FreeHyTE: theoretical bases and developer's manual

Ionuţ Dragoş Moldovan[1,]

CERIS, ICIST, Instituto Superior Técnico, Universidade de Lisboa, Lisbon, Portugal

Ildi Cismaşiu

Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa, Monte de Caparica, Portugal

---

Keywords:
Hybrid-Trefftz finite elements, Free-source software, User-friendly interface, Boundary value problem

---

1. Introduction

Hybrid-Trefftz finite elements approximate the solution of a boundary value problem using trial functions that satisfy *exactly* the homogeneous form of the governing differential equation in the domain of the element. An exact particular solution is also added to the trial basis, if available. The trial functions are subsequently combined such as to enforce the initial and boundary conditions, either on average or by collocation. However, as opposed to conforming finite elements (also labelled here as *conventional*), neither of the boundary conditions needs to be exactly satisfied.

Hybrid-Trefftz finite elements are designed to combine the best features of the boundary and finite elements, namely the boundary integral formulation that typifies the former and the sparse, Hermitian solving system featured by the latter. Moreover, all approximation bases are regular, as fundamental solutions of the governing equations need not be included in the trial bases.

The first applications of the Trefftz concept in the context of the finite element method date back to 1973 [1], followed by significant contributions

Email addresses: `dragos.moldovan@tecnico.ulisboa.pt` (Ionuţ Dragoş Moldovan), `ildi@fct.unl.pt` (Ildi Cismaşiu)

[1]Corresponding author. Ph. +351-218418249

by Jirousek [2], Herrera [3], Piltner [4], Qin [5] and Freitas [6], among many other authors. Such research efforts have led to the development of variants of hybrid-Trefftz finite elements for a wide breadth of physical problems, ranging from heat conduction [7] to structural elasticity [2], and from plate bending [8] to poroelasticity [9]. A comprehensive review of some key contributions is given in a recent book on Trefftz finite elements [10].

Besides endorsing boundary (rather than domain) integral formulations, hybrid-Trefftz finite elements feature physically meaningful approximation bases, tailored specifically for the problem under analysis. This enables highly accurate solutions to be obtained with very coarse meshes with relatively few degrees of freedom and renders these elements virtually insensitive to issues known to hinder the behaviour of conventional finite elements, as for instance gross mesh distortions, nearly-incompressible materials, high solution gradients and high frequency content in transient problems [11]. However, despite the advantages they provide over the conventional variants of the finite element method, public and user-friendly software featuring hybrid-Trefftz finite elements is still mostly unavailable. A notable contribution in this field is due to Qin, who includes some excellent C and MATLAB routines in his 2009 monograph on the Trefftz finite elements [10], but the collection of functions still falls short of providing the level of user-friendliness required by an inexperienced analyst. The scarcity of software featuring hybrid-Trefftz elements is somewhat surprising, as it places them in stark contrast with the boundary elements, for which a wide range of publicly available software exists (e.g. [12]), although user-friendliness is not the forte of most of it.

The hybrid-Trefftz finite element platform FreeHyTE is introduced in this paper. The investment into its development stems from the gradual realization that vast amounts of code developed in higher education are simply lost when the main developer moves out, rendering most of the subjacent research virtually unreproducible; and, that a change in the research paradigm is required to preserve and disseminate the results of our work. FreeHyTE is designed to be straightforward to use, even by analysts unacquainted with Trefftz elements, and amenable to expansion by researchers willing to test their new ideas without having to code common procedures from scratch.

To support the users, FreeHyTE features a sequence of Graphical User Interfaces (GUIs), regular and non-regular automatic mesh generators and adaptive p-refinement procedures. Moreover, its distribution is free under the terms of the GNU General Public License [13] and supported by a user's and developer's manuals to quickly get new users started [14].

To support developers, FreeHyTE approaches Trefftz finite elements through a unified perspective, applicable to hyperbolic, parabolic and elliptic boundary value problems alike. It features standardized procedures in all phases of the algorithm, including unified data structures, system assembling, preconditioning and solving, plotting of the results and debriefing reports. Moreover, the modular structure of FreeHyTE endorses the inclusion of existing procedures into newly developed ones with minimal coding effort. The vision driving the development of FreeHyTE is to enable Trefftz finite element researchers to dedicate more time to new ideas rather than to new code. One should only need to code the *specific* aspects of his/her formulation, and simply reuse pre-programmed procedures for everything else. The reproducibility (and visibility) of new scientific developments should also be fostered by their inclusion in the FreeHyTE framework, since their implementation can take advantage of the pre-programmed GUIs with minimal effort, putting them at the fingertips of other researchers and analysts worldwide.

FreeHyTE is very much work in progress (and we hope it remains so for a long time). New modules are constantly added to its distribution and new versions of the existing modules are constantly released. Therefore, instead of describing the FreeHyTE modules released to date, this paper focuses on the unifying theoretical perspective that guides its development and on the pre-programmed GUIs, data structures and procedures that can be straightforwardly adapted to new extensions.

The document is organized in eight sections. This introduction is followed by a brief comparison between conforming and hybrid-Trefftz finite elements, to lay the foundation of the subsequent presentation. The theoretical grounds of the hybrid-Trefftz finite elements implemented in FreeHyTE are described in Section 3. The following sections are completely devoted to the presentation of the computational implementation of these elements. An overview of the structure of the FreeHyTE modules is given in Section 4. Following this structure, the pre-processing, processing and post-processing phases of FreeHyTE are presented in Sections 5, 6 and 7, respectively. All essential procedures implemented to date in FreeHyTE are presented, to support their combination for the creation of new modules for new applications. However, it should not be assumed that all procedures described here are (or should be) applied to all modules of FreeHyTE. They should rather be regarded as pre-programmed construction blocks that can be readily combined to support the effort of future developers. The last section of the paper presents some conclusions.

2. Conventional vs. hybrid-Trefftz finite elements

A general comparison between conventional (conforming) and hybrid-Trefftz finite elements is given here. To preserve generality, the main physical quantity of the problem is referred to as the *state field*, while its (generalized) gradient is coined as the *flux field*. Depending on the specific application, the state field could represent the acoustic pressure (or the velocity potential) in acoustic problems, the temperature in heat conduction problems or the displacement field in structural applications. Likewise, the flux field could represent the particle velocity field, the heat flux or the stress field, for the same applications, respectively.

The comparison focuses the way conventional and hybrid-Trefftz formulations enforce the domain and boundary equations, the features of the approximation functions, the nature of the solving system, and the handling of the mesh and basis refinements. While the features described next refer mainly to the hybrid-Trefftz formulations implemented in FreeHyTE, most of them can be immediately adapted to other types of hybrid-Trefftz elements, especially to those involving compatible state field frames [2], which are quite widespread nowadays (e.g. [15, 16]).

2.1. Enforcement of the governing equations

Conventional finite elements. The conventional elements implemented in the vast majority of commercial codes are strictly compatible. Essentially, this means that they satisfy exactly the Dirichlet boundary conditions on the exterior Dirichlet and interior (inter-element) boundaries. The domain balance equation is enforced weakly and the Neumann boundary conditions are typically enforced at the nodes of the mesh. As a consequence, the state field is typically approximated with much superior precision as compared to the flux field.

Hybrid-Trefftz finite elements. Hybrid-Trefftz elements satisfy strongly the (homogeneous form of the) domain equations. The Dirichlet and Neumann boundary conditions are enforced weakly on the exterior boundaries of the structure. Either the Dirichlet or the Neumann boundary conditions are also enforced weakly on the interior boundaries of the mesh. Since domain equations are enforced strongly, the predictions of the state and flux fields are much more balanced in terms of quality than in the case of conventional elements, although with some slight bias towards the field whose continuity is enforced on the interior boundaries.

2.2. Approximation functions

Conventional finite elements. Nodes are the pivotal concept in the definition of the approximation functions in conventional finite elements. The nodal values of the state field are the main unknowns (degrees of freedom) and all fields are completely determined by their values. The number and location of the nodes also determine the expressions of the (polynomial) approximation functions for the state field in the domain of the element. Consequently, the redefinition of the nodes calls for the recalculation of all approximation functions. In order to ensure the strong compliance with the Dirichlet 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 state field through the partition of unity property, meaning that the state field will converge to the exact solution as the elements grow smaller (i.e. the mesh gets more refined).

Hybrid-Trefftz finite elements. Hybrid-Trefftz elements approximate not only the state and flux fields in the domain of the element, but also one of the fields on the essential boundaries of the element. Both approximations abandon the concept of nodes altogether. The nodal values of the fields are no longer the main unknowns of the problem and the redefinition of the nodes does not call for any redefinition of the approximation functions. Since nodes lose their significance, the 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.

However, compatible boundary approximations as used by alternative hybrid-Trefftz element formulations [2, 15, 16] are a straightforward particularization of this general approach and require no modifications of the code whatsoever.

In the domain of the element, the approximation functions embody relevant physical information on the phenomenon they model. They are tailored for each problem being solved and account for most super-convergent features of the hybrid-Trefftz elements, as discussed in Reference [11]. However, they are generally more difficult to integrate than their conventional element counterparts, especially when their order is high. By default, the domain variables (i.e. the weights of the approximation bases) do not have any physical meaning, being generalized variables instead. If desired, however, the domain shape functions can easily be combined such that their weights are actual physical quantities using the technique presented in Reference [17].

The approximation functions are intrinsically able to recover exactly a constant state field through the inclusion of the explicit unit monomial in the domain approximation basis, meaning that the state field converges to the exact solution as the elements grow smaller (i.e. the mesh gets more refined).

2.3. Solving system

Conventional finite elements. The assemblage of the solving system is based on the enforcement of the nodal flux balance. As a consequence, the solving system is sparse and symmetric, but the nodal state variables are shared by all finite elements that converge in the respective node. Summation of stiffness[2] matrices of adjacent elements occurs on the main diagonal of the solving system. The system is always kinematically indetermined (meaning that all nodal state variables cannot be recovered using only the Dirichlet boundary condition), provided there is at least a single node with free state variables. The system is not singular provided the state field is enforced in at least one node and generally not ill-conditioned.

Hybrid-Trefftz finite elements. By default, the solving system of hybrid-Trefftz finite elements is assembled in its full (uncondensed) form, with no summation of stiffness matrices. The domain variables are thus element-specific and not shared between neighbouring elements. This trait enables localized refinements of the approximation bases, which are instrumental for adaptive p-refinement procedures (see Section 6.3), but it also means that the systems are larger than those of the conventional elements for the same number of finite elements and the same order of the domain bases. On the other hand, since very large Trefftz elements are affordable, the solving systems generally result smaller, in practice, than in conventional elements. If desired, condensation on the boundary variables can be performed directly from the full form of the system, to secure the coupling with conventional finite elements [2]. Sparseness and symmetry properties of the conventional systems are preserved.

[2]Where no generic (and widely accepted) denomination is available, the structural mechanics nomenclature is used.

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 boundaries, as discussed in Section 3.4.2. The system is not singular provided the state field is specified on at least a boundary, but can be more ill-conditioned than its conventional counterpart.

2.4. Improvement of the solutions

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

Hybrid-Trefftz 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 [11]. 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 matrices in the solving system enable distinct definitions for the orders of approximation on each finite element and essential boundary, meaning that localized p-refinement is affordable. For these reasons, hybrid-Trefftz bases are typically of higher order as compared to the conventional elements. 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. Theoretical grounds

The theoretical fundamentals of FreeHyTE are based on a unifying perspective over the formulation of hybrid-Trefftz finite elements for different kinds of physical problems. According to this perspective, elliptic, parabolic and hyperbolic (initial) boundary value problems can be casted in the same (spectral-like) form, either directly or after the discretization in time of the governing equations. If the problem is time-dependent, the spectral-like form holds whether the time variation is harmonic, periodic or transient. This approach is instrumental for the standardization of the procedures implemented in FreeHyTE, as it means that their adaptation from one application to another is straightforward.

The spectral problem can be casted in terms of the state field or the flux field. The alternative forms are well suited to the formulation of two (dual) models of the hybrid-Trefftz finite elements, namely the state model and the flux model. Each model endorses the recovery of the respective field with slightly more precision as compared to its dual, so choosing one model over the other depends on the problem that is being solved and on the preferences of the analyst.

The description of the spectral problem in terms of the state and flux fields is given next, followed by brief descriptions of the main solution lines built into FreeHyTE. The presentation takes a very general stance, without getting into the details of specific applications. Instead, reader is referred to other publications that deal with specific applications whenever available.

3.1. Description of the problem

The problem FreeHyTE focuses on is defined on an arbitrary domain V, bounded by the complementary Dirichlet and Neumann boundaries Γ_u and Γ_σ, where the state and flux fields, $\boldsymbol{u}(\boldsymbol{x})$ and $\boldsymbol{\sigma}(\boldsymbol{x})$ are prescribed, respectively.

The differential equation governing the problem can be casted in one of the following forms:

Statement 1. *Find the state field $\boldsymbol{u}(\boldsymbol{x})$ that satisfies the (Navier-type) differential equation,*

$$\boldsymbol{\mathcal{D}} \cdot [\boldsymbol{k}\boldsymbol{\mathcal{D}}\boldsymbol{u}(\boldsymbol{x})] + \omega^2 \boldsymbol{\rho}\boldsymbol{u}(\boldsymbol{x}) = \boldsymbol{f_0}(\boldsymbol{x}) \tag{1}$$

and the boundary conditions,

$$\boldsymbol{n}\left[\boldsymbol{k}\boldsymbol{\mathcal{D}}\boldsymbol{u}(\boldsymbol{x})\right]=\boldsymbol{n}\boldsymbol{\sigma}(\boldsymbol{x})=\boldsymbol{t_\Gamma}(\boldsymbol{x}),\quad \textit{on } \Gamma_\sigma, \textit{ and} \tag{2}$$

$$\boldsymbol{u}(\boldsymbol{x})=\boldsymbol{u_\Gamma}(\boldsymbol{x}),\quad \textit{on } \Gamma_u \tag{3}$$

where $\boldsymbol{\mathcal{D}}$ and $\boldsymbol{\mathcal{D}}\cdot$ are some generalized gradient and divergence operators, $\boldsymbol{k}$ and $\boldsymbol{\rho}$ are some physical parameters coined here using the structural mechanics terminology as the generalized material stiffness and mass matrices, ω is a (possibly complex) generalized frequency, $\boldsymbol{f_0}$ is a source term, $\boldsymbol{n}$ collects the components of the outward normal to the Neumann boundary Γ_σ, and $\boldsymbol{t_\Gamma}$ and $\boldsymbol{u_\Gamma}$ are the enforced fluxes and state fields on the boundaries of the domain.

Alternatively, the same problem can be formulated in terms of the flux field as,

Statement 2. *Find the flux field* $\boldsymbol{\sigma}(\boldsymbol{x})$ *that satisfies the (Beltrami-type) differential equation,*

$$\boldsymbol{\mathcal{D}}\left[\boldsymbol{\rho}^{-1}\,\boldsymbol{\mathcal{D}}\cdot\boldsymbol{\sigma}(\boldsymbol{x})\right]+\omega^2\,\boldsymbol{f}\,\boldsymbol{\sigma}(\boldsymbol{x})=\boldsymbol{\mathcal{D}}\,\boldsymbol{\rho}^{-1}\boldsymbol{f_0}(\boldsymbol{x}),\ \ \textit{in } V \tag{4}$$

and the boundary conditions (2) and (3).

where $\boldsymbol{f}=\boldsymbol{k}^{-1}$ is the generalized material flexibility matrix.

Statements of type 1 and 2 are able to encode problems of remarkably diverse physical nature, ranging from simple Laplace boundary value problems [18] to shock wave propagation in complex multi-phase materials [9]. The generalized frequency ω is null in time-independent problems, real in harmonic time-dependent problems, and purely imaginary or complex in transient applications (see Section 5.2). The source term $\boldsymbol{f_0}$ depends on the initial conditions of the current time step in transient applications and may be used to collect non-linear terms as reported in Reference [19] (the applications included in FreeHyTE are currently limited to linear). Accordingly, the boundary value problem defined by Statements 1 and 2 is homogeneous in linear and time-independent or harmonic applications without internal flux generation, or non-homogeneous otherwise.

3.2. Formulation of the hybrid-Trefftz state elements

Hybrid-Trefftz state elements solve the problem posed according to Statement 1. If the problem is non-homogeneous and a closed-form particular solution is not straightforward to find, FreeHyTE uses two protocols to construct an approximate particular solution. In the first, the particular and complementary solution approximations are coupled in the same basis and their weights determined in a single step. In the second, the two approximations are uncoupled. The particular solution is approximated first, using a novel Dual Reciprocity Method variant, and the complementary solution is derived in a subsequent step. The two variants are fully discussed in References [20] and [21], respectively. Here, their presentation is limited to the extent relevant to their implementation in FreeHyTE.

3.2.1. Approximation bases

Hybrid-Trefftz state elements are built on two independent approximations (hence the *hybrid* label), namely the state field in the domain (V^e) of the element and the flux field on its essential (i.e. Dirichlet and interior) boundary, (Γ_e^e).

The state field is approximated as,

$$\boldsymbol{u} = \boldsymbol{U_c}\boldsymbol{x_c} + \boldsymbol{u_p}, \text{ in } V^e \tag{5}$$

where basis $\boldsymbol{U_c}$ collects the complementary solution trial functions, vector $\boldsymbol{x_c}$ its unknown weights, and $\boldsymbol{u_p}$ is a particular solution of equation (1).

The trademark feature of the Trefftz elements is that basis $\boldsymbol{U_c}$ collects functions that belong to the solution space of the homogeneous form of equation (1),

$$\boldsymbol{\mathcal{D}} \cdot (\boldsymbol{k}\boldsymbol{\mathcal{D}}\boldsymbol{U_c}) + \omega^2\boldsymbol{\rho}\boldsymbol{U_c} = \boldsymbol{0} \tag{6}$$

The solutions of equation (6) are application-dependent and rich in physical information regarding the modelled phenomenon. This property is essential for the high convergence rates of the Trefftz elements [24] and for their robustness to gross mesh distortion, incompressibility of constituents and predominant wavelength content [11].

The particular solution $\boldsymbol{u_p}$ may have a closed-form expression for relatively simple source terms $\boldsymbol{f_0}$. This is the case, for instance, of elastostatic problems in structural mechanics if $\boldsymbol{f_0}$ is the (constant) own weight of the (sub-)structure [22], or of steady-state heat conduction problems with constant heat generation. For more complex problems, such as those arising from the time discretization of transient problems, the source term $\boldsymbol{f_0}$ does not have a simple expression and a closed-form particular solution is not readily derivable. Two main strategies are typically applied for the recovery of the particular solution $\boldsymbol{u_p}$ in such cases, namely direct approaches, based on the evaluation of the particular solution using Green's functions, and indirect approaches, where the particular solution is approximated using additional trial functions. For a comprehensive review of these strategies, reader is referred to Reference [23]. The *indirect* approach is adopted in FreeHyTE, meaning that the particular solution is approximated using a separate basis $\boldsymbol{U_p}$, to which correspond the generalized state variables $\boldsymbol{x_p}$,

$$\boldsymbol{u} = \boldsymbol{U_c x_c} + \boldsymbol{U_p x_p}, \text{ in } V^e \tag{7}$$

The restraints applied to the particular solution basis $\boldsymbol{U_p}$ depend on the protocol used to determine its weights and are detailed in Sections 3.2.2 and 3.2.3.

An independent flux approximation is made on the essential boundary of the element, reading,

$$\boldsymbol{n\sigma} = \boldsymbol{Zy}, \text{ on } \Gamma_e^e \tag{8}$$

No restraints (besides completeness and linear independence) are enforced on the flux basis $\boldsymbol{Z}$. Chebyshev bases of arbitrary orders are implemented in FreeHyTE, but any other polynomial bases may effortlessly be embedded.

3.2.2. Finite element equations - the coupled approach

Described at length in Reference [20], the coupled approach integrates Trefftz-compliant ($\boldsymbol{U_c}$) and particular solution ($\boldsymbol{U_p}$) trial functions in the same basis, without making any distinction on their roles in the solution recovery process. This option enables Trefftz-compliant functions to improve the effectiveness of the particular solution basis, meaning that accurate total solutions can be obtained with relatively poor particular solution approximations.

To simplify the resulting formulation, the particular solution basis is built using functions that satisfy exactly the static problem,

$$\boldsymbol{\mathcal{D}} \cdot (\boldsymbol{k\mathcal{D}U_p}) = \boldsymbol{0} \tag{9}$$

The domain finite element equations are obtained by enforcing weakly the domain equation (1), using the functions collected in bases $\boldsymbol{U_c}$ and $\boldsymbol{U_p}$ for weighting,

$$\int \boldsymbol{U_i^*} \left[\boldsymbol{\mathcal{D}} \cdot (\boldsymbol{k\mathcal{D}u}) + \omega^2 \boldsymbol{\rho u}\right] \, dV^e = \int \boldsymbol{U_i^*} \boldsymbol{f_0} \, dV^e \tag{10}$$

where $i = \{c, p\}$ and $\boldsymbol{U}_{\boldsymbol{i}}^*$ designates the complex conjugate of the basis $\boldsymbol{U}_{\boldsymbol{i}}$.

The first term of equation (10) is integrated by parts to force the emergence of boundary integral terms, where the Neumann boundary conditions (2) and the flux approximation (8) are inserted. Further mathematical manipulation yields,

$$\boldsymbol{D}_{\boldsymbol{ic}}\, \boldsymbol{x}_{\boldsymbol{c}} + \boldsymbol{D}_{\boldsymbol{ip}}\, \boldsymbol{x}_{\boldsymbol{p}} - \boldsymbol{B}_{\boldsymbol{i}}\, \boldsymbol{y} = \boldsymbol{x}_{\boldsymbol{\Gamma}_{\boldsymbol{i}}} - \boldsymbol{x}_{\boldsymbol{0}_{\boldsymbol{i}}} \tag{11}$$

where the following definitions are used,

$$\boldsymbol{D}_{\boldsymbol{ic}} = \boldsymbol{D}_{\boldsymbol{ci}}^* = \int \boldsymbol{U}_{\boldsymbol{i}}^*\, \boldsymbol{n}\, \left(\boldsymbol{k}\boldsymbol{\mathcal{D}}\boldsymbol{U}_{\boldsymbol{c}}\right)\, d\Gamma^e \tag{12}$$

$$\boldsymbol{D}_{\boldsymbol{pp}} = \int \boldsymbol{U}_{\boldsymbol{p}}^*\, \boldsymbol{n}\, \left(\boldsymbol{k}\boldsymbol{\mathcal{D}}\boldsymbol{U}_{\boldsymbol{p}}\right)\, d\Gamma^e - \omega^2 \int \boldsymbol{U}_{\boldsymbol{p}}^*\, \boldsymbol{\rho}\, \boldsymbol{U}_{\boldsymbol{p}}\, dV^e \tag{13}$$

$$\boldsymbol{B}_{\boldsymbol{i}} = \int \boldsymbol{U}_{\boldsymbol{i}}^*\, \boldsymbol{Z}\, d\Gamma_e^e \tag{14}$$

$$\boldsymbol{x}_{\boldsymbol{\Gamma}_{\boldsymbol{i}}} = \int \boldsymbol{U}_{\boldsymbol{i}}^*\, \boldsymbol{t}_{\boldsymbol{\Gamma}}\, d\Gamma_\sigma^e \tag{15}$$

$$\boldsymbol{x}_{\boldsymbol{0}_{\boldsymbol{i}}} = \int \boldsymbol{U}_{\boldsymbol{i}}^*\, \boldsymbol{f}_{\boldsymbol{0}}\, dV^e \tag{16}$$

and Γ^e and Γ_σ^e are the total and Neumann boundaries of the finite element, respectively.

Finally, the Dirichlet boundary condition (3) is enforced separately on each essential boundary of the element, using the functions listed in the flux basis $\boldsymbol{Z}$ for weighting.

$$\int \boldsymbol{Z}^*\, \left(\boldsymbol{u} - \boldsymbol{u}_{\boldsymbol{\Gamma}}\right)\, d\Gamma_e^e = \boldsymbol{0} \tag{17}$$

Substitution of the state field approximation (7) into equation (17) yields,

$$\boldsymbol{B}_{\boldsymbol{c}}^* \boldsymbol{x}_{\boldsymbol{c}} + \boldsymbol{B}_{\boldsymbol{p}}^* \boldsymbol{x}_{\boldsymbol{p}} = \boldsymbol{y}_{\boldsymbol{\Gamma}} \tag{18}$$

where,

$$\boldsymbol{y}_{\boldsymbol{\Gamma}} = \int \boldsymbol{Z}^*\, \boldsymbol{u}_{\boldsymbol{\Gamma}}\, d\Gamma_e^e \tag{19}$$

The solving system of the (coupled) hybrid-Trefftz state element collects the domain and boundary statements (11) and (18), to yield,

$$\left[\begin{array}{cc|c} \boldsymbol{D_{cc}} & \boldsymbol{D_{cp}} & -\boldsymbol{B_c} \\ \boldsymbol{D_{pc}} & \boldsymbol{D_{pp}} & -\boldsymbol{B_p} \\ \hline -\boldsymbol{B_c^*} & -\boldsymbol{B_p^*} & \boldsymbol{0} \end{array}\right] \left(\begin{array}{c} \boldsymbol{x_c} \\ \boldsymbol{x_p} \\ \hline \boldsymbol{y} \end{array}\right) = \left(\begin{array}{c} \boldsymbol{x_{\Gamma_c}} - \boldsymbol{x_{0_c}} \\ \boldsymbol{x_{\Gamma_p}} - \boldsymbol{x_{0_p}} \\ \hline -\boldsymbol{y_\Gamma} \end{array}\right) \tag{20}$$

The price to be paid for coupling the complementary and particular solution bases is having to cope with domain integrals in the definition of the dynamic matrix $\boldsymbol{D_{pp}}$ and source vectors $\boldsymbol{x_{0_i}}$. However, as opposed to the direct approaches, where the particular solution is calculated using Green's functions [23], the domain integrals present here do not involve fundamental, but regular solutions of the Navier equation (1).

The properties of system (20) are further explored in Section 3.4.1. Its solution yields a unique estimate for the domain state field, according to definition (7).

3.2.3. Finite element equations - the Dual Reciprocity approach

The Dual Reciprocity Method (DRM) is an indirect approach to the computation of the particular solution in boundary value problems. In the DRM, the particular solution is obtained enforcing that a linear combination of its trial functions recovers the source term at some collocation points. The method typically involves two (dependent) approximations: one for the source term $\boldsymbol{f_0}$ and another for the particular solution $\boldsymbol{u_p}$.

They are obtained sequentially, in two steps. In the first step, the source function $\boldsymbol{f_0}$ is approximated by collocating a trial basis $\boldsymbol{F_p}$ into some collocation points in the domain of the element, $\overline{\boldsymbol{x}} \in V^e$,

$$\boldsymbol{F_p}(\overline{\boldsymbol{x}})\, \boldsymbol{x_p} = \boldsymbol{f_0}(\overline{\boldsymbol{x}}) \tag{21}$$

In the second step, the particular solution trial functions $\boldsymbol{U_p}$ are obtained by analytically solving the non-homogeneous equation having each of the functions collected in basis $\boldsymbol{F_p}$ as non-homogeneous terms,

$$\boldsymbol{\mathcal{D}} \cdot [\boldsymbol{k}\, \boldsymbol{\mathcal{D}}\, \boldsymbol{U_p}(\boldsymbol{x})] + \omega^2 \boldsymbol{\rho}\, \boldsymbol{U_p}(\boldsymbol{x}) = \boldsymbol{F_p}(\boldsymbol{x}) \tag{22}$$

If equations (21) and (22) are simultaneously satisfied, then the particular solution $\boldsymbol{u_p} = \boldsymbol{U_p}\, \boldsymbol{x_p}$ satisfy precisely the Navier equation (1) in the collocation points,

$$\left\{\boldsymbol{\mathcal{D}} \cdot [\boldsymbol{k}\, \boldsymbol{\mathcal{D}}\, \boldsymbol{U_p}(\overline{\boldsymbol{x}})] + \omega^2 \boldsymbol{\rho}\, \boldsymbol{U_p}(\overline{\boldsymbol{x}})\right\} \cdot \boldsymbol{x_p} = \boldsymbol{f_0}(\overline{\boldsymbol{x}}) \tag{23}$$

This approach was introduced in 1982 [25] and consistently studied ever since [23]. It presents the advantage of completely avoiding the need for domain integration, but the expressions of the functions included in basis $\boldsymbol{U_p}$ typically result extremely complex and sometimes singular.

The technique used in FreeHyTE is fully described in Reference [21]. It differs from the mainstream DRM in that it uses the same trial functions (except for a scalar multiplier) for both source term and particular solution approximations, and all trial functions have simple analytic expressions.

The particular solution basis is built using functions that satisfy exactly the Helmholtz equation,

$$\boldsymbol{\mathcal{D}} \cdot (\boldsymbol{k}\boldsymbol{\mathcal{D}}\boldsymbol{U_p}) + \lambda^2 \boldsymbol{\rho} \boldsymbol{U_p} = \boldsymbol{0} \tag{24}$$

where λ is some arbitrary, generalized wave number. Constraint (24), enforced on the particular solution basis, is a Trefftz-type condition similar to (6), where the generalized frequency ω is replaced by the wave number λ. Therefore, no additional effort is required for the solution of equation (24). The choice $\lambda = 0$ is acceptable and recovers the particular solution basis used in the coupled approach (Section 3.2.2). The choice $\lambda = \omega$ is, however, not acceptable, because of the linear independence requirement. Finally, the choice of multiple wave numbers to generate the particular solution basis improves its convergence and numerical robustness. Three wave numbers are implemented in FreeHyTE, as further detailed in Section 6.1.

Substitution of definition (24) into equation (22) yields the corresponding source function basis,

$$\boldsymbol{F_p}(\boldsymbol{x}) = \left(\omega^2 - \lambda^2\right) \boldsymbol{\rho}\ \boldsymbol{U_p}(\boldsymbol{x}) \tag{25}$$

meaning that the source term and particular solution bases only differ by the constant multiplier $(\omega^2 - \lambda^2)\ \boldsymbol{\rho}$.

Equation (21) is collocated in the Gauss-Legendre or Gauss-Chebyshev quadrature points of each element, but other collocation meshes can be easily implemented if needed. The number of collocation points is automatically controlled to match the dimension of the particular solution basis as close as possible, while making sure that system (21) is not under-determined. Its solution defines the particular solution approximation for each finite element.

The complementary solution is subsequently obtained by enforcing weakly the Navier equation (1) in the domain of the element, using the functions collected in the complementary solution basis $\boldsymbol{U_c}$ for weighting,

$$\int \boldsymbol{U_c^*} \left[\boldsymbol{\mathcal{D}} \cdot (\boldsymbol{k \mathcal{D} u}) + \omega^2 \boldsymbol{\rho u} \right] dV^e = \int \boldsymbol{U_c^*} \boldsymbol{f_0} \, dV^e \tag{26}$$

The same type of manipulations as described in Section 3.2.2 are performed on equation (26). As opposed to the coupled approach, however, substitution of property (23) into the resulting expressions cancel out all domain integrals, to yield,

$$\boldsymbol{D_c x_c} - \boldsymbol{B_c y} = \boldsymbol{x_\Gamma} - \boldsymbol{D_p x_p} \tag{27}$$

where the following definitions hold,

$$\boldsymbol{D_c} = \int \boldsymbol{U_c^*} \boldsymbol{n} \left(\boldsymbol{k \mathcal{D} U_c} \right) d\Gamma^e \tag{28}$$

$$\boldsymbol{D_p} = \int \boldsymbol{U_c^*} \boldsymbol{n} \left(\boldsymbol{k \mathcal{D} U_p} \right) d\Gamma^e \tag{29}$$

$$\boldsymbol{x_\Gamma} = \int \boldsymbol{U_c^*} \boldsymbol{t_\Gamma} \, d\Gamma^e_\sigma \tag{30}$$

and $\boldsymbol{B_c}$ is given by expression (14), with $i \equiv c$.

The boundary state field compatibility equation is identical to equation (17) used in the coupled approach. It is restated here for reader's convenience,

$$\int \boldsymbol{Z}^* \left(\boldsymbol{u} - \boldsymbol{u_\Gamma} \right) d\Gamma^e_e = \boldsymbol{0}$$

Substitution of the state field approximation (7) into the above equation yields,

$$\boldsymbol{B_c^* x_c} = \boldsymbol{y_\Gamma} - \boldsymbol{B_p^* x_p} \tag{31}$$

where $\boldsymbol{y_\Gamma}$ and $\boldsymbol{B_p}$ are defined by expressions (19) and (14), the latter with $i \equiv p$.

The solving system of the DRM-based hybrid-Trefftz state element is constructed on the domain and boundary statements (27) and (31),

$$\left[\begin{array}{c:c} \boldsymbol{D_c} & -\boldsymbol{B_c} \\ \hdashline -\boldsymbol{B_c^*} & \boldsymbol{0} \end{array} \right] \left(\begin{array}{c} \boldsymbol{x_c} \\ \hdashline \boldsymbol{y} \end{array} \right) = \left(\begin{array}{c} \boldsymbol{x_\Gamma} - \boldsymbol{D_p x_p} \\ \hdashline -\boldsymbol{y_\Gamma} + \boldsymbol{B_p^* x_p} \end{array} \right) \tag{32}$$

No domain integrals are present in system (32), which is a considerable advantage of this approach as compared to the coupled technique presented in Section 3.2.2. However, the complete separation of the complementary and particular solution bases mean that one can no longer compensate the other's weaknesses, which is a drawback.

The properties of system (32) are further explored in Section 3.4.1. Its solution yields a unique estimate for the domain state field, according to definition (7).

3.3. Formulation of the hybrid-Trefftz flux elements

Hybrid-Trefftz flux elements solve the problem posed according to Statement 2. For non-homogeneous problems with no closed-form particular solution, the same type of coupled and uncoupled solution protocols as described above are implemented. The coupled approach is discussed in detail in Reference [26], but some features relevant to the implementation in FreeHyTE are revisited next.

3.3.1. Approximation bases

Hybrid-Trefftz flux elements are built on the independent approximations of the flux field in the domain (V^e) of the element and the state field on its essential (i.e. Neumann and interior) boundary, (Γ_e^e).

The flux field is approximated as,

$$\boldsymbol{\sigma} = \boldsymbol{S_c} \boldsymbol{x_c} + \boldsymbol{\sigma_p}, \text{ in } V^e \tag{33}$$

where basis $\boldsymbol{S_c}$ is used to approximate the complementary solution, and $\boldsymbol{\sigma_p}$ is a particular solution of equation (4). As typical of Trefftz formulations, the approximation functions collected in basis $\boldsymbol{S_c}$ must satisfy the homogeneous form of the same equation.

The solution process of equation (4) is simplified by considering the auxiliary state field $\boldsymbol{U_c}$, constructed such as to satisfy the Navier equation (6). Indeed, it is immediate to show that any flux field defined as,

$$\boldsymbol{S_c} = \boldsymbol{f}^{-1} \boldsymbol{\mathcal{D}} \boldsymbol{U_c} \tag{34}$$

satisfies exactly the Beltrami equation (4). Consequently, the construction of the Trefftz-compliant flux basis $\boldsymbol{S_c}$ requires the solution of the same differential equation as the state field basis, which further simplifies the implementation of the models. The flux model also shares the convergence and robustness properties with the state model, since their (problem-dependent) approximation bases are both rich in physical information.

For problems where the source term $\boldsymbol{f_0}$ is simple enough for a closed-form particular state solution $\boldsymbol{u_p}$ to be found, a Beltrami-compliant particular flux solution can be found in the same way as above,

$$\boldsymbol{\sigma_p} = \boldsymbol{f}^{-1}\,\boldsymbol{\mathcal{D}}\boldsymbol{u_p} \tag{35}$$

Otherwise, the particular solution $\boldsymbol{\sigma_p}$ is approximated using a particular solution basis $\boldsymbol{S_p}$, to which correspond the generalized flux variables $\boldsymbol{x_p}$, to yield,

$$\boldsymbol{\sigma} = \boldsymbol{S_c}\boldsymbol{x_c} + \boldsymbol{S_p}\boldsymbol{x_p}, \text{ in } V^e \tag{36}$$

An independent state field approximation is enacted on the essential boundary of the elements, reading,

$$\boldsymbol{u} = \boldsymbol{Z}\boldsymbol{y}, \text{ on } \Gamma_e^e \tag{37}$$

The same (Chebyshev) polynomials that are used to approximate the boundary fluxes in the state model are included in the boundary state field approximation basis $\boldsymbol{Z}$, to preserve the consistency of their implementation in FreeHyTE.

3.3.2. Finite element equations - the coupled approach

Described at length in Reference [26], the coupled approach integrates Trefftz-compliant ($\boldsymbol{S_c}$) and particular solution ($\boldsymbol{S_p}$) trial functions in the same basis, without making any distinction on their roles in the solution recovery process. As for the corresponding state model (Section 3.2.2), the option of merging the two bases enables Trefftz-compliant functions to improve the effectiveness of the particular solution basis, at the cost of allowing domain integrals to arise in the solving system.

The particular solution basis is built using functions that satisfy exactly the static problem,

$$\boldsymbol{\mathcal{D}} \cdot \boldsymbol{S_p} = \boldsymbol{0} \tag{38}$$

The domain finite element equations are obtained by enforcing weakly the domain equation (4), using the functions collected in bases $\boldsymbol{S_c}$ and $\boldsymbol{S_p}$ for weighting,

$$\int \boldsymbol{S_i^*}\left[\boldsymbol{\mathcal{D}}\left(\boldsymbol{\rho}^{-1}\,\boldsymbol{\mathcal{D}}\cdot\boldsymbol{\sigma}\right) + \omega^2\,\boldsymbol{f}\,\boldsymbol{\sigma}\right]\,dV^e = \int \boldsymbol{S_i^*}\boldsymbol{\mathcal{D}}\left(\boldsymbol{\rho}^{-1}\boldsymbol{f_0}\right)\,dV^e \tag{39}$$

where $i = \{c, p\}$.

The first term of equation (39) is integrated by parts to force the emergence of boundary integral terms, where the Dirichlet boundary conditions (3) and the state field approximation (37) are inserted.

Some mathematical manipulation, detailed in Reference [26] yields,

$$\boldsymbol{D_{ic}}\,\boldsymbol{x_c} + \boldsymbol{D_{ip}}\,\boldsymbol{x_p} - \boldsymbol{B_i}\,\boldsymbol{y} = \boldsymbol{x_{\Gamma_i}} - \boldsymbol{x_{0_i}} \tag{40}$$

where,

$$\boldsymbol{D_{ic}} = \boldsymbol{D^*_{ci}} = \int (\boldsymbol{n}\boldsymbol{S_i})^*\,\boldsymbol{U_c}\,d\Gamma^e \tag{41}$$

$$\boldsymbol{D_{pp}} = \int (\boldsymbol{n}\boldsymbol{S_p})^*\,\boldsymbol{U_p}\,d\Gamma^e \tag{42}$$

$$\boldsymbol{B_i} = \int (\boldsymbol{n}\boldsymbol{S_i})^*\,\boldsymbol{Z}\,d\overline{\Gamma}^e_e \tag{43}$$

$$\boldsymbol{x_{\Gamma_i}} = \int (\boldsymbol{n}\boldsymbol{S_i})^*\,\boldsymbol{u_\Gamma}\,d\Gamma^e_u \tag{44}$$

$$\boldsymbol{x_{0_c}} = \omega^{-2}\int (\boldsymbol{\mathcal{D}}\cdot\boldsymbol{S_c})^*\,\boldsymbol{\rho}^{-1}\boldsymbol{f_0}\,dV^e = -\int \boldsymbol{U^*_c}\,\boldsymbol{f_0}\,dV^e \tag{45}$$

$$\boldsymbol{x_{0_p}} = \boldsymbol{0} \tag{46}$$

Next, the Neumann boundary condition (2) is enforced separately on each essential boundary of the element, using the functions listed in the flux basis $\boldsymbol{Z}$ for weighting.

$$\int \boldsymbol{Z}^*\,(\boldsymbol{n}\boldsymbol{\sigma} - \boldsymbol{t_\Gamma})\;d\Gamma^e_e = \boldsymbol{0} \tag{47}$$

Substitution of the flux approximation (36) into equation (47) yields,

$$\boldsymbol{B^*_c}\,\boldsymbol{x_c} + \boldsymbol{B^*_p}\,\boldsymbol{x_p} = \boldsymbol{y_\Gamma} \tag{48}$$

$$\boldsymbol{y_\Gamma} = \int \boldsymbol{Z}^*\,\boldsymbol{t_\Gamma}\,d\Gamma^e_e \tag{49}$$

Finally, the solving system of the (coupled) hybrid-Trefftz flux element collects the domain and boundary statements (40) and (48), to yield,

$$\left[\begin{array}{cc|c} \boldsymbol{D_{cc}} & \boldsymbol{D_{cp}} & -\boldsymbol{B_c} \\ \boldsymbol{D_{pc}} & \boldsymbol{D_{pp}} & -\boldsymbol{B_p} \\ \hline -\boldsymbol{B^*_c} & -\boldsymbol{B^*_p} & \boldsymbol{0} \end{array}\right] \left(\begin{array}{c} \boldsymbol{x_c} \\ \boldsymbol{x_p} \\ \hline \boldsymbol{y} \end{array}\right) = \left(\begin{array}{c} \boldsymbol{x_{\Gamma_c}} - \boldsymbol{x_{0_c}} \\ \boldsymbol{x_{\Gamma_p}} \\ \hline -\boldsymbol{y_\Gamma} \end{array}\right) \tag{50}$$

As compared to the coupled hybrid-Trefftz state element, the solving system (50) contains less domain integral terms, none of which in the matrix of the coefficients.

The solution of system (50) yields a unique estimate for the domain flux field, according to definition (36). Its algebraic properties are further explored in Section 3.4.1.

3.3.3. Finite element equations - the Dual Reciprocity approach

The DRM approach to the formulation of the flux model uses the same procedure for the approximation of the particular solution as described in Section 3.2.3. The objective is to find the flux field $\boldsymbol{\sigma_p}\left(\boldsymbol{x}\right)$ that satisfies the Beltarmi equation (4) in the collocation points $\overline{\boldsymbol{x}} \in V^e$. To achieve this, the following steps are taken:

- Construct a state field basis $\boldsymbol{U_p}\left(\boldsymbol{x}\right)$ such as to satisfy exactly the Helmholtz equation (24) for some arbitrary values of the wave number λ,

$$\boldsymbol{\mathcal{D}} \cdot \left[\boldsymbol{f}^{-1}\boldsymbol{\mathcal{D}}\boldsymbol{U_p}\left(\boldsymbol{x}\right)\right] + \lambda^2 \boldsymbol{\rho}\,\boldsymbol{U_p}\left(\boldsymbol{x}\right) = \boldsymbol{0}$$

- Construct a source field basis $\boldsymbol{F_p}\left(\boldsymbol{x}\right)$ using definition (25),

$$\boldsymbol{F_p}(\boldsymbol{x}) = \left(\omega^2 - \lambda^2\right)\, \boldsymbol{\rho}\; \boldsymbol{U_p}(\boldsymbol{x})$$

- Collocate the basis $\boldsymbol{F_p}$ according to equation (21), such as to recover the source function $\boldsymbol{f_0}$ in the collocation points $\overline{\boldsymbol{x}}$,

$$\boldsymbol{F_p}(\overline{\boldsymbol{x}})\, \boldsymbol{x_p} = \boldsymbol{f_0}(\overline{\boldsymbol{x}})$$

  and solve the resulting algebraic system for the weights $\boldsymbol{x_p}$.

- Define the particular flux field $\boldsymbol{\sigma_p}\left(\boldsymbol{x}\right)$ as,

$$\boldsymbol{\sigma_p}(\boldsymbol{x}) = \boldsymbol{S_p}(\boldsymbol{x})\, \boldsymbol{x_p} \tag{51}$$

$$\boldsymbol{S_p}(\boldsymbol{x}) = \boldsymbol{f}^{-1}\, \boldsymbol{\mathcal{D}}\boldsymbol{U_p}\left(\boldsymbol{x}\right) \tag{52}$$

Under the above definitions, the particular solution approximation $\boldsymbol{\sigma_p}\left(\boldsymbol{x}\right)$ satisfies the Beltrami equation,

$$\boldsymbol{\mathcal{D}}\left[\,\boldsymbol{\rho}^{-1}\,\boldsymbol{\mathcal{D}} \cdot \boldsymbol{\sigma}(\overline{\boldsymbol{x}})\right] + \omega^2\, \boldsymbol{f}\,\boldsymbol{\sigma}(\overline{\boldsymbol{x}}) = \boldsymbol{\mathcal{D}}\,\boldsymbol{\rho}^{-1}\boldsymbol{f_0}(\overline{\boldsymbol{x}}) \tag{53}$$

in the collocation points $\overline{\boldsymbol{x}}$.

The complementary solution is now obtained by enforcing weakly the Beltrami equation (4) over the domain of the element, using the functions collected in basis $\boldsymbol{S_c}$ for weighting,

$$\int \boldsymbol{S_c^*} \left[\boldsymbol{\mathcal{D}} \left(\boldsymbol{\rho}^{-1} \boldsymbol{\mathcal{D}} \cdot \boldsymbol{\sigma} \right) + \omega^2 \boldsymbol{f} \, \boldsymbol{\sigma} \right] \, dV^e = \int \boldsymbol{S_c^*} \boldsymbol{\mathcal{D}} \left(\boldsymbol{\rho}^{-1} \boldsymbol{f_0} \right) \, dV^e \tag{54}$$

The same type of manipulations as described in Section 3.3.2 are performed on equation (54). As opposed to the coupled approach, however, substitution of property (53) into the resulting expressions cancel out all domain integrals, to yield,

$$\boldsymbol{D_c} \, \boldsymbol{x_c} - \boldsymbol{B_c} \, \boldsymbol{y} = \boldsymbol{x_\Gamma} - \boldsymbol{D_p} \, \boldsymbol{x_p} \tag{55}$$

where the following definitions hold,

$$\boldsymbol{D_c} = \int (\boldsymbol{n} \boldsymbol{S_c})^* \, \boldsymbol{U_c} \, d\Gamma^e \tag{56}$$

$$\boldsymbol{D_p} = \int (\boldsymbol{n} \boldsymbol{S_c})^* \, \boldsymbol{U_p} \, d\Gamma^e \tag{57}$$

$$\boldsymbol{x_\Gamma} = \int (\boldsymbol{n} \boldsymbol{S_c})^* \, \boldsymbol{u_\Gamma} \, d\Gamma_u^e \tag{58}$$

and $\boldsymbol{B_c}$ is given by expression (43), with $i \equiv c$.

The Neumann boundary equation is identical to equation (47) used in the coupled approach. It is restated here for reader's convenience,

$$\int \boldsymbol{Z}^* \left(\boldsymbol{n} \boldsymbol{\sigma} - \boldsymbol{t_\Gamma} \right) \, d\Gamma_e^e = \boldsymbol{0}$$

Substitution of the flux field approximation (36) into the above equation yields,

$$\boldsymbol{B_c^*} \, \boldsymbol{x_c} = \boldsymbol{y_\Gamma} - \boldsymbol{B_p^*} \, \boldsymbol{x_p} \tag{59}$$

where $\boldsymbol{y_\Gamma}$ and $\boldsymbol{B_p}$ are defined by expressions (49) and (43), the latter with $i \equiv p$.

The solving system of the DRM-based hybrid-Trefftz flux element is constructed on the domain and boundary statements (55) and (59),

$$\left[\begin{array}{c|c} \boldsymbol{D_c} & -\boldsymbol{B_c} \\ \hline -\boldsymbol{B_c^*} & \boldsymbol{0} \end{array} \right] \left(\begin{array}{c} \boldsymbol{x_c} \\ \hline \boldsymbol{y} \end{array} \right) = \left(\begin{array}{c} \boldsymbol{x_\Gamma} - \boldsymbol{D_p} \, \boldsymbol{x_p} \\ \hline -\boldsymbol{y_\Gamma} + \boldsymbol{B_p^*} \boldsymbol{x_p} \end{array} \right) \tag{60}$$

As was the case of the corresponding state field formulation, system (60) is free of domain integral terms, but the sequential computation of the complementary and particular solutions mean that one can no longer compensate the other's weaknesses.

The properties of system (60) are further explored in Section 3.4.1. Its solution yields a unique estimate for the domain flux field, according to definition (36).

3.4. General comments on the formulations

The formulations presented in Sections 3.2 and 3.3 constitute the backbone of FreeHyTE. They are structurally uniform, yet simple to adapt to a variety of boundary value problems, of static, harmonic and transient nature, by adequately calibrating the differential operators and the involved parameters. Examples of such adaptations in the current framework are available for Laplace and Poisson problems [18], Helmholtz problems [20], hyperbolic boundary values problems [21], elastostatic problems [27], spectral elastodynamic problems [28], biphasic elastostatics [24], spectral and transient biphasic elastodynamics [6, 11], and transient wave propagation through multi-phase materials [9].

3.4.1. Approximation bases and solving systems

All formulations presented here use domain bases which are hierarchical, of arbitrary orders, and disconnected from the nodes of the elements. Unlike conventional boundary element method, the bases implemented in FreeHyTE are regular and present relatively simple expressions, including the (typically cumbersome) particular solution basis under the Dual Reciprocity approach. The computation of all domain bases (complementary and particular solutions, and the source field) requires the solution of a single (Helmholtz) type of equation, with different wave numbers.

The boundary bases are only subjected to completeness and linear independency constraints and can be easily adapted to be linked to the nodal values of the fields, in order to endorse the coupling with conventional finite elements (e.g. [2]). However, since the nodeless approach to the construction of the boundary bases endorses the use of localized p-refinement procedures, this option is implemented in all FreeHyTE modules.

The uniform structure of the formulations extends to solving systems (20), (32), (50) and (60). All systems are sparse, Hermitian and present a block-diagonal structure in the region corresponding to the dynamic matrices

($\boldsymbol{D_{ij}}$) of the elements. Since the domain variables $\boldsymbol{x_i}$ are strictly element-dependent and boundary variables $\boldsymbol{y}$ can be shared by at most two neighbouring elements, the solving systems are also very localized, with blocks of entries whose dimensions and insertion points depend only on the orders of the approximation bases. This trait endorses the memory pre-allocation and mapping of the system, increasing the cost efficiency of the algorithms. However, the most important feature of the solving systems is the lack of summation of 'stiffness' coefficients over adjacent elements, as typical of conventional finite elements. This feature enables the localized p-refinement of all finite elements and essential boundaries, considerably increasing the flexibility of the model's definition as compared with the conventional elements. The straightforward definition of localized p-refinement also endorses the development of automatic p-adaptive algorithms, with the further advantage that a refinement increment only requires the calculation of the new system entries, since the approximation bases are hierarchical.

The solving systems are always assembled in full in FreeHyTE. Naturally, they can subsequently be condensed on the boundary 'frame' variables [10], if desired, at the price of loosing most of the advantages related to the localized p-refinement of the essential boundaries.

3.4.2. Indeterminacy numbers

Since the orders of all (domain and boundary) approximation bases are completely free to choose, care must be taken to ensure that the solving system is both statically and kinematically indeterminate. This is a necessary condition to avoid dependencies in the solving system and may not be straightforward to enforce for inexperienced users. Consequently, FreeHyTE checks and, if necessary, automatically redefines the orders of the bases to ensure that every finite element in the mesh presents positive static and kinematic indeterminacy numbers. These numbers are defined next, for the state and flux models.

State model. Let N_D represent the total number of coefficients collected in the state field vectors $\boldsymbol{x_i}$ in systems (20) and (32), N_{RB} the total number of rigid body (i.e. flux-free) modes, and N_Γ the number of generalized fluxes collected in vector $\boldsymbol{y}$.

Systems (20) and (32) are statically indeterminate if all flux (static) variables $\boldsymbol{y}$ cannot be determined from the domain equations (11) and (27). The total number of domain equations is equal to N_D, while the total number of unknowns in these equations is $N_D + N_\Gamma - N_{RB}$ (note that the weights associated to the rigid body modes are absent from the domain equations). The static indeterminacy condition is thus expressed as,

$$\alpha = (N_D + N_\Gamma - N_{RB}) - N_D = N_\Gamma - N_{RB} > 0 \tag{61}$$

where α is the indeterminacy number. The solving system of the state formulation is thus statically indeterminate if there are enough Dirichlet boundary conditions to restrict the rigid body modes of the element, which is a well-known result in the structural mechanics context.

The kinematic indeterminacy condition requires that the kinematic variables $\boldsymbol{x_i}$ in systems (20) and (32) cannot be determined from the boundary equations (18) and (31) alone. This means that the number of boundary equations (N_Γ) should be smaller than the number of domain equations (N_D),

$$\beta = N_D - N_\Gamma > 0 \tag{62}$$

where β is the kinematic indeterminacy number. The kinematic indeterminacy condition (62) ensures that the domain approximation basis is not over-constrained by an excessive enforcement of the boundary conditions, which would render the boundary equations (18) and (31) overdetermined. The kinematic indeterminacy is controlled by the choice of the orders of the domain and boundary bases, but it may be quite delicate to ensure in some types of physical problems, as discussed in Reference [24].

Flux model. Let N_D represent the total number of coefficients collected in the flux field vectors $\boldsymbol{x_i}$ in systems (50) and (60), and N_Γ the number of state variables collected in vector $\boldsymbol{y}$ (note that, as opposed to the state model, rigid body modes do not enter the flux bases, since they cause no fluxes).

Systems (50) and (60) are kinematically indeterminate if all state (kinematic) variables $\boldsymbol{y}$ cannot be determined from the domain equations (40) and (55). The total number of domain equations is equal to N_D, while the total number of unknowns in these equations is $N_D + N_\Gamma$. The kinematic indeterminacy condition is thus expressed as,

$$\beta = (N_D + N_\Gamma) - N_D = N_\Gamma > 0 \tag{63}$$

and is trivial to observe if the domain of the problem has at least one Neumann boundary.

The static indeterminacy condition requires that the static variables $\boldsymbol{x_i}$ in systems (50) and (60) cannot be determined from the boundary equations (48) and (59) alone. This means that the number of boundary equations (N_Γ) should be smaller than the number of domain equations (N_D),

$$\alpha = N_D - N_\Gamma > 0 \tag{64}$$

As for the kinematic indeterminacy condition in the state model, the static indeterminacy condition (64) ensures that the domain approximation basis is not overconstrained by an excessive enforcement of the boundary conditions, which would render the boundary equations (48) and (59) overdetermined. The choice of the orders of the domain and boundary bases should ensure that condition (64) is observed for each finite element in the mesh.

4. Computational architecture of FreeHyTE

The unified formulation framework presented in Section 3 enables the development of unified solution procedures, easily adaptable to physically different applications. A considerable range of such procedures has been implemented to date, and their presentation is the main objective of this paper. They include user-friendly GUIs for the problem's definition, uniform data structures, the implementation of the coupled and uncoupled models (see Section 3), adaptive p-refinement procedures, along with numerous auxiliary procedures for the assessment of the soundness of the input data, system conditioning and numerical robustness, and various safety nets.

It should be noted, however, that not all procedures are implemented for all problems, as our intention has been to create the tools needed to support the future expansion of FreeHyTE, as opposed to exhaustively solve one (or a set of) given application(s). Therefore, FreeHyTE should be regarded as a collection of procedures aimed at supporting the efforts of the hybrid-Trefftz research community, and not as a 'marketable' product designed to efficiently solve a given practical problem (although it should gradually develop into that as well).

FreeHyTE is entirely implemented in the Matlab environment. The choice of Matlab is mainly justified by the simplicity of its programming language and by the availability of highly efficient procedures for dealing with multi-dimensional arrays, which ideally suits the finite element development (many matrix procedures are implicitly parallelized). Moreover, Matlab offers a variety of deployment procedures, including as standalone applications not requiring access to its proprietary components. On the other hand, Matlab codes tend to be less efficient than other (possibly lower-lever) programming languages, but the gain in terms of ease of implementation is rather high if the bulk of the coding duties relies on the engagement of (short stayed) senior year students, as have been our case. These short internships are also reflected in the modular structure of FreeHyTE [14].

The general structure of a FreeHyTE module is described in Figure 1. It consists of the following parts:

- *User input.* The user input is organized in a sequence of GUIs, where the problem and the solution parameters are defined. The definition of the problem includes the geometry of the domain, and the initial and boundary conditions. The solution parameters include the discretization in time and space, and the definition of the orders of the approximation bases. Some advanced definitions must be made in a code file.

- *Time integration.* This solution step only exists in transient analyzes. According to the time discretization parameters, FreeHyTE generates the input data for each of the spectral problems of type (1) or (4), according to a process described in Section 5.2.

- *Generation of data structures.* Three (edge-based) data structures are used in FreeHyTE. The `Edges` data structure is used to store the geometrical, topological and algorithmic information relative to all edges of the mesh. The `Loops` data structure stores the topological, physical and algorithmic information for each finite element. The `BConds` data structure contains the information required to define the boundary conditions. In p-adaptive analyses, the additional `List` data structure is used to store the information produced by the iterative p-refinement process.

- *Particular solution computation.* This step only exists in the solution of non-homogeneous boundary value problems and is only independent from the complementary solution computation in Dual Reciprocity-based procedures. It performs the procedures described in Sections 3.2.3 (for the state model) or 3.3.3 (for the flux model).

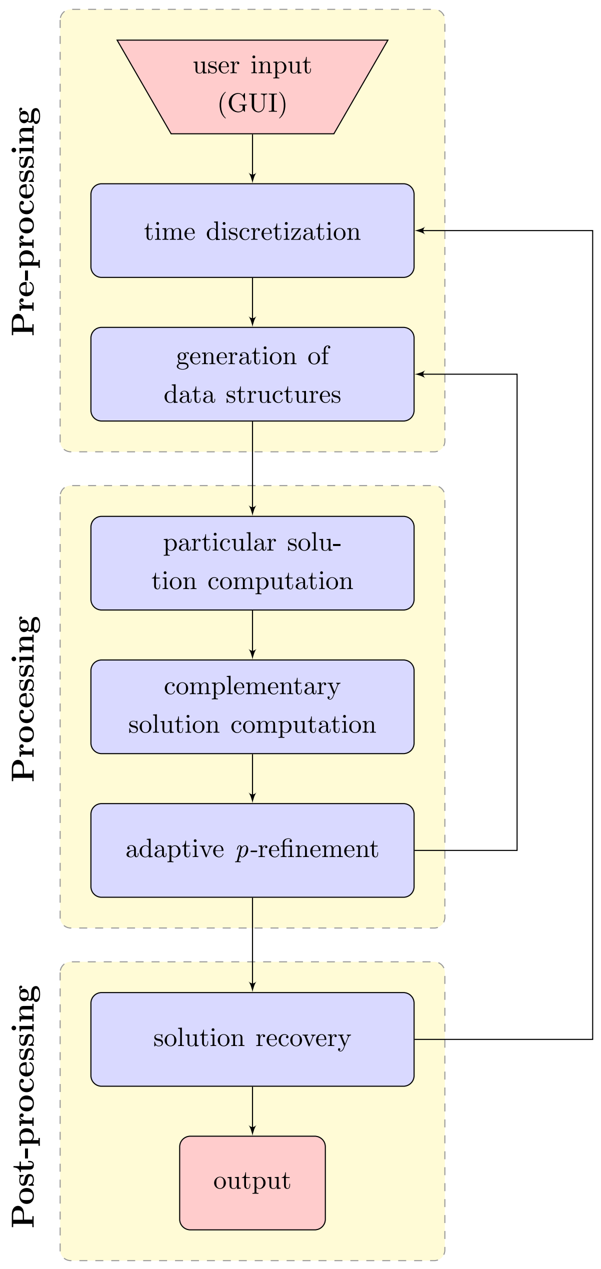


Figure 1: General structure of FreeHyTE.

- *Complementary solution computation.* In this step, the solving systems

(20), (32), (50) or (60) are constructed and solved. Pre-conditioning procedures are also in place. In p-adaptive analyses, special care is taken to ensure the numerical soundness of the solving system, to avoid the inclusion of spurious modes that would compromise the stability of the ensuing iterations.

- *Adaptive* p-*refinement*. The goal of the p-adaptive refinement algorithm is to enable inexperienced analysts to take advantage of the capabilities of the hybrid-Trefftz elements by automatically choosing adequate refinements for each approximation basis. The computational cost of the adaptive analysis is considerably larger than that of a single run, however.

- *Post-processing*. The post-processing phase consists of the reconstruction of the field approximations according to definitions (5) or (33), where the particular solution can either be analytic (if available) or approximated.

The implementation of the general solution process described above is presented in more detail in the next three sections, structured along the pre-processing, processing and post-processing stages identified in Figure 1.

5. Pre-processing

This section gives a general description of the GUI and data structures in FreeHyTE, along with a short discussion on how the spectral (time-independent) equations described by Statements 1 and 2 of Section 3.1 are obtained from the original (time-dependent) parabolic and hyperbolic problems. Specific details that may interest potential developers, but probably not general users, are confined to the appendices.

5.1. Graphical user interfaces

The general structure of the GUI is presented in Figure 2, in a flowchart format. It consists of five main interfaces, complemented by the Matlab-native `pdetool` interface 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.

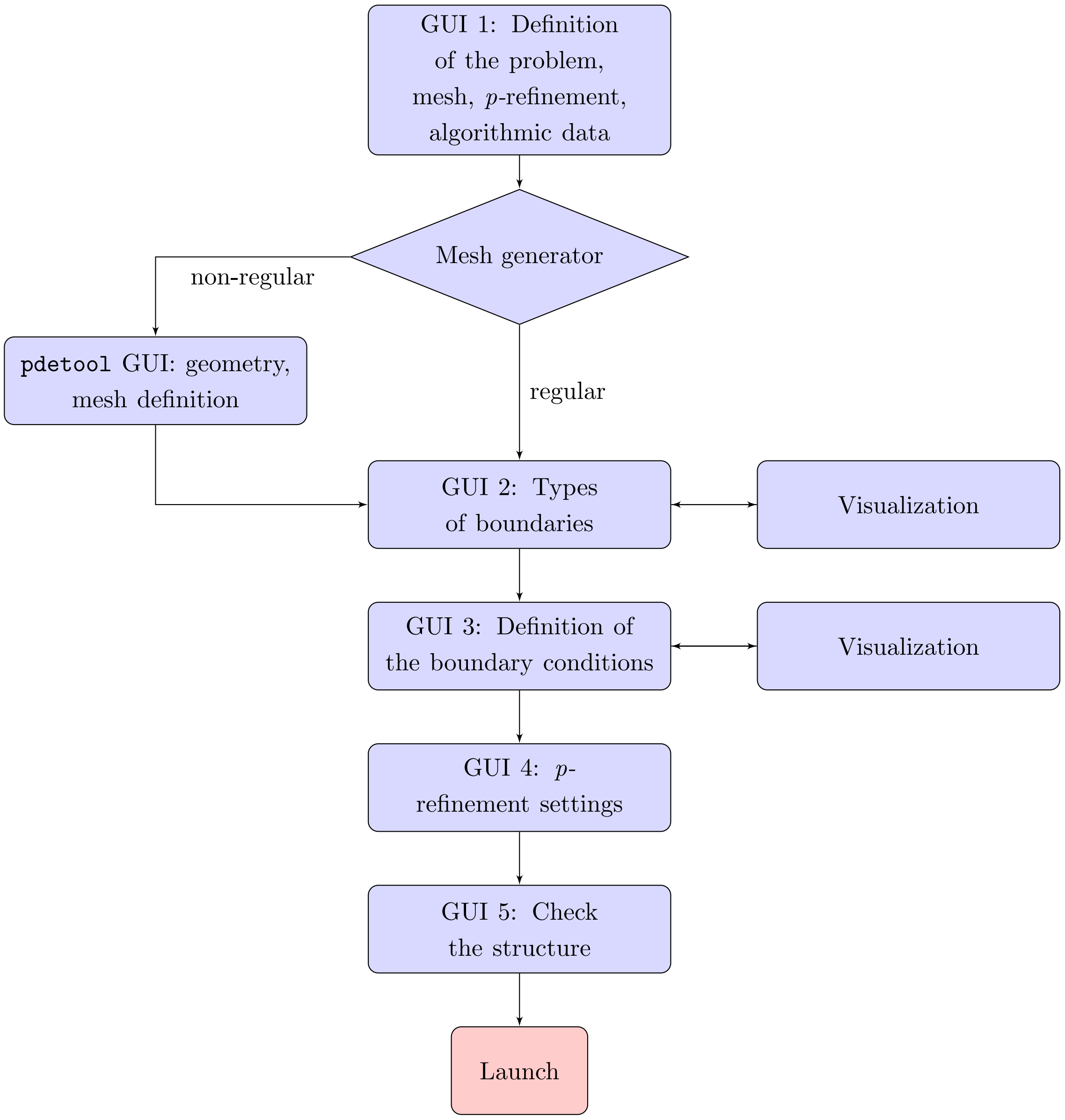

GUI 1: Definition of the problem, mesh, p-refinement, algorithmic data
Mesh generator
non-regular
pdetool GUI: geometry, mesh definition
regular
GUI 2: Types of boundaries
Visualization
GUI 3: Definition of the boundary conditions
Visualization
GUI 4: p-refinement settings
GUI 5: Check the structure
Launch

Figure 2: GUI structure in FreeHyTE.

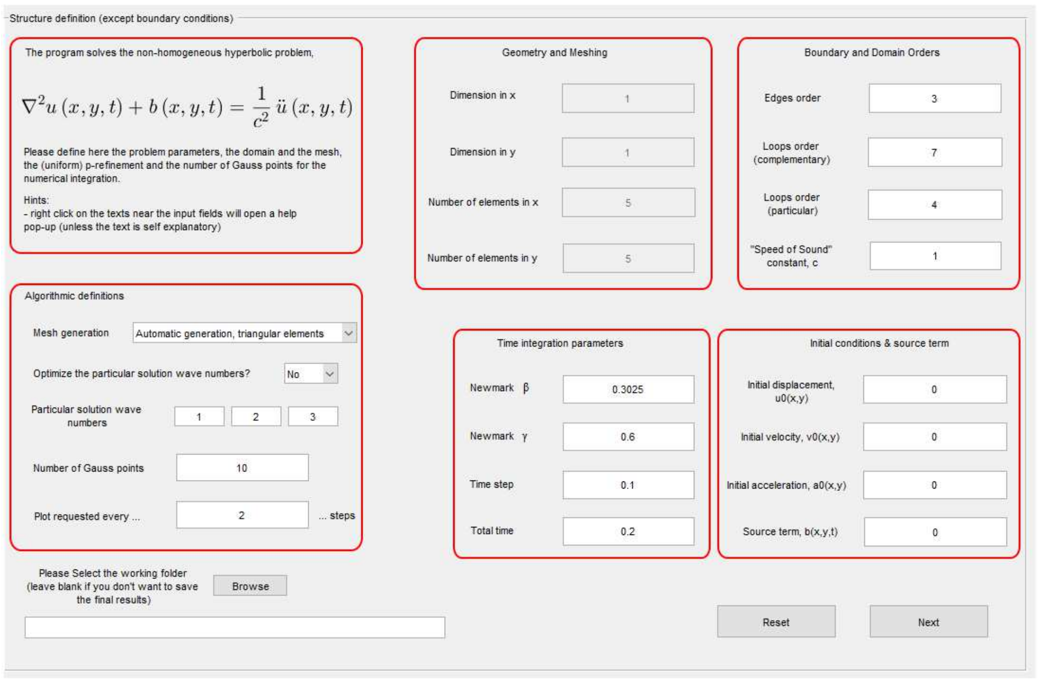
The five main GUI are described in the next subsections. Since the details of each GUI vary slightly from one module to the next, the most general form available to date is presented here for each interface. The pdetool GUI is described in the Matlab's documentation [29] and is not revisited here.

5.1.1. *GUI 1: Definition of the problem, mesh,* p-*refinement and algorithmic data*

The first GUI is used to define the geometry of the structure, initial conditions, source term, h- and p-refinements, material characteristics and algorithmic options.

The definitions of the geometry of the structure, source terms and initial conditions are made in a global, Cartesian referential. The boundary conditions are defined in local (Cartesian) referentials associated to each edge, in the normal and tangential directions. The systems of reference used in FreeHyTE are described in more detail in Appendix A.

A typical configuration of GUI 1 is presented in Figure 3. It is taken from the FreeHyTE module that solves hyperbolic boundary value problems using the Dual Reciprocity approach. The main data zones of the interface are identified with red frames. Each of these zones is briefly described below.


Structure definition (except boundary conditions)

The program solves the non-homogeneous hyperbolic problem,

$$\nabla^2 u(x,y,t) + b(x,y,t) = \frac{1}{c^2}\ddot{u}(x,y,t)$$

Please define here the problem parameters, the domain and the mesh, the (uniform) p-refinement and the number of Gauss points for the numerical integration.

Hints:
- right click on the texts near the input fields will open a help pop-up (unless the text is self explanatory)

Geometry and Meshing

Dimension in x	1
Dimension in y	1
Number of elements in x	5
Number of elements in y	5

Boundary and Domain Orders

Edges order	3
Loops order (complementary)	7
Loops order (particular)	4
"Speed of Sound" constant, c	1

Algorithmic definitions

Mesh generation	Automatic generation, triangular elements
Optimize the particular solution wave numbers?	No
Particular solution wave numbers	1, 2, 3
Number of Gauss points	10
Plot requested every ...	2 ... steps

Time integration parameters

Newmark β	0.3025
Newmark γ	0.6
Time step	0.1
Total time	0.2

Initial conditions & source term

Initial displacement, u0(x,y)	0
Initial velocity, v0(x,y)	0
Initial acceleration, a0(x,y)	0
Source term, b(x,y,t)	0

Please Select the working folder (leave blank if you don't want to save the final results)

Figure 3: Sample layout of GUI 1.

Algorithmic definitions. The most important algorithmic definition is the choice of the mesh generator to use. Two mesh generators are currently implemented in FreeHyTE, namely a regular mesh generator and a non-regular mesh generator.

The regular mesh generator can be used only for rectangular structures, which are discretized in an array of uniformly sized, rectangular finite elements. The geometry of the structure is defined by simply specifying its length and width. Likewise, for the definition of the mesh it suffices to specify the number of finite elements in the (global) X and Y directions. The origin of the global referential is taken by default in the lower left corner of the body and cannot be changed. Despite the inherent limitations, the use of the regular mesh generator is recommended wherever possible, as it ensures a faster and simpler structural definition, with no risk of mesh distortion.

The non-regular mesh generator 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. The GUI `pdetool` is one of the best ways to create a Matlab geometry at the starting level [29]. It features a simple click-and-drag interface and user gets to see the geometry as he/she creates it. Basic shapes as rectangles, ellipses and polygons can be freely combined to obtain fairly complex geometries. However, the `pdetool` mesh generator is limited to triangular elements.

Apart from choosing the mesh generator to use, the user must define the protocol for the generation of the particular solution basis (with or without optimization, see Section 6.1) and the wave numbers (λ) present in the Helmholtz equation (24), the number of Gauss-Legendre quadrature points to perform the integrations and the result plotting frequency. In the modules where available, the option to use or not automatic p-refinement is also specified in this zone.

Geometry and meshing. This area is only editable if the regular mesh generator is used to define the structure. The fields correspond to the dimensions of the mesh and number of finite elements in the Cartesian (X and Y) directions. If the non-regular mesh generator is chosen in the algorithmic definitions, the `pdetool` GUI is launched automatically when the user advances to the next interface.

Boundary and domain orders. The orders of p-refinement of the essential boundaries and finite elements are the main definitions in this area. The orders of p-refinement are assumed uniform in all elements and on all essential boundaries, although this is not a requirement, as shown in Section 3.4.1. Advanced users may define separate orders of p-refinement for each mesh entity by editing a code file, as explained in the FreeHyTE user's manuals [14]. For automatic p-refinement analyses, the orders of p-refinement inserted in this area are used as staring points for the iterative process.

Time integration parameters. The Newmark implicit scheme is implemented for the time discretization of hyperbolic (second order) boundary value problem. First order problems are discretized in time using the generalised midpoint family of methods [30], which are able to recover a variety of implicit schemes (e.g. Crank Nicolson, Galerkin, backward Euler) by the suitable choice of a single parameter. Further details on the time integration schemes implemented in FreeHyTE are given in Section 5.2.

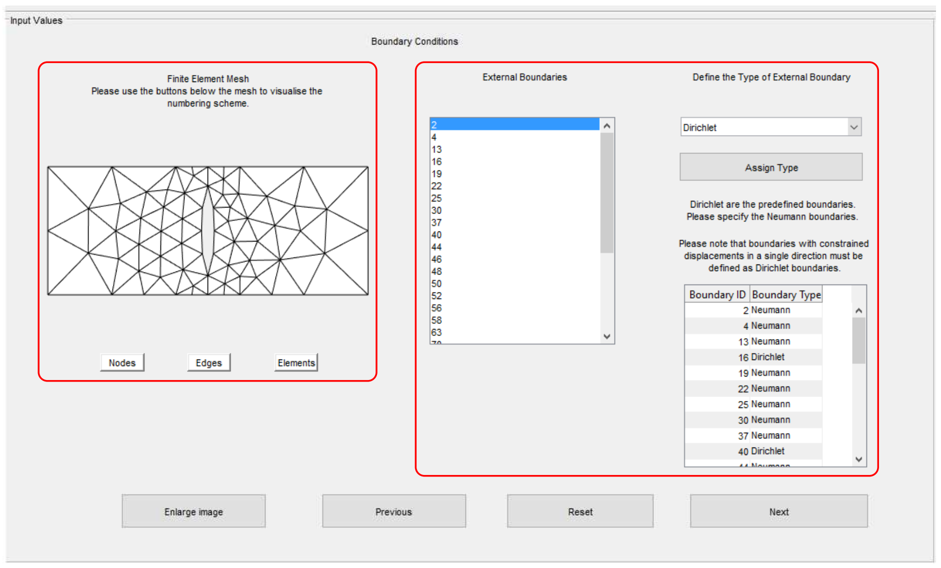
Initial conditions and source terms. The initial conditions and source terms can be defined by any mathematically valid expression in variables X, Y and, for the latter, time (t). For modules involving more complex materials, the material parameters are also defined in this region.

5.1.2. GUI 2: Types of boundaries

The second GUI is used to define the type (Dirichlet or Neumann) for each exterior side of the structure. The configuration of GUI 2 (for a plane elasticity problem) is presented in Figure 4. The main data zones are identified with red frames and briefly described below.

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. This information consists of the node, edge and element numbers, according to the buttons that are pressed. In some cases, the visualization area in GUI 2 is too small to support a clear read of the structural data. If so, 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 5).

Definition of the external boundary types. The external boundaries of the structure are listed in the *External boundaries* table. The boundary types


Input Values

Boundary Conditions

Finite Element Mesh
Please use the buttons below the mesh to visualise the numbering scheme.

External Boundaries

2
4
13
16
19
22
25
30
37
40
44
46
48
50
52
56
58
63

Define the Type of External Boundary

Dirichlet

Dirichlet are the predefined boundaries.
Please specify the Neumann boundaries.

Please note that boundaries with constrained displacements in a single direction must be defined as Dirichlet boundaries.

Boundary ID	Boundary Type
2	Neumann
4	Neumann
13	Neumann
16	Dirichlet
19	Neumann
22	Neumann
25	Neumann
30	Neumann
37	Neumann
40	Dirichlet

Figure 4: Sample layout of GUI 2.

assigned to each side are listed in the table situated on the right side of GUI 2.

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 automatically adjust to reflect the changes.

5.1.3. 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. Its typical configuration is presented in Figure 6, for the same plane elasticity problem as in previous section. The main data zones of the interface are identified with red frames. Besides a structure visualization zone essentially identical to that described in the previous section, GUI 3 presents two panels where the Dirichlet and Neumann boundary conditions should be defined in the boundary normal and tangential directions.

The boundary conditions can be described by polynomials of any order. The definition of a boundary condition is made by specifying its values in

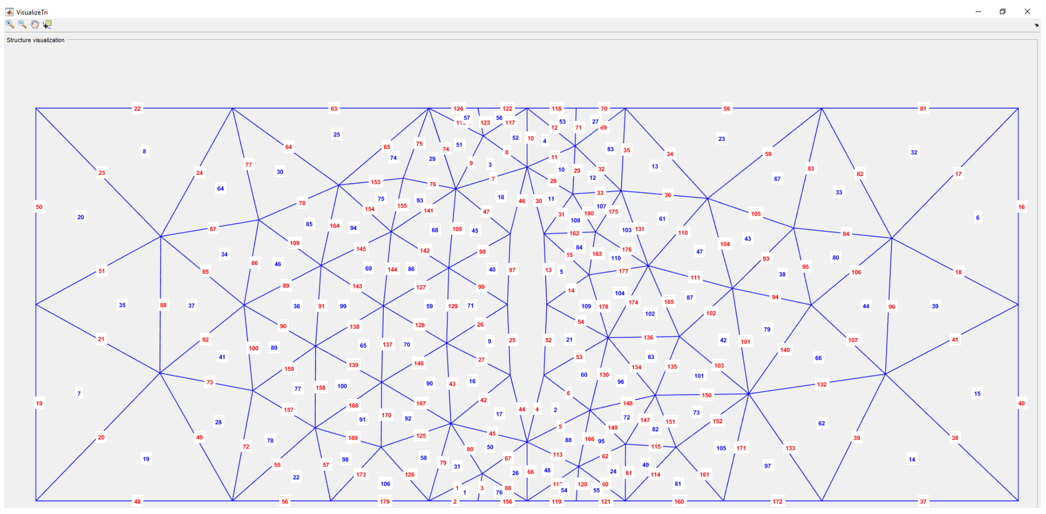


Figure 5: Sample layout of the enlarged visualization GUI.

as many equally spaced points along the boundary as needed to define its polynomial variation. For instance, a single value for a boundary condition implies that it remains constant over its length. However, in the example presented in Figure 6, the Neumann boundary condition on edge 19 is specified by two values, which indicate a linear distribution.

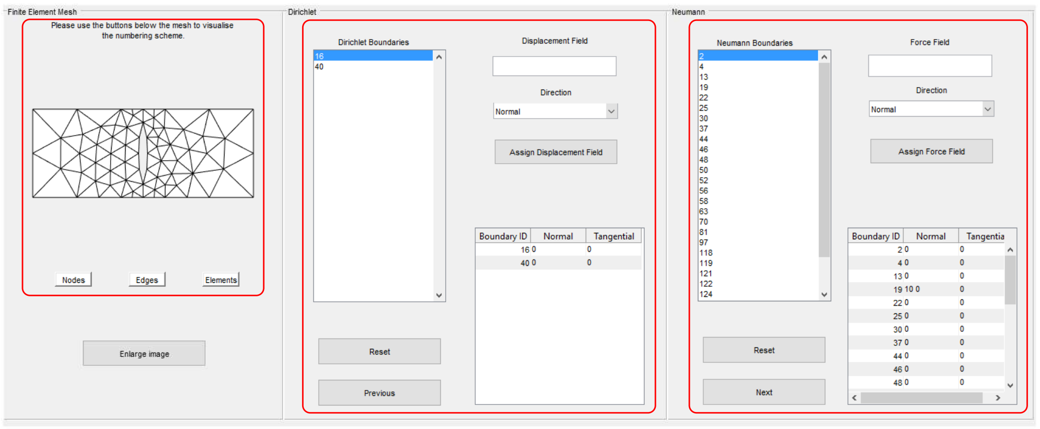
The positive directions of the boundary conditions and the order in which the boundary values are specified are in accordance with the side referentials defined in Figure A.1 of Appendix A.

5.1.4. GUI 4: Automatic p*-refinement settings*

The automatic refinement GUI shown in Figure 7 only emerges if an adaptive basis refinement analysis was requested in GUI 1, in modules where available. It is used to define the adaptive procedures and the parameters that steer the iterative basis refinement process described in more detail in Section 6.3. Besides choosing the desired selection and stopping criteria (see Sections 6.3.2 and 6.3.3, respectively), user is asked to input the algorithmic parameters that tune the p-adaptive process (see Section 6.3.5).

5.1.5. GUI 5: Check the structure

GUI 5 (Figure 8) is the last stop before launching the execution of the FreeHyTE calculation module. It is meant to allow the user to verify the



Dirichlet Boundaries: 16, 40

Direction: Normal

Boundary ID	Normal	Tangential
16	0	0
40	0	0

Neumann Boundaries: 2, 4, 13, 19, 22, 25, 30, 37, 44, 46, 48, 50, 52, 56, 58, 63, 70, 81, 97, 118, 119, 121, 122, 124

Direction: Normal

Boundary ID	Normal	Tangentia
2	0	0
4	0	0
13	0	0
19	10 0	0
22	0	0
25	0	0
30	0	0
37	0	0
44	0	0
46	0	0
48	0	0

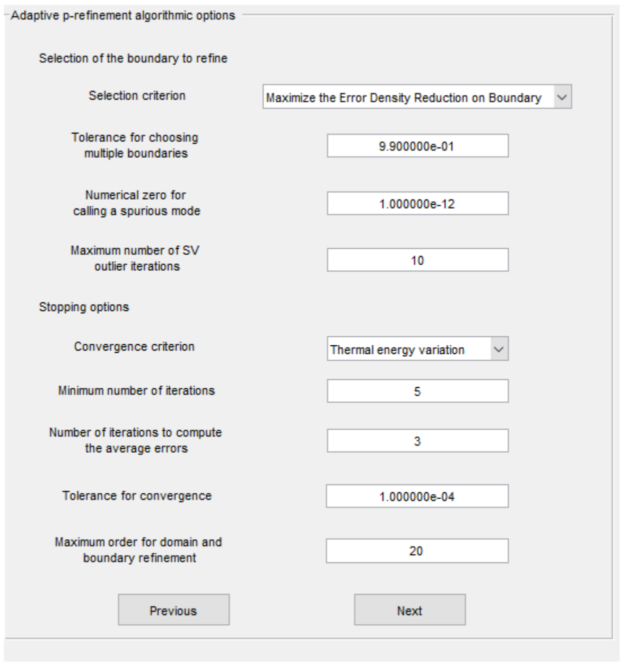
Figure 6: Sample layout of GUI 3.

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 (in the normal and tangential directions). Dirichlet sides are plotted in black, while Neumann sides are plotted in red, according to the definitions made in Figure 6. It features zoom, pan and data cursor capabilities, to support the visualization of the structure in more detail.

5.2. Time discretization

The general framework used in FreeHyTE to bring first and second order *transient* problems to the spectral form (1) is detailed in this section. While many options are available for the time discretization of transient problems [31], only a generalized mid-point technique and the Newmark method are currently implemented in FreeHyTE, for first and second order problems, respectively.

Note that for non-transient (e.g. harmonic and periodic) problems, the spectral form is easily acquired using the discrete Fourier transform, and the source term $\boldsymbol{f}_0$ present in the Navier equation (1) is free from initial condition-dependent constituents [28]. Therefore, the application of the techniques described in Section 3 is straightforward.


Adaptive p-refinement algorithmic options

Selection of the boundary to refine

Selection criterion: Maximize the Error Density Reduction on Boundary

Tolerance for choosing multiple boundaries: 9.900000e-01

Numerical zero for calling a spurious mode: 1.000000e-12

Maximum number of SV outlier iterations: 10

Stopping options

Convergence criterion: Thermal energy variation

Minimum number of iterations: 5

Number of iterations to compute the average errors: 3

Tolerance for convergence: 1.000000e-04

Maximum order for domain and boundary refinement: 20

Figure 7: Sample layout of GUI 4.

5.2.1. First order problems

Let a general first order, transient problem be written as,

$$\boldsymbol{\mathcal{D}} \cdot [\boldsymbol{k}\boldsymbol{\mathcal{D}}\boldsymbol{u}(\boldsymbol{x}, t)] + \boldsymbol{b}(\boldsymbol{x}, t) = \boldsymbol{c}\dot{\boldsymbol{u}}(\boldsymbol{x}, t) \tag{65}$$

where $\boldsymbol{b}(\boldsymbol{x}, t)$ is a source term defined on the domain and $\boldsymbol{c}$ is a generalized capacity term. The flux and state boundary conditions are consistent with equations (2) and (3), where the involved fields are now time-dependent, and complemented by the initial conditions,

$$\boldsymbol{u}(\boldsymbol{x}, 0) = \boldsymbol{u_0}(\boldsymbol{x}) \tag{66}$$

$$\dot{\boldsymbol{u}}(\boldsymbol{x}, 0) = \boldsymbol{v_0}(\boldsymbol{x}) \tag{67}$$

where $\boldsymbol{u_0}(\boldsymbol{x})$ denotes the initial state field and $\boldsymbol{v_0}(\boldsymbol{x})$ its initial rate of change.

Let now the total time of the analysis be divided into an arbitrary number of time steps of size Δt. The time derivative of the solution $\boldsymbol{u}(\boldsymbol{x}, t)$ at the

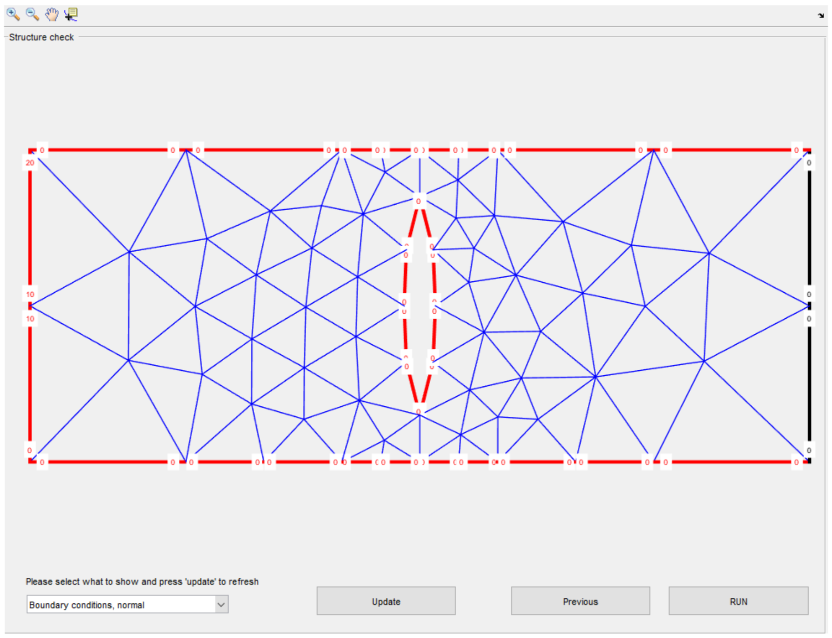

Structure check
Please select what to show and press 'update' to refresh
Boundary conditions, normal
Update
Previous
RUN

Figure 8: Sample layout of GUI 5.

end of each time interval is approximated as,

$$\dot{\boldsymbol{u}}_{\boldsymbol{\Delta t}} \simeq \frac{1}{\alpha \Delta t} \left(\boldsymbol{u}_{\boldsymbol{\Delta t}} - \boldsymbol{u}_{\boldsymbol{0}} \right) - \frac{1-\alpha}{\alpha} \boldsymbol{v}_{\boldsymbol{0}} \tag{68}$$

A variety of implicit time integration schemes are recovered by setting different values for parameter α in equation (68), including the Crank Nicolson scheme $\left(\alpha = \frac{1}{2}\right)$, the Galerkin scheme $\left(\alpha = \frac{2}{3}\right)$, and the backward Euler scheme $(\alpha = 1)$.

Substitution of approximation (68) into the governing equation (65) immediately recovers equation (1), with

$$\boldsymbol{\rho} = \boldsymbol{c} \tag{69}$$

$$\omega^2 = -\frac{1}{\alpha \Delta t} \tag{70}$$

$$\boldsymbol{f}_{\boldsymbol{0}} = -\boldsymbol{b}_{\boldsymbol{\Delta t}} - \boldsymbol{c} \frac{1}{\alpha \Delta t} \left[\boldsymbol{u}_{\boldsymbol{0}} + (1-\alpha) \Delta t \boldsymbol{v}_{\boldsymbol{0}} \right] \tag{71}$$

where $\boldsymbol{b}_{\boldsymbol{\Delta t}}$ is the source field $\boldsymbol{b}$ prescribed at the end of the time interval.

Unlike problems discretized using the discrete Fourier transform, the generalized frequency (70) is now purely imaginary and the spectral problem (1) results inhomogeneous even if $\boldsymbol{b} = \boldsymbol{0}$, because of the presence of initial condition terms.

5.2.2. Second order problems

Let a general second order, transient problem be written as,

$$\boldsymbol{\mathcal{D}} \cdot [\boldsymbol{k}\boldsymbol{\mathcal{D}}\boldsymbol{u}(\boldsymbol{x},t)] + \boldsymbol{b}(\boldsymbol{x},t) = \boldsymbol{d}\dot{\boldsymbol{u}}(\boldsymbol{x},t) + \boldsymbol{m}\ddot{\boldsymbol{u}}(\boldsymbol{x},t) \tag{72}$$

where $\boldsymbol{b}(\boldsymbol{x},t)$ is a source term defined on the domain, and $\boldsymbol{d}$ and $\boldsymbol{m}$ are some generalized damping and mass terms. The boundary conditions (2) and (3) are complemented by the initial conditions,

$$\boldsymbol{u}(\boldsymbol{x},0) = \boldsymbol{u_0}(\boldsymbol{x}) \tag{73}$$
$$\dot{\boldsymbol{u}}(\boldsymbol{x},0) = \boldsymbol{v_0}(\boldsymbol{x}) \tag{74}$$
$$\ddot{\boldsymbol{u}}(\boldsymbol{x},0) = \boldsymbol{a_0}(\boldsymbol{x}) \tag{75}$$

After the division of the total duration of the analysis into time steps of size Δt, the time derivatives of the state field $\boldsymbol{u}(\boldsymbol{x},t)$ at the end of a time step are approximated using the Newmark scheme as,

$$\dot{\boldsymbol{u}}_{\boldsymbol{\Delta t}} \backsimeq \frac{\gamma}{\beta\Delta t}\boldsymbol{u}_{\boldsymbol{\Delta t}} - \left[\frac{\gamma}{\beta\Delta t}\boldsymbol{u_0} + \left(\frac{\gamma}{\beta} - 1\right)\boldsymbol{v_0} + \left(\frac{\gamma}{2\beta} - 1\right)\boldsymbol{a_0}\Delta t\right] \tag{76}$$
$$\ddot{\boldsymbol{u}}_{\boldsymbol{\Delta t}} \backsimeq \frac{1}{\beta\Delta t^2}\boldsymbol{u}_{\boldsymbol{\Delta t}} - \left[\frac{1}{\beta\Delta t^2}\boldsymbol{u_0} + \frac{1}{\beta\Delta t}\boldsymbol{v_0} + \left(\frac{1}{2\beta} - 1\right)\boldsymbol{a_0}\right] \tag{77}$$

where β and γ are calibration parameters.

As shown in detail in Reference [20], the collocation of the governing equation (72) at the end of the time interval and substitution of approximations (76) and (77) in the resulting expressions recovers the spectral form (1) under the following definitions,

$$\omega^2 = -\frac{1}{\beta\Delta t^2} \tag{78}$$
$$\boldsymbol{\rho} = \boldsymbol{m} - \frac{\gamma}{\sqrt{\beta}}\hat{\imath}\omega^{-1}\boldsymbol{d} \tag{79}$$
$$\boldsymbol{f_0} = -\left[\boldsymbol{b}_{\boldsymbol{\Delta t}} + \frac{1}{\beta\Delta t}\left(\gamma\boldsymbol{d}\overline{\boldsymbol{u}}_{\boldsymbol{0}} + \boldsymbol{m}\overline{\boldsymbol{v}}_{\boldsymbol{0}}\right)\right] \tag{80}$$

where $\hat{\imath}$ is the imaginary unit and,

$$\overline{\boldsymbol{u}}_{\mathbf{0}} = \boldsymbol{u}_{\mathbf{0}} + \left(1 - \frac{\beta}{\gamma}\right) \boldsymbol{v}_{\mathbf{0}} \Delta t + \left(\frac{1}{2} - \frac{\beta}{\gamma}\right) \boldsymbol{a}_{\mathbf{0}} \Delta t^2 \tag{81}$$

$$\overline{\boldsymbol{v}}_{\mathbf{0}} \Delta t = \boldsymbol{u}_{\mathbf{0}} + \boldsymbol{v}_{\mathbf{0}} \Delta t + \left(\frac{1}{2} - \beta\right) \boldsymbol{a}_{\mathbf{0}} \Delta t^2 \tag{82}$$

It is noted that the time integration techniques presented in Section 5.2.1 and the current section are designed to avoid the recomputation of systems (20), (32), (50) and (60) at each time step (provided the time step is constant). Only the right-hand side vectors are updated, and the solution recomputed using the (pre-stored) LU-factorized form of the matrix of coefficients.

5.3. Data structures

The FreeHyTE engine gets data from two main sources: the GUIs and the mesh generators. Data received from the GUIs are directly included in the data structures. Conversely, data received from the mesh generators need to be treated to build the topological information, which is then included in the data structures.

The mesh data and data structures used in FreeHyTE are briefly described in this section. A more complete description of the data structures may interest potential developers and is presented in Appendix B.

5.3.1. Mesh data

At this stage of development, FreeHyTE handles *automatically* regular meshes of rectangular finite elements and non-regular meshed of triangular finite elements.

When the regular mesh generator is used to define the structure, the generation of topological data requires no interpretation.

Any mesh generator can be used to define non-regular meshes of triangular elements provided that it is able to output the following information:

- a list of the nodal coordinates in the global Cartesian referential;
- a list of the indices of the nodes belonging to each triangular element.

This information is included in the default mesh data generated by the Matlab's `pdetool` mesh generator, both in the old (`pet`) and in the new (`FEMesh`) formats. The data is then processed by an interpreter which generates the topological information of the mesh, including the initial and final nodes of each edge and their left and right elements. In accordance to the default orientation of the edges (see Appendix A), there always exists a left element for an edge. The edges without an element to their right are exterior edges. A list with the edges belonging to each element is also produced at this stage.

It is noted that, except for the mesh interpreter, FreeHyTE routines are not conditioned in any way by the shape and number of edges of the elements. Therefore, if the user is able to provide the geometrical and topological information mentioned above (and further detailed in Appendix B.1), any mesh configuration is acceptable.

5.3.2. Data structures

The user data input through the GUIs and the topological data gathered from the mesh generator enable the construction of the `Edges`, `Loops` and `BConds` data structures, where the geometrical, topological and algorithmic data referring to the edges, finite elements and boundary conditions are stored.

Brief descriptions of the fields of these data structures are given in Tables 1 to 3, with more details to be found in Appendix B.2.

6. Processing

This section covers the procedures used to recover the particular and complementary solutions of the spectral problems defined by Statements 1 and 2. They occur after the definition of the input data and the construction of the data structures and do not require the user's intervention.

The calculation of the particular solution in the Dual Reciprocity procedure presented in Sections 3.2.3 and 3.3.3 is covered in Section 6.1, followed by the computation of the particular solution in Section 6.2, with a focus on the construction and conditioning control of the solving system. Finally, the adaptive p-refinement procedure is presented in Section 6.3.

6.1. Computation of the particular solution

This section describes the implementation of the Dual Reciprocity technique for the computation of the particular solution of equations (1) and (4). When the particular and complementary solution bases are coupled, as described in Section (3.2.2) and (3.3.2), the process presented here is not called

Geometrical data	`Edges.nini`, `Edges.nfin`	Initial and final nodes of the edge
	`Edges.parametric`	Global coordinates of the starting node of the edge and extensions in X and Y
Topological data	`Edges.lleft`, `Edges.lright`	Elements at the left and right of the edge
Algorithmic data	`Edges.type`	Type of edge (Neumann or Dirichlet)
	`Edges.order`	Order of the edge basis
	`Edges.insert`	Insertion point of the edge in the solving system
	`Edges.dim`	Dimension of the edge in the solving system

Table 1: `Edges` data structure

for, and both solutions are recovered using the technique given in Section (6.2).

The particular solution basis is constructed on the free-field solutions of the Helmholtz-type equation (24). While obviously problem-dependent, equations of type (24) generally have known analytic solutions, of both regular and singular natures. Singular solutions may foster the convergence of problems involving field concentrations [18] if the singularity point is consistently defined by the user, and damage the numerical stability of the procedure otherwise. In FreeHyTE, only regular solutions are currently implemented. Even so, the choice of the particular solution basis remains considerably flexible, as it allows for multiple definitions of the wave number λ, present in equation (24).

The procedure to compute the particular solution is conducted one finite element at a time, following the workflow presented in Figure 9. User is only requested to provide the desired order of the particular solution basis and three wave numbers ($\boldsymbol{\lambda}$) to construct the basis with. The only restriction

Geometrical data	`Loops.center`	Global coordinates of the barycenter
	`Loops.area`	Area of the element
Topological data	`Loops.nodes`	Nodes of the element
	`Loops.edges`	Edges of the element
Algorithmic data	`Loops.material`	Physical data of the element
	`Loops.order`	Order of the element basis
	`Loops.insert`	Insertion point of the element in the solving system
	`Loops.dim`	Dimension of the element in the solving system

Table 2: `Loops` data structure

to the choice of the wave numbers is that they should be different from the generalized frequency ω, either resulted from the discrete Fourier transform of the original, time-dependent problem, or computed according to expressions (70) or (78). Large values of the wave numbers should also generally be avoided, as they may render the approximation functions $\boldsymbol{F_p}$ highly oscillatory, affecting the numerical stability of the collocation process. In Reference [21], it was shown that the approximation of the source term is fairly robust to sub-optimal choices of the wave numbers as long as the numerical stability of the collocation system (21) is not compromised, so simply choosing some small wave numbers should be enough in most cases. Alternatively, user can opt for an adaptive choice of the the wave numbers such as to minimize the error of the source function approximation in the domain of each element, in which case the supplied wave numbers function as a starting point for the minimization process.

`BConds.Neumann`	Applied fluxes on the exterior Neumann sides
`BConds.Dirichlet`	Enforced state fields on the exterior Dirichlet sides

Table 3: `BConds` data structure

Based on the number of functions in the particular solution basis, a grid of collocation points $\overline{\boldsymbol{x}}$ is automatically generated. The number of collocation points is chosen to be square number which is closest to, but higher than the total number of functions in the particular solution basis. This condition ensures that the collocation system (21) is not underdetermined. The position of the collocation points in the domain of the element corresponds to a Gauss-Chebyshev quadrature.

Next, the values of the source term $\boldsymbol{f_0}$ are calculated in the collocation points. In transient problems with null initial conditions, it is common to have null source terms in some elements. This may lead to serious numerical instability if an adaptive choice of wave numbers is requested, so before proceeding further, FreeHyTE checks for elements with null source terms and just sets their particular solutions to zero ($\boldsymbol{x_p} = \boldsymbol{0}$).

If the current element has non-null source term, FreeHyTE constructs the collocation system (21), and solves it using the Moore-Penrose pseudoinverse. If an adaptive choice of the wave numbers is requested, the process does not end here, but enters an iterative Nedler-Mead minimization scheme to optimize the wave numbers. The objective function to be minimized is defined as the norm of the approximation residual, normalized to the norm of the source function,

$$\epsilon = \frac{\|\boldsymbol{F_p}\,\boldsymbol{x_p} - \boldsymbol{f_0}\|}{\|\boldsymbol{f_0}\|} \tag{83}$$

The function norms are calculated numerically, using a richer Gauss quadrature than the one used for the collocation.

The particular solution approximation is then stored as FreeHyTE advances to the computation of the complementary solution.

6.2. Computation of the complementary solution

This section describes the construction and solution procedures of the solving systems (20), (32), (50) and (60). As shown in Section 3, these sys-

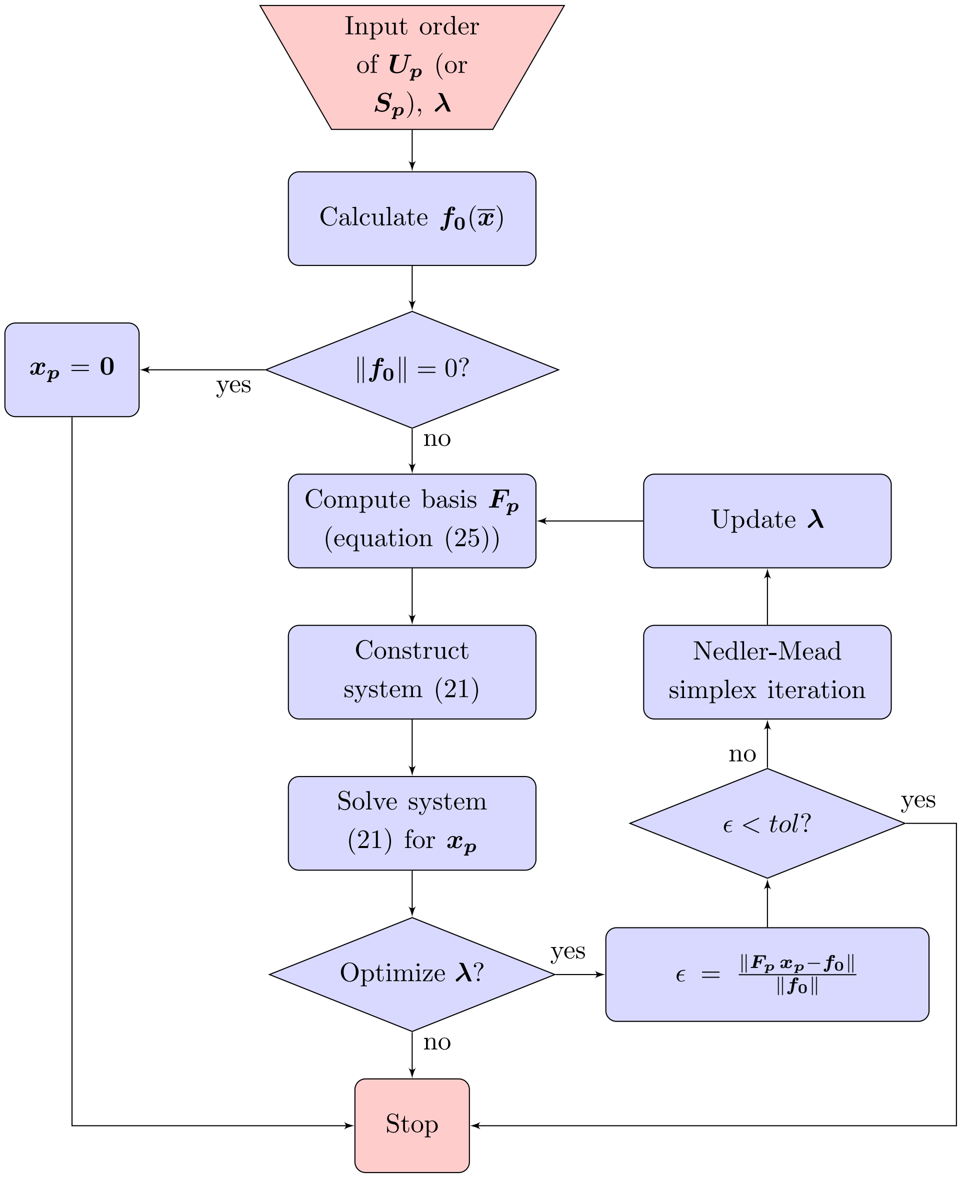

Input order of $\boldsymbol{U_p}$ (or $\boldsymbol{S_p}$), $\boldsymbol{\lambda}$
Calculate $\boldsymbol{f_0}(\overline{\boldsymbol{x}})$
$\|\boldsymbol{f_0}\| = 0$?
yes
$\boldsymbol{x_p} = \boldsymbol{0}$
no
Compute basis $\boldsymbol{F_p}$ (equation (25))
Update $\boldsymbol{\lambda}$
Construct system (21)
Nedler-Mead simplex iteration
no
Solve system (21) for $\boldsymbol{x_p}$
$\epsilon < tol$?
yes
Optimize $\boldsymbol{\lambda}$?
yes
$\epsilon = \frac{\|\boldsymbol{F_p}\,\boldsymbol{x_p} - \boldsymbol{f_0}\|}{\|\boldsymbol{f_0}\|}$
no
Stop

Figure 9: Particular solution computation in FreeHyTE.

tems enable the computation of the complementary solutions of the spectral problems (1) and (4) and, in the case of the coupled solution procedures, of the particular solution as well. Since the solving systems share the same layout and properties, they are assembled and solved in the same manner irrespectively of the formulation they belong to.

6.2.1. Assembly of the solving system

A typical layout of a solving system is presented in Figure 10. It features a logical block structure, directly related to the topological structure of the mesh, which leads to an efficient organization of the variables and of the associated matrices.

The dynamic matrices belonging to each finite element are distributed in blocks along the main diagonal. The blocks are square in shape and have as many lines as functions in the domains' bases. The boundary matrices may be shared by at most two adjacent elements (edge 2 in Figure 10), or may belong to a single element, if the boundary is exterior (edges 1 and 3 in Figure 10). They share one of the dimensions with the element(s) they belong to. The other dimension is equal to the number of functions in the boundary basis.

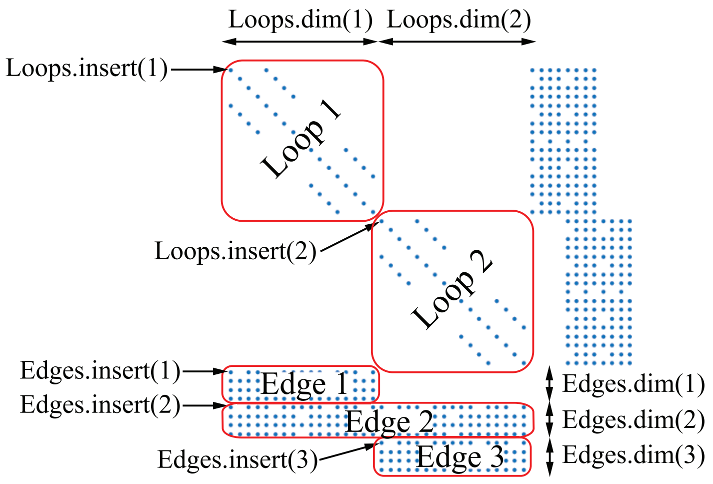


Figure 10: Sample layout a solving system. The blocks that store the dynamic and boundary matrices are marked with red frames.

The insertion points and dimensions of the blocks corresponding to the finite elements and essential edges are stored in the fields `insert` and `dim` of the respective data structures (see Section 5.3.2), as illustrated in Figure 10. The disjunction of the domain and boundary matrices (e.g. no summation of coefficients are required for the assembly of the solving system) mean that different bases can be easily accommodated by simply setting different dimensions for the respective blocks in the solving system.

All integrations required for the computation of the coefficients of the solving systems are performed numerically. The Gauss-Legendre quadrature rule is used to compute the integrals, with the number of integration points specified by the user in GUI 1 (Section 5.1.1). The integration domains are mapped to $[-1, 1]$ spaces, whether they are one- or two-dimensional. As typical of the Trefftz methods, domain integrations only occur in the coupled formulations presented in Sections 3.2.2 and 3.3.2, and even in those cases they are confined to few terms of the solving system. FreeHyTE takes full advantage of the vectorial programming paradigm of Matlab and opts for a block-wise (ass opposed to coefficient-wise) construction of the solving system. All terms of a block are computed at once, using two-, three- and four-dimensional arrays to list the values of the integrands in the Gauss points, according to Table 4.

Blocks	Structure of the integrand	Dimensions
Free vector, boundary integral	Two-dimensional	Orders of the test basis (lines), Gauss abscissas (columns)
Free vector, domain integral	Three-dimensional	Orders of the test basis (lines), Gauss abscissas (columns, pages)
Matrix block, boundary integral	Three-dimensional	Orders of the test and trial bases (lines, columns), Gauss abscissas (pages)
Matrix block, domain integral	Four-dimensional	Orders of the test and trial bases (lines, columns), Gauss abscissas (pages, blocks)

Table 4: Vectorialized integrand structures

The only exception to the block-wise system construction is the integration of the $\boldsymbol{D_{pp}}$ coefficients on the domains of triangular elements, in the state model. There, the non-regular distribution of the Gauss points renders the construction of the required four-dimensional structures too slow and a coefficient-wise approach (involving two nested `for` cycles) is used instead.

6.2.2. Solution of the solving system

Due to the high convergence rates of the Trefftz finite elements under p-refinement [24], high order bases are typically used in Trefftz models. The drawback of this strategy is that the solving systems may be more prone to numerical instability than in models involving conventional finite elements (though typically less than for boundary elements).

This issue is tackled at two levels. At the first level, all solving systems are preconditioned. At the second level, the numerical stability of the systems is diagnosed to support the choice of an adequate solver.

Preconditioning of the solving system. Consider a linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{p}$, where $\boldsymbol{A}$ is a square, possibly Hermitian matrix and $\boldsymbol{x}$ and $\boldsymbol{p}$ are vectors. Then, the system may be written in the scaled form,

$$\overline{\boldsymbol{A}}\overline{\boldsymbol{x}} = \overline{\boldsymbol{p}} \tag{84}$$

where the scaled arrays are defined as:

$$\begin{aligned}
&\overline{\boldsymbol{A}} = \boldsymbol{S}^{*}\boldsymbol{A}\boldsymbol{S}\\
&\overline{\boldsymbol{x}} = \boldsymbol{S}^{-1}\boldsymbol{x}\\
&\overline{\boldsymbol{p}} = \boldsymbol{S}^{*}\boldsymbol{p}
\end{aligned}$$

In the above definitions, $\boldsymbol{S}$ represents a diagonal scaling matrix, whose terms are defined as the square roots of the diagonal terms of the matrix $\boldsymbol{A}$. When a null diagonal term is encountered in matrix $\boldsymbol{A}$, the respective scaling term is equal to one, meaning that no scaling is applied to the respective lines and columns. This scaling procedure preserves the symmetry of the original system and was shown to be highly effective in lowering the condition number of the solving system it is applied to [11].

Solution of the solving system. The default Matlab solver (`mldivide`) is used in FreeHyTE for the solution of well-conditioned solving systems. It is a highly efficient and adaptive solver, which analyses the structure of the system and than launches the most appropriate solution procedure.

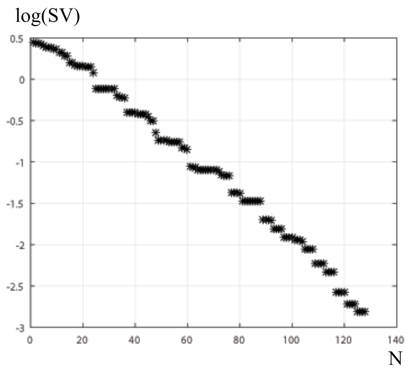
A truncated singular value decomposition technique [32] is used to find the solution of ill-conditioned systems. To detect ill-conditioned systems, two procedures are currently implemented in FreeHyTE. The first procedure is based on the calculation of an estimate of the condition number of the matrix of coefficients. If the reciprocal of the condition number is smaller than the precision of the machine, the system is considered ill-conditioned and its solution is based on the computation of the Moore-Penrose pseudoinverse of the matrix. Matlab's default tolerance for eliminating (truncating) small singular values is used,

$$tol = \varepsilon \, N \max(\mathbf{\Sigma}) \tag{85}$$

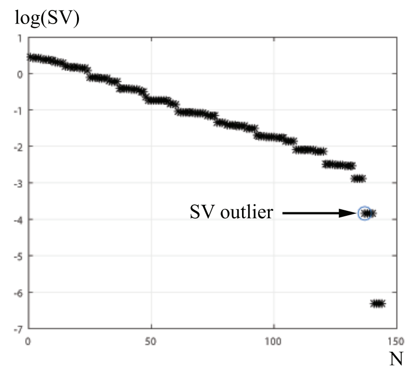
where ε is the machine precision (machine epsilon), N is the dimension of the system and $\mathbf{\Sigma}$ is the singular value matrix.

Unfortunately, a large reciprocal of the condition number (as compared to the precision of the machine) is a necessary, but not a sufficient condition for the numerical stability of the solving system. To illustrate this issue, the logarithmic plot of the singular values of a well-conditioned coefficient matrix is presented in Figure 11(a). The plot reveals a rather good alignment of the (logarithms of the) singular values, which, in our experience, is an important indicator of a numerically stable system. Indeed, the numerical solution of the problem is smooth and continuous. A one unit increase of the order of the approximation bases on the boundaries, however, leads to a matrix of coefficients whose singular values are distributed according to the plot in Figure 11(b). Here, the addition of functions to the boundary bases leads to the emergence of singular values which are unaligned with the previous ones and separated from them by gaps of at least one order of magnitude. While the condition number in the second configuration does suffer a considerable increase from the first case (more than three orders of magnitude), its reciprocal ($\sim 10^7$) is still very far from the machine precision. However, the solutions corresponding to the second case are pronouncedly unstable and discontinuous.

Therefore, the second procedure for the detection of ill-conditioned systems is based on the analysis of its singular values. Marked discontinuities and misalignments between consecutive singular values are sought for and when found, the first 'outlier' singular value (Figure 11(b)) is used as a truncation tolerance instead of the Matlab's default (85). This procedure is computationally more expensive than the one based on the assessment of the condition number, but implicitly provides a threshold for the truncation.



(a) Singular values of a well-conditioned matrix.



(b) Singular values of an ill-conditioned matrix.

Figure 11: Comparison of the singular values of two matrices with seemingly robust condition numbers.

6.3. Adaptive p*-refinement*

The objective of the adaptive *p*-refinement procedure is twofold. On the one hand, it aims to take full advantage of the flexible structure of FreeHyTE, by optimizing the choice of basis refinement for each essential boundary and finite element. On the other hand, it attempts to do so automatically, therefore lifting the burden of such choice from the user's shoulders. When the adaptive *p*-refinement option is invoked, the orders of the approximation bases specified in GUI 1 (see Section 5.1.1) are treated as initial orders of the *p*-refinement process.

In essence, the adaptive *p*-refinement algorithm searches, at each iteration, for the basis refinement that would cause the largest impact on the finite element solution according to some selection criterion. The process stops when further basis refinement fails to significantly improve the solution for a few consecutive steps, according to a stopping criterion. The hierarchical structure of the approximation bases enables FreeHyTE to avoid re-computing the terms of the solving system anew in each iteration. This feature is instrumental to improve the computational efficiency of the algorithm. However, its cost is still considerably superior to that of single-step analyses, with fixed basis refinements.

The theoretical bases of the automatic *p*-refinement process are given in Section 6.3.1, followed by the description of its refinement and stopping criteria (Sections 6.3.2 and 6.3.3). The implementation of the *p*-refinement algorithm is described in Section 6.3.4. The input data and the data structure specific to the p-refinement algorithm are briefly described in Section 6.3.5.

6.3.1. Theoretical grounds

The p-refinement process presented here generalizes the technique first presented in Reference [33] to non-homogeneous boundary value problems, and extends it to accommodate various selection and stopping criteria.

Consider that the finite element solving system was assembled with some given orders of approximations in the domains of the elements and on the essential boundaries. In order to improve the finite element solution, the orders of one or more boundary approximations must be increased. The objective of the p-adaptive algorithm is to improve the finite element solution by locally increasing the orders of the approximation functions on certain essential boundaries, selected according to some consistent and meaningful criteria. A selection criterion is labelled as *consistent* if its application does not depend on some conditions prone to change during the execution of the algorithm, and *meaningful* if its application is expectable to lead to a convergent solution. Moreover, the algorithm should have the least computational cost possible. Therefore, the 'brute force' algorithm involving repeatedly constructing and solving the governing system for each possible refinement increment and then comparing the new set of solutions with the previous ones should be avoided. Finally, the complex data created during the p-refinement process must be handled efficiently by an adequate data structure.

Let the solving system corresponding to some given orders of p-refinement be written in the generic form,

$$\begin{bmatrix} \boldsymbol{D} & -\boldsymbol{B} \\ -\boldsymbol{B}^* & \boldsymbol{0} \end{bmatrix} \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{pmatrix} = \begin{pmatrix} \boldsymbol{x}_{\boldsymbol{\Gamma}} \\ -\boldsymbol{y}_{\boldsymbol{\Gamma}} \end{pmatrix} \tag{86}$$

which is, of course, straightforwardly adaptable to any of the systems (20), (32), (50) or (60).

Let the approximation basis $\boldsymbol{Z}$ of an essential boundary be enriched with a new shape function (mode), $\overline{Z}$. For simplicity, the theoretical presentation in this section assumes that a single function is added to the basis, but more than one may actually be added in FreeHyTE.

The new mode has two effects on the solving system (86). First, it causes the emergence of a new line and column in the solving system, to which a new generalized variable, $\overline{y}$, and a new free term $\overline{y}_\Gamma$ are associated. Second, it causes the solutions of system (86) to change by quantities $\boldsymbol{\Delta x}$ and $\boldsymbol{\Delta y}$.

The solving system for the augmented boundary basis is thus,

$$\left[\begin{array}{cc:c} \boldsymbol{D} & -\boldsymbol{B} & -\overline{\boldsymbol{B}} \\ -\boldsymbol{B}^* & \mathbf{0} & \mathbf{0} \\ \hdashline -\overline{\boldsymbol{B}}^* & \mathbf{0} & 0 \end{array}\right] \left(\begin{array}{c} \boldsymbol{x}+\boldsymbol{\Delta x} \\ \boldsymbol{y}+\boldsymbol{\Delta y} \\ \hdashline \overline{y} \end{array}\right) = \left(\begin{array}{c} \boldsymbol{x}_{\boldsymbol{\Gamma}} \\ -\boldsymbol{y}_{\boldsymbol{\Gamma}} \\ \hdashline \overline{y}_{\Gamma} \end{array}\right) \tag{87}$$

Taking into account that solution vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ obtained in the previous refinement step satisfy equation (86), system (87) can be rewritten in its equivalent incremental form,

$$\left[\begin{array}{cc:c} \boldsymbol{D} & -\boldsymbol{B} & -\overline{\boldsymbol{B}} \\ -\boldsymbol{B}^* & \mathbf{0} & \mathbf{0} \\ \hdashline -\overline{\boldsymbol{B}}^* & \mathbf{0} & 0 \end{array}\right] \left(\begin{array}{c} \boldsymbol{\Delta x} \\ \boldsymbol{\Delta y} \\ \hdashline \overline{y} \end{array}\right) = \left(\begin{array}{c} \mathbf{0} \\ \mathbf{0} \\ \hdashline \overline{y}_{\Gamma} - \overline{\boldsymbol{B}}^* \boldsymbol{x} \end{array}\right) \tag{88}$$

Analysis of system (88) immediately reveals that the free term, $\overline{y}_{\Gamma} - \overline{\boldsymbol{B}}^* \boldsymbol{x}$, is an indicator of how much the solution from the previous iteration will change in the current iteration. From a mathematical perspective, it represents the residual of the previous solution measured in the direction of the new basis function, and reflects the error in the boundary balance equation that is corrected by the addition of the new function. If the residual is null, the addition of the new approximation function brings no improvement to the solution, and the added mode is said to be *spurious*. In FreeHyTE, a normalized form of the boundary balance residual is used, defined as,

$$R_{\Gamma} = \left\| \frac{\overline{y}_{\Gamma} - \overline{\boldsymbol{B}}^* \boldsymbol{x}}{L_{\Gamma}} \right\| \tag{89}$$

where L_{Γ} is the length of the refined boundary.

It is noted that only one line of coefficients has to be calculated to update the solving system from an iteration to the next, since all approximation functions from the previous iteration are reused in the current one. Also, the solution of system (88) takes advantage of that of the previous iteration, as the coefficient matrix is stored in its LU-factorized form at the end of each iteration to speed up computations in the next.

After the computation of the solution increment $\boldsymbol{\Delta x}$, the solution energy corresponding to the iteration i is computed using the quadratic norm,

$$E_i = \frac{1}{2} (\boldsymbol{x}+\boldsymbol{\Delta x})^* \; \boldsymbol{D} \; (\boldsymbol{x}+\boldsymbol{\Delta x}) \tag{90}$$

Finally, the energy residual between two successive iterations is computed as,

$$R_E = \left\| \frac{E_i - E_{i-1}}{E_{i-1}} \right\| \tag{91}$$

6.3.2. Refinement criteria

The *p*-adaptive algorithm implemented in FreeHyTE refines iteratively the essential boundaries of the mesh. A global and a local selection criteria support the choice of the boundary (or boundaries) to refine at each iteration. The refinement of the domain bases is a consequence of the boundary refinement, and aimed at correcting the indeterminacy and numerical stability of the solving system.

In some situations (e.g. symmetric structures), more than one boundary yield the same values of the selection criterion (SC). In order to avoid choosing the boundary to refine based on some floating point error, FreeHyTE requires user to specify a selection tolerance ($tol_{SC} \leq 1.0$), such that all essential boundaries that present selection criteria,

$$SC \geq tol_{SC} \cdot \max(SC) \tag{92}$$

are selected for refinement. In inequality (92), $\max(SC)$ is the maximum selection criterion for all boundaries. As explained before, this presentation assumes that a single boundary is selected for refinement at the current iteration, for simplicity.

Boundary refinement using the energy variation criterion. The energy variation criterion selects for refinement the boundary that causes the largest relative variation R_E of the solution energy. The energy variation is computed according to expression (91). This is a global refinement criterion, as the solution energy is a global quantity. It is important to note that since neither the state, nor the flux hybrid-Trefftz formulations produce kinematically or statically admissible solutions, the monotonous convergence of the energy that typify such solutions does generally not occur in FreeHyTE. Instead, the energy recovered by the model oscillates in the vicinity of the exact solution as it converges [34]. This means that the maximum energy variation in one iteration is not necessarily smaller than in the previous iteration, although the overall tendency should be to decrease as the solution converges.

Boundary refinement using the boundary balance residual criterion. The boundary balance residual criterion selects for refinement the boundary that causes the largest balance residual in the right hand side of system (88). The boundary balance residual is computed according to expression (89). This is a local refinement criterion, essentially aiming at maximizing the improvement of the boundary balance caused by the added mode. As for the energy variation criterion, the decrease in the boundary balance residual cannot be guaranteed from one iteration to the next, although it should occur in an overall sense.

Domain refinement criteria. The domain refinements are always triggered by boundary refinements. The domain refinement criteria are aimed at avoiding the over-constrainment of the domain bases and the instability of the solving system that could be caused by the addition of boundary modes.

Regarding the over-constrainment of the domain bases, the finite elements are refined whenever the addition of boundary modes would render them statically or kinematically (over-)determinate, that is, whenever one of the indeterminacy conditions (61) to (64) is infringed (see Section 3.4.2). Moreover, to avoid the local over-constrainment of the domain bases, the elements are so refined as to ensure that the orders of their bases are larger than the orders of all neighbouring essential boundaries.

Regarding the instability of the solving system, whenever the refinement of a boundary basis causes the emergence of singular value outliers, in the sense defined in Section 6.2.2, new modes are added to the finite elements adjacent to that boundary until all outliers vanish. It is noted that, in adaptive processes, FreeHyTE uses a very strict definition of 'outliers', making it much more prone to detecting false positives than false negatives. False positive outliers are singular values that are not detrimental to the stability of the solving system, but get treated as if they were. Conversely, false negative outliers are singular values that damage the stability of the solutions, but as not identified as such. The (strong) bias towards false positives is aimed at reducing the odds of unstable solutions occurring in the iterative process, since they may cause the following iteration to diverge. This reflects the user-friendly implementation paradigm common to all FreeHyTE modules, as it aims at yielding meaningful solutions to inexperienced users, even though imperfect, as opposed to risking the divergence of the solution by trying to refine it as much as possible. The drawback of this approach is that the iterative process may exit because of excessive singular value outliers (see Section 6.3.3), even if their presence is not detrimental to the solution.

6.3.3. Stopping criteria

Three types of stopping criteria are implemented to control the execution of the p-refinement algorithm: the convergence criteria, the maximum order criterion and the system instability criteria. A brief description of each stopping criterion follows.

Convergence criteria. These are the most desirable stopping criteria, essentially meaning that the solution convergence has been reached within the desired accuracy. The user must choose between the energy convergence criterion and the boundary residual convergence criterion. Besides these criteria, the execution stops if all available refinements fail to improve the solution.

- *Energy convergence criterion.* The convergence in energy is achieved when the average energy variation, computed according to definition (91) over the last n iterations is less than a convergence tolerance tol_{conv},

  $$\epsilon_E = \frac{1}{n}\left(\sum_{a=i-n+1}^{i} R_E^a\right) \leq tol_{conv} \tag{93}$$

  where i is the current iteration. The energy variation is averaged over the last n iterations to avoid spurious early convergence. Both n and tol_{conv} need to be specified by the user (in GUI 4).

- *Boundary residual convergence criterion.* The boundary residual convergence is achieved when the boundary residual of the refined boundary, computed according to definition (89) over the last n iterations and normalized to its value in the first iteration (R_Γ^1), is less than a convergence tolerance tol_{conv},

  $$\epsilon_\Gamma = \frac{1}{n \cdot R_\Gamma^1}\left(\sum_{a=i-n+1}^{i} R_\Gamma^a\right) \leq tol_{conv} \tag{94}$$

  Since the boundary residual of the side selected for refinement is the largest of all sides at each iteration, this criterion stops the execution when the maximum boundary residual has consistently reached values inferior to $tol_{conv} \cdot R_\Gamma^1$.

- *No improvement convergence criterion.* In rare cases, all available refinements may fail to improve the solution. This happens when the reduction of the boundary residual (89) falls short of some (user-defined) numerical zero threshold on all essential boundaries. This is typically the case when the exact solution has been achieved by the algorithm, so further refinement is futile. However, it may also occur, in principle, in situations where all available modes are orthogonal to the error on the respective boundaries.

Maximum order criterion. Increasing the order of p-refinement is the best way to improve the quality of the solution when using hybrid-Trefftz finite elements. However, our testing experience has shown that there is only so much accuracy that one can gain from increasing the refinement of the bases [11]. After this tipping point is reached, further increasing the order of p-refinement results in a loss, rather than a gain in accuracy, because of the difficulties associated with the numerical handling of very high order functions and the ill-conditioning of the solving system. In order to avoid the numerical problems caused by exceedingly refined approximation functions, user is required to define a maximum allowable order. If this order is reached, the execution ends and the most refined solution is output. This situation is typically reached if the convergence threshold (tol_{conv}) is set too small, but may also mean that the problem is physically ill posed, as is the case, for instance, when discontinuous state or flux fields are enforced of the boundaries.

System instability criteria. In p-adaptive analyses, FreeHyTE uses the singular value analysis described in Section 6.2.2 to diagnose the numerical stability of the solving system, rather then the more straightforward assessment of the condition number. The algorithm does not accept systems where outliers are encountered, and automatically increases the refinement of the elements adjacent to the refined boundary in an attempt to improve the stability of the system (see Section 6.3.2). The numerical instability of the solving system can lead to a forced exit under the following circumstances:

- if outlier singular values are detected in the first analysis of an automatic p-refinement process. Before the iterative process starts, a preliminary analysis of the structure is made using the starting refinement orders supplied by the user. If singular values outliers are detected in this analysis, the iterative process does not even start and FreeHyTE exits with an error message.

- if outlier singular values continue to be detected after a user-specified number of successive attempts to improve the stability of the system. If this number is reached, the execution ends and the last stable (i.e. outlier-free) solution is output.

6.3.4. Flowchart of the algorithm

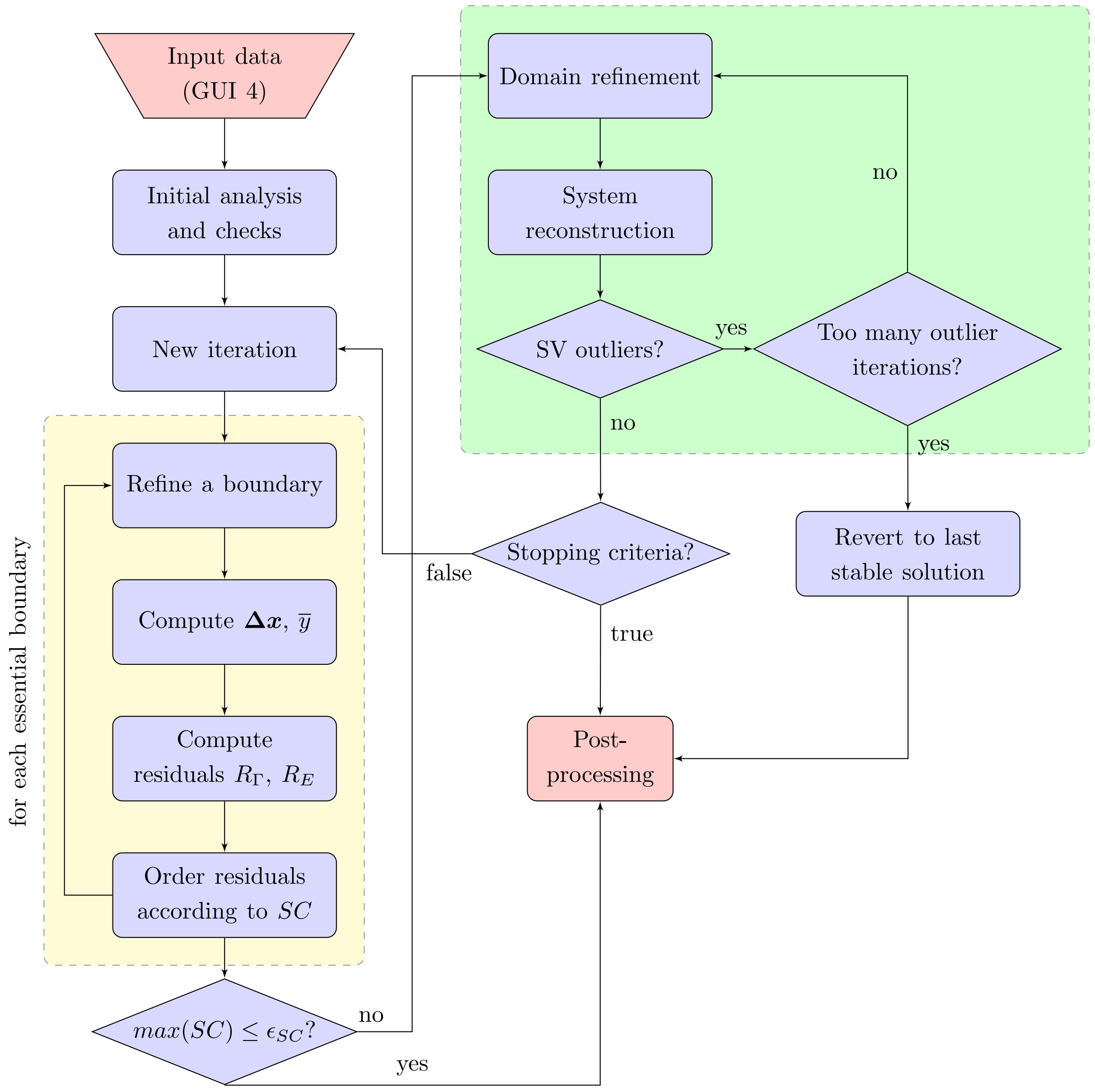
Because of the various checks, selection, and stopping criteria involved, the adaptive p-refinement algorithm is rather complex. An attempt to its systematization is made in Figure 12, where a flowchart of the algorithm is presented (note that SC stands for Selection Criterion, and SV stands for Singular Value). A few comments on some of the steps is also given below.

Input data. The data supplied by the user to steer the p-adaptive process is described in more detailed in Section 6.3.5.

Initial analysis and checks. Before proceeding to the iterative p-adaptive process, an initial analysis is attempted with the initial orders of the bases, supplied by the user in GUI 1 (Section 5.1.1). If elements with negative indeterminacy numbers are found (Section 3.4.2), the orders of the domain bases in these elements are increased as necessary. If singular values outliers are found in the first run, FreeHyTE exits with an error message.

The iterative process. The iterative process consists of three phases: the boundary refinement (yellow zone in Figure 12), the domain refinement (green zone in Figure 12), and the check of the stopping criteria.

Boundary refinement zone. The boundary refinement phase is aimed at selecting the boundary to refine in the current step, by maximizing the value of the selection criterion supplied by the user (Section 6.3.2). This value is computed for each essential boundary of the mesh, by incrementing its basis and solving system (88), using a procedure that does not call for the reconstruction of the pre-existing part. This way, only the vector $\overline{\boldsymbol{B}}$ and term $\left(\overline{y}_\Gamma - \overline{\boldsymbol{B}}^* \boldsymbol{x}\right)$ need to be computed anew for each essential boundary. The sides with selection criteria within the selection tolerance from the maximum value are selected for refinement, according to expression (92).



Input data (GUI 4)
Initial analysis and checks
New iteration
Refine a boundary
Compute $\boldsymbol{\Delta x}$, $\overline{y}$
Compute residuals R_Γ, R_E
Order residuals according to SC
for each essential boundary
$max(SC) \leq \epsilon_{SC}$?
no
yes
Domain refinement
System reconstruction
SV outliers?
yes
no
Too many outlier iterations?
no
yes
Stopping criteria?
false
true
Revert to last stable solution
Post-processing

Figure 12: p-Adaptive algorithm in FreeHyTE.

No improvement convergence check. As shown in Section 6.3.3, if the maximum value of the selection criterion is less then a numerical zero threshold (ϵ_{SC}), the algorithm assumes that the exact solution of the problem was reached and launches the post-processing. Otherwise, it proceeds to the domain refinement.

Domain refinement zone. Upon entry in this zone, the indeterminacy criteria are checked and elements refined as necessary. After the solving system is reconstructed, the singular value analysis described in Section 6.2.2 is performed and if outliers are found, the elements adjacent to the refined boundaries are iteratively refined. If this process fails to eliminate the singular value outliers in a maximum number of iteration prescribed by the user, FreeHyTE reverts to the last stable solution and launches the post-processing. If no (more) outliers are found, the algorithm proceeds to the check of the stopping criteria.

Stopping criteria. The stopping criteria checked at this stage are described in Section 6.3.3. If any of those is true, FreeHyTE launches the post-processing. If not, a new iteration is started.

6.3.5. Input data and data structure

Input data. The user data for the p-adaptive process is input using GUI 4 (Section 5.1.4). There, the user must choose the selection and stopping criteria, and input the values of the parameters that steer the algorithm. These parameters are listed in Table 5, along with their default/recommended values. Their use in the algorithm is documented in detail in Sections 6.3.1 to 6.3.4.

Data structure. In p-adaptive analyses, a specific data structure (`List`) is designed to handle the information during the iterative process. This data structure contains two types of fields: the local fields, that contain variables that are overwritten at every iteration, and the global fields, that store the key information regarding each iteration and are therefore not overwritten. The global variables are mainly used to log the decisions and operations made during the process.

Brief descriptions of the fields of the `List` data structure are given in Tables 6 (local variables) and 7 (global variables), with more details to be found in Appendix B.2.

7. Post-processing

This section describes the reconstruction of the state and flux solutions after the processing phase is completed, including the recovery of the flux-free

Description	Notation	Default value
selection tolerance for choosing multiple boundaries for refinement	tol_{SC}	0.99
selection tolerance (energy variation criterion)	tol_{conv}	10^{-4}
selection tolerance (boundary balance residual criterion)	tol_{conv}	10^{-2}
numerical zero threshold	ϵ_{SC}	10^{-12}
minimum number of iterations that must be performed to avoid spurious early convergence	-	5
number of iterations to compute the averages required by the stopping criteria	n	3
maximum number of consecutive domain refinement iterations with singular value outliers	-	10
maximum order for the approximation bases	-	20

Table 5: User-defined parameters for the p-refinement process

part of the state solution. The typical output data provided by FreeHyTE is also presented here.

7.1. Computation of the solution

7.1.1. State model

Following the solution of systems (20) or (32), the state solution at the end of the current time step (or final solutions, in static or steady-state analyses) is recovered using approximation (7). Expression (5) is also used in problems where an exact particular solution can be found.

If desired, the flux field in the domain of each element can also be recovered by enforcing explicitly the state-flux relation on the state approximations (5) or (7). While obviously problem-dependent, the state-flux relation involves the derivation of the former, so the domain flux solution results unique.

Multiple estimates of the flux solution can be obtained, however, on the essential boundaries of the mesh, namely by projecting the flux solution from the domain, or by using the independent boundary flux approximation (8). The two solutions should be consistent upon convergence, so their difference may be used as a convergence indicator. However, the difference does not

`List.Edge`	Lists the indices of the essential boundaries and the corresponding residuals R_E and R_Γ
`List.EdgesToRefine`	Lists the indices of the edges to refine according to the selection tolerance tol_{SC}
`List.SpuriousEdges`	Lists the edges where the addition of a mode brings no improvement to the solution, $R_\Gamma = 0$
`List.SpuriousEdgesToRefine`	Intersection of the `List.EdgesToRefine` and `List.SpuriousEdges` sets. If not empty, it means that all available refinements fail to reduce the boundary residual (Section 6.3.3)
`List.LoopsToRefine`	Lists the elements selected for refinement according to the criteria presented in Section 6.3.2

Table 6: Local variables in the `List` data structure

make for an efficient selection criterion of the boundary to be refined in a p-adaptive refinement process (see Section 6.3), since the flux balance is not enforced separately on each boundary (as is the state field continuity), but rather 'smeared out' in the domain equation (11). The consequence of this is that, while the refinement of a boundary does improve the *global* flux balance over the finite element, it does not necessarily guarantee its local improvement on that essential boundary in particular.

In multi-step, transient analyses, the solutions at the end of each time step are used to construct the source fields $\boldsymbol{f_0}$ for the next, according to definitions (71) and (80), for first and second order problems, respectively. The solutions are computed and stored in the Gauss quadrature points, to be straightforwardly usable in the next time step.

7.1.2. Flux model

For the flux model, the domain flux estimates at the end of the time step are recovered using definitions (33) or (36), depending on whether an analytical particular solution can be found or not. The weights of the flux bases correspond to the solutions of systems (50) or (60).

Dependent state field solutions may not be immediately recoverable from the flux field approximation, however, since they may require some (general-

`List.EnergyIt`	Lists the solution energy for each iteration
`List.EnergyVariationIt`	Lists the relative energy variation (91) for each iteration
`List.ErrorEdgeNormIt`	Lists the boundary balance residual (89) of the boundary selected for refinement for each iteration
`List.ErrorEdgeVariationIt`	Lists the boundary balance residual `List.ErrorEdgeNormIt`, normalized to its value in the first iteration
`List.RefinedEdgesIt`	Lists the edges selected for refinement at each iteration
`List.EdgesOrderIt`	Lists the orders of approximation for all edges of the mesh, at each iteration
`List.RefinedLoopsIt`	Lists the elements selected for refinement at each iteration
`List.LoopsOrderIt`	Lists the orders of approximation for all elements of the mesh, at each iteration
`List.DoF_It`	Lists the total number of degrees of freedom (or the dimension of the solving system) at each iteration
`List.BetaIt`	For the state (flux) model, lists the kinematic (static) indeterminacy number at each iteration

Table 7: Global variables in the `List` data structure

ized) integration process, and thus the flux-free part of the state field cannot be accounted for. This is typical, for instance, of static structural problems, where the displacement field cannot be fully recovered from the stress field, since it lacks information on the rigid body displacements of the element, but this issue does not occur, for instance, in harmonic problems.

When flux-free state modes exist, their weights are computed in the post-processing phase, by enforcing the boundary conditions (3) on the exterior Dirichlet sides, along with the state field continuity condition on the interior sides of the mesh,

$$\int \boldsymbol{W}^{*} \boldsymbol{U}_{\boldsymbol{r}} \, d\Gamma_u \, \boldsymbol{x}_{\boldsymbol{r}} = \int \boldsymbol{W}^{*} \left(\boldsymbol{u}_{\boldsymbol{\Gamma}} - \overline{\boldsymbol{u}} \right) \, d\Gamma_u \tag{95}$$

$$\int \boldsymbol{W}^{*} \boldsymbol{U}_{\boldsymbol{r}}^{i} \, d\Gamma_i \, \boldsymbol{x}_{\boldsymbol{r}}^{i} - \int \boldsymbol{W}^{*} \boldsymbol{U}_{\boldsymbol{r}}^{j} \, d\Gamma_i \, \boldsymbol{x}_{\boldsymbol{r}}^{j} = \int \boldsymbol{W}^{*} \left(\overline{\boldsymbol{u}}^{j} - \overline{\boldsymbol{u}}^{i} \right) \, d\Gamma_i \tag{96}$$

where $\boldsymbol{W}$ is an arbitrary set of test functions, $\boldsymbol{U}_{\boldsymbol{r}}$ and $\boldsymbol{x}_{\boldsymbol{r}}$ are the flux-free state modes and their corresponding weights, and $\overline{\boldsymbol{u}} = \boldsymbol{U}_{\boldsymbol{c}} \, \boldsymbol{x}_{\boldsymbol{c}} + \boldsymbol{U}_{\boldsymbol{p}} \, \boldsymbol{x}_{\boldsymbol{p}}$ is the displacement solution without including the flux-free modes, as known from the processing phase. In expression (96), superscripts i and j are the generic indices of two elements that share the interior side Γ_i.

Boundary equations of type (95) and (96) lead to a (generally over-determined) system of equations, which is solved using a least-square solver for the weights $\boldsymbol{x}_{\boldsymbol{r}}$.

Besides the option of projecting on the boundary the domain state field computed according to the procedure described above, an independent estimate of the boundary state field can be obtained from the definition (37). As for the boundary flux case in the previous section, the difference between the boundary state fields computed using the domain and boundary approximations can be used as an convergence indicator, but not as a selection criterion in adaptive p-refinement analyses.

7.2. Output

FreeHyTE produces three types of output: graphical representation of the solutions, solution files, and convergence plots (only available in adaptive p-refinement analyses).

Graphical representation of the solutions. Graphical representation of the solutions is available in all modules of FreeHyTE. It typically consists in colour map plots of the state and flux fields, but it may also contain additional information, where relevant, as the deformed shape of a structure, in structural mechanics applications (Figure 13), or the final orders of the domain and boundary bases in adaptive p-refinement runs.

In transient dynamic applications, user is required to input the number of time steps the solutions should be plotted at, in order to avoid tedious plots of solutions at every time step. This information is supplied by the user in the *Algorithmic definitions* zone of GUI 1 (see Figure 3 and Section (5.1.1)).

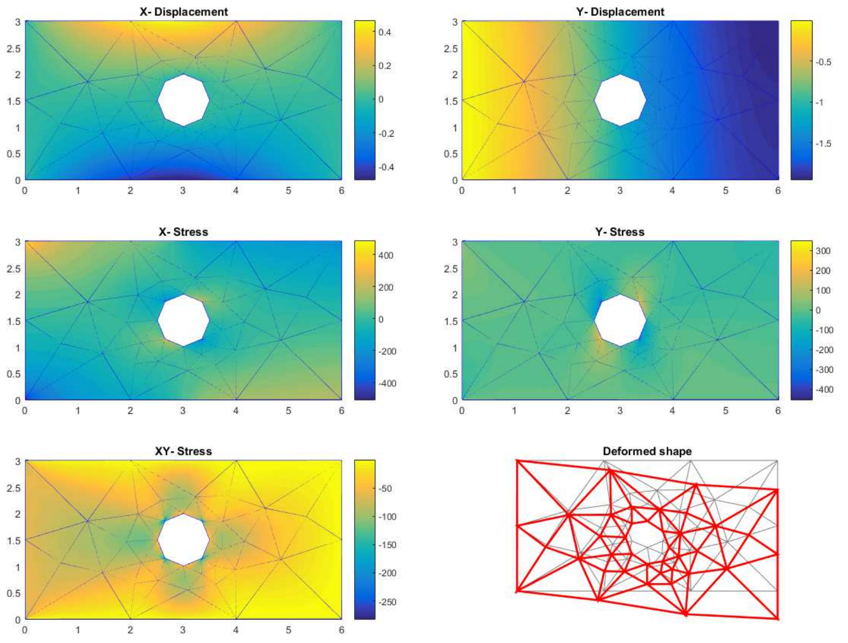


Figure 13: Sample graphical output from a structural mechanics application.

Solution files. Lists of the solution fields in some grid of points (typically Gauss quadratures) in the domains of the elements are also produced by FreeHyTE, if requested by the user in GUI 1 (Figure 3). Each list has as many lines as sampling points. On each line, the global Cartesian coordinates of the sampling point are followed by as many (tab-separated) values as needed to completely describe the state and flux fields. Therefore, complex quantities are allocated two result fields, for their real and imaginary parts, respectively.

In transient applications, separate solution files are produced at every time step. Likewise, in adaptive *p*-refinement analyses, the results are stored at each iteration.

Header rows are added to render the solution files directly compatible with the visualization software Tecplot, but their contents are, of course, compatible with most mainstream visualization packages.

Convergence plots. In adaptive *p*-refinement analyses, FreeHyTE also plots the convergence graphs, as shown in Figure 14. Each of the four plots are briefly described next.

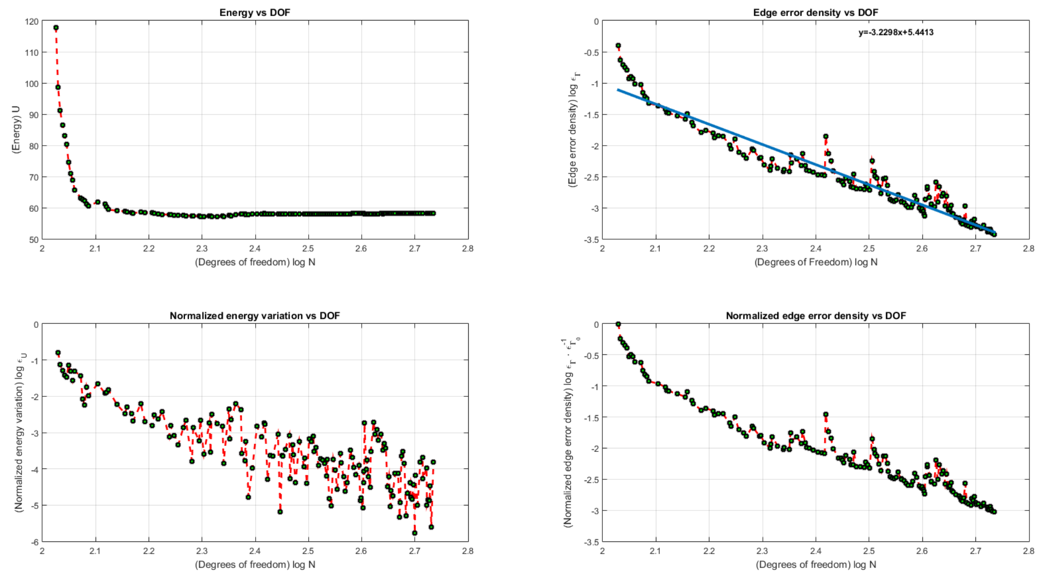


Figure 14: Sample convergence plots in a p-adaptive analysis.

- *Energy vs DOF.* Plots the variation of the solution energy (90) with the total number of degrees of freedom of the model. Each dot corresponds to an iteration. The abscissa is logarithmic. The stabilization of this quantity is an important indicator of the convergence;

- *Normalized energy variation norm vs DOF.* Plots the energy residual (91) between two successive iterations. Both axes are logarithmic. This quantity is used both as a refinement and as a stopping criterion, as discussed in Sections 6.3.2 and 6.3.3;

- *Edge error density vs DOF.* Plots the maximum boundary balance residual (89) at each iteration against the total number of degrees of freedom of the model. Both axes are logarithmic. This quantity is used as a selection criterion (see Section 6.3.2). The blue line in the plot is a linear fit of the boundary residual decay and the function $y(x)$ is its mathematical expression;

- *Normalized edge error density vs DOF.* This plot contains the same information as the previous plot, but the maximum error density on the boundary is normalized to its value in the first iteration. The plotted quantity is used as a stopping criterion (see Section 6.3.3).

8. Funding source

This research was supported by Fundação para a Ciência e a Tecnologia through grant SFRH/BPD/87317/2012.

Bibliography

[1] Ruoff G. Die praktische Berechnung der Kombination der Trefftzschen Methode und bei flachen Schalen. In: Finite Elemente in der Statik, Berlin;1973,p.242-59.

[2] Jirousek J, Teodorescu P. Large finite elements method for the solution of problems in the theory of elasticity. Comp Struct 1982;15:575-87.

[3] Herrera I. Boundary Methods - an Algebraic Theory. Pitman Advanced Publishing Program. Boston, London, Melbourne;1984.

[4] Piltner R. Special finite elements with holes and internal cracks. Int J Numer Methods Eng 1985;21:1471-85.

[5] Qin QH. Postbuckling analysis of thin plates by a hybrid Trefftz finite element method. Comput Method Appl M 1995;128:123-36.

[6] Freitas JAT, Moldovan ID, Cismaşiu C. Hybrid-Trefftz displacement element for bounded and unbounded poroelastic media. Comput Mech 2011;48:659-73.

[7] Cheung YK, Jin WG, Zienkiewicz OC. Direct solution procedure for solution of harmonic problems using complete, non-singular, Trefftz functions. Commun Appl Numer M 1989;5(3):159-69.

[8] Piltner R. The application of a complex 3-dimensional elasticity solution representation for the analysis of a thick rectangular plate. Acta Mech 1988;75:77-91.

[9] Moldovan ID, Cao DT, Freitas JAT. Elastic wave propagation in unsaturated porous media using hybrid-Trefftz stress element. Int J Numer Methods Eng 2014;97:32-67.

[10] Qin QH, Wang H. MATLAB and C Programming for Trefftz Finite Element Methods. CRC Press. Boca Raton, London, New York;2009.

[11] Moldovan ID, Freitas JAT. Hybrid-Trefftz displacement and stress elements for bounded poroelasticity problems. Comp Geo 2012;42:129-44.

[12] Beer G, Smith I, Duenser C. The Boundary Element Method with Programming: For Engineers and Scientists. Springer;2008.

[13] Free Software Foundation. GNU General Public License, https://www.gnu.org/licenses/gpl.html; 2007 [accessed 01.01.2017].

[14] FreeHyTE Release Page, https:/ /www.sites.google.com/site/ionutdmoldovan/freehyte; 2016 [accessed 01.01.2017].

[15] Cao C, Qin QH, Yu A. A novel boundary integral based finite element method for 2D and 3D thermo-elasticity problems. J Therm Stresses 2012;35:849-76.

[16] Leconte N, Langrand B, Markiewicz E. On some features of a plate hybrid-Trefftz displacement element containing a hole. Finite Elem Anal Des 2010;46:819-28.

[17] Soh AK, Long ZF. A high precision element with a central circular hole. Int J Solids Struct 1999;36:5485-97.

[18] Cismaşiu C. The hybrid-Trefftz displacement element for static and dynamic structural analysis problems. PhD thesis, Universidade Técnica de Lisboa;2000.

[19] Qin QH. Formulation of hybrid Trefftz finite element method for elastoplasticity. Appl Math Model 2005;29:235-52.

[20] Moldovan ID. A new particular solution strategy for hyperbolic boundary value problems using hybrid-Trefftz displacement elements. Int J Numer Methods Eng 2015;102:1293-315.

[21] Moldovan ID, Radu L. Trefftz-based Dual Reciprocity Method for hyperbolic boundary value problems. Int J Numer Methods Eng 2016;106:1043-70.

[22] Qin QH. Trefftz Finite Element Method and Its Applications. Appl Mech Rev 2005;58(5):316-37.

[23] Cho HA, Golberg MA, Muleshkov AS, Li X. Trefftz Methods for Time Dependent Partial Differential Equations. CMC Comput Mater Con 2004;1(1):1-37.

[24] Moldovan ID, Cao DT, Freitas JAT. Hybrid-Trefftz elements for biphasic elastostatics. Finite Elem Anal Des 2013;66:68-82.

[25] Nardini D, Brebbia CA. A new approach to free vibration analysis using boundary elements. In: CA Brebbia, Ed. Boundary Element Method in Engineering. Springer-Verlag;1982.

[26] Moldovan ID. A new approach to non-homogeneous hyperbolic boundary value problems using hybrid-Trefftz stress finite elements. Eng Anal Bound Elem 2016;69:57-71.

[27] Freitas JAT. Formulation of elastostatic hybrid-Trefftz stress elements. Comput Method Appl M 1998;153:127-51.

[28] Freitas JAT. Hybrid-Trefftz displacement and stress elements for elastodynamic analysis in the frequency domain. Comp Assisted Meth Eng Sci 1997;4:345-68.

[29] MathWorks. Partial Differential Equation Toolbox, http://www.mathworks.com/help/pde/index.html; 2016 [accessed 01.01.2017].

[30] Hughes TJR. Analysis of transient algorithms with particular reference to stability behaviour. In: Computational methods for transient analysis. Elsevier Science Publishers;1983.

[31] Tamma KK, Zhou X, Sha D. The time dimension: A theory towards the evolution, classification, characterisation and design of computational algorithms for transient/dynamic applications. Arch Comput Method E 2000;7(2):67-290.

[32] Hansen PC. Truncated singular value decomposition solutions to discrete ill-posed problems with ill-determined numerical rank. SIAM J Sci Stat Comput 1990;11(3):503-18.

[33] Freitas JAT, Cismasiu C. Adaptive *p*-refinement of hybrid-Trefftz finite element solutions. Finite Elem Anal Des 2003;39(11):1095-121.

[34] Moldovan ID, Cao DT, Freitas JAT. Hybrid-Trefftz displacement finite elements for elastic unsaturated soils. Int J Comp Meth-Sing 2014;11(2):1342005.

Appendix A. Systems of reference in FreeHyTE

The model definition takes place in the global Cartesian referential (X, Y), and in the Cartesian, normal-tangential referentials (n_i, t_i) associated to each exterior side of the mesh. For the construction of the domain approximation bases, two local referentials, (xy) and (r, θ) are defined in each finite element. The boundary bases are constructed using the side coordinate s. These referentials are presented in Figure A.1 and shortly described next. The local referentials are presented in a sample finite element, to keep the figure legible.

- *Global* (X, Y) *referential.* Cartesian referential, used by the user to define the geometry of the model, the source terms and the initial conditions. FreeHyTE uses the global referential to store geometrical data and to present the solutions in the post-processing phase.

- *Local* (x, y) *referential.* Cartesian referential, associated to each finite element of the mesh. Has its origin in the barycenter of the element and its axes are parallel to the global X and Y directions. It is hidden from the user. FreeHyTE uses the local Cartesian referential to store geometrical data and for the graphical output.

- *Local* (r, θ) *referential.* Polar referential, associated to each element of the mesh. Has its origin in the barycenter of the element and its θ axis measured from the local x axis. It is invisible to users and used to compute the domain approximation functions.

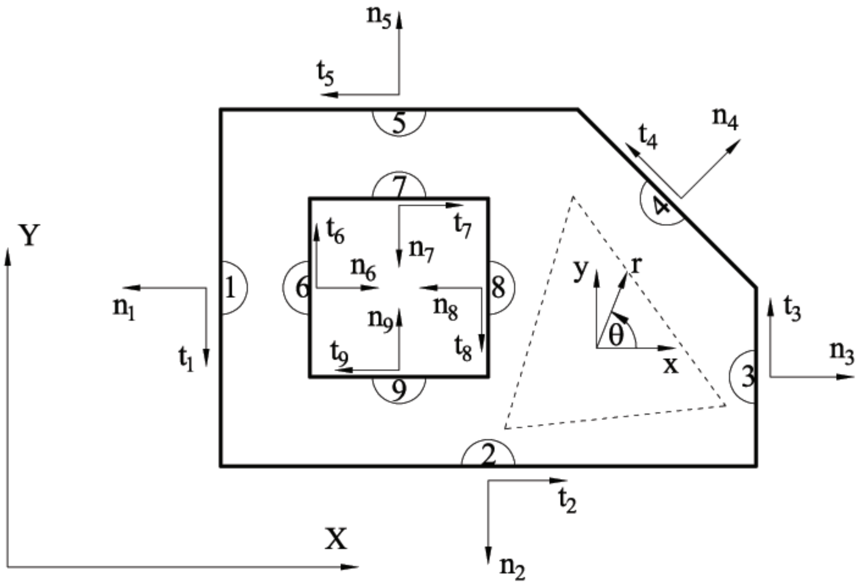


Figure A.1: Finite elements, Neumann, Dirichlet and interior boundaries.

- *Local boundary* (n, t) *referential.* Cartesian referential, associated to each side of the mesh. Has its origin in the center of the edge. Its n axis is normal to the edge and points outwards from the element. Its t axis is oriented such as to encircle encircle the structure in a counter clockwise direction on the outside and in a clockwise direction on the interior openings (see Figure A.1). It is used by the user to define the boundary conditions.

- *Local side* (s) *referential.* Side, one-dimensional referential, associated to each side of the mesh. Has its origin in the center of the edge and its orientation is identical to that of the t axis of the local boundary referential. The s coordinate maps t to the $[-1, 1]$ interval. It is invisible to users. Used by FreeHyTE to perform the numerical integrations on the boundaries.

Appendix B. Data structures in FreeHyTE

To assist potential developers, a brief description of the typical data structures used in FreeHyTE is given here. They include the mesh and topological data (see Section 5.3.1), the three main data structures that store the information regarding the edges, finite elements and boundary conditions (Section 5.3.2), and the data structure that stores the data generated by the p-adaptive process (Section 6.3.5). For information regarding the structural definitions that lead to these data structures, the reader is referred to the user's manuals of the FreeHyTE modules [14].

Appendix B.1. Mesh and topological data

The following mesh and topological data are returned by the (regular and non-regular) mesh generators:

- `Nodes`: Matrix with as many lines as nodes and two columns. Each line stores the global X and Y coordinates of the respective node.

- `Loops_nodes`: Matrix with as many lines as finite elements, and as many columns as the nodes of each element (three for non-regular meshes and four for regular meshes). Each line stores the indices of the nodes of the respective element, listed in a clockwise order.

- `Edges_nodes`: Matrix with as many lines as edges and two columns. Each line contains the indices of the initial and final nodes of the respective edge.

- `Edges_loops`: Matrix with as many lines as edges and two columns. Each line contains the indices of the finite elements located left and right of the respective edge. The orientation of the edges is chosen such as to always have a finite element on the left side of the edge. Exterior edges have no right element and a null entry is listed in matrix `Edges_loops`.

- `Loops_edges`: Matrix with as many lines as finite elements, and as many columns as the edges of each element (three for non-regular meshes and four for regular meshes). Each line contains the indices of the edges of the respective finite element, listed in no particular order.

As specified in Section 5.3.1, mesh configurations different from those automatically generated by FreeHyTE are also acceptable, if the user provides directly the data listed in this section.

Appendix B.2. Data structures

***`Edges`** data structure.* The `Edges` data structure collects information regarding the edges of the mesh. Its members are vectors or matrices with as many lines as edges. They store the following information:

- `Edges.nini`: Vector, lists the index of the initial node of the respective edge.
- `Edges.nfin`: Vector, lists the index of the final node of the respective edge.
- `Edges.lleft`: Vector, lists the index of the finite element on the left of the respective edge.
- `Edges.lright`: Vector, lists the index of the finite element on the right of the respective edge.
- `Edges.type`: Vector of characters. Each term lists 'N' or 'D' to designate a Neumann or Dirichlet edge, respectively. It is noted that the internal boundaries are listed as Neumann boundaries in the flux models and as Dirichlet boundaries in the state models.
- `Edges.oders`: Vector, lists the order of the (Chebyshev) approximation basis of the respective edge.
- `Edges.insert`: Vector or matrix, depending on the type of application. Lists the insertion points of the boundary block(s) corresponding to the current edge in the solving system. The structure of the solving system is discussed in Section 6.2.1 and illustrated in Figure 10.
- `Edges.dim`: Vector or matrix, depending on the type of application. Lists the dimension of the boundary block(s) corresponding to the current edge in the solving system. The reader is again referred to Section 6.2.1 and Figure 10.

***`Loops`** data structure.* The `Loops` data structure collects information regarding the finite elements. Its members are vectors or matrices with as many lines as elements. They store the following information:

- `Loops.nodes`: Matrix, identical to the `Loops_nodes` matrix returned by the mesh generators.

- `Loops.edges`: Matrix, identical to the `Loops_edges` matrix returned by the mesh generators.
- `Loops.center`: Matrix with two columns. Each line lists the global X and Y coordinates of the barycenter of the respective finite element.
- `Loops.area`: Vector, lists the area of each finite element.
- `Loops.order`: Vector or matrix, depending on the type of application. Each line lists the order(s) of the approximation basis(es) defined in the domain of the respective element. Multiple orders are typical of non-homogeneous problems, where the particular and complementary solutions are approximated using distinct bases (Section 3).
- `Loops.insert`: Vector or matrix, depending on the type of application. Each line lists the insertion point(s) of the $\boldsymbol{D_{ij}}$ block(s) corresponding to the current element (Section 3) in the solving system. The structure of the solving system is discussed in Section 6.2.1 and illustrated in Figure 10.
- `Loops.dim`: Vector or matrix, depending on the type of application. Each line lists the dimension(s) of the $\boldsymbol{D_{ij}}$ block(s) corresponding to the current element in the solving system (see Section 6.2.1 and Figure 10.
- `Loops.materials`: Matrix with as many columns as necessary to define the (application-specific) physical properties of the medium. The physical properties may be different between elements.

`BConds` data structure. The `BConds` data structure collects information regarding the boundary conditions. Its members are *cell* arrays with as many lines as the external boundaries of the structure. They store the following information:

- `BConds.Dirichlet`: Lists the values of the enforced state fields on each Dirichlet boundary and NaN (not a number) on the Neumann boundaries. Boundary conditions are defined as explained in Section 5.1.3 and, with more detail, in the user's manuals of the FreeHyTE modules [14].
- `BConds.Neumann`: Lists the values of the enforced flux fields on each Neumann boundary and NaN on the Dirichlet boundaries.

`List` data structure. The `List` data structure collects information regarding the adaptive p-refinement processes. Its members store the following information:

- `List.Edge`: Matrix with as many lines as essential boundaries and three columns. Each line stores the index of the essential boundary it corresponds to and the values of the two boundary refinement criteria presented in Section 6.3.2.

- `List.EdgesToRefine`: Vector, lists the edges selected for refinement according to the selection criterion (92).

- `List.SpuriousEdges`: Vector, lists the edges with boundary balance residuals (89) smaller than the numerical zero threshold ϵ_{SC} (see Table 5).

- `List.SpurEdgesToRefine`: Vector, lists the edges that are present in both `List.EdgesToRefine` and `List.SpuriousEdges` vectors. It should be empty at all times and is used to identify numerical inconsistency in the execution of the adaptive p-refinement procedure.

- `List.LoopsToRefine`: Vector, lists the finite elements selected for refinement according to the domain refinement criteria presented in Section 6.3.2.

- `List.EnergyIt`: Vector with as many terms as the number of iterations. Stores the solution energy (90) at each iteration and is used to compute the energy residual (91). At the end of the iterative process, its variation is plotted in the first convergence plot presented in Figure 14.

- `List.EnergyVariationIt`: Vector with as many terms as the number of iterations. Stores the values of the energy residual (91) at each iteration. At the end of the iterative process, its variation is plotted in the third convergence plot presented in Figure 14.

- `List.ErrorEdgeNormIt`: Vector with as many terms as the number of iterations. Stores the values of the largest boundary balance residual (89) at each iteration. At the end of the iterative process, its variation is plotted in the second convergence plot presented in Figure 14.

- `List.ErrorEdgeVariationIt`: Vector with as many terms as the number of iterations. Stores the values of the largest boundary balance residual (89) at each iteration, normalized to the corresponding value at the first iteration. At the end of the iterative process, its variation is plotted in the fourth convergence plot presented in Figure 14.

- `List.GDL_It`: Vector with as many terms as the number of iterations. Stores the total number of degrees of freedom (that is, the dimension of the solving system), at each iteration.

- `List.RefinedEdgesIt`: Vector of *cells* with as many terms as the number of iterations. Each entry lists the boundaries selected for refinement at each iteration.

- `List.EdgesOrderIt`: Vector of *cells* with as many terms as the number of iterations. Each entry lists the orders of the approximation bases of the essential boundaries, at each iteration.

- `List.RefinedLoopsIt`: Vector of *cells* with as many terms as the number of iterations. Each entry lists the finite elements selected for refinement at each iteration.

- `List.LoopsOrderIt`: Vector of *cells* with as many terms as the number of iterations. Each entry lists the orders of the approximation bases of the finite elements, at each iteration.

- `List.BetaIt`: Vector with as many terms as the number of iterations. For the state model, each entry lists the kinematic indeterminacy numbers, computed according to definition (62), at each iteration. For the flux model, each entry lists the static indeterminacy numbers, computed according to definition (64), at each iteration.