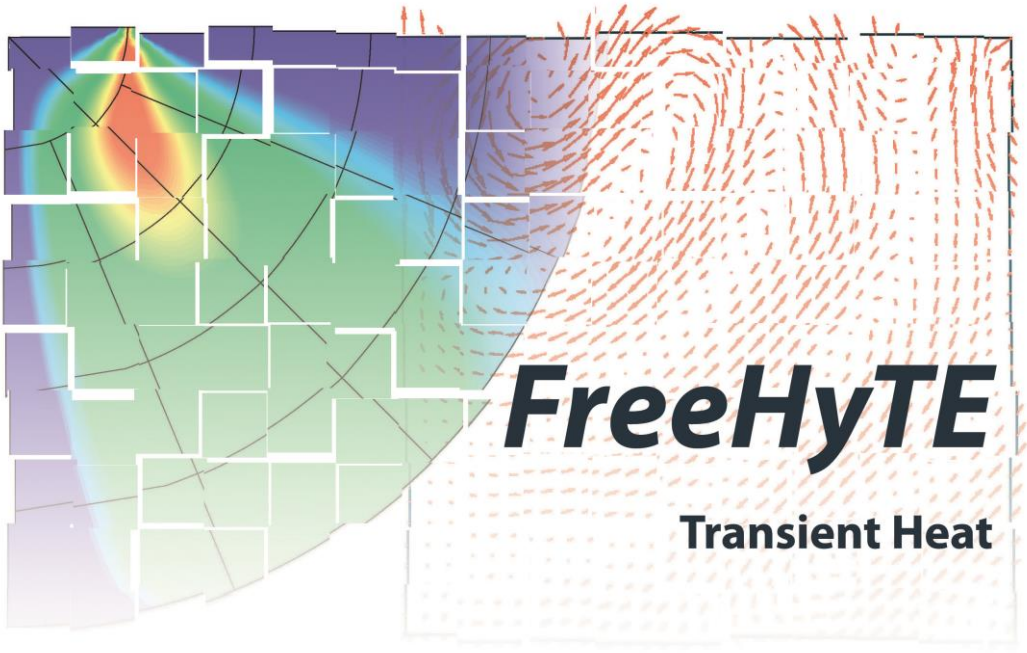
CERIS : Civil Engineering Research and Innovation for Sustainability



USER'S MANUAL

V1.2

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

DISCLAIMER

FreeHyTE – Transient heat

Hybrid-Trefftz Temperature Finite Elements for Transient Heat Conduction

CERIS : Civil Engineering Research and Innovation for Sustainability

TÉCNICO LISBOA

ACKNOWLEDGEMENTS

FreeHyTE – Transient heat

Hybrid-Trefftz Temperature Finite Elements for Transient Heat Conduction

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 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 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 2018, 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 – TRANSIENT HEAT

The **FreeHyTE – Transient heat** module uses the temperature model of the hybrid-Trefftz finite elements for the solution of transient heat conduction problems in two dimensions.

Unlike conforming temperature (conventional) finite elements implemented in the vast majority of commercial codes, hybrid-Trefftz temperature finite elements (HTTE) use independent approximations of the temperature in the domains of the elements and of the normal heat flux on the Dirichlet and interior boundaries (hence the *hybrid* label of the elements). Moreover, the domain temperature 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 relatively large finite elements as compared to the conventional models and can handle problems involving high solution gradients and discontinuous solution fields without cumbersome local mesh refinements. Awkward geometries 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 temperatures 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 or convection/Robin, 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 – Transient heat 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 the use of **FreeHyTE – Transient heat** module for the solution of transient heat diffusion problems. Consequently, the manual follows a strictly need-to-know, user-oriented perspective and should provide enough information for the needs of beginner and advanced users alike. For in-depth information regarding the implementation of the hybrid-Trefftz elements in FreeHyTE, please refer to the Developer's Manual.

Chapter 2 focuses on *when* you should use **FreeHyTE – Transient heat**. 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 as much about the hybrid-Trefftz temperature elements (HTTE) 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 HTTE elements and explains the conventional positive directions of the involved referentials.

Chapter 5 focuses on *how* **FreeHyTE – Transient heat** module should be used. It presents the Graphical User Interface (GUI), and illustrates its use for the analysis of a rectangular plate and an I-beam subjected to a fire.

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 – Transient heat Installation Manual**, which introduces the various module deployment options and the respective installation steps.

Please note that the module was tested in Matlab versions 2016a. Kindly let us know if you experience difficulties in using it under other versions of Matlab.

2. STRUCTURES AND 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 result*. 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. MATHEMATICAL MODEL

2.1.1. Simplifying assumptions

The behaviour of solids exposed to intense heat 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 – Transient heat**:

- the material behaviour is **physically linear**, meaning that the heat flux and the gradient of the temperature are related by a constant conductivity coefficient;
- the material is **piecewise homogeneous and uniform**, meaning that its thermal 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 interfaces between different materials have **negligible contact resistance**, meaning that no temperature drop occurs at such boundaries;
- the material is **isotropic**, meaning that its mechanical characteristics are the same in all directions.

Besides these simplifying assumptions, the following limitation specific to the **FreeHyTE – Transient heat** module is in place:

- the structure has a *plane* behaviour. This does not necessarily mean that the structure must be plane, but that its behaviour can be fully captured by knowing the states of temperature and heat flux in a single plane. This situation is typical to structural elements subjected to a uniform temperature along their length, but not along their width and height. 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 is not on our short-term agenda.

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

2.1.2. Governing equations

Based on the simplifying assumptions presented above, the mathematical model expresses the *thermal energy balance* (1) of a differential element in the domain (Ω) of the heated body presented in Figure 1, and the linear relationship between the heat flux and the temperature gradient, that is, the *Fourier law* (2).

$$k \cdot \nabla^2 T(x,y,t) + Q(x,y,t) = \rho \cdot c \cdot \dot{T}(x,y,t) \text{ , in } \Omega \tag{1}$$

$$\mathbf{q}(x,y,t) = -k \cdot \nabla T(x,y,t) \text{ , in } \Omega \tag{2}$$

In equations (1) and (2), $T(x,y,t)$ and $\mathbf{q}(x,y,t)$ are the (scalar) temperature and (vectorial) heat flux fields, $Q(x,y,t)$ is the internal heat source, and k, ρ and c are the thermal conductivity, density and specific heat capacity of the material. The differential operators ∇^2 and ∇ designate the gradient and Laplace operators, respectively, and the dot over a field variable denotes its differentiation in respect to time.

Differential equations (1) and (2) are complemented by the initial condition,

$$T(x,y,0) = T_0(x,y) \text{ , in } \Omega \tag{3}$$

where $T_0(x,y)$ is the initial temperature field in the domain of the problem.

Finally, for the full definition of the problem, the temperatures and normal heat fluxes applied to the boundaries of the domain must also be specified. Three types of external boundaries can be defined in **FreeHyTE – Transient heat**. They are:

- the *Dirichlet* boundary condition, where the temperature field T_Γ is enforced on the boundary Γ_d (see Figure 1),

$$T(x,y,t) = T_\Gamma(x,y,t) \text{ , on } \Gamma_d \tag{4}$$

- the *Neumann* boundary condition, where the normal heat flux field q_Γ is enforced on the boundary Γ_n ,

$$\mathbf{n}\cdot\mathbf{q}(x,y,t) = q_\Gamma(x,y,t) \text{ , on } \Gamma_n \tag{5}$$

In expression (5), $\mathbf{n}$ collects the components of the normal to the boundary.

- the *Robin* (convection) boundary condition, where the linear relationship (6) between the temperature and normal flux fields is enforced on the boundary Γ_r ,

$$\mathbf{n}\cdot\mathbf{q}(x,y,t) = h\cdot\left(T(x,y,t) - T_f(t)\right) \text{ , on } \Gamma_r \tag{6}$$

In expression (6), T_f is the temperature of the surrounding fluid in the vicinity of the Robin boundary, and h is the convection coefficient.

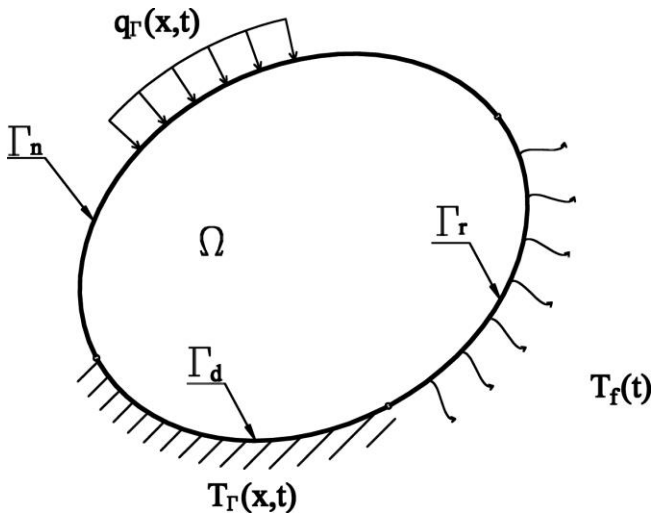


Figure 1. Problem's domain, Dirichlet, Neumann and Robin boundary conditions.

2.2. APPROXIMATE SOLUTIONS

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

In this context, **FreeHyTE – Transient heat** is a hybrid-Trefftz finite element solver of the differential equations describing the heat conduction through a solid body. Upon completion of the analysis, **FreeHyTE – Transient heat** should be able to predict the temperature and heat flux fields that occur throughout the body at any given time.

Since it *approximates* the solution of the governing equations, **FreeHyTE – Transient heat** 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 should be controlled by the user through a process known as the *refinement* of the model, as discussed in Section 4.5. 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.

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 temperature) 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 transient plane heat conduction problems are described in Section 3.2.

3.1. CONVENTIONAL vs. HYBRID-TREFFTZ TEMPERATURE FINITE ELEMENTS

The comparison between conventional and hybrid-Trefftz temperature finite elements focuses the following aspects:

- the techniques employed by these formulations to 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 – Transient heat**.

3.1.1. Enforcement of the governing equations

The description of the transient heat conduction problem is given in Section 2.1.2. To recover its solution, three domain equations need to be solved, namely equations (1) to (3), subjected to three boundary conditions, expressed by equations (4) to (6). Because of the division of the domain into finite elements, the flux balance and temperature compatibility on the internal (inter-element) boundaries also need to be enforced.

These eight equations are originally expressed in both time and space variables. Regardless on the type of finite element that is used, their solution involves the approximation of the time variation of the unknown (temperature and flux) fields. Several methods can be used for this approximation. In **FreeHyTE – Transient heat**, it is handled using a generalized mid-point (finite difference) algorithm. This time discretization process reduces the original problem in time and space to a series of problems defined only in space, one for each time step the duration of the problem was divided into.

As exact solutions for these problems cannot be found, in general, some of the governing 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 Fourier law in the domain of the element and the Dirichlet and Robin boundary conditions on the exterior Dirichlet and Robin boundaries.

On the interior boundaries, the temperature compatibility is also exactly satisfied. The domain energy balance equation is enforced weakly and the Neumann and flux balance boundary conditions are enforced at the nodes of the mesh. As a consequence, the temperature field is typically approximated with considerably superior precision as compared to the heat flux field.

Hybrid-Trefftz temperature finite elements. Hybrid-Trefftz temperature elements satisfy strongly *all domain*[2] equations. The Dirichlet, Neumann and Robin boundary conditions are enforced weakly on the exterior boundaries of the structure. Temperature compatibility is also enforced weakly on the interior boundaries of the mesh. However, the heat flux balance is not enforced at all on the interior boundaries. Since temperature and heat flux-related equations are enforced in the same way (except for the interior boundaries), the predictions of the respective fields are much more balanced in terms of quality than in the case of conventional elements, although some slight bias towards the temperature field is still present.

[2] This is actually an overstatement, as hybrid-Trefftz elements only satisfy the homogeneous form of the governing equations. The particular solution is only exact in certain points in the domain (collocation).

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 temperatures are the main unknowns (degrees of freedom) of the problem and all fields in all points of the domain are completely determined by their values. The number and localization of the nodes also determine the expressions of the (polynomial) approximation functions for the temperature 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 Dirichlet and temperature continuity 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 generally requires the redefinition of all elements of the mesh. The approximation functions are intrinsically able to recover exactly a constant temperature field, through the partition of unity property, meaning that the temperature field will converge to the exact solution as the elements grow smaller (i.e. the mesh gets more refined).

Hybrid-Trefftz temperature finite elements. Hybrid-Trefftz temperature elements approximate not only the **temperature field in the domain of the element**, but also the **heat flux field on its essential** (i.e. exterior Dirichlet, Robin and interior) **boundaries**. Moreover, the temperature approximation in the domain is split into two parts: one that approximates the particular solution of the governing equation and one that approximates its complementary solution.

All approximations abandon the concept of nodes altogether. The nodal temperatures are no longer the main unknowns of the problem and the redefinition of the nodes does not call for the 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 temperature approximation functions embody relevant physical information on the phenomenon they model. They are tailored for each problem and account for most super-convergent features of the hybrid-Trefftz elements, as discussed in Section 3.2. However, they are generally more difficult to integrate than their conventional element counterparts, especially when their order is high. The approximation functions are intrinsically able to recover exactly a constant temperature field, through the inclusion of the explicit unit monomial in the domain approximation basis, meaning that the temperature 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 energy balance condition. As a consequence, the solving system is sparse and symmetric, but the nodal temperature variables are shared by all finite elements that converge in the respective node. Summation of conductivity matrices of neighbouring elements occurs on the main diagonal of the system. The system is always *kinematically indetermined* (meaning that all nodal temperatures cannot be recovered using only the Fourier law and the temperature continuity boundary condition), provided there is at least a single node with a free (undetermined) temperature. The system is not *singular* provided the temperature in at least one point is enforced and generally not *ill-conditioned*.

Hybrid-Trefftz temperature finite elements. The solving system of hybrid-Trefftz finite elements is assembled with no summation of conductivity matrices on the main diagonal. The generalized temperature 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 temperature bases. On the other hand, since large Trefftz elements are affordable, the solving systems generally result smaller, in practice, than the conventional ones. 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, Robin and interior) boundaries, as discussed in Section 4.5.2. The system is not *singular* provided the temperature is specified on at least a boundary, but can be more *ill-conditioned* than its conventional counterpart. Ill-conditioning is controlled in **FreeHyTE – Transient heat** by calculating the condition number of the matrix of coefficients and by applying pre-conditioning to the system. If the system is still 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), or 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 limit, in practice, the domain bases to low-order polynomials.

Hybrid-Trefftz temperature 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 conductivity 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 elements 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 to 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 temperatures and heat flux results;
- shape functions that embed the physics of the problem;
- large finite elements are affordable. Leading dimensions can be 5-10 times larger than those of conventional elements;
- very high convergence under p-refinement. High-order approximation bases are mainstream;
- flexibility in choosing the orders of the approximation bases in each element and on each essential boundary. Localized p-refinement is affordable;
- results are insensitive to high solution gradients and gross mesh distortions;
- 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 can be tough to handle for inexperienced users;
- users need to ensure the kinematic indeterminacy of the solving system (this drawback is avoided if the predefined sets of refinement orders presented in Section 4.5.2 are used);
- solving system is prone to ill-conditioning, especially if the orders of approximation bases are too high.

4. STRUCTURE DEFINITION IN FreeHyTE – Transient Heat

4.1. INTRODUCTION

This chapter is a hands-on guide to the structure definition in **FreeHyTE – Transient heat**. 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, and the definition of the boundary conditions.

FreeHyTE – Transient heat 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 or other geometric non-regularities** (generates **triangular elements**).

These concepts are applied to the solution of two problems, one featuring a square steel plate and the other an I steel profile, both subjected to a fire. The first structure is analysed using a regular mesh with square elements, while for the second structure, the mesh features triangular elements and is obtained using an automatic generator built into Matlab. 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 obtained with **FreeHyTE – Transient heat** are presented, compared and analysed (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 computational effort.

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

4.2. SYSTEMS OF REFERENCE

Before moving on to the presentation of the two case studies, 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 a **boundary normal–tangential referential**. Both referentials are Cartesian.

To illustrate this principle, Figure 2 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.

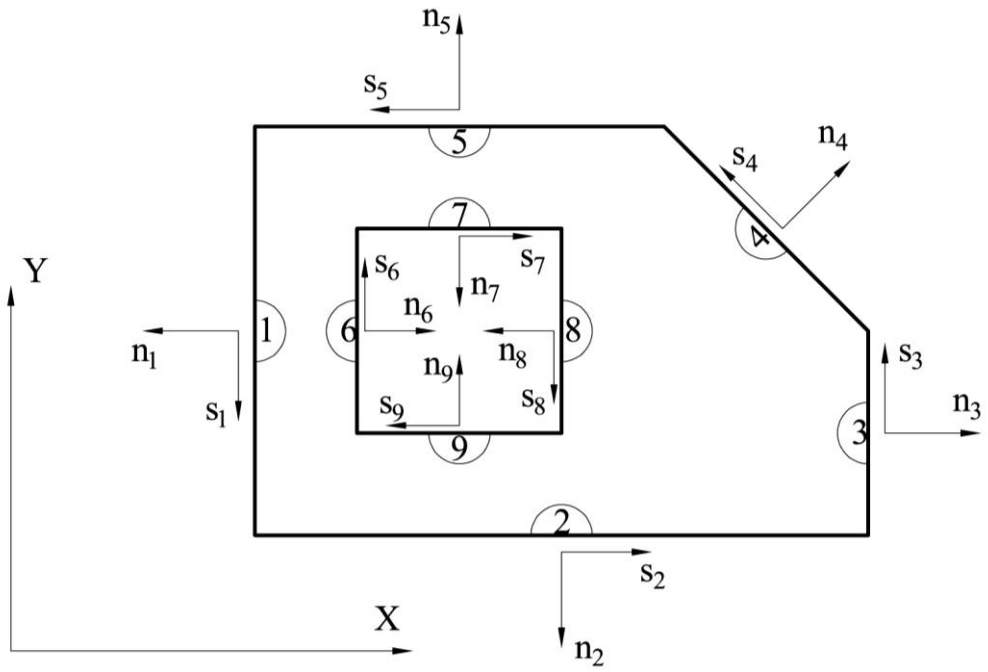


Figure 2. Positive directions of global and side referentials

The conventional positive directions of the boundary normal – tangential referential, are as follows:

- the **boundary normal axis** (n) is oriented **outwards from the element**;
- the **boundary tangential axes** (s) 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**.

When defining normal heat fluxes on the Neumann boundaries of the mesh, their signs must be tuned according to the orientation of the boundary normal axis (n). 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, involving a rectangular steel plate and a steel I-beam, subjected to a fire. The two examples are presented in this section.

4.3.1. The rectangular plate

Consider the rectangular plate presented in Figure 3, made of European structural steel, with a thermal conductivity $k = 40\,[W/(m\,^{\circ}C)]$, mass density $\rho = 7860[W/m^3]$, and specific heat $k = 490\,[J/(kg\,^{\circ}C)]$. The internal heat generation is null. The initial conditions correspond to a constant temperature $T_0 = 32.5\,^{\circ}C$ throughout the plate and the surrounding air. The rectangular plate is subjected to a fire which causes the heating of the air underneath its lower boundary, while the air around its upper boundary remains at the initial temperature (Figure 4). The lateral boundaries of the steel plate are thermally insulated, which corresponds to a null flux boundary condition.

The fire event is described by the time-temperature curve presented in Figure 5. It is a customized fire curve for tunnel design, which takes into account the activation of the sprinklers and the response of the fire squad, after roughly 25-30 minutes. The effect of the fire is applied by means of Robin (convection) boundary conditions of type (6). The fluid temperature $(T_f(t))$ is defined by the time-temperature curve in Figure 5, on the lower boundary, and maintained constant at $T_f = 32.5\,^{\circ}C$ on the upper boundary. The heat transfer coefficient is set to $h = 15\,W/m^2\,^{\circ}C$ on both convection boundaries.

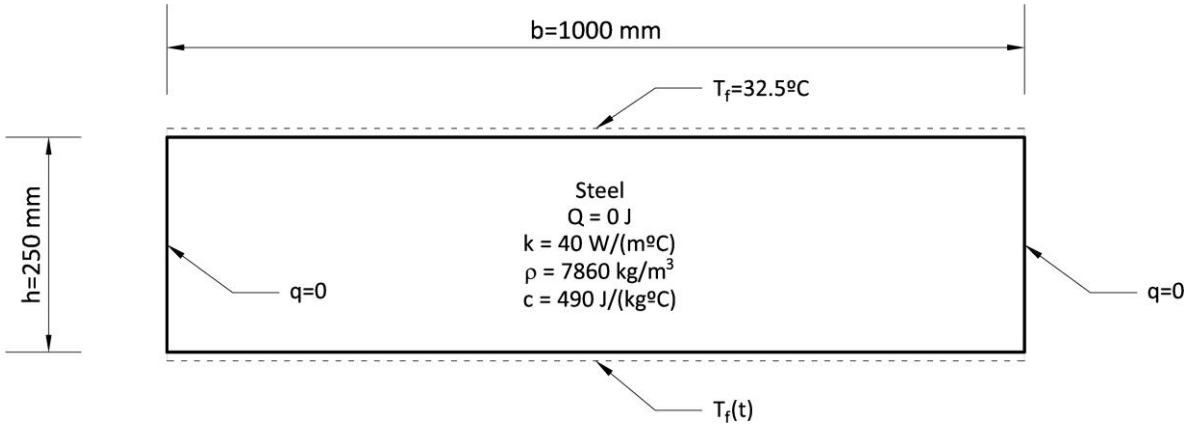


Figure 3. Rectangular plate: geometry and material

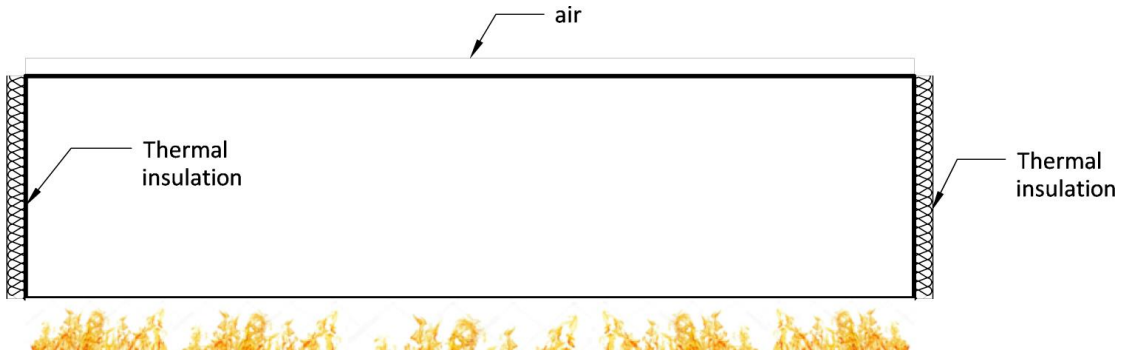


Figure 4. Rectangular plate: boundary conditions

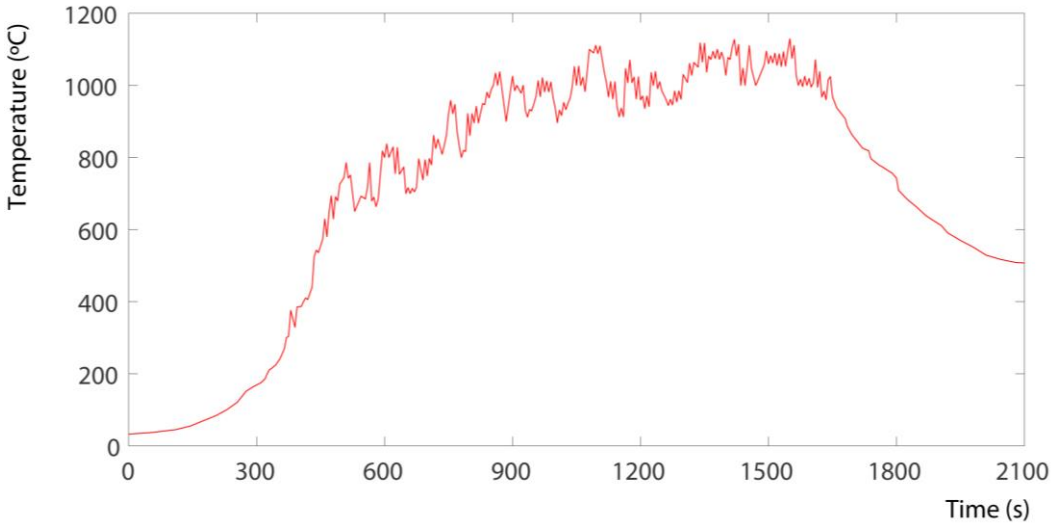


Figure 5. Time-temperature graph of the fire event

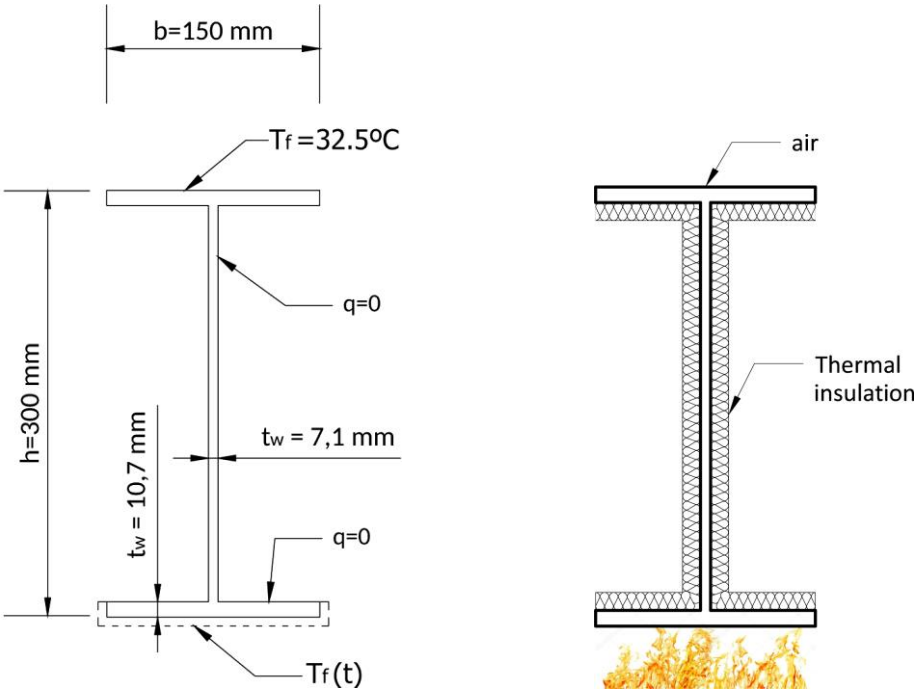
The objective of the analysis is to compute the temperature field throughout the steel plate at 500s, 1000s, 1500s, and 2000s, using **FreeHyTE – Transient heat**.

4.3.2. The I-beam

The second numerical example involves a steel I-beam subjected to a fire. The beam is a European profile IPE300, its cross section presented in Figure 6(a). The material parameters of the I-beam are identical to those of the steel plate presented in the previous section. The initial conditions correspond to a constant temperature $T_0 = 32.5\,^{o}C$ throughout the cross section and the surrounding air. The I-beam is thermally insulated along its web and on the interior side of its flanges, and exposed to the surrounding air on the exterior side of its flanges, Figure 6(b). The beam is subjected to the same fire as described in the previous section (Figure 5). The increase in the air temperature is applied to the lower part

of its bottom flange, while the air around its upper flange remains at the initial temperature.

For both convective boundaries, the heat transfer coefficient is $h = 15\ W/m^2{}^\circ C$.



(a) Cross section of the I-beam (b) Boundary conditions

Figure 6. I-beam subjected to fire: geometry and boundary conditions

The objective of the analysis is to compute the temperature field throughout the I-beam every eight minutes, using **FreeHyTE – Transient heat**.

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 – Transient heat**, 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.

- 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 bodies 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 7). The **origin of the global referential** is taken by default **in the lower left corner of the body** and cannot be changed.

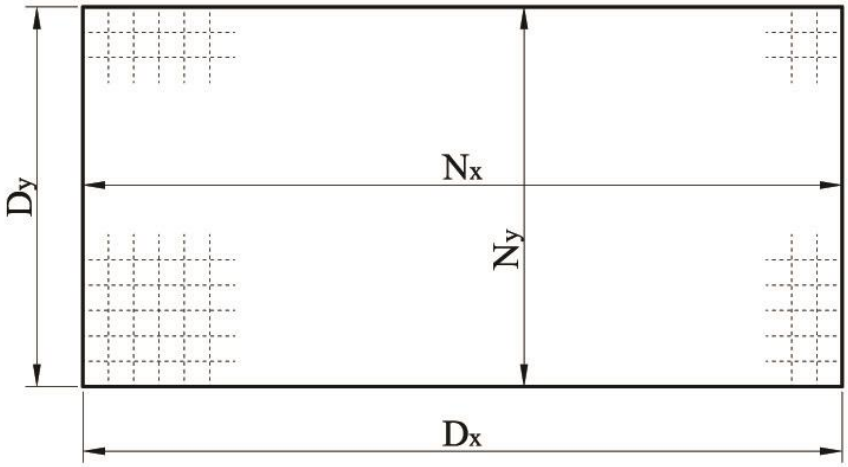
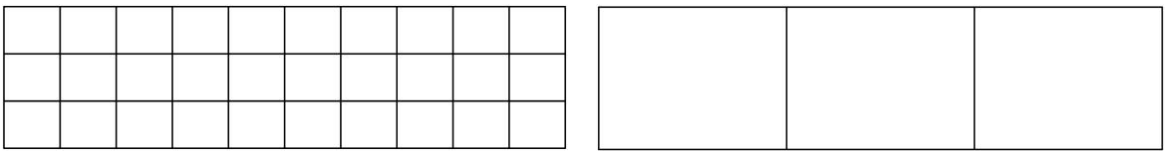


Figure 7. Regular mesh definition of a rectangular structure

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 **5 to 10 times larger than** that of **conventional elements** without compromising the quality of the results.

The rectangular plate described in Section 4.3.1 is discretized using a fine mesh with 30 finite elements (Figure 8a) and a coarse mesh with three finite elements (Figure 8b).



a. Fine mesh b. Coarse mesh

Figure 8. Meshes used for the rectangular plate

4.4.2. Non-regular mesh generator

The non-regular mesh generation protocol in **FreeHyTE – Transient heat** 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 is probably not worthwhile.

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 I-beam presented in Section 4.3.2.

Geometry definition using `pdetool`. The `pdetool` command is **automatically invoked** during the execution of **FreeHyTE – Transient heat** (see Section 5.2.3). The command opens the GUI dialog presented in Figure 9. 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 9). For the case, these shapes are the three rectangles constituting the top and bottom flanges and the web. Each shape receives an automatic label (*R1*, *R2* and *R3*), as shown in Figure 10;

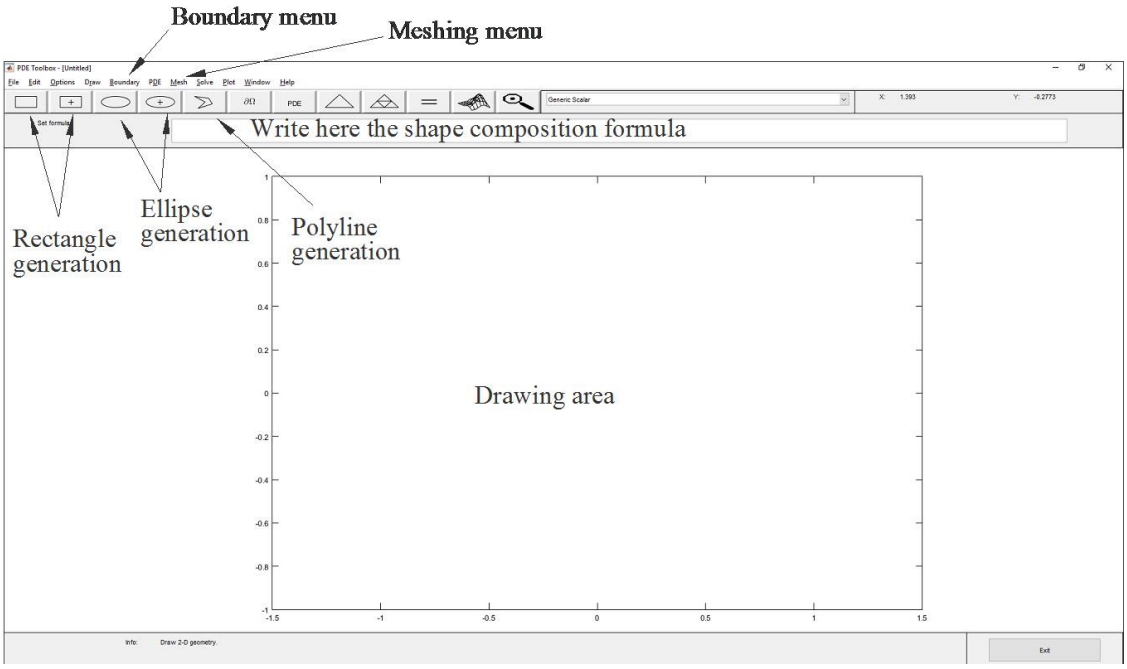


Figure 9. Initial `pdetool` screen

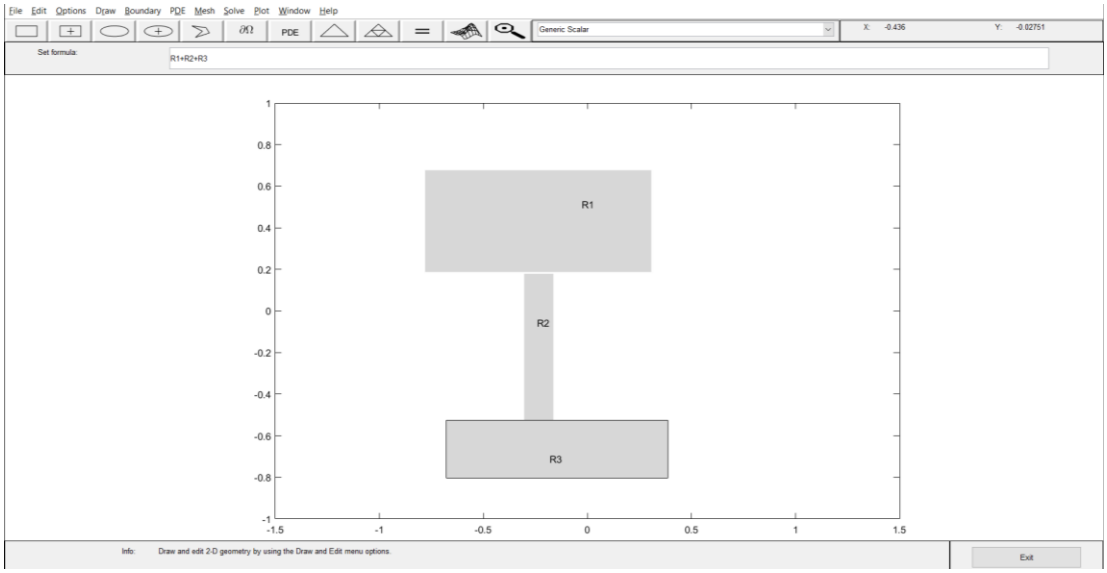


Figure 10. Initial sketch of the geometry

- *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 11). Change the size of the window from *Options->Axes Limits* to fully visualize your structure;

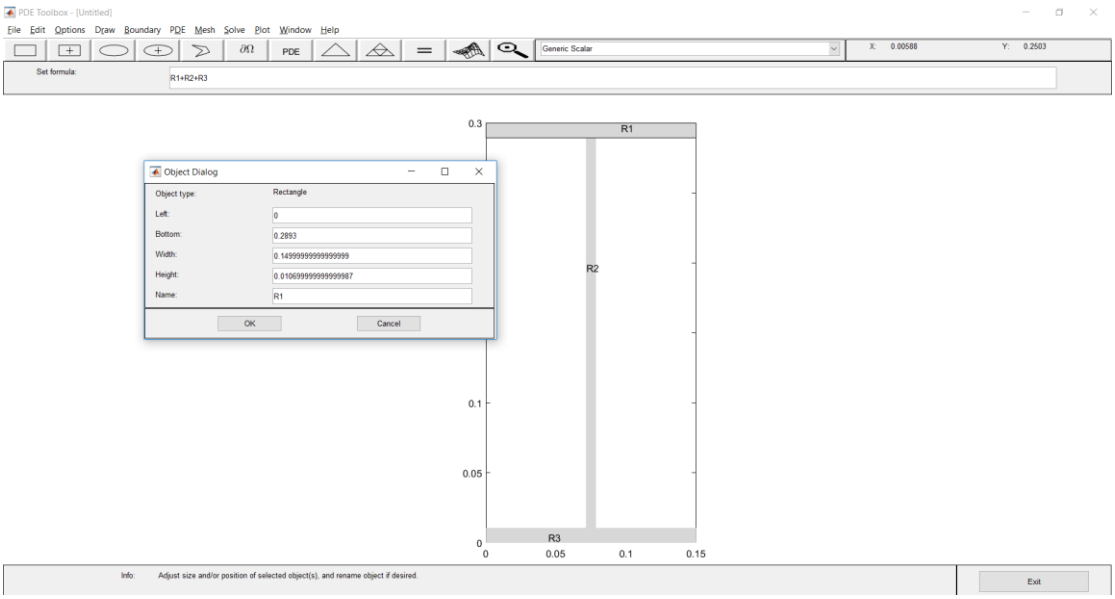


Figure 11. Object definition dialog. Constituent shapes in final form

- *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, we just wish to add the areas together, so the correct formula should read *R1+R2+R3*. After setting up the composition formula, click on *Boundary->Boundary Mode* to apply it to your structure (Figure 12).

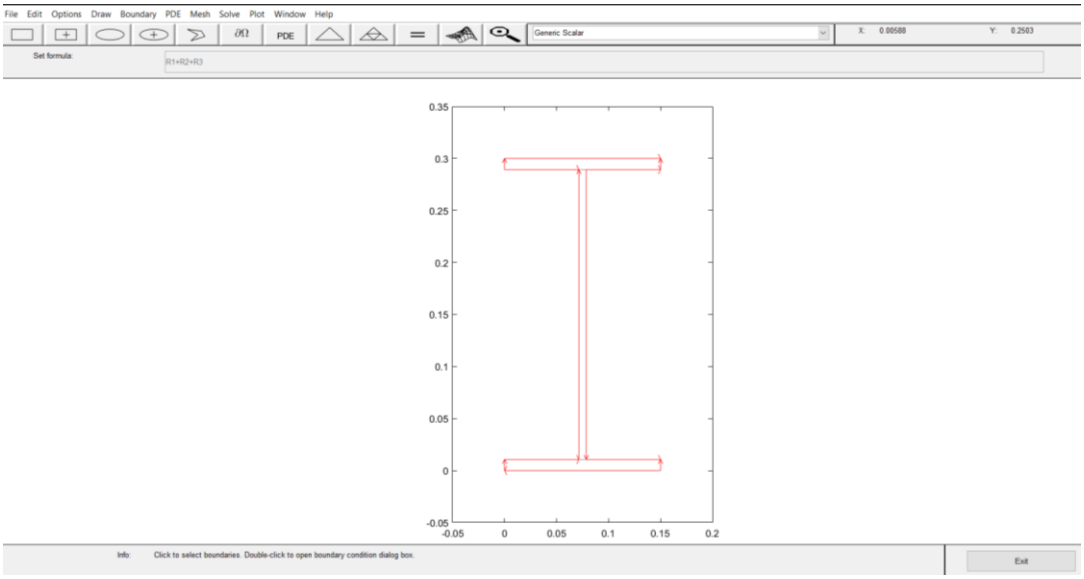
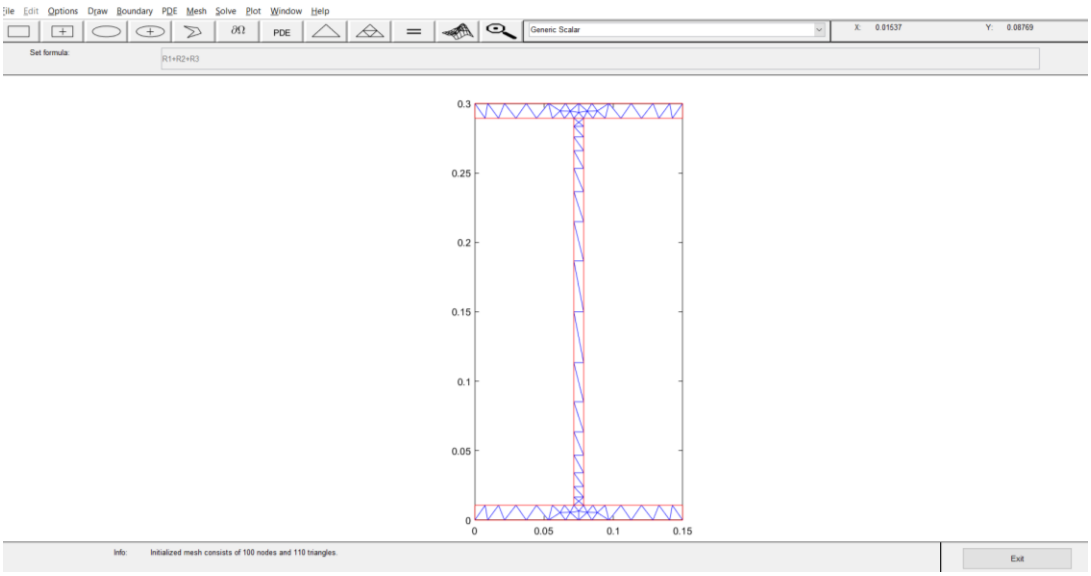


Figure 12. Shape composition formula. Final geometry concluded

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 I-beam, are:
    - *maximum edge size*: Defines the largest triangle edge length. For our case, we chose not to define a restrictive limit for the maximum edge size, and set it to 1.0. This means that the element size is controlled by the thickness of the web and flanges and by the automatic mesh refinement generated by `pdetool` at the intersections of the web and flanges;
    - *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). We use the value of 1.3 for our I-beam;
- *generate the mesh*: Use the *Initialize Mesh* option in the *Mesh* menu of the `pdetool` interface to generate the mesh presented in



- Figure *13*.

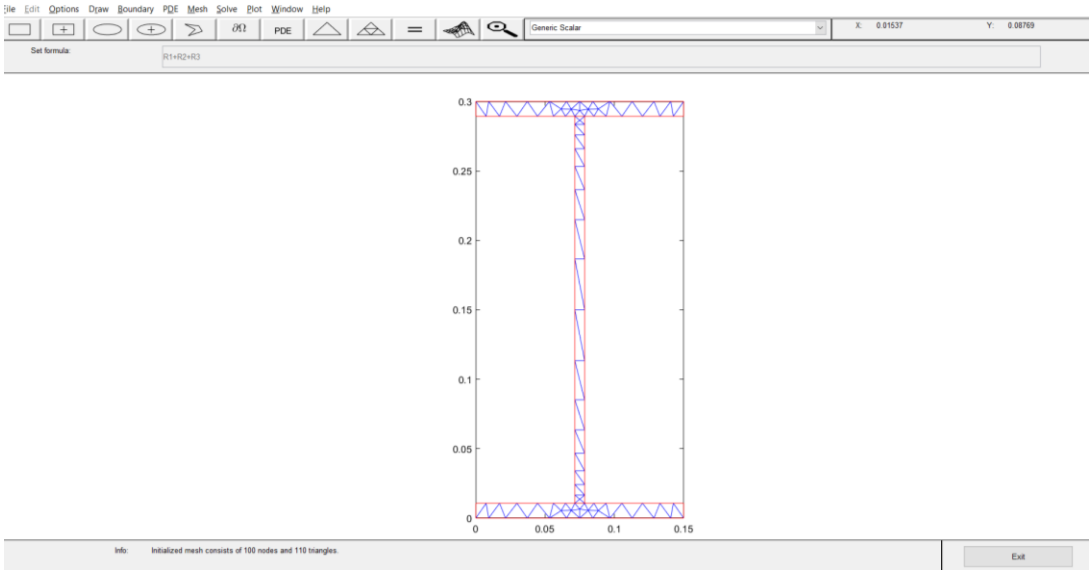


Figure 13. Mesh used for the I-beam

- *export the mesh*: The mesh information must be exported to the `base` workspace in order to be read by **FreeHyTE – Transient heat**. 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 – Transient heat**;
- *save the mesh as an *.m file* (*File->Save As*) if you wish to be able to modify it later using `pdetool`.

You can call `pdetool` from the Matlab command window before executing **FreeHyTE – Transient heat**, or you can execute **FreeHyTE – Transient heat** 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 Matlab version 2015a. For backward compatibility, **FreeHyTE – Transient heat** 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. Strategies for basis refinement

As discussed in Section 3.1.2, 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 – Transient heat**, 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 boundary flux 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 coarse meshes and relatively 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 – Transient heat features two alternatives for defining the refinement of the bases:**

- the **manual definition** of the orders of p-refinement; and,
- the **automatic basis refinement** mode;

In the **manual p-refinement mode, the orders of refinement of all finite elements and all essential boundaries are uniform.** The difficulties associated with securing a kinematically indeterminate system are eliminated by choosing these orders according to the rules presented in the next section.

In the **automatic p-refinement mode, the orders of refinement of the finite elements and essential boundaries are chosen by FreeHyTE – Transient heat.** The program also makes sure that the system is kinematically indeterminate, but the execution time is, usually much larger than in the manual p-refinement mode.

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 – Transient heat** 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.

Hybrid-Trefftz finite elements implemented in **FreeHyTE – Transient heat** approximate independently the particular and complementary solutions of the temperature field in the domain of the elements and the normal heat flux fields on their essential (Dirichlet, Robin, and interior) boundaries. Let n_D^P and n_D^C denote the (uniform) orders of p-refinement in the finite elements (for the particular and complementary solutions, respectively), and let n_Γ denote the order of p-refinement on the essential boundaries. These orders are defined in the GUI, as explained in Section 5.2.3.

The orders n_D^C and n_Γ must be chosen to ensure that, for each finite element, **the total number of temperature degrees of freedom** in the domain **is larger than the total number of heat flux degrees of freedom** on its essential boundary. 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,

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

The approximation basis of the particular solution is not subjected to strong constraints. Since three wave numbers are used to construct it, it would make sense for its order to be roughly two to three times smaller than that of the complementary solution, that is, $\boxed{2n_D^P \le n_D^C \le 3n_D^P}$, but not smaller than three, $\boxed{n_D^P \ge 3}$. However, the best approach to set the order of the particular solution basis is probably the trial and error strategy.

Poor particular solution bases may cause completely meaningless results. If this occurs, the user is advised to set the wave numbers to smaller values (e.g. 0.1, 0.2 and 0.3) and/or to increase the order n_D^P of the particular solution approximation.

Besides the observation of the above inequalities, it is recommended that $\boxed{1 \le n_\Gamma \le 5}$ 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 – Transient heat**, are listed in Table 1, along with the recommended number of Gauss-Legendre quadrature points for the integration of the solving 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	$\min\left(n_D^C\right)$ (triangular)	$\min\left(n_D^C\right)$ (rectangular)	n_D^P	Nº of Gauss points
1	4	4	2-3	10
2	5	6	2-3	10
3	7	8	3-4	10
4	8	10	3-4	15
5	10	12	3-4	15
6	11	14	3-5	15

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

p-refinement for the rectangular beam structure. For the solid 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 8a) and higher for the coarser (Figure 8b). Accordingly, $n_\Gamma = 2$, $n_D^C = 5$, and $n_D^P = 2$ is the choice for the finer mesh and $n_\Gamma = 6$, $n_D^C = 16$, and $n_D^P = 6$ is the choice for the coarse mesh. Note that the orders of the domain bases are very large for the coarse mesh, to compensate for its lack of refinement. Such large orders may cause numerical instability and are not recommended, in general.

p-refinement for the I-beam. The approximation bases of the I-beam presented in Section 4.3.2 are constructed using $n_\Gamma = 2$, $n_D^C = 6$, and $n_D^P = 3$. The low orders are acceptable due to the fine refinement of the mesh (Figure 13).

In all cases, 10 Gauss-Legendre quadrature points were used for the computation of the solving system's coefficients.

The basis refinement data and the number of Gauss-Legendre quadrature points are input in the first GUI (see Section 5.2.4).

4.6. BOUNDARY CONDITIONS

4.6.1. General definition

Three types of boundary conditions can be defined in **FreeHyTE – Transient heat**:

- **Dirichlet boundary conditions**, where the **boundary temperatures** are specified;
- **Neumann boundary conditions**, where the **boundary-normal heat fluxes** are specified; and,
- **Robin boundary conditions**, where the **temperature of the fluid near the boundary** is specified.

All boundary conditions are variable in time. Dirichlet and Neumann boundary conditions are also variable in space. The convective fluid temperature is constant over each boundary but can be different on different boundaries.

The boundary temperatures and normal heat fluxes on the Dirichlet and Neumann boundaries are defined by the product of a function of time (t) and a function of the side coordinate (s),

$$T_\Gamma(s,t)=T_\Gamma^t(t)\cdot T_\Gamma^s(s), \text{ on } \Gamma_d \tag{7}$$

$$q_\Gamma(s,t)=q_\Gamma^t(t)\cdot q_\Gamma^s(s), \text{ on } \Gamma_n \tag{8}$$

These two functions are input by the user following the procedures presented in the next sections.

4.6.2. Definition of the time variation

The time variations of the surface temperatures on the Dirichlet boundaries, heat fluxes on the Neumann boundaries, and fluid temperatures on the Robin (convection)

boundaries are specified separately for each boundary. Their definition can be performed in two ways:

- as an analytic expression of time; or,
- as a time-value list to be loaded from a file.

Definition using analytic expressions. Any analytic expression involving the time variable t is acceptable for the definition of the time variation functions $T_\Gamma^t(t)$, $q_\Gamma^t(t)$, and $T_f(t)$, present in expressions (7), (8), and (6), respectively.

Logical operations can define discontinuous patterns for the time variation functions. For instance, $T_\Gamma^t(t) = 10 * t * (t <= 1)$ would generate a linearly increasing function for $t \leq 1$, followed by a drop to zero and silence afterwards.

Definition using data files. In transient heat conduction problems, it is frequent to use environmental measurements as input. In such cases, the time variation functions are defined by time-temperature pairs, and not by analytic expressions. The time-temperature pairs must be listed in a text file, which is loaded using the GUI (see Section 5.4). The text file should have no header. The data should be organized in two columns, the first containing the time, and the second the corresponding values of the time variation functions. An example of a data file is presented in Figure 14.

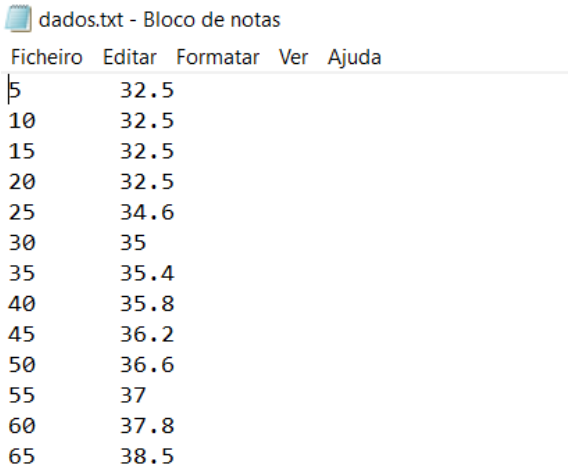


Figure 14. Example of an input data file

The definition of time-temperature pairs is independent of the choice of the time step**. FreeHyTE – Transient heat interpolates linearly between the values specified in the text file** to obtain the time variation function at the current time.

4.6.3. Definition of the space variation

The definition of the space variation of the boundary temperature $T_\Gamma^s(s)$ or normal heat flux $q_\Gamma^s(s)$ (see expressions (7) and (8) in Section 4.6.1) 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 2) and the last point must correspond to the final point of the side.

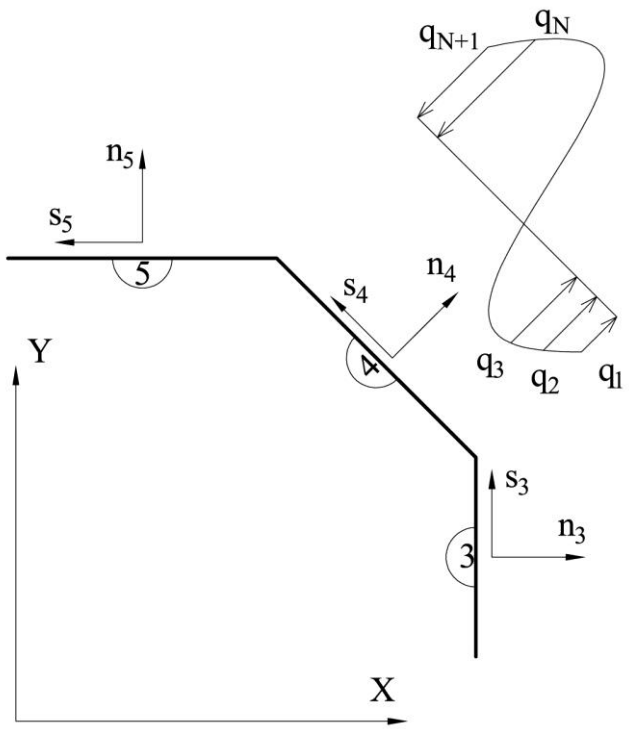


Figure 15. Normal polynomial flux applied to side 4

Consider, for example, that on edge 4 of the structure presented in Figure 2 a heat flux is applied in the normal direction, as shown in Figure 15. The flux is defined by a polynomial

of degree N. In **FreeHyTE – Transient heat**, such flux is input as a list containing its values in $N+1$ equally spaced points on side 4, including the first and the last, for instance,

$$q_{\Gamma}^{s}(s_4) \equiv \begin{pmatrix} q_1 & q_2 & q_3 & \cdots & q_N & q_{N+1} \end{pmatrix}$$

The values are given in the order prescribed by the orientation of axis s_4, and the signs are in accordance with the orientation of the normal axis, n_4 (meaning, for instance, that q_1 to q_3 are positive and q_N and q_{N+1} are negative).

If the boundary condition is constant, it is, of course, sufficient to specify its value in a single point.

4.6.4. Boundary conditions for the rectangular plate and I-beam

Following the recipes given in Section 4.6.1, the boundary conditions for the structures defined in Sections 4.3.1 (Figure 3) and 4.3.2 (Figure 6) are listed in Table 2 and Table *3*.

Location	Boundary type	Space variation (or *h*, for Robin)	Time variation
Left & right boundaries	Neumann	0	1
Bottom boundary	Robin	$h=15$	from file (Figure 5)
Top boundary	Robin	$h=15$	32.5

Table 2. Boundary conditions for the rectangular plate

Location	Boundary type	Space variation (or *h*, for Robin)	Time variation
Web boundaries	Neumann	0	1
Interior flange sides	Neumann	0	1
Bottom flange, exterior and lateral sides	Robin	$h=15$	from file (Figure 5)
Top flange, exterior and lateral sides	Robin	$h=15$	32.5

Table 3. Boundary conditions for the I-beam

5. GRAPHICAL USER INTERFACE

5.1. INTRODUCTION

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

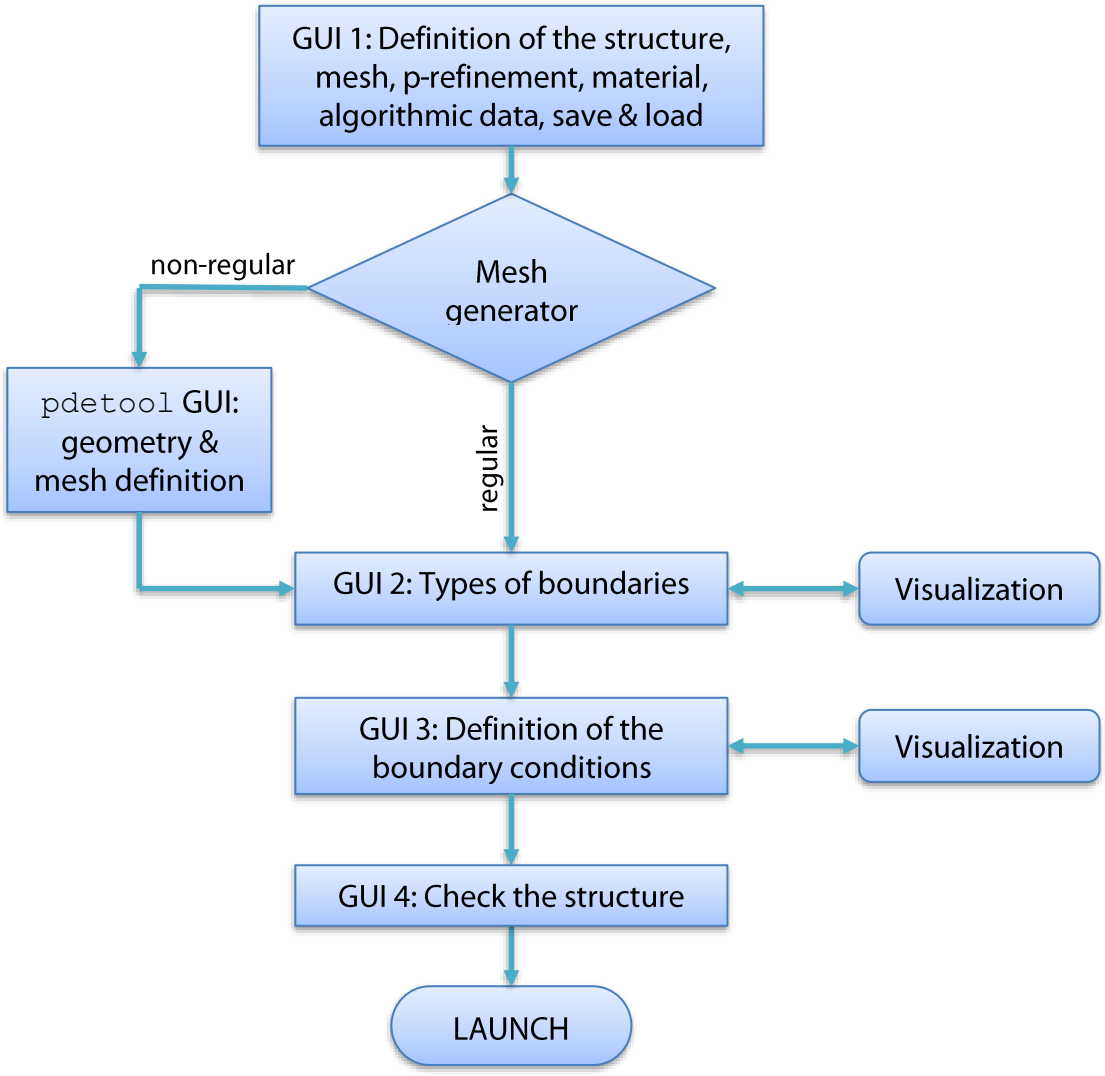


Figure 16. GUI flowchart

The general structure of the GUI is presented in Figure 16, in a flowchart format. It consists of four main GUI, complemented by the `pdetool` GUI for the definition of

non-regular bodies 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 – Transient heat** writes the data acquired from the ended interface to a **.mat* file. Consequently,

- you must run **FreeHyTE – Transient heat** from a folder where you have writing rights; and,
- when **FreeHyTE – Transient heat** is ran from the source (*.m) files, it is possible to start its execution from *any point* of the flowchart in Figure 16, provided you want to reuse all data from the interfaces you skipped.

The latter capability is not available when you run **FreeHyTE – Transient heat** 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 – Transient heat** is initiated is presented in Figure 17. The main data zones of the interface are identified with red frames. Each of these zones is briefly described below.

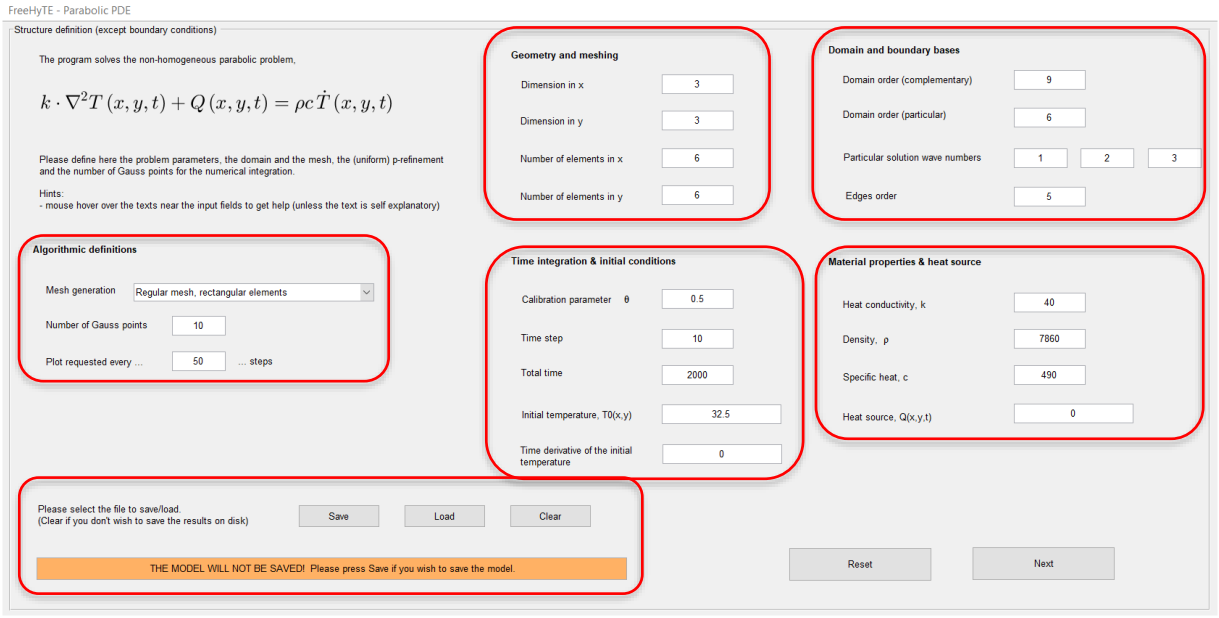


Figure 17. Layout of GUI 1

5.2.1. General features of the interface

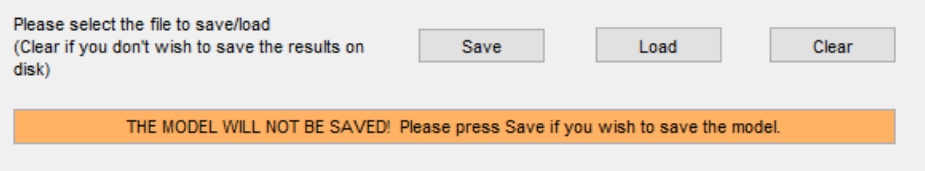
- GUI 1 is **pre-filled with the data from the last run** before it is made visible, if such data is available in the working folder. The Reset button deletes pre-filled data;
- hovering the mouse over the text accompanying the edit boxes opens **context help** explaining in more detail what input is 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

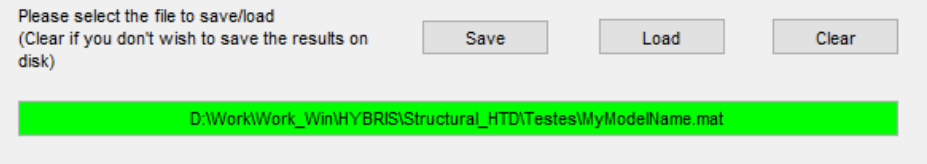
FreeHyTE – Transient heat offers **two levels of save & load**, namely an implicit level and an explicit level.

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

Explicit save & load. Only the data from the last run is automatically 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 – Transient heat** 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 choose to 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 series of files with the values of the temperatures and heat fluxes in the Gauss points of each element, at each time step. These files are stored in a folder named *MyModelName*, and are named *TS_i.dat*, where *i* is the time step counter.

The output files are formatted for direct loading in the post-processing software Tecplot, but can be used in other visualization software as well since they simply contain a list of points and temperature/flux values. **The results are not saved in an output file if you choose not to save your model, but colour maps of temperatures and heat fluxes are plotted at the required time steps** (see Section 5.2.3) regardless of the save option.

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 again.

The mesh data is saved with both explicit and implicit processes, **enabling the reuse of the same mesh in subsequent runs**. **However, the saved 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. User must choose between the regular and non-regular geometry and mesh generators using the *Mesh generation* popup menu. Additionally, user must specify the number of Gauss-Legendre quadrature points to perform the integrations needed to compute the conductivity matrix. 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. Finally, user can request the solutions to be plotted at a certain number of time steps. Since Matlab plotting function is notoriously slow, plotting of the solution should be as rare as possible. If you wish to use the plots from all time steps to make an animation of the solution, for instance, it is recommended that you use a dedicated post-processing software on the data files produced by **FreeHyTE – Transient heat** (see Section 5.2.2).

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 is 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 bases. Here, user should insert the (uniform) orders of p-refinement in the domain of the elements and on their essential boundaries. Hybrid-Trefftz finite elements implemented in **FreeHyTE – Transient heat** approximate independently the particular and complementary solutions of the temperature field in the domain of the elements and the normal heat flux fields on their essential (Dirichlet, Robin, and interior) boundaries. The orders of these approximations should be defined here. For some guidance on the choice of the p-refinement orders, please consult Section 4.5.2 of this manual. It is recommended that the particular solution wave numbers be defined as 1, 2, and 3, unless the user is familiar with the Dual Reciprocity Method used in **FreeHyTE – Transient heat** to approximate the particular solution.

Material properties & heat source. Specify the values of the thermal conductivity, density and specific heat of the material and the internal heat source in the body. The heat source must be defined by an analytic expression in variables *x*, *y* and *t*. The use of basic operations (+, -, *, /) is recommended, but logical operations and most of the built-in Matlab functions should also be acceptable as input.

Time integration & initial conditions. Specify the parameters to be used for the time discretization, and the initial conditions of the problem. The time discretization is the process through which the original problem, in time and space, is reduced to a series of problems in space variables only. In **FreeHyTE – Transient heat**, the time discretization is handled using a generalized mid-point (finite difference) algorithm. The total duration of the problem is split into an arbitrary number of time steps and the governing equations are enforced at the end of each time step. A calibration parameter (denoted by θ in the GUI) is used to control the numerical damping and stability of the algorithm.

A variety of implicit time integration schemes are recovered by setting different values for parameter θ, including the Crank-Nicolson scheme $\left(\theta = 1/2\right)$, Galerkin scheme $\left(\theta = 2/3\right)$ and backward Euler scheme $\left(\theta = 1\right)$. Values inferior to $1/2$ are not recommended for the calibration parameter θ, as they may cause numerical instability in the algorithm if the time step is not carefully controlled (i.e. the algorithm becomes conditionally stable).

It is not straightforward to guide the user into the choice of the 'best' value of θ for a specific problem. As a rule, the larger the value of θ, the larger the numerical damping of the solution. This means that using the backward Euler scheme would be expected to reduce the spurious vibrations of the solution, but would also damp its sudden variations. Conversely, the Crank-Nicolson scheme would tend to be more precise (especially for relatively small time steps), but is also more prone to spurious vibrations.

The duration of the time step also influences considerably the quality of the solution. While choosing small time steps is generally preferable for the stability of the time discretization scheme, the approximation basis of the particular solution is hindered by very small time steps. Therefore, in highly transient analyses, the time step size may need to be calibrated empirically.

Besides the total duration of the analysis, time step and calibration parameter, the user is expected to input the analytic expressions of the initial temperature field, $T(x,y,0)$, and the initial time derivative of the temperature field, $\overset{o}{T}(x,y,0)$. Although the latter value is not strictly required, since it can be automatically calculated by enforcing the balance equation (1) at $t = 0$, it is requested from the user, to avoid the inclusion of an additional Matlab toolbox in the pre-requisites of **FreeHyTE – Transient heat**. The initial fields must be defined by analytic expressions in variables *x* and *y*. The use of basic operations (+, -, *, /) is recommended, but logical operations and most of the Matlab built-in functions should also be acceptable as input. **FreeHyTE – Transient heat** does not perform any check on the consistency of the initial conditions, so inconsistent expressions can be input for the initial temperature and its time derivative.

5.2.4. GUI 1 data for the rectangular plate and I-beam problems

The GUI 1 data for the problems defined in Sections 4.3.1 (Figure 3) and 4.3.2 (Figure 6) are listed in Figure 18 and Figure 19, respectively. For the rectangular plate problem, the data refers to the finer mesh shown in Figure 8a.

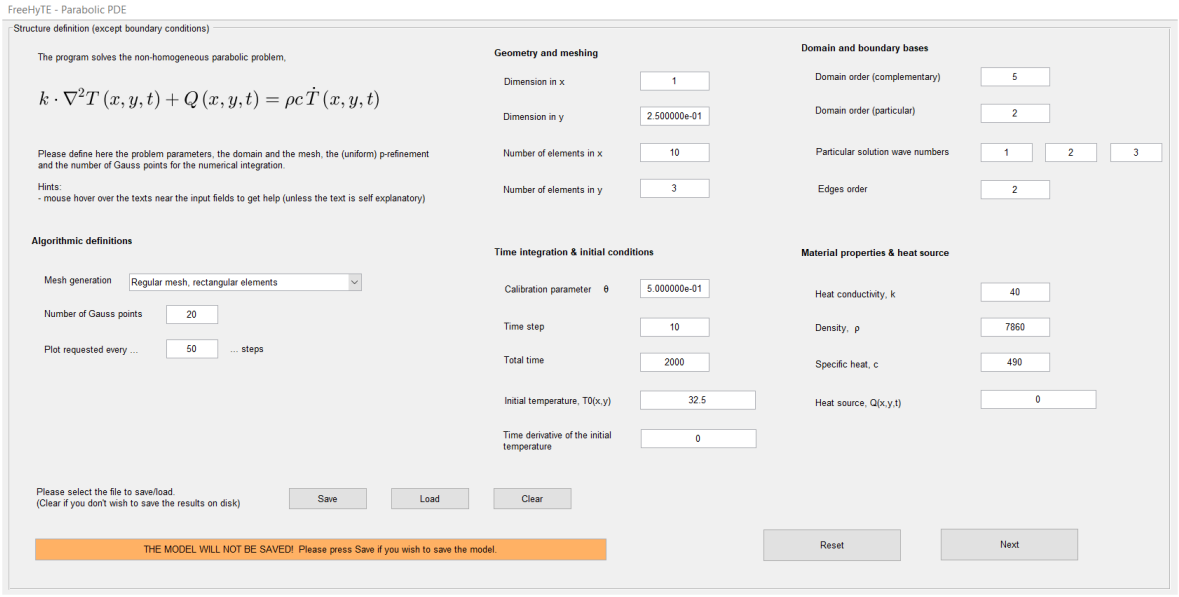


Figure 18. GUI 1 data for the rectangular plate (finer mesh)

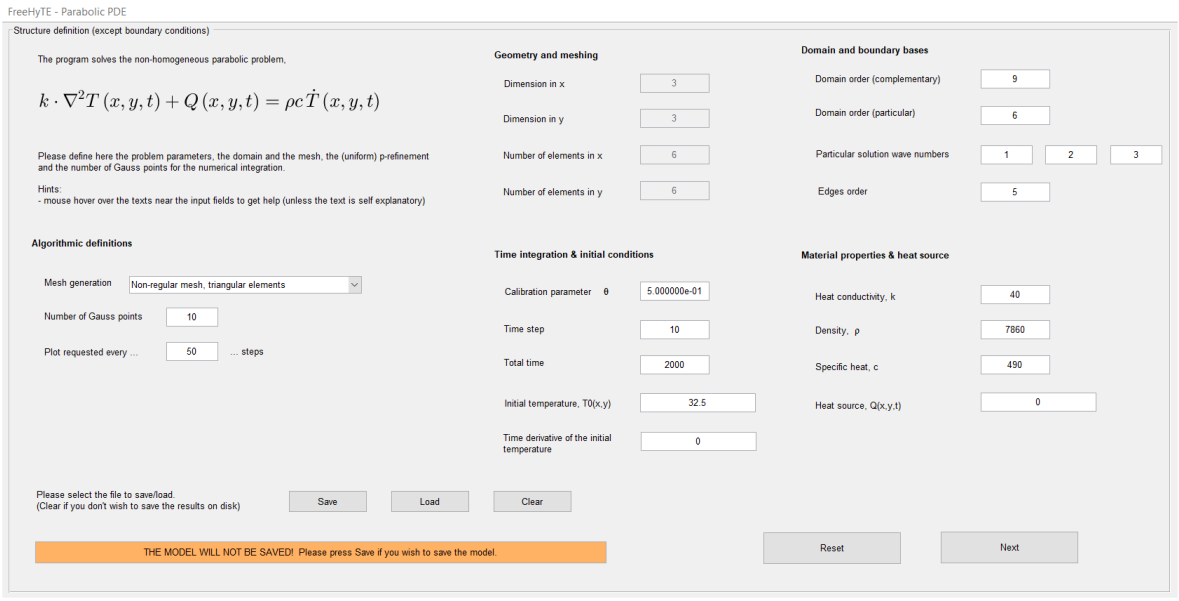


Figure 19. GUI 1 data for the I-beam

5.3. GUI 2: DEFINITION OF THE BOUNDARY TYPES

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

The configuration of GUI 2 for the finer mesh of the rectangular plate problem is presented in Figure 20. The main data zones of the interface are identified with red frames. Each of these zones is briefly described below.

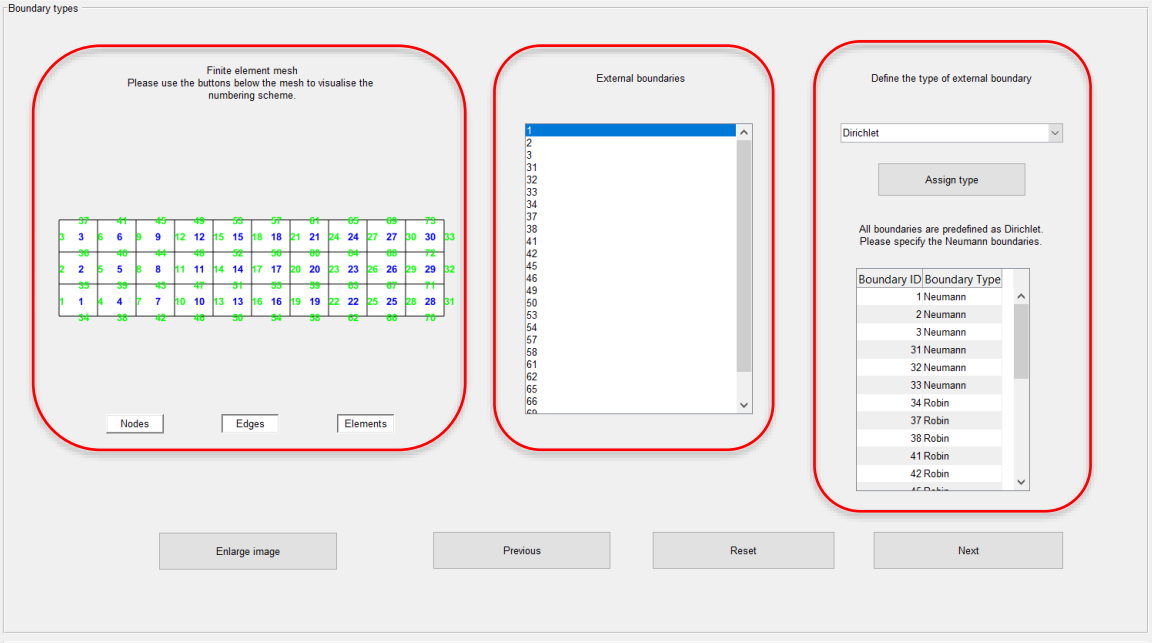


Figure 20. Layout of GUI 2

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 20, 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 21).

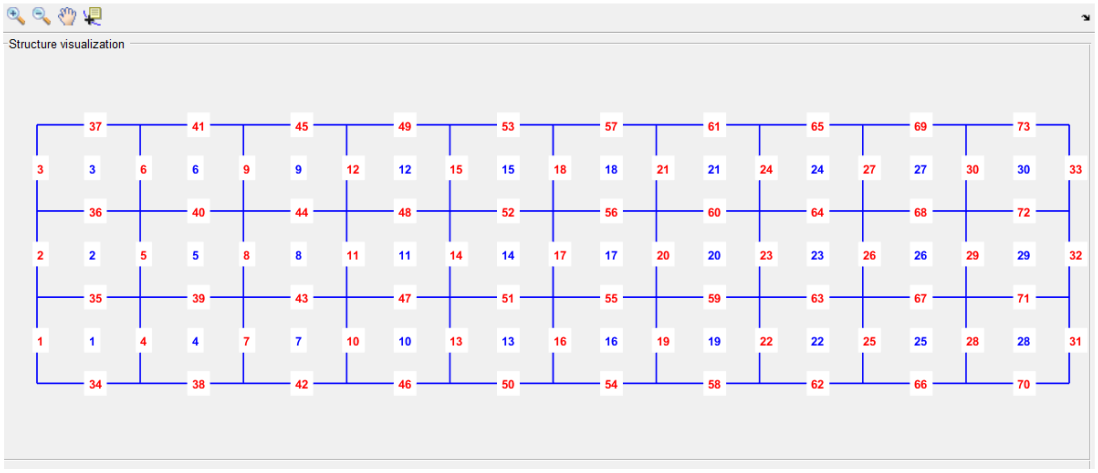


Figure 21. Layout of the visualization interface

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 (Dirichlet, Neumann, or Robin) 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 – Transient heat** 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 rectangular plate problem is presented in Figure 22. The main data zones of the interface are identified with red frames. Each of these zones is briefly described below.

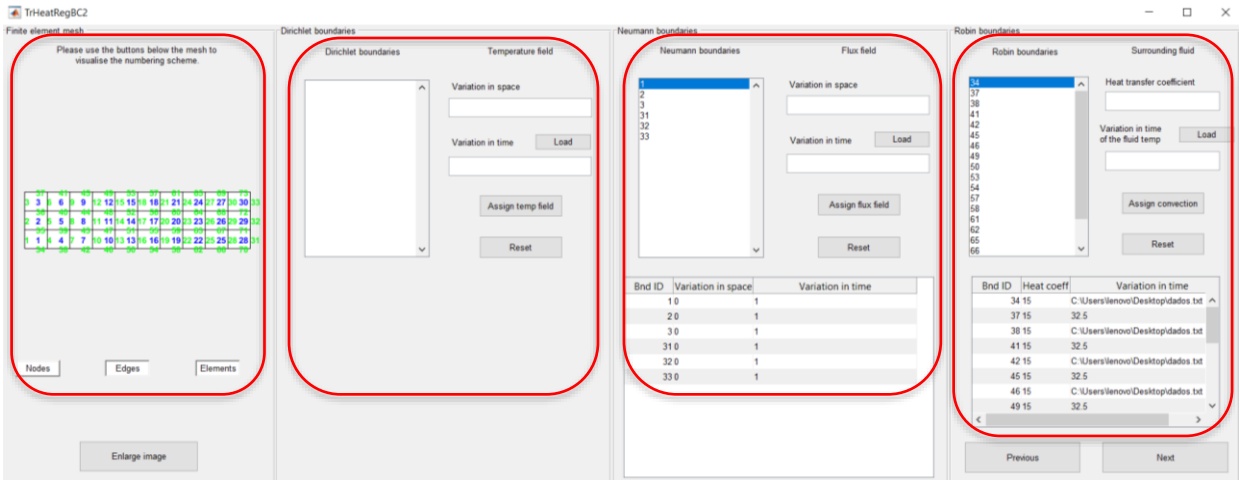


Figure 22. Layout of GUI 3 for the rectangular plate problem

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

The other three input zones are used to define the Dirichlet, Neumann and Robin 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*.

For each boundary type, the exterior boundaries of that type (according to the specifications input in GUI 2) are listed in the *Dirichlet/Neumann/Robin boundaries* tables.

In these tables, the user must select the boundaries where he/she wishes to specify the enforced temperature/heat flux/convection (multiple selection is possible).

As explained in Section 4.6, Dirichlet and Neumann boundary conditions may vary in both time and space. Robin boundary conditions may vary only in time. The variations in time and space of the boundary conditions are defined as described in Sections 4.6.2 and 4.6.3, respectively.

The analytic expression of the time variation is inserted in the *Variation in time* field. It must be defined by an analytic expression in variable *t*. The use of basic operations (+, -, *, /) is recommended, but logical operations and most of the built-in Matlab functions should also be acceptable as input. Alternatively, use the Load button next to the *Variation in time* field to load the time variation from a file.

The variation in space of the applied temperature or heat flux field is specified in the *Variation in space* field, according to the procedure detailed in Section 4.6.3. **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). In the case of Robin boundaries, the heat transfer coefficient must be input in the respective field.

Press the Assign temp/flux/convection buttons when you are done. The values in the tables at the bottom of each region should update automatically.

For completeness, the configuration of GUI 3 for the I-beam problem is also presented in Figure 23.

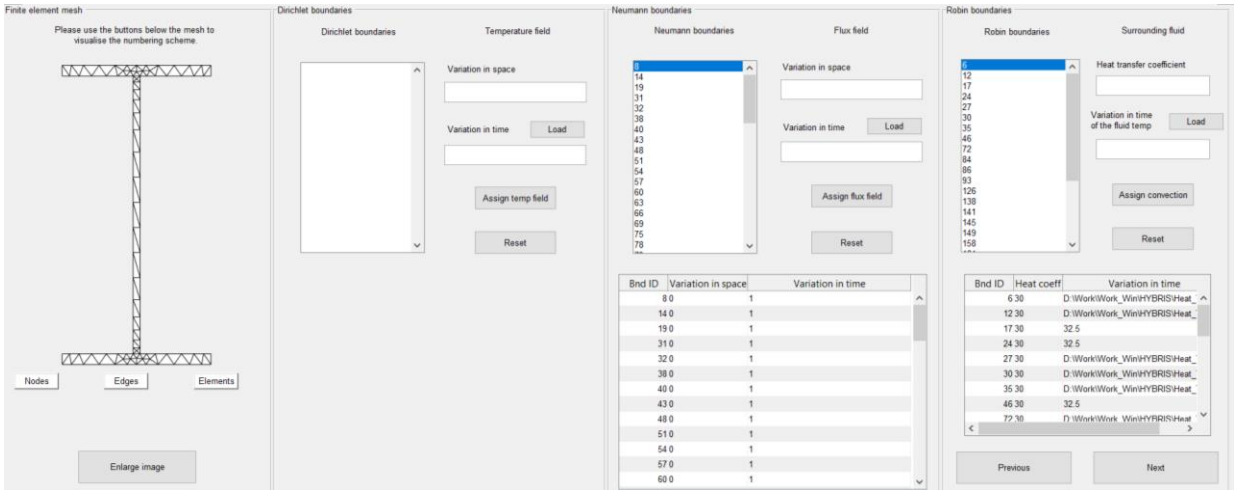


Figure 23. Layout of GUI 3 for the I-beam problem

After completing the definition of the boundary conditions, press the Next button to move to the next 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 – Transient heat** calculation module. It is meant to allow 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 and boundary conditions.

The interface is launched with the mesh numbering visualization on, as presented in Figure 24 for the finer mesh of the rectangular plate problem.

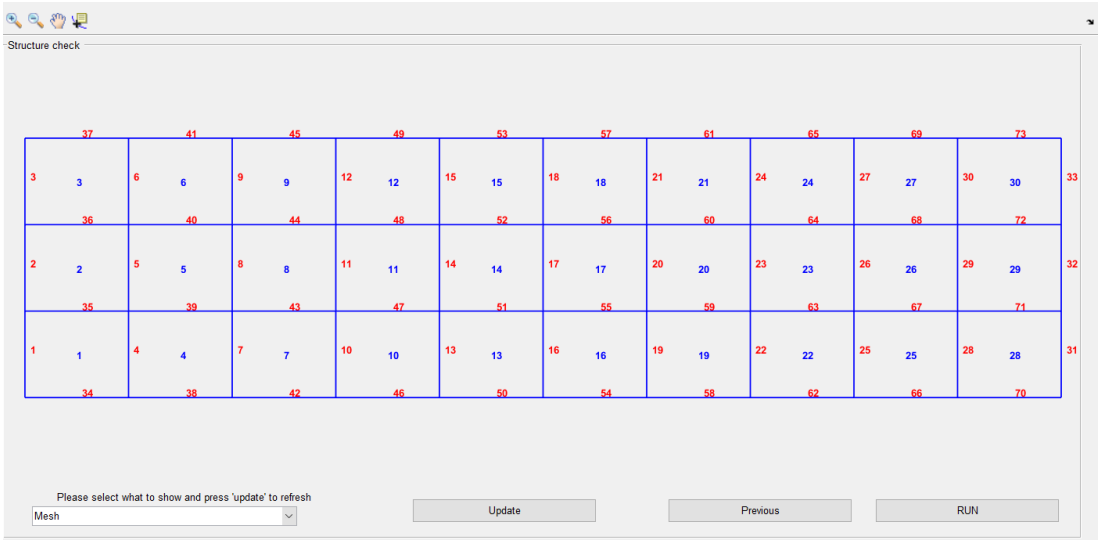


Figure 24. Visualization GUI in the mesh numbering mode

Selecting the *Boundary conditions* option in the pop-up menu and then pressing the Update button triggers the boundary conditions visualization mode, where the enforced boundary values are plotted at the beginning and the end of each exterior side. **Dirichlet sides** are plotted in **black**, **Neumann sides** are plotted in **red**, and **Robin sides** are plotted in **yellow**. The visualization interface for the boundary conditions of the mesh presented in Figure 24 is shown in Figure 25. A combined view of the mesh numbering and boundary conditions is also available.

Press the RUN button to launch the analysis of the model.

Figure 25. Visualization GUI in the boundary conditions mode

5.6. POST-PROCESSING

Post-processing in **FreeHyTE – Transient heat** is limited to the writing of the output data files (if the user choses to save the model, see Section 5.2.2), and the plotting of the analysis results at the required time steps (see *Algorithmic definitions* paragraph in Section 5.2.3). The temperature and heat flux plots obtained for the rectangular plate and I-beam presented in Sections 4.3.1 and 4.3.2 are given in Sections 5.6.1 and 5.6.2.

5.6.1. The rectangular plate results

Two refinement schemes were used for the solution of the rectangular plate problem, as detailed in Sections 4.4.1 and 4.5.2. The first scheme involved a 30-element mesh and refinement orders of $n_\Gamma = 2$, $n_D^C = 5$, and $n_D^P = 2$ on the essential boundaries and in the finite element domains. The second scheme used three finite elements, but increased the refinement orders to $n_\Gamma = 6$, $n_D^C = 16$, and $n_D^P = 6$ to compensate for the coarseness of the mesh.

The temperature and heat flux predictions after 1000s of exposure to the fire described in Figure 5 are presented in Figure 26 and Figure *27*, respectively. The plots are pasted directly from the **FreeHyTE – Transient heat** post-processor.

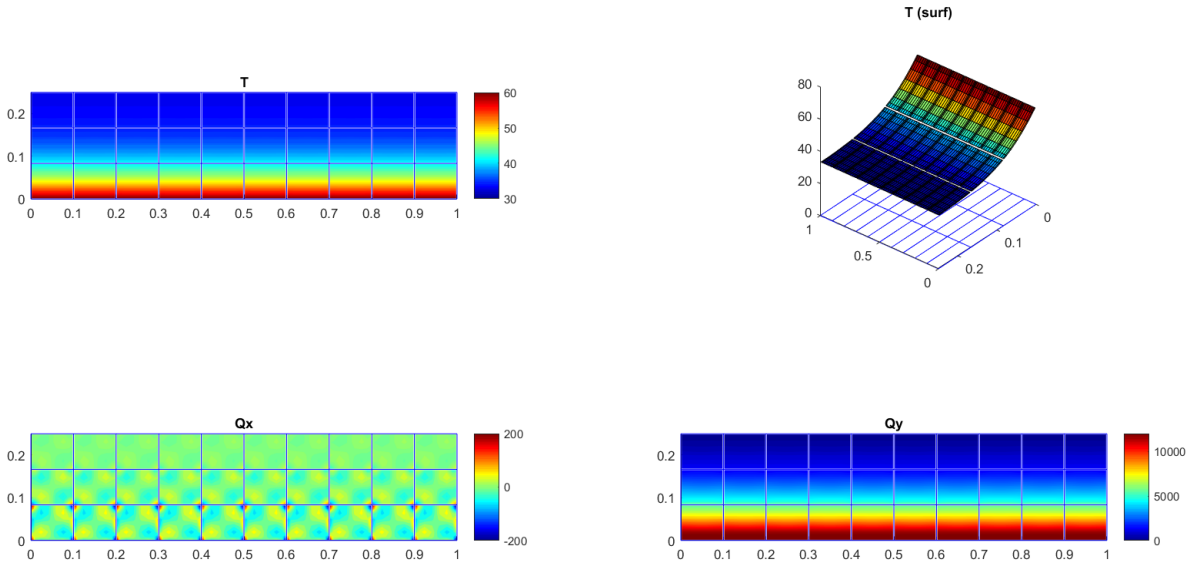


Figure 26. Temperature and heat flux plots for the rectangular plate problem, 30 finite elements, at t=1000s.

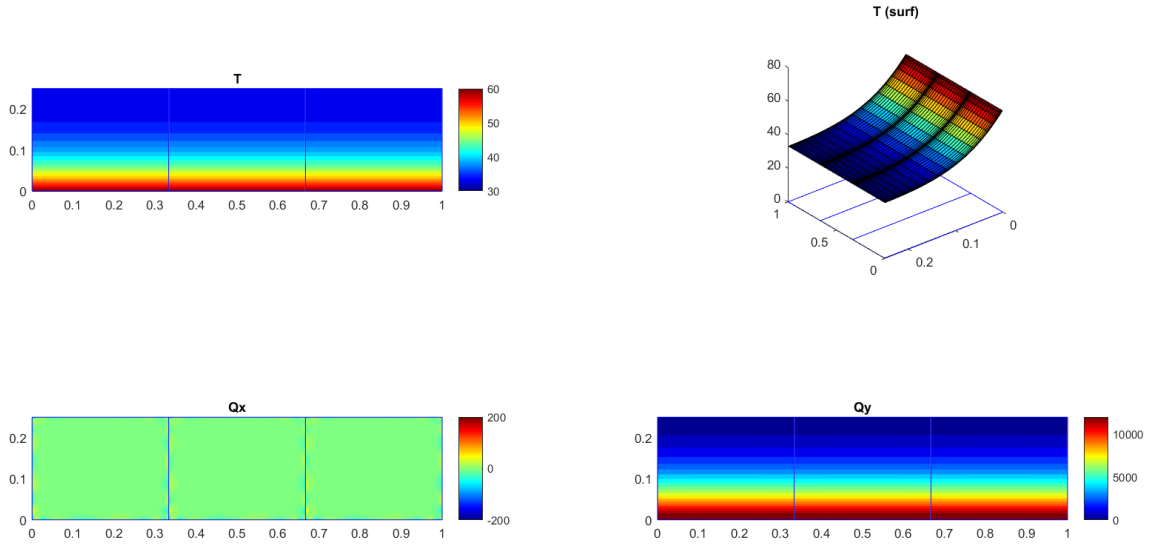


Figure 27. Temperature and heat flux plots for the rectangular plate problem, 3 finite elements, at t=1000s

It is clear that the results are very similar in both cases for both temperature and heat flux fields. Some spurious heat fluxes in the horizontal direction are obtained with the 30 element mesh (Figure 26) because of the weakness of the temperature compatibility enforcement on the interior boundaries (note that $n_\Gamma = 2$). However, the values of the horizontal heat flux are residual as compared to those obtained in the vertical direction. No flux averaging is used in the plots.

The total analysis times for the whole duration of the fire event (210 time steps covering 2100s, see Figure 5) were 39s for the finer mesh case and 14s for the coarser mesh case. No ill-conditioning of the solving system was reported in either case.

The results obtained with the two levels of refinement illustrate well the superior convergence of the Trefftz elements under basis refinement, as compared to mesh refinement. Indeed, the fine mesh model (with lower basis refinement) requires more time to compute the results, and these results are not better than those obtained with the coarser mesh, with higher basis refinement. However, the basis refinement should not be so high as to cause ill-conditioning of the solving system, as this may considerably hinder the results.

5.6.2. The I-beam results

The model used for the solution of the I-beam problem is detailed in Sections 4.4.2 and 4.5.2. The refinement scheme involved a 110-element mesh and refinement orders of $n_\Gamma = 2$, $n_D^C = 6$, and $n_D^P = 3$ on the essential boundaries and in the finite element domains.

The temperature and heat flux predictions after 1000s of exposure to the fire are plotted in Figure 28.

The temperature and heat flux continuity are well recovered in all points of the domain. The heat flux discontinuities occurring at the inner corners (meeting of the web and flanges) are sharply recovered. The enforced boundary conditions are well respected. The convective heat transmitted by the fire sharply increases the temperature of the bottom flange, particularly near its edges, where the upper heat flux is prevented by the insulation, see Figure 6(b). Conversely, the temperature is slightly lower near the centre of the flange, where a large vertical heat flux occurs. The temperature values drop swiftly through the web and stabilize close to the initial temperature at its upper part and throughout the top flange. The total analysis time was 157s, for all 210 time steps.

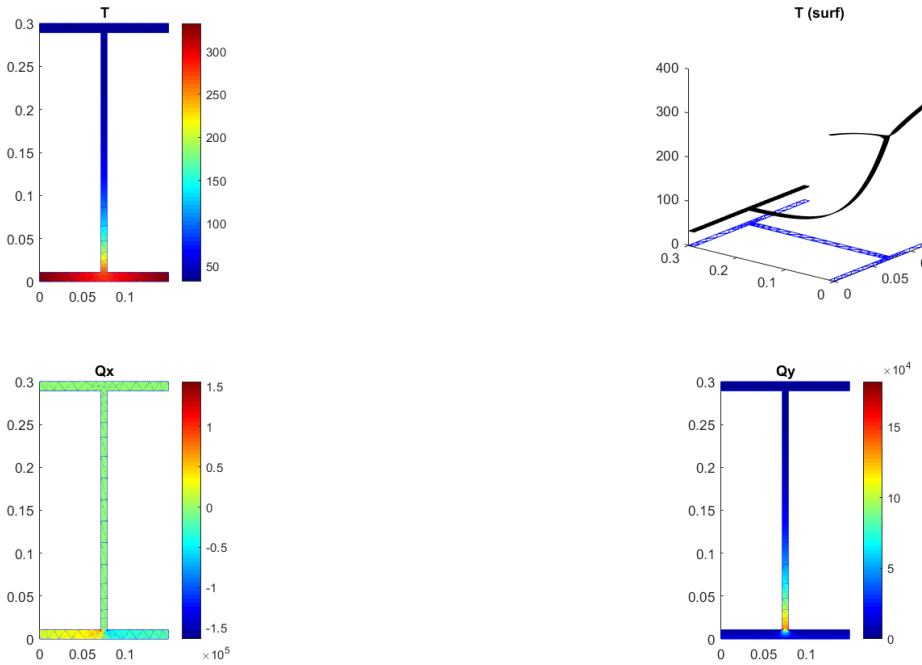


Figure 28. Temperature and heat fluxes for the I-beam, t= 1000s of exposure

6. ADVANCED STRUCTURAL DEFINITION

6.1. INTRODUCTION

While appropriate for the vast majority of applications, the standard operation of **FreeHyTE – Transient heat** does not provide the user with a way to choose different p-refinements for different mesh entities. However, while the Graphical User Interface (GUI) of **FreeHyTE – Transient heat** only allows uniform basis refinements to be defined, these definitions can be programmatically overwritten to specify different orders of refinement for different finite elements and essential boundaries. A simple procedure for this is described in Section 6.2.

Another limitation of the GUI-based **FreeHyTE – Transient heat** 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.

Please note, however, that **no contact resistance can be prescribed between layers of different materials**, meaning that temperature continuity is enforced on the interface boundaries. This limitation may be removed in future versions of **FreeHyTE – Transient heat**.

6.2. LOCALIZED P-REFINEMENT

The concept of localized p-refinement was introduced in Sections 3.1.4 and 4.5. The procedure may improve the finite element solutions in areas where localized effects like temperature or heat flux concentrations/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 – Transient heat**. 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 29);
- 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*, the input line should be something like,

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

  To set the order *ap* for the particular solution basis of element *b* and the order *ac* for the complementary solution basis of the same element, the input lines should be,

  ```
  Loops.orderP(b,:) = ap;

  Loops.orderC(b) = ac;
  ```

- 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.

```
%% EDGE REFINEMENT DATA
% Allocate the refinement order defined in the GUI to all Dirichlet
% boundaries ...
Edges.order(Edges.type=='D') = EdgesOrder;
Edges.order(Edges.type=='R') = EdgesOrder;
% ... or overwrite orders manually, if you need to have different orders
% for different essential boundaries
% Edges.order(:) = 4;      % <------- Examples
% Edges.order(3:4) = 0;    % <------- Examples
% Edges.order(8:3:11) = 0; % <------- Examples


%% ELEMENT DATA
% Allocate the refinement orders defined in the GUI to all elements
% 1. Particular solution basis
Loops.orderP(:,1) = LoopsOrderP; % order of the first Bessel basis
Loops.orderP(:,2) = LoopsOrderP; % order of the second Bessel basis
Loops.orderP(:,3) = LoopsOrderP; % order of the third Bessel basis
Loops.lambda(:,1) = lambda1; % lambda for the first Bessel functions
Loops.lambda(:,2) = lambda2; % lambda for the second Bessel functions
Loops.lambda(:,3) = lambda3; % lambda for the third Bessel functions
% 2. Complementary solution basis
Loops.orderC(:) = LoopsOrderC; % order of the complementary solution basis
% ... or overwrite orders manually, if you need to have different orders
% for different elements
% Loops.orderP(1,:) = 3;  % <------- Examples
% Loops.orderC(1) = 9;    % <------- Examples

% Computing the number of Gauss collocation points in each direction
Loops.gc(:) = ceil(sqrt(3*(2*LoopsOrderP+1)));
```

Figure 29. 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 4.5.2. 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 Dirichlet and interior boundaries**. Denoting by n_D^C 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**,

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

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

Since three wave numbers are used to construct the approximation basis for the particular solution, the order of the particular solution basis should be roughly two to three times

smaller than that of the complementary solution basis, that is, $2n_D^P \leq n_D^C \leq 3n_D^P$. It is noted that the wave numbers used for the particular solution computation in each element can also be modified by simply editing the '`% ELEMENT DATA`' area in Figure 29.

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 30);
- set the new material characteristics for the desired finite elements according to the examples indicated in the **.m* file. For instance, to set the conductivity *a* and specific heat *b* for the element *c*, the input lines should be something like,

```
Loops.material(c,1) = a; % '1' designates the conductivity

Loops.material(c,2) = b; % '2' designates the specific heat
```

- 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.

```
54
55  %% MATERIAL DATA
56  % Loads the material data and stores them in the Loop structure. The
57  % material data is assumed uniform for all elements. However, users may
58  % overwrite the data loaded from the GUI to define elements with distinct
59  % material properties. An example is given below.
60
61  % Loading the material data
62  load('TrHeatStructDef','k','c','rho');
63
64  % Allocate the material characteristics defined in the GUI to all elements
65  Loops.material(:,1) = k; %thermal condutivity
66  Loops.material(:,2) = c; %specific heat
67  Loops.material(:,3) = rho; %density
68  % Computing the thermal diffusivity based on the input data
69  Loops.material(:,4) = k/c/rho; %thermal diffusivity (alpha)
70  % ... or overwrite the material characteristics manually
71  % Loops.material(5,1) = 1;      <------- Examples
72  % Loops.material(5,2) = 2;      <------- Examples
73  % Loops.material(5,3) = 3;      <------- Examples
74  % Loops.material(5,4) = Loops.material(5,1)/Loops.material(5,2)/Loops.material(5,3)
75
76
77
78
```

Figure 30. Code areas to modify for defining different materials