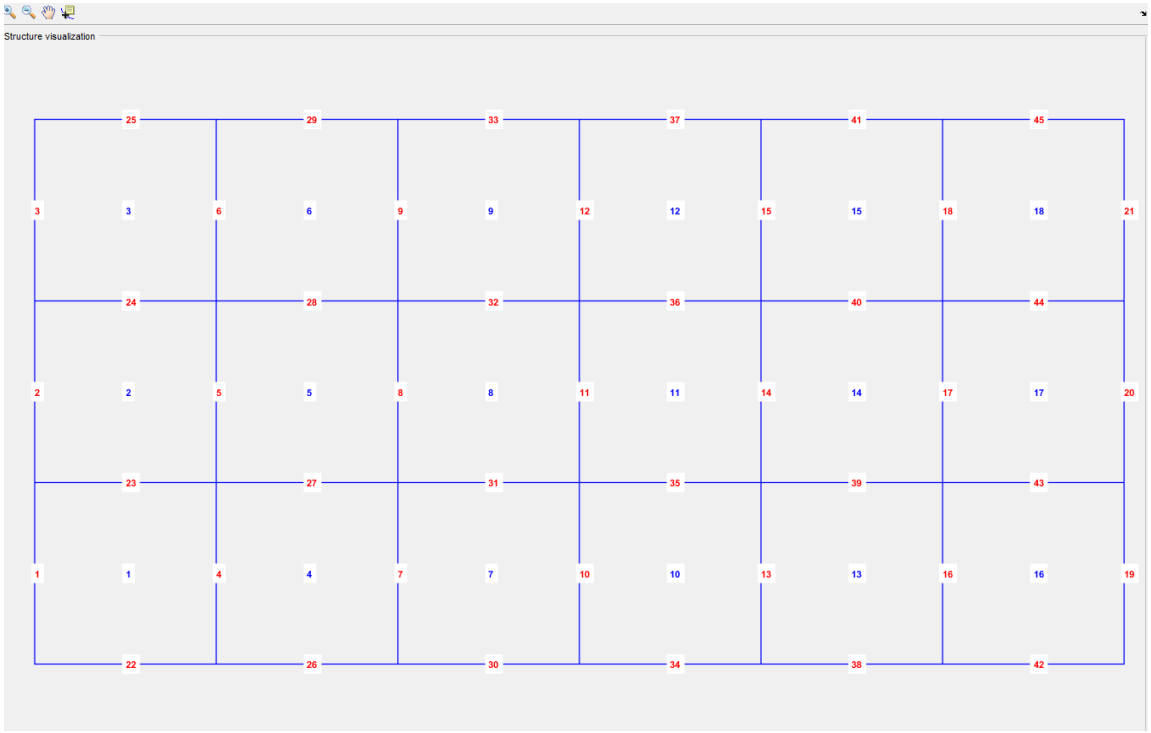


Figure 22. Layout of the visualization interface

5.3.2. Definition of the external boundary types

The external boundaries of the structure are listed in the *External boundaries* table, in the central zone of GUI 2. The boundary types are listed in the table situated in the right side of the interface.

At startup, **GUI 2 pre-loads the boundary types that were auto-saved in the previous run or loaded from a save file if no changes were made to the mesh in GUI 1**. If changes were made, all boundary types are reset as Dirichlet.

To define the boundary types, user must select the boundaries in the *External boundaries* table (multiple selection is possible), select the desired boundary type from the pop-up menu on the right side of the interface and click the Assign type button. The boundary types in the table on the right should automatically adjust to reflect the changes. Press the Next button to move to GUI 3 or the Previous button to return to GUI 1.

5.4. GUI 3: DEFINITION OF THE BOUNDARY CONDITIONS

The third GUI is used to define the boundary conditions according to the boundary types defined in the previous interface. A more insightful discussion on the definition of the boundary conditions in **FreeHyTE – STRUCTURAL HTD** is given in Section 4.6, so this section is restricted to the presentation of the interface used to input those definitions.

The configuration of GUI 3 for the finer mesh of the solid beam problem is presented in Figure 23. The main data zones of the interface are identified with red frames. Each of these zones is briefly described below.

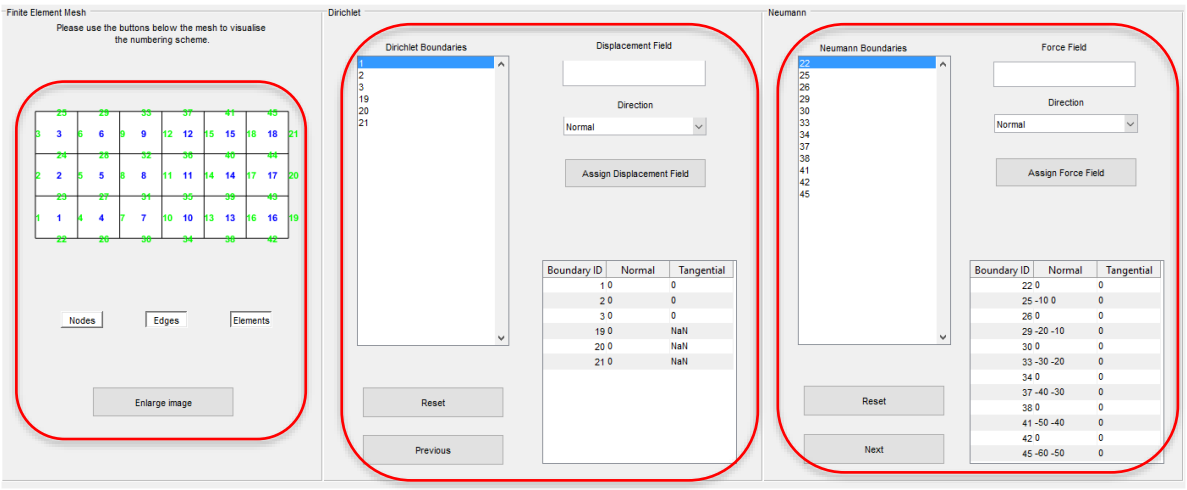


Figure 23. Layout of GUI 3 for the solid beam problem

The structure visualization zone is identical to that of GUI 2 and is presented in Section 5.3.1.

The centre and right input zones are used to define the Dirichlet and Neumann boundary conditions, respectively.

At startup, **GUI 3 pre-loads the boundary conditions that were auto-saved in the previous run or loaded from a save file if no changes were made to the mesh in GUI 1, nor to the boundary types in GUI 2**. If changes were made, all boundary conditions are predefined as *NaN*.

Regarding the definition of the Dirichlet boundary conditions, the exterior boundaries specified as Dirichlet in GUI 2 are listed in the *Dirichlet boundaries* table. In this table, the user must select the boundary where he/she wishes to specify the enforced displacements (multiple selection is possible). In the *Direction* pop-up menu, the user must specify if the displacements are enforced in the boundary normal or tangential direction. Finally, the applied displacements are specified in the *Displacement Field* area, according to the procedure detailed in Section 4.6.1. **If the field to be applied is not constant, its values along the boundary must be inserted all at once, separated by space** (comma also works). Press the Assign displacement field button when you are done. The displacement values in the table below should update automatically. Please note that on edges 19 to 21, sliding supports were prescribed by leaving *NaN* in the tangential displacement field (see Section 4.6.2 for details on how it works).

The procedure for the definition of the Neumann boundary conditions is exactly the same as for the Dirichlet boundary conditions. On boundaries 25, 29, 33, 37, 41 and 45, the linearly distributed normal load (see Figure 4) is specified using its values at the beginning and at the end of each side (see Figure 23). No *NaN* Neumann boundary conditions are acceptable and **FreeHyTE – STRUCTURAL HTD** issues an error message if any is found, prompting the user to check the respective definitions.

For completeness, the configuration of GUI 3 for the finer mesh of the hollow beam problem is also presented in Figure 24.

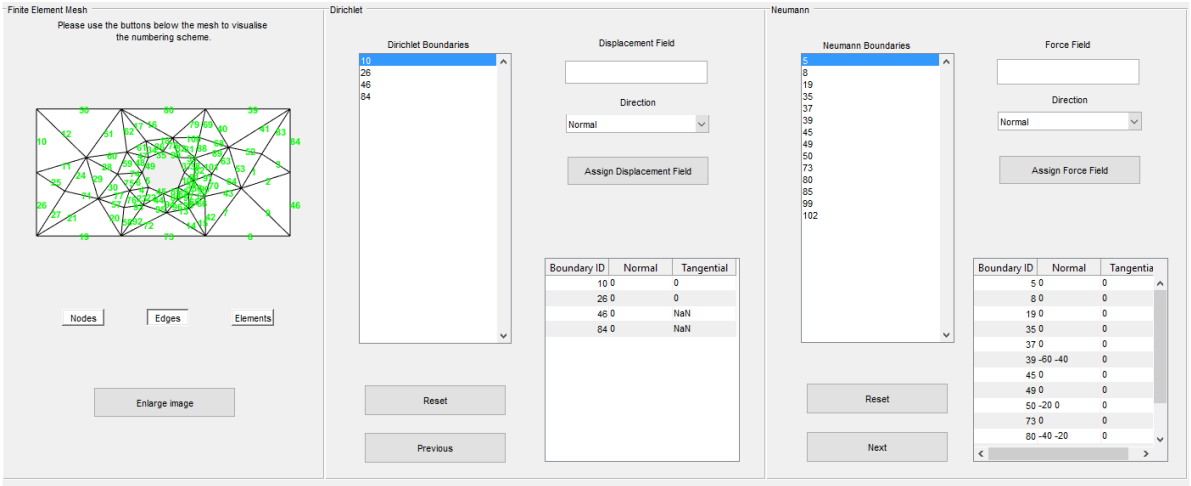


Figure 24. Layout of GUI 3 for the hollow beam problem

After completing the definition of the boundary conditions, press the Next button to move to the Verification GUI or the Previous button to return to GUI 2.

5.5. VERIFICATION GUI

Verification GUI is the last stop before launching the execution of **FreeHyTE – STRUCTURAL HTD** calculation module. It is meant to allow the user to verify the definitions of the structure and boundary conditions and features a simple interface with a single pop-up menu to choose from the visualizations of the mesh numbering, normal and tangential boundary conditions.

The interface is launched with the mesh numbering visualization on, as presented in Figure 25 for the finer mesh of the hollow beam problem.

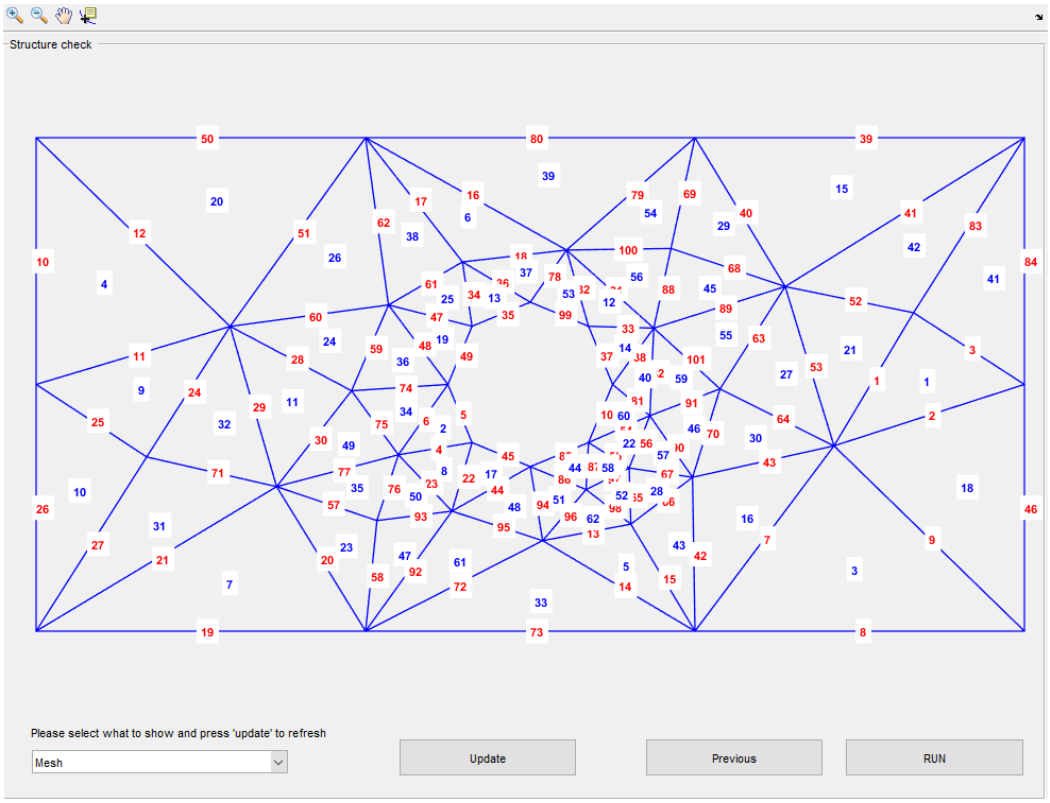


Figure 25. Visualization GUI in the mesh numbering mode

Selecting the *Boundary conditions, normal* and *Boundary conditions, tangential* options in the pop-up menu and then pressing the Update button triggers the boundary conditions visualization modes, where the enforced boundary values are plotted at the beginning and the end of each exterior side. **Dirichlet sides** are plotted in **black**, while **Neumann sides** are plotted in **red**. The visualization interfaces for the normal and tangential boundary conditions of the problem presented in Figure 25 are shown in Figure 26 and Figure 27, respectively. Press the RUN button to launch the analysis of the model.

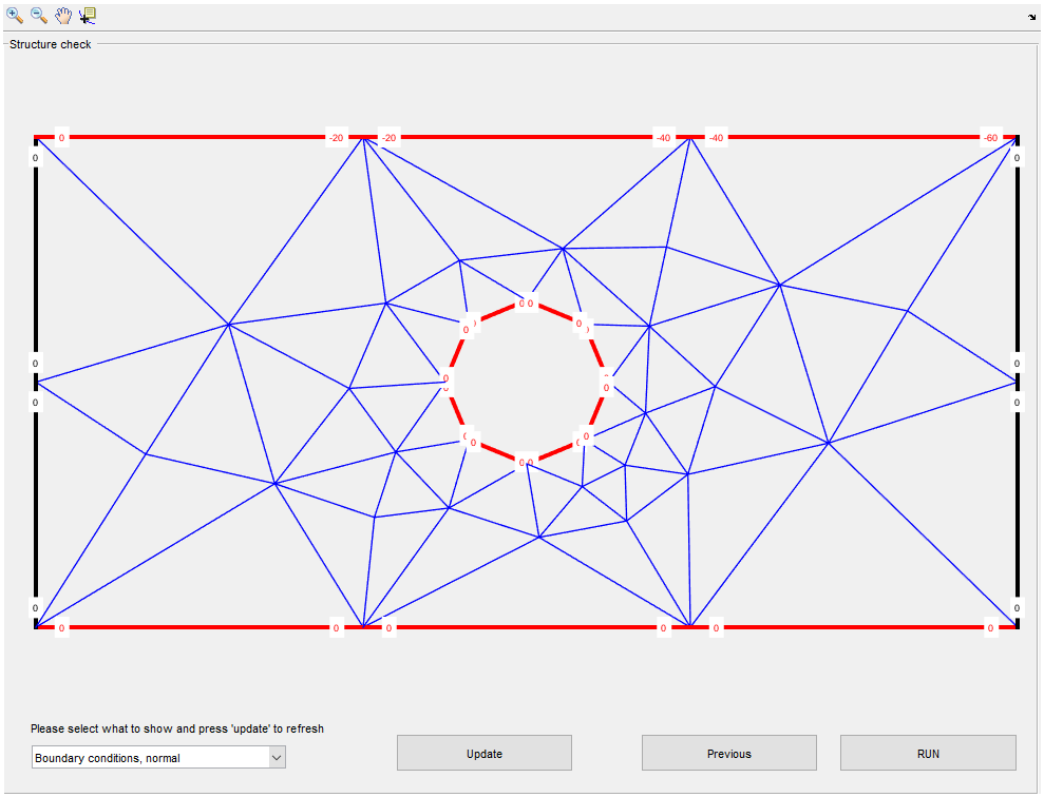


Figure 26. Visualization GUI in the boundary normal mode

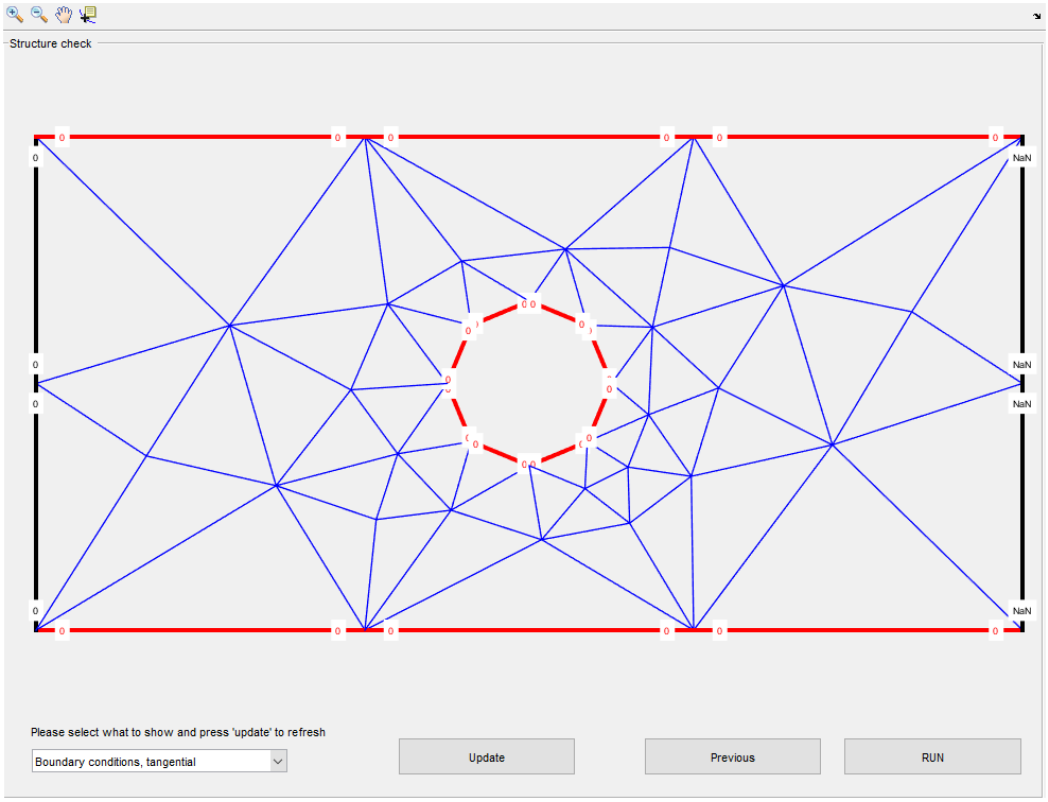


Figure 27. Visualization GUI in the boundary tangential mode

5.6. POST-PROCESSING

Post-processing in **FreeHyTE – STRUCTURAL HTD** is limited to the writing of the output data file (if the user choses to save the model, see Section 5.2.2) and the plotting of the analysis results. The displacement and stress plots obtained for the solid and hollow beams presented in Sections 4.3.1 and 4.3.2 are given below, to endorse the comparison between the various h- and p-refinement options.

5.6.1. The solid beam results

Two refinement schemes were used for the solution of the solid beam problem, as detailed in Sections 4.4.1 and 4.5.2. The first scheme involved an 18-element mesh and refinement orders of $n_\Gamma = 6$ and $n_D = 15$ on the essential boundaries and in the finite element domains, respectively. The second scheme used one single finite element, but increased the refinement orders to $n_\Gamma = 13$ and $n_D = 29$ to compensate for the coarseness of the mesh.

The displacement and stress predictions as plotted by the **FreeHyTE – STRUCTURAL HTD** post-processor are presented in Figure 28 and Figure 29, respectively. The same scales were used in both plots to enable the direct comparison.

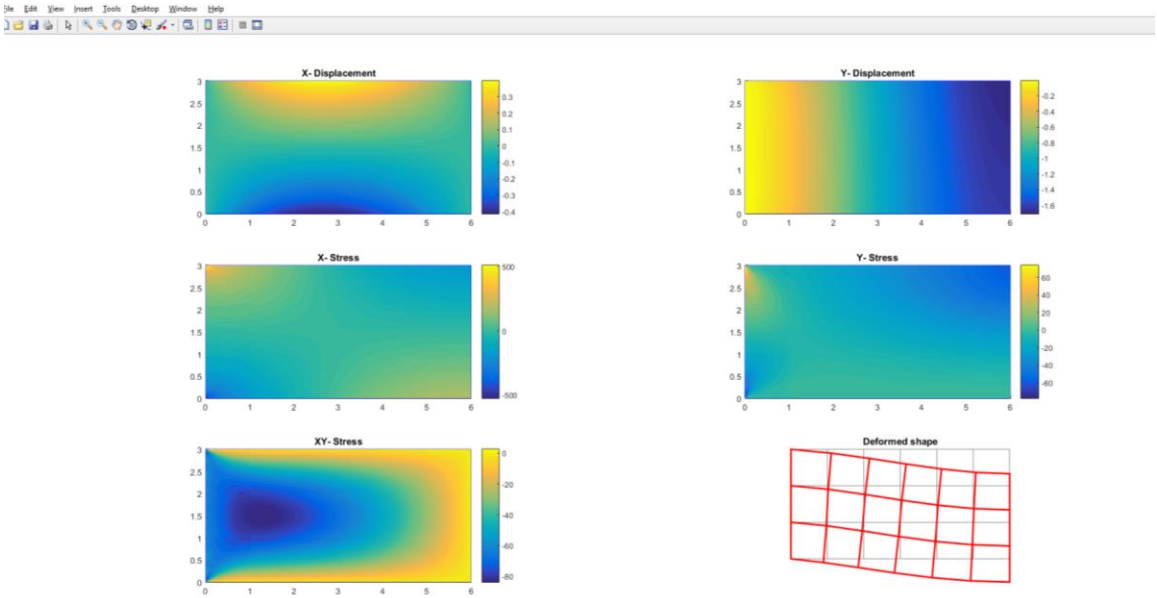


Figure 28. Displacement and stress plots for the solid beam problem, 18 finite elements

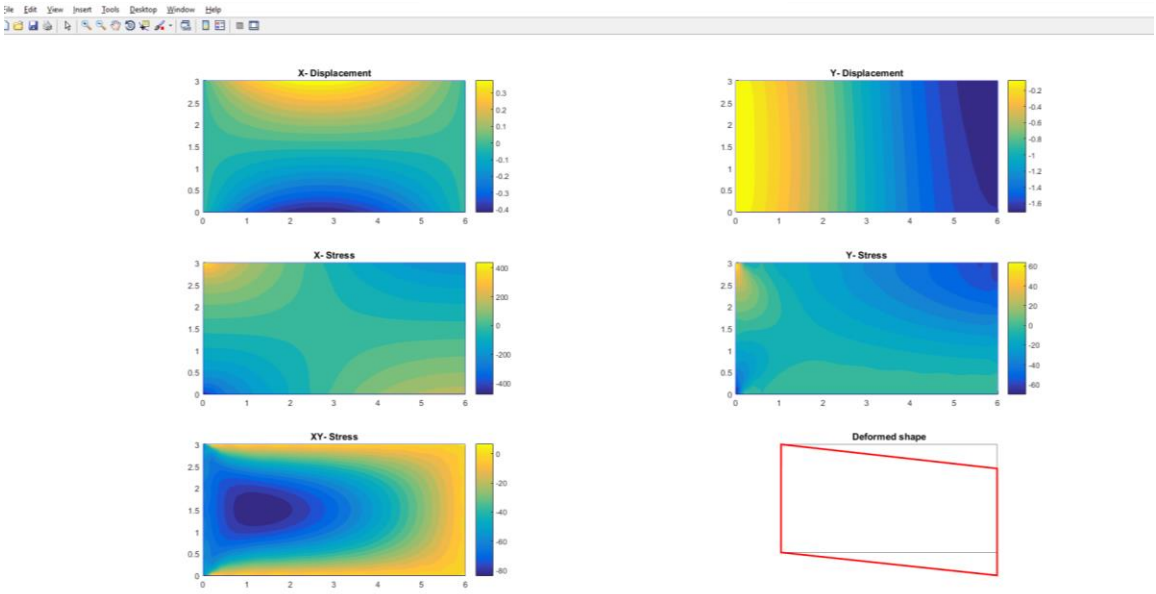


Figure 29. Displacement and stress plots for the solid beam problem, one finite element

It is clear that the results are practically identical in both cases for both displacement and stress fields (the deformed shape is just a linear interpolation of the corner displacements, so it is irrelevant for the coarse mesh). Reader's attention is called to the smoothness in the stress transitions between neighbouring elements in Figure 28, despite having used no stress averaging in the plots. As explained in more detail in Section 3.1.1, this is one of the most important advantages in using hybrid-Trefftz finite elements.
The total analysis times were 3.42 sec for the finer mesh case (with 1557 degrees of freedom) and 0.73 sec for the coarser mesh case (with 160 degrees of freedom). No ill-conditioning of the solving system was reported in either case.

5.6.2. The hollow beam results

Two refinement schemes were used for the solution of the hollow beam problem, as detailed in Sections 4.4.2 and 4.5.2. The first scheme involved a 62-element mesh and refinement orders of $n_\Gamma = 6$ and $n_D = 11$ on the essential boundaries and in the finite element domains, respectively. The second scheme used a 12-element mesh, but increased the refinement orders to $n_\Gamma = 13$ and $n_D = 22$ to compensate for is coarseness.

The displacement and stress predictions as plotted by the **FreeHyTE – STRUCTURAL HTD** post-processor are presented in Figure 30 and Figure 31, respectively. The same scales were used in both plots for direct comparison.

Except for some slight discontinuity in the vertical displacements predicted by the coarse mesh model, the results are practically identical for both refinements, despite the very strong mesh distortion and large element sizes present in the coarse model.

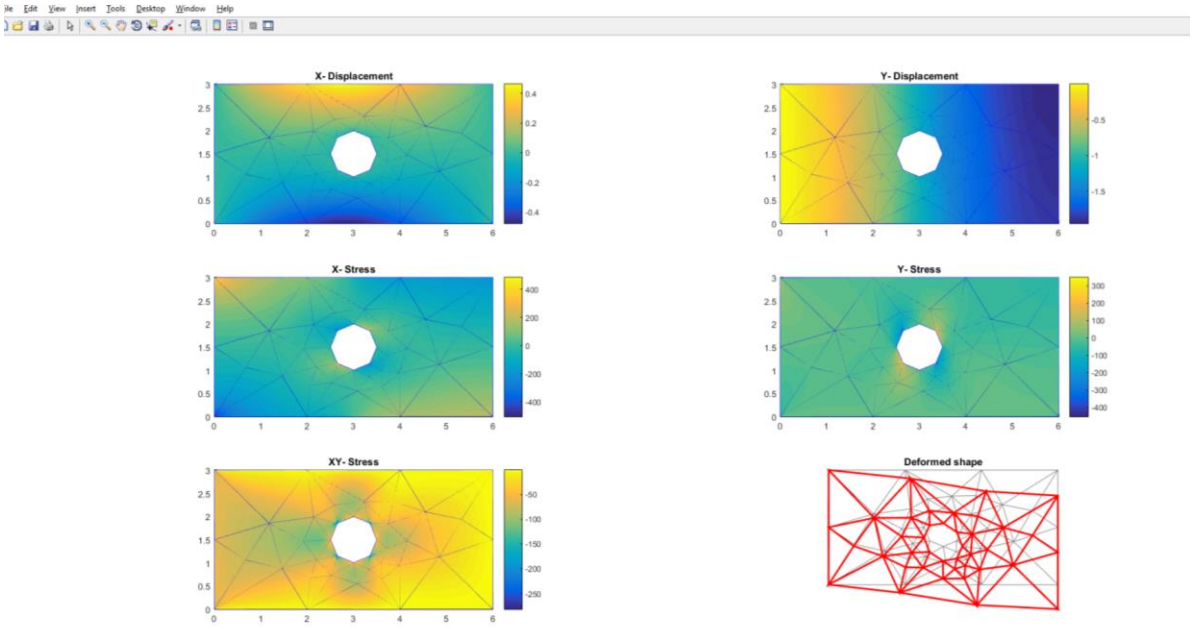


Figure 30. Displacement and stress plots for the hollow beam problem, 62 finite elements

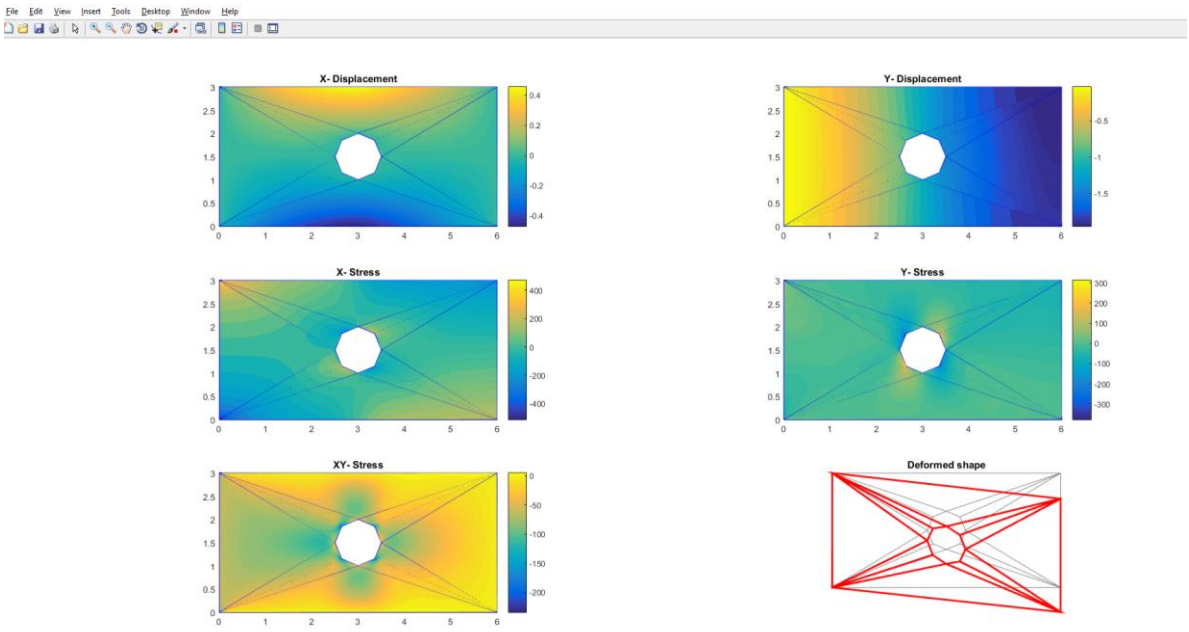


Figure 31. Displacement and stress plots for the hollow beam problem, 12 finite elements

The total analysis times were 9.19 sec for the finer mesh case (with 4070 degrees of freedom) and 2.79 sec for the coarser mesh case (with 1458 degrees of freedom). Ill-conditioning of the solving system was reported in the case of the coarse mesh due to the high orders of the approximation bases, but had no significant effect on the solution. However, in such situations, **the user is strongly advised to avoid ill-conditioned systems by refining the mesh and lowering the orders of p-refinement**.

6. ADVANCED STRUCTURAL DEFINITION

6.1. INTRODUCTION

While appropriate for the majority of applications, the standard operation of **FreeHyTE – STRUCTURAL HTD** fails to take advantage of one of the most important advantages of hybrid-Trefftz finite elements, namely the localized p-refinement capability. Indeed, as discussed in Section 4.5, the Graphical User Interface (GUI) of **FreeHyTE – STRUCTURAL HTD** only allows uniform basis refinements to be defined, but these definitions can be programmatically overwritten to specify different orders of refinement for different finite elements and essential boundaries. A simple procedure to do so is described in Section 6.2.

Another limitation of the GUI-based **FreeHyTE – STRUCTURAL HTD** is that a single material type can be used in the structure definition. This limitation can be overcome following the same procedure as for the localized p-refinement. This topic is covered in Section 6.3.

6.2. LOCALIZED P-REFINEMENT

The concept of localized p-refinement was introduced in Sections 3.1.4 and 4.5.1. The procedure may improve the finite element solutions in areas where localized effects like stress concentrations or discontinuities are expected to occur, without the need of incrementing the order of the bases in areas where doing so would bring no significant improvement.

Localized p-refinement can be defined programmatically, by slightly modifying the *InputProcReg.m* file if you use the regular mesh generator or *InputProcTri.m* file if you use the non-regular mesh generator.

Both of these files are found in the installation folder of **FreeHyTE – STRUCTURAL HTD**. The procedure for the definition of localized p-refinement involves the following steps:

- define your model as usual, inserting the typical orders of boundary and finite element refinements in GUI 1;
- at any point between the creation of the mesh and pressing the RUN button in the verification interface, open the *InputProcReg.m* or *InputProcTri.m* file, according to the mesh generator you use;
- in the Matlab code, look for the areas marked with '`% EDGE REFINEMENT DATA`' or '`% ELEMENT DATA`' (see Figure 32);
- set the new orders of approximation for the desired Dirichlet boundaries and finite elements according to the examples indicated in the **.m* file. For instance, to set the order *a* for the boundary *b* and element *c*, the input lines should be something like,

```
Edges.order(b) = a;

Loops.order(c) = a;
```

- use as many lines as needed to overwrite all orders that you wish to change;
- remember to change the code back after the analysis is completed.

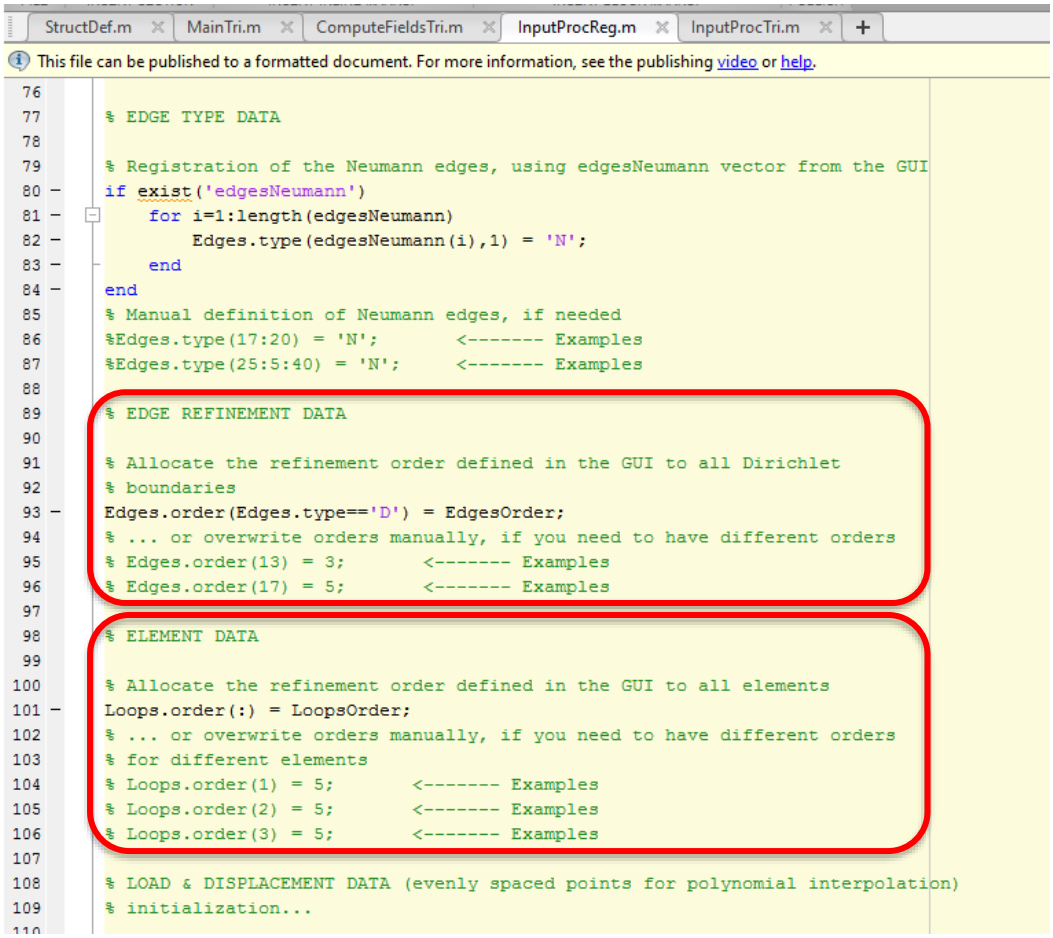


Figure 32. Code areas to modify for localized p-refinement

The localized p-refinement must be defined such as to **satisfy the kinematic indeterminacy condition**, as explained in Section 4.5. This means that, for every finite element, **the number of degrees of freedom inside the element must be superior to the sum of the degrees of freedom on its essential** (Dirichlet and interior) **boundaries**. Denoting by n_D the order of the domain basis of a finite element and by n_Γ^i the order of the approximation basis on its essential boundary *i*,

the **kinematic indeterminacy criterion is always satisfied if**,

$$2n_D - 1 \geq \sum_{i=1}^{S} \left(n_\Gamma^i + 1 \right)$$

where S is the total number of essential boundaries of the element.

6.3. DEFINITION OF DIFFERENT MATERIALS

The same procedure described in Section 6.2 can be applied to overwrite the uniform material description given in GUI 1. The code (**.m*) files to change are the same and the necessary steps are listed below:

- define your model as usual, inserting the typical material characteristics in GUI 1;
- at any point between the creation of the mesh and pressing the RUN button in the verification interface, open the *InputProcReg.m* or *InputProcTri.m* file, according to the mesh generator you use (regular or non-regular);
- in the Matlab code, look for the area marked with '`% MATERIAL DATA`' (see Figure 33);
- set the new material characteristics for the desired finite elements according to the examples indicated in the **.m* file. For instance, to set the Young's modulus *a* and Poisson's coefficient *b* for the element *c*, the input lines should be something like,

```
Loops.materials(c,1) = a; % '1' designates the Young's modulus

Loops.materials(c,2) = b; %'2' designates the Poisson's coefficient
```

- use as many lines as needed to overwrite the material characteristics for all elements that you wish to change;
- remember to change the code back after the analysis is completed.

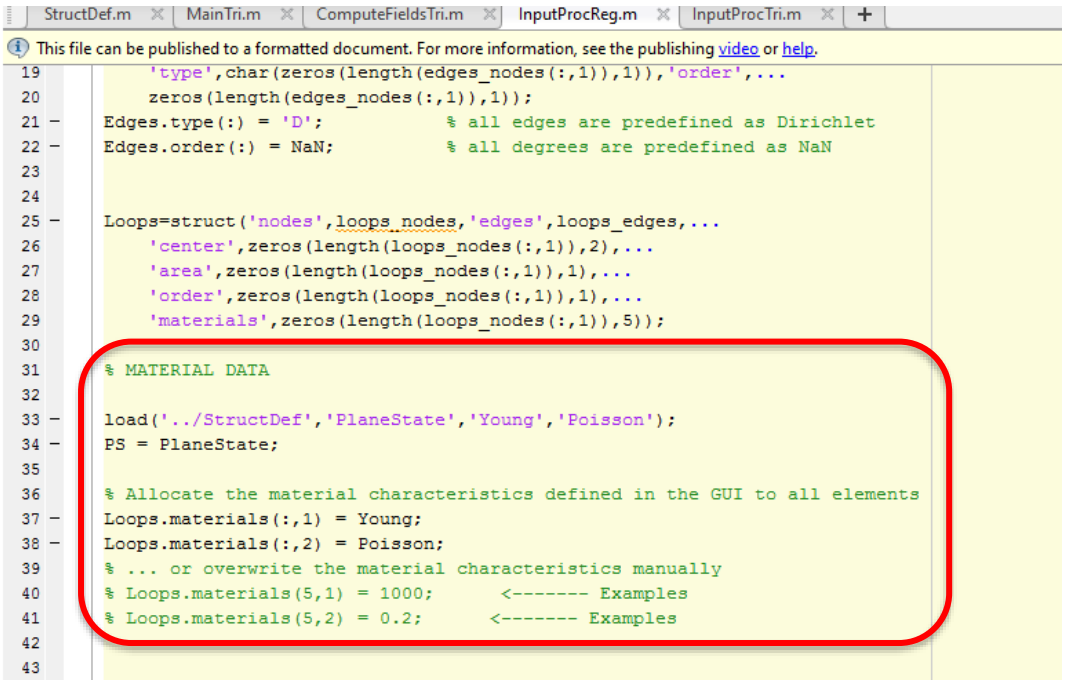


Figure 33. Code areas to modify to overwrite material properties