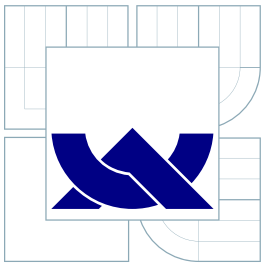


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV RADIOELEKTRONIKY
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF RADIO ELECTRONICS

NUMERICAL SOLUTIONS OF EMC PROBLEMS OF SMALL AIRPLANES

NUMERICKÁ ŘEŠENÍ PROBLEMATIKY EMC MALÝCH LETADEL

DIZERTAČNÍ PRÁCE
DOCTORAL THESIS

AUTOR PRÁCE — Ing. VLADIMÍR ŠEDĚNKA
AUTHOR

VEDOUCÍ PRÁCE — prof. Dr. Ing. ZBYNĚK RAIDA
SUPERVISOR

BRNO 2013

ABSTRACT

The dissertation describes actual problems in certifying small airplanes, which are to be solved by numerical modeling of the airplanes. The work is closely linked to the European project HIRF-SE, which deals with this problem. The essential part of the work is devoted to describing design of two modules within the HIRF-SE framework: time-domain finite-element solver BUTFE and its excitation module BUTFE_EXC. The thesis describes solution of absorbing boundary conditions, dispersive media, anisotropy and thin wire approximation. Special attention is devoted to a proper approximation of thin wires with sharp bends. Current implementation of the approximation leads to overlaps of wire segments.

KEYWORDS

electromagnetic compatibility, finite element method, high intensity radiated pulse, HIRF-SE, Amelet-HDF, HDF, absorbing boundary condition, dispersion, anisotropy, thin wire approximation

ABSTRAKT

Disertace popisuje současné problémy v certifikaci malých letadel, které by se měly v budoucnu řešit numerickým modelováním. Tento postup má zefektivnit návrh a zlevnit certifikaci letadel. Práce je úzce spjata s projektem HIRF-SE, který se problematikou certifikace letadel numerickými metodami zabývá. Podstatná část práce je věnována popisu dvou modulů pro platformu HIRF-SE: řešič BUTFE založený na metodě konečných prvků v časové oblasti a budicí nástroj BUTFE_EXC. Práce popisuje řešení pohlcujících okrajových podmínek, modelování disperzních a anizotropních materiálů a aproximaci tenkých drátů. Speciální pozornost je věnována řešení aproximace tenkých drátů s ostrými ohyby, jejíž současná formulace způsobuje překryvy mezi jednotlivými segmenty drátu.

KLÍČOVÁ SLOVA

elektromagnetická kompatibilita, metoda konečných prvků, vyzařované pole s vysokou intenzitou, HIRF-SE, Amelet-HDF, HDF, pohlcujících okrajové podmínky, disperze, anizotropie, aproximace tenkého drátu

ŠEDĚNKA, Vladimír *Numerical solutions of EMC problems of small airplanes*: doctoral thesis. Brno: Brno University of Technology, Faculty of Electrical Engineering and Communication, Department of Radio Electronics, 2013. 86 p. Supervised by prof. Dr. Ing. Zbyněk Raida

DECLARATION

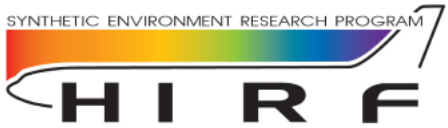
I declare that I have written my doctoral thesis on the theme of "Numerical solutions of EMC problems of small airplanes" independently, under the guidance of the doctoral thesis supervisor and using the technical literature and other sources of information which are all quoted in the thesis and detailed in the list of literature at the end of the thesis.

As the author of the doctoral thesis I furthermore declare that, as regards the creation of this doctoral thesis, I have not infringed any copyright. In particular, I have not unlawfully encroached on anyone's personal and/or ownership rights and I am fully aware of the consequences in the case of breaking Regulation § 11 and the following of the Copyright Act No. 121/2000 Coll., and of the rights related to intellectual property right and changes in some Acts (Intellectual Property Act) and formulated in later regulations, inclusive of the possible consequences resulting from the provisions of Criminal Act No. 40/2009 Coll., Section 2, Head VI, Part 4.

Brno ....

(author's signature)

ACKNOWLEDGEMENTS



The work described in this thesis has received funding from EC FP7 under grant no. 205294 (the HIRF SE project).
Further financing was provided by the Czech Ministry of Education (the grant no. 7R09008).

The described research was performed in laboratories supported by the SIX project; the registration number CZ.1.05/2.1.00/03.0072, the operational program Research and Development for Innovation.

A support of the project CZ.1.07/2.3.00/20.0007 Wireless Communication Teams financed by the operational program Education for Competitiveness is also gratefully acknowledged.

This research used resources of the National Supercomputing Center IT4Innovations, which is supported by the Op VaVpI project number CZ.1.05/1.1.00/02.0070.

LIST OF SYMBOLS, PHYSICAL CONSTANTS AND ABBREVIATIONS

3D	three-dimensional
ABC	absorbing boundary condition
API	application program interface
CPU	central processor unit
EM	electromagnetic
EMC	electromagnetic compatibility
ERPCF	epoxy resin prepreg carbon fiber
FDM	finite difference method
FDTD	finite difference time-domain
FEM	finite element method
FIT	finite integration technique
GPU	graphics processing unit
GUI	graphical user interface
HDF	Hierarchical Data Format
HIRF	High-Intensity Radiated Fields
HIRF-SE	High-Intensity Radiated Fields - Synthetic Environment
HT	Hyper-Threading technology
IPC	inter-process communication
MCR	MATLAB Compiler Runtime
MoM	method of moments
TLM	transmission line matrix
PCM	prepreg copper mesh
PDE	partial differential equation
PEC	perfectly electric conductor
PED	personal electronic device

PEMF	pulsed electromagnetic field
PMC	perfect magnetic conductor
PML	perfectly matched layer
PO	physical optics
SE	shielding effectiveness
TDFE	time-domain finite element
WPED	wireless personal electronic device

CONTENTS

1 PREFACE

A growing involvement of electronic devices inside an aircraft and especially employment of electrical devices in critical flight systems makes *electromagnetic compatibility* (EMC) a more and more important part of the aircraft design process. Internal electronic systems interfere with each other. In addition, all the systems are exposed to influences of radio and TV transmitters, satellite transmitters, radars, lightning strikes, etc. Such an environment is known as *High-Intensity Radiated Fields* (HIRF).

Beyond the usage of electronic devices inside the aircraft, growing interest in EMC is also given by an effort to reduce costs and increase in-flight comfort. Conductive parts are increasingly being replaced by composite materials which makes the aircraft lighter but also more vulnerable to HIRF. Portable electronic devices become a common part of our lives. However, their use during the flight makes the EMC problem even more challenging.

Before final production, each aircraft must pass various EMC tests. This typically consists in simulating the airplane (or its model) at an EMC test site trying to model the worst case scenario. These tests are expensive in terms of space, time and money. Sometimes they may be also destructive.

Fig. 1.1 *Test chamber for EMC purposes[1].*

Previously mentioned facts lead to the effort of numerical modeling before testing. Testing costs are replaced by computational costs which are decreasing due to advances in computer technology. This improves the efficiency of the design process and makes the certification cheaper.

2 STATE OF THE ART

Methods employed by software for numerical modeling of an electromagnetic field most often are the following: *finite difference method* (FDM), *finite element method* (FEM), *transmission line matrix* (TLM), *method of moments* (MoM), *finite integration technique* (FIT) and *physical optics* (PO). None of them is a universal method, each one has specific benefits and drawbacks. A comparison between previously mentioned methods was studied by N.O. Sadiku [2, 3]. The work aims at FEM, so a brief state of the art of this numerical method will follow.

Finite element method

The method converts a *partial differential equation* (PDE) into a system of linear equations. The computational domain is divided into smaller parts (elements). The desired physical quantity over each element is approximated by selected base functions. The system of linear equations can be formed by connecting all the elements. By solving this system, piecewise interpolation of the quantity over the entire structure can be obtained. Compared with FDM, the finite element method is more versatile and can better handle complex geometries and inhomogenous media. The approximation is more accurate than in the case of FDM.

Origin of the method

Development of FEM began approximately in 1942 by R. Courant [4]. The term "finite element" has been used since 1960. The first application of FEM to electromagnetism was done by P.P. Silvester [5].

Software tools

FEM is used by many commercial and open-source programs [11, 12]. However, *electromagnetic* (EM) problems are solved by multiphysical packages like ANSYS© or COMSOL Multiphysics©.

Perfectly matched layer

One of the most challenging problems in numerical methods is a reflectionless truncation of the computational domain in the case of open-region problems. Before using a *perfectly matched layer* (PML), there were several techniques used for the truncation, but none of them was as accurate. The first PML was proposed by Berenger [13] for the *finite difference time-domain* (FDTD) method. However, this approach used modified Maxwell equations thus it was not suitable for FEM. Afterwards Sacks et al. [14] proposed a PML employing anisotropic media, which could be easily implemented in frequency-domain finite element code. Mathis [15] used this approach to derive PML for FEM in the time domain. Performance of this PML was examined in [16]. However, good absorption was achieved only for a limited frequency band. The frequency band limitation was figured

out in [17] and [18]. Rylander and Jin [19] engaged in conformal PML, which reduced the amount of elements in the computational domain. The approach published in [17, 18] was later outperformed by second-order PML [20].

Computation

Since the 1990s, it has been more effective to gain computational power by increase of the number of *central processor units* (CPUs) rather than increasing clock frequency. Due to this fact, attention was focused on the development of efficient programs for multi-processor [21] and multi-core [22] systems. The idea to compute the problems on a *graphics processing unit* (GPU) is beginning to assert itself. In fact, a GPU is well designed for matrix operations and can achieve higher computational power than CPUs due to hundreds of cores inside. Implementation of FEM on a GPU has already been done [23].

Another way how to speed-up the computation, is using preconditioners. Convergence properties of three preconditioning techniques in combination with two iterative solvers are investigated in [24].

Adaptive FEM provides better distribution of degrees of freedom within the computational domain using an h-refinement (varying element size) or/and a p-enrichment (varying polynomial order). In [25], authors focus on problems regarding very low frequencies and extremely small elements relative to the wavelength. Giannacopoulos [26] reduces communication among processors in parallel adaptive FEM computation. In [27], authors compare the adaptation methods on FEM, MoM and PO.

Books and reviews

Among the important books dealing with applying FEM in electromagnetics worth mentioning are Silvester et. al. [6, 7] and Jian-Ming Jin [8], [9] and [10] (written with Riley).

There were also written several summarizing articles dealing with FEM. Silvester et al. made a summarizing article containing nearly 240 selected citations related to FEM in electromagnetism [28][1]. Other articles were written by Jin et al. [29] and Teixeira [30]. Time domain formulations were discussed in [31].

[1] This paper was also his last one, it contains a remembrance of P.P.Silvester.

3 AIMS OF THE DISSERTATION

Although various software tools for the EM field analysis are already developed, they usually do not cover the whole area of possible EMC problems related to the aircraft. Moreover, some differences can appear in results obtained from software based on different numerical methods due to the strengths and weaknesses of the particular method. Basically, user should pick the right method based on his knowledge. However the computation of the EM field inside a complex object such as an airplane makes the choice quite difficult.

Project *High-Intensity Radiated Fields - Synthetic Environment* (HIRF-SE) [32] aimed to create a numerical modeling computer framework for the aeronautic industry. The framework should improve the design phase of the aircraft and also reduce costs of the EMC tests (due to the reasons mentioned on page 8). The HIRF-SE considered [33]:

- the external EM environment to which the air vehicle has to be certified. This environment is defined in regulatory documents currently available and about to be released;
- the internal EM environment generated by air vehicle electronic installations;
- the internal EM environment whose complexity will increase in the future due to allowing *personal electronic devices* (PEDs) and *wireless personal electronic devices* (WPEDs) to be used in the cockpit and the cabin during all phases of the flight.

The project involved 44 partners in 11 European countries. Each partner was developing a specific module making use of the particular method. All the modules were connected with each other via the format Amelet-HDF [34], which was also being developed within the project. The Amelet-HDF format is based on *Hierarchical Data Format* (HDF) [35].

Aims of the dissertation are defined as follows:

- **TDFE solver:** create a three-dimensional *time-domain finite element* (TDFE) solver aiming at EM simulations
- **Bent wires:** improve the solver's capabilities towards solving EMC problems concerning small airplanes (thin wire approximation of bent wires)
- **File interface:** design an input/output wrapper that will adapt the TDFE solver for application in the HIRF-SE framework
- **Development tools:** create development tool(s) for simplifying the design and test phase and facilitating further improvements of the TDFE solver

In order to save computational time, the solver can be (under specific conditions) combined together with the *pulsed electromagnetic field* (PEMF) method [36]. It is an analytical method that can be utilized for determining the excitation for the solver in the case of propagation of an electromagnetic wave through a perforated slab.

4 ACHIEVEMENTS

This chapter is divided into four sub-chapters where each one focuses on a particular aim of the dissertation. Each sub-chapter starts with a brief introduction of its content.

4.1 TDFE solver

This chapter is especially dedicated to explaining the core functionality of the TDFE solver called BUTFE and its excitation module BUTFE_EXC. The history of its development justifies the presence of some chapters in this work. It is briefly outlined in chapter 4.1.1. The involvement of the solver and the excitation module within the HIRF-SE framework is shown in chapter 4.1.2. The semantic location within the framework is shown on a sample simulation. The description of both modules from the user's point of view follows in chapter 4.1.3. Chapter 4.1.4 focuses on the mathematical formulation of the problem leading to the final system of linear equations of the TDFE solver. It is assumed that the reader is familiar with the finite element method itself. Types of basis functions involved in the solver are described in chapter 4.1.5. Chapter 4.1.6 shows the essential functionality of the solver on an earlier version written in MATLAB© environment. Chapter 4.1.7 shows some results of the BUTFE solver. Chapter 4.1.9 concludes this topic.

4.1.1 History of its development

The core of the finite element program was taken from my diploma thesis [37]. The program performed modal analysis of a cavity resonator in three-dimensional space in MATLAB©. Originally, the program was designed to solve one component of a selected field quantity. The remaining components were computed analytically through the critical wave number thanks to the longitudinal homogeneity of considered structure. Spurious solutions were removed from the list of the solutions in compliance with the computed field distribution. At the beginning of my doctoral study, versatility of the original solver was improved and the solver was employed in the multi-objective synthesis of a cavity resonator [38].

The program was enhanced to carry out the analysis in the frequency domain and later also in the time domain according to [39].

The original scalar nodal-oriented basis functions were replaced by the vector (edge-oriented) ones [42]. This allowed modeling of structures containing material junctions due to tangential continuity of the edge basis functions. It suppressed the problem with the spurious solutions as well. Usage of the vector basis functions also improved the accuracy of the solver. All the components[1] of a particular field quantity could be computed at once.

Movement from the longitudinally homogeneous structure towards a general structure and the use of the edge-oriented basis functions required a graphical tool for entering boundary conditions. This was essential for the development phase of the solver[2].

[1] Number of the components depends on the dimensionality of the problem.
[2] It evolved in a separate module BUTFE_EXC afterwards.

Attention was also paid to the mesh generator. The original program employed an external open-source mesher GMSH© [51]. Due to some problems discovered during the resonator synthesis, its substitution was looked for. Finally, special GiD [52] licenses were acquired for HIRF-SE purposes. The mesh generation was another development related issue[3], caused by lack of any input data for testing and a tool to create them. The ability of a new mesh generator to export the mesh into the Amelet-HDF file was an argument to start development of data retrieval using the Amelet-HDF format.

Due to the computational demands and memory limitations of the MATLAB© environment (and also due to the requirements of the HIRF-SE project)[4], the MATLAB code had to be rewritten into C language. Along with this challenge, the code was enhanced to accept all the inputs given by the Amelet-HDF input file, not just the mesh. In order to employ the anisotropic PML in the future, the program was improved to handle material properties given by a tensor.

Due to the fact that the HIRF-SE framework did not provide a tool to set excitation in a specific part of the mesh, a new module (called BUTFE_EXC) had to be created for that purpose.

A thin-wire approximation was employed in order to compute wire currents preventing an excessive increase of computational demands.

4.1.2 Connecting the modules within the framework

This chapter describes work in the HIRF-SE framework from the user's perspective. The entire simulation is formed by a chain of sub-simulations performed by various modules. For each sub-simulation, the HIRF-SE framework prepares an input Amelet-HDF file, executes the main executable of a given module and reads the outputs from an output Amelet-HDF file created by the module. A schematic diagram of each sub-simulation shows inputs (left side), module used for the simulation (middle) and outputs (right side). Notation "(a..b)" represents minimal, resp. maximal allowed amount of data of a given data type.

Mesh generation

The first sub-simulation is responsible for mesh generation. It uses the module GiD, which was developed by the group CIMNE [52]. Figure 4.1 shows that the module accepts up to five different types of input data and can produce up to two types of output data. All of them are optional.

In this case, the module is used as a mesh generator. A window, opened within the sub-simulation, allows the user to define the geometry, generate an output mesh and store it into an object *mesh_gid*.

[3] The mesh is part of the input data that should be already presented in the Amelet-HDF input file supplied by the HIRF-SE framework.

[4] The requirement for writing the main code in C was announced in the course of the project.

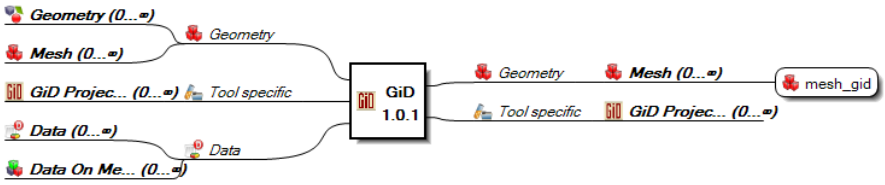


Fig. 4.1 *The HIRF-SE framework: Mesh generation.*

Sources determination

The previously generated mesh *mesh_gid* can be linked with the excitation module BUTFE_EXC (see figure 4.2). The module requires one object of type *Mesh* and produces 3 types of output data: *Mesh*, *Global environment* and one or more objects of type *Electromagnetic source*.

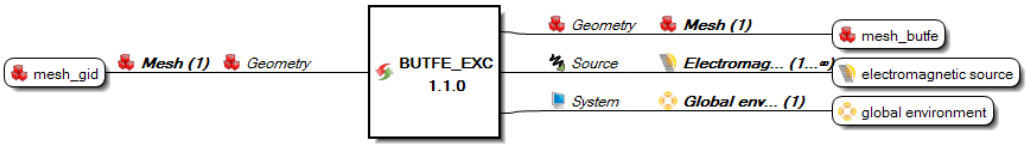


Fig. 4.2 *The HIRF-SE framework: Sources determination.*

Based on the user defined excitation(s), the module slightly modifies the input *mesh_gid* into *mesh_butfe* and creates a list of requested time samples *global environment* and excitation samples *electromagnetic source*[5].

Computation

This sub-simulation employs the TDFE solver module BUTFE (see figure 4.3). The module expects *Mesh*, *Global environment* and *Electromagnetic source(s)* from the BUTFE_EXC module. The user has to define the remaining input data of the simulation: materials and physical quantities to compute. Materials are set via object *Link* and objects of the *Material* category. The module accepts predefined materials like vacuum, *perfectly electric conductor* (PEC), *perfect magnetic conductor* (PMC), classical materials and dispersive materials given by the Debye model. The materials can be either isotropic or anisotropic defined by a material tensor.

The requested physical quantities are defined via *Output request*. The BUTFE module supports the following output quantities:

- electric and/or magnetic field intensity in the middle of a given tetrahedron,
- electric and/or magnetic field intensity in a predefined point,
- electric current flowing through a predefined wire in the middle of wire segments.

Only an electric/magnetic field in the domain and electric field in two specific groups of points are requested in this simulation.

Note that the BUTFE module solves the intensity of the electrical field. The intensity of the magnetic field is obtained using post-processing (see chapter 4.1.4 for more details).

[5] The Amelet-HDF objects will be discussed in chapter 4.3. Detailed specification can be found in [34].

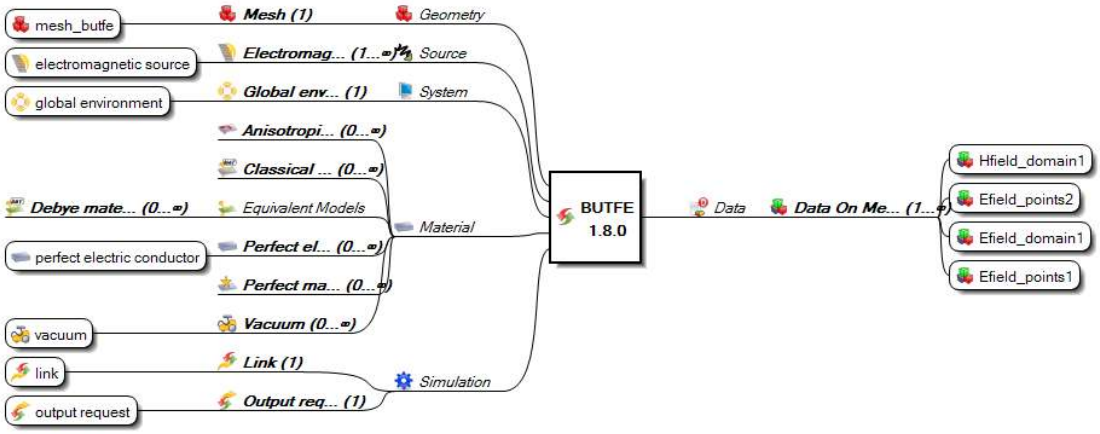


Fig. 4.3 *The HIRF-SE framework: EM field computation.*

Visualization

The last part of the simulation chain could be visualization. The module Paraview is developed by the company AxesSim. It is a wrapper for a known visualization program of the same name developed by Kitware, Inc. [53].

Another visualization tool (to be considered as the main tool) in the HIRF-SE framework is GiD.

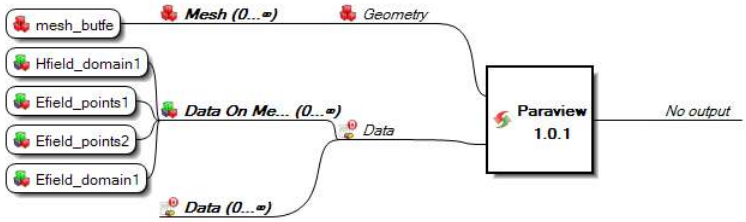


Fig. 4.4 *The HIRF-SE framework: EM field visualization by Paraview.*

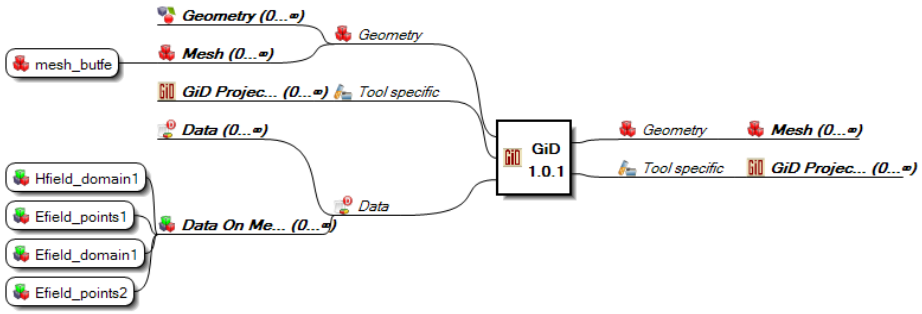


Fig. 4.5 *The HIRF-SE framework: EM field visualization by GiD.*

Figures 4.4 and 4.5 are only informative. The module Paraview can handle only inputs that cover all the elements of the geometry. In this case the Paraview would return that there are more points in the geometry than just the five predefined groups. The module GiD was not adapted to be compatible with output data of the BUTFE solver.

4.1.3 Description of the modules

BUTFE_EXC module

Excitation is closely connected with the mesh so in order to achieve easy operation, the module has to contain a *graphical user interface* (GUI). Based on my programming skills, I have decided to make this tool in the MATLAB© environment. The limitations of the MATLAB© code mentioned in chapter 4.1.1 do not have an influence (considering also lightened requirements of the HIRF-SE project).

An executable for the HIRF-SE module was generated using MATLAB© Compiler™ [49]. A special set of libraries called *MATLAB Compiler Runtime* (MCR) package must be installed on a target machine in order to run the executable [50]. The MCR installation package can be freely distributed together with the module.

The excitation tool can work in three different modes: classical, hybrid and eigenmode. Classical excitation sets distribution in the nearest neighborhood of a given plane. Hybrid excitation computes the EM field behind a perforated slab using the PEMF method [36] and uses the results for excitation. Eigenmode solves a 2D eigenmode analysis (using 2D FEM) of a given plane to get desired field distribution.

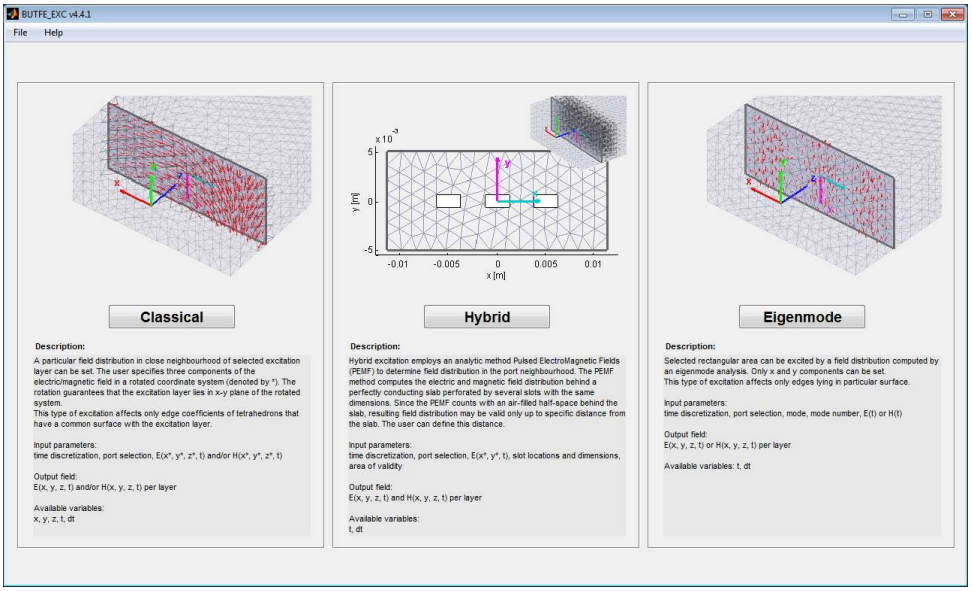


Fig. 4.6 *The excitation tool: choosing operation mode.*

All these three cases are restricted to using a plane layer from the input mesh. The layer is supposed to contain triangle elements. The tool extracts edges from tetrahedrons forming the neighborhood of the layer and adds the edges into the output mesh (that is the only difference between *mesh_gid* and *mesh_butfe*). The resulting excitation coefficients are related to the edges. Paths for the input and output files are hard-coded as *simulation/workingDir/input.h5*, resp. *simulation/workingDir/output.h5*. These paths are common for all modules in the HIRF-SE framework.

The general layout of the tool remains the same in the case of all the three modes. The tool contains the following sections: *Main table*, *Status*, *Time scale*, *Time domain*

and *Excitation layer*. The *Main table* allows to set the excitation formula (time-domain dependency), *Status* informs about performed actions. Once the program starts, the input mesh is analyzed and the optimal time step is determined. The user can modify time step, number of steps or total computational time in the *Time scale* panel. The *Time domain* box displays selected excitation signal in both the time and frequency domain. The *Excitation layer* lets the user to check space distribution of the excitation. Geometry can be manipulated through tools in the *View* or *Camera* tool bar. Output data is written using the menu option *File*→*Save* (or by pressing CTRL+S).

BUTFE_EXC: classical

Figure 4.7 shows a screenshot of the module in classical mode. The *Main table* displays two mesh layers that are suitable for the excitation. The layer *EXC* was created by the mesher to act as an excitation layer. The user can set all three components of the electric field separately[6]. The column *Output* sets a target location of the excitation within the output Amelet-HDF file. Each target(*Output*) must be predefined by the user in the HIRF-SE framework and the location must exist in the input file.

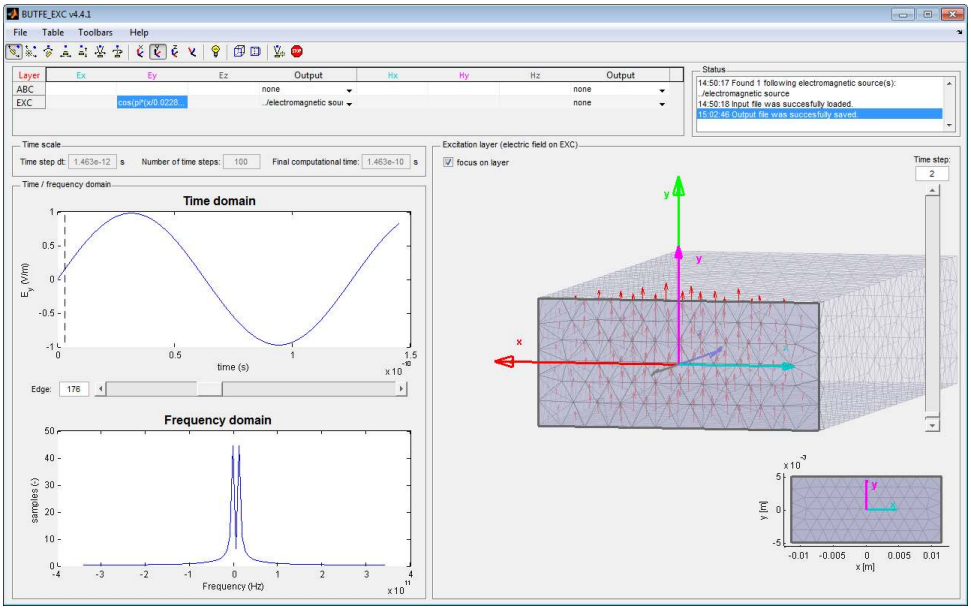


Fig. 4.7 *The excitation tool: classical.*

The *Excitation layer* highlights this layer in the whole geometry (top) and shows as an extracted plane transformed into a new x-y plane (bottom right). Note that the original coordinate system is colored red-green-blue (x-y-z) but transformed coordinates are cyan-magenta (x-y). The definition of the pulse in the *Main table* is related to the transformed coordinate system. The relationship between the coordinate systems is visible through the general view in the *Excitation layer* (top).

Note that this mode supports both the spatial and temporal definition of the excitation pulse.

[6] Excitation by magnetic field is not supported by future versions of BUTFE solver.

BUTFE_EXC: hybrid

The *Excitation layer* box contains a list of rectangular perforations of the slab. The user can add, edit and remove slots and modify width and height of a slot (all slots have the same dimensions). Parameter *Distance* sets a perpendicular distance from the slot where the PEMF method still influences the excitation. Basically, it defines a coupling volume between PEMF and FEM.

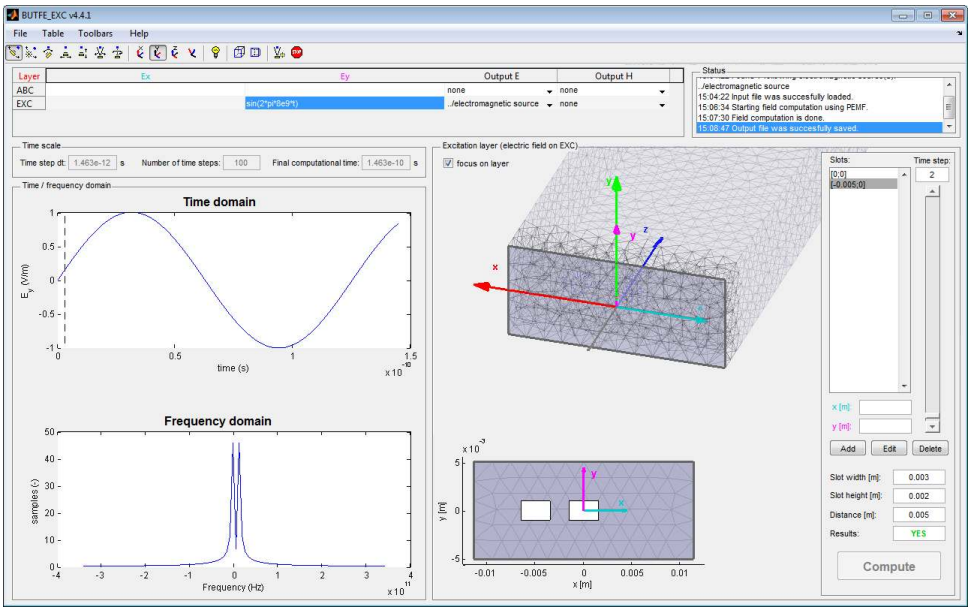


Fig. 4.8 *The excitation tool: hybrid.*

The edit box *Results* (bottom right) shows "NO" at the beginning. This means that the resulting field distribution from the PEMF method is not yet computed. After setting all necessary parameters and clicking the *Compute* button, computation of the wave propagation through the perforated slab starts. The finished computation is notified in two ways: using a status window and by changing *Results* to "YES". The button *Compute* is disabled. Any change of input parameters can enable this button again. Change of the field *Results* to "OLD" informs the user that current computed data does not correspond with given inputs and thus it is necessary to compute them again.

BUTFE_EXC: eigenmode

This mode is based on the code described in section 4.1.6. In the *Main table*, the user can select transversal *Mode* (TE/TM) and *Mode number* (1 to 5) to set space behavior and *Exc formula* to set time behavior. The remaining controls are the same as in the classical mode.

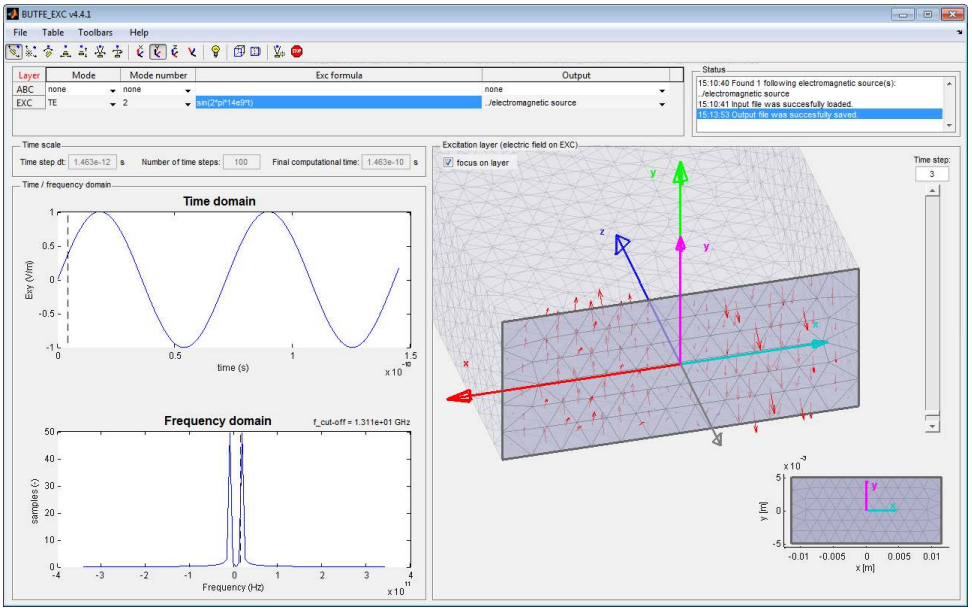


Fig. 4.9 *The excitation tool: eigenmode.*

The module shows cut-off frequency for the given mode on the top right corner of the frequency domain plot. The user can check whether the excitation fits the cut-off frequency.

BUTFE module

The BUTFE module has no graphical interface. It only produces a text log, which is visible in the framework after the simulation (see figure 4.10).

Firstly, the input file is processed. The program can continue only if all required inputs are read successfully. The status of the read processes is reported in the log file. The category */simulation/sim_butfe* is the so-called entry point of the input file. It stores paths to all necessary data and simulation parameters. Requested physical quantities are set in */outputRequest*. The category */mesh* stores the computational mesh, */physicalModel* definition of materials and */link* connection between these two. Time samples are defined in */globalEnvironment* and sources in */electromagneticSource*. The Amelet-HDF objects will be discussed in chapter 4.3. Detailed specification can be found in [34].

An *AH5* library[7] reads Amelet-HDF data into raw variables. These variables are converted into a form that suits the computation by mesh conversion. Geometry checks are performed within the mesh conversion.

[7] The *AH5* will be described later in chapter 4.3.3.

```
################################### BUTFE 1.9.0 ###################################
###################################################################################

Reading input file input.h5...
  /simulation/sim_butfe:
    /outputRequest/output request...                                        done
    /mesh/mesh_butfe...                                                     done
    /physicalModel/perfectElectricConductor...                              done
    /globalEnvironment/global environment...                                done
    /electromagneticSource/sourceOnMesh/electromagnetic source...           done
    /link/link...                                                           done
    /physicalModel/vacuum...                                                done
    /physicalModel/volume/ABC...                                            done

Mesh conversion...
    Line length range:                          3.24E-03 to 1.45E-02
    Triangle area range:                        9.57E-07 to 1.27E-04
    Tetrahedron volume range:                   7.27E-10 to 1.09E-06

Sources determination...
    Edges used for E-field excitation:                      15283/15507

Reading simulation parameters...
    Number of threads/cores:                                        8/8
    Preconditioner/solver:                                  jacobi/cg
    Matrix assembly using 8 thread(s)...                        done
    Computation using 8 thread(s)...                            done
###################################################################################
Total computational time: 04:05:52.
```

Fig. 4.10 *BUTFE module: Simulation log.*

In sources determination, the solver identifies excitation(s) given by the BUTFE_EXC module. It merges more excitations of the same kind together and adapts them to match boundary conditions. That is why the number of used excitation edges can be lower than the total number of edges of the excitation layer. In the moment when the user sets the excitation (in BUTFE_EXC), the boundary conditions are not yet known.

Information about applied simulation parameters is followed by brief information about the progress and total computational time.

Actual progress of the simulation can be monitored in the framework through the window "Follow simulation" (see figure 4.11).

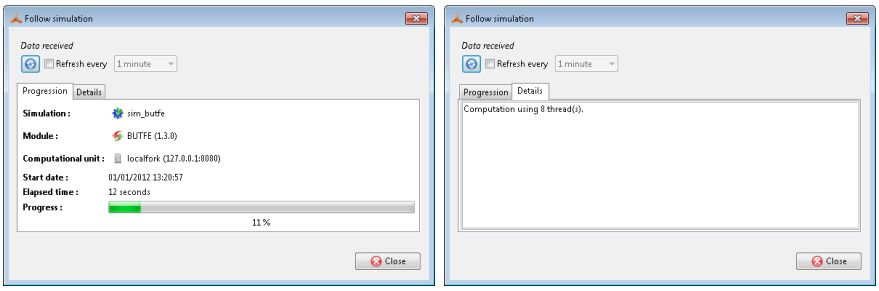


Fig. 4.11 *BUTFE module: Follow simulation.*

The module offers the possibility of a controlled end of the simulation from the framework. In this case, the output file will only contain results computed so far. Information about a premature end of the simulation is noted in the log file.

4.1.4 Mathematical background

The BUTFE solver is based on time domain Maxwell equations in differential form [10]:

$$\nabla \times \mathbf{E}(\mathbf{r},t) = -\mu_0\hat{\mu}_r \frac{\partial \mathbf{H}(\mathbf{r},t)}{\partial t} \tag{4.1a}$$

$$\nabla \times \mathbf{H}(\mathbf{r},t) = \epsilon_0\hat{\epsilon}_r \frac{\partial \mathbf{E}(\mathbf{r},t)}{\partial t} + \hat{\sigma}\mathbf{E}(\mathbf{r},t) + \mathbf{J}_{imp}(\mathbf{r},t) \tag{4.1b}$$

$$\nabla \cdot [\epsilon_0\hat{\epsilon}_r \mathbf{E}(\mathbf{r},t)] = \rho \tag{4.1c}$$

$$\nabla \cdot [\mu_0\hat{\mu}_r \mathbf{H}(\mathbf{r},t)] = 0 \tag{4.1d}$$

where the meaning of the symbols is as follows

$\mathbf{E}(\mathbf{r},t), \mathbf{H}(\mathbf{r},t)$	electric (V/m) and magnetic (A/m) field intensity
$\mathbf{J}_{imp}(\mathbf{r},t)$	electric current density (A/m^3)
ϵ_0, μ_0	free-space permittivity $(8.854 \cdot 10^{-12}\, F/m)$
	and permeability $(4\pi \cdot 10^{-7}\, H/m)$
$\hat{\epsilon}_r, \hat{\mu}_r$	relative permittivity and permeability tensors
$\hat{\sigma}$	electric conductivity tensor (S/m)
ρ	electric charge density (C/m^3)

Equation 4.1 describes behavior of the electromagnetic field in the computation domain. Behavior on a PEC surface is given by Dirichlet boundary condition [10]:

$$\vec{n} \times \mathbf{E}(\mathbf{r},t) = 0 \quad \mathbf{r} \in \text{PEC} \tag{4.2}$$

where $\vec{n}$ is outward unit vector normal to the surface. We employ also a first-order absorbing boundary condition, where the scattered electric field $\mathbf{E}_{sc}(\mathbf{r},t)$ is required to satisfy [9]:

$$\vec{n} \times \left[\frac{1}{\mu_0\hat{\mu}_r}\nabla \times \mathbf{E}_{sc}(\mathbf{r},t)\right] + Y_0\frac{\partial}{\partial t}\left[\vec{n} \times \vec{n} \times \mathbf{E}_{sc}(\mathbf{r},t)\right] = 0 \quad \mathbf{r} \in ABC \tag{4.3}$$

where $Y_0 = \sqrt{\frac{\epsilon_0}{\mu_0}}$ is free-space admittance. Assuming total electric field as sum of scattered and incident field $\mathbf{E} = \mathbf{E}_{sc} + \mathbf{E}_{inc}$, we can write

$$\begin{aligned} \vec{n} \times \left[\frac{1}{\mu_0\hat{\mu}_r}\nabla \times \mathbf{E}(\mathbf{r},t)\right] + Y_0\frac{\partial}{\partial t}\left[\vec{n} \times \vec{n} \times \mathbf{E}(\mathbf{r},t)\right] = \\ \vec{n} \times \left[\frac{1}{\mu_0\hat{\mu}_r}\nabla \times \mathbf{E}_{inc}(\mathbf{r},t)\right] + Y_0\frac{\partial}{\partial t}\left[\vec{n} \times \vec{n} \times \mathbf{E}_{inc}(\mathbf{r},t)\right] \quad \mathbf{r} \in ABC \end{aligned} \tag{4.4}$$

Now we start from Maxwell equations 4.1. Dividing 4.1a by the permeability tensor $\hat{\mu} = \mu_0\hat{\mu}_r$, applying rotation and substituting time derivative of 4.1b into equation 4.1a yields a time-domain wave equation

$$\nabla \times \left[\frac{1}{\mu_0\hat{\mu}_r}\nabla \times \mathbf{E}(\mathbf{r},t)\right] + \epsilon_0\hat{\epsilon}_r\frac{\partial^2 \mathbf{E}(\mathbf{r},t)}{\partial t^2} + \hat{\sigma}\frac{\partial \mathbf{E}(\mathbf{r},t)}{\partial t} + \frac{\partial \mathbf{J}_{imp}(\mathbf{r},t)}{\partial t} = 0 \quad \mathbf{r} \in V \tag{4.5}$$

Within the residual minimization, we multiply 4.5 by the testing function $\mathbf{N}_i(\mathbf{r})$ (same as the basis function - the Galerkin's approach) and integrate over the computation domain

$$\iiint_V \mathbf{N}_i(\mathbf{r}) \cdot \left\{ \nabla \times \left[\frac{1}{\mu_0 \hat{\mu}_r} \nabla \times \mathbf{E}(\mathbf{r},t) \right] \right\} \mathrm{d}V + \iiint_V \mathbf{N}_i(\mathbf{r}) \cdot \epsilon_0 \hat{\epsilon}_r \frac{\partial^2 \mathbf{E}(\mathbf{r},t)}{\partial t^2} \mathrm{d}V + \iiint_V \mathbf{N}_i(\mathbf{r}) \cdot \hat{\sigma} \frac{\partial \mathbf{E}(\mathbf{r},t)}{\partial t} \mathrm{d}V + \iiint_V \mathbf{N}_i(\mathbf{r}) \cdot \frac{\partial \mathbf{J}_{imp}(\mathbf{r},t)}{\partial t} \mathrm{d}V = 0 \quad \mathbf{r} \in V \tag{4.6}$$

Involving vector identity $\mathbf{a} \cdot (\nabla \times \mathbf{b}) = (\nabla \times \mathbf{a}) \cdot \mathbf{b} - \nabla \cdot (\mathbf{a} \times \mathbf{b})$ in the first term, we can write

$$\begin{aligned} &\iiint_V \mathbf{N}_i(\mathbf{r}) \cdot \left\{ \nabla \times \left[\frac{1}{\mu_0 \hat{\mu}_r} \nabla \times \mathbf{E}(\mathbf{r},t) \right] \right\} \mathrm{d}V = \\ &\quad \iiint_V \left[\nabla \times \mathbf{N}_i(\mathbf{r}) \right] \cdot \left[\frac{1}{\mu_0 \hat{\mu}_r} \nabla \times \mathbf{E}(\mathbf{r},t) \right] \mathrm{d}V \\ &\quad - \iiint_V \nabla \cdot \left\{ \mathbf{N}_i(\mathbf{r}) \times \left[\frac{1}{\mu_0 \hat{\mu}_r} \nabla \times \mathbf{E}(\mathbf{r},t) \right] \right\} \mathrm{d}V \end{aligned} \tag{4.7}$$

where the last term can be rewritten using divergence theorem $\iiint_V \nabla \cdot \mathbf{f} \mathrm{d}V = \oiint_S \vec{n} \cdot \mathbf{f} \mathrm{d}S$ and vector identity $(\mathbf{a} \times \mathbf{b}) \cdot \mathbf{c} = -\mathbf{a} \cdot (\mathbf{c} \times \mathbf{b})$ leading the first term of equation 4.5 into

$$\begin{aligned} &\iiint_V \mathbf{N}_i(\mathbf{r}) \cdot \left\{ \nabla \times \left[\frac{1}{\mu_0 \hat{\mu}_r} \nabla \times \mathbf{E}(\mathbf{r},t) \right] \right\} \mathrm{d}V = \\ &\quad \iiint_V \left[\nabla \times \mathbf{N}_i(\mathbf{r}) \right] \cdot \left[\frac{1}{\mu_0 \hat{\mu}_r} \nabla \times \mathbf{E}(\mathbf{r},t) \right] \mathrm{d}V \\ &\quad + \oiint_S \mathbf{N}_i(\mathbf{r}) \cdot \left\{ \vec{n} \times \left[\frac{1}{\mu_0 \hat{\mu}_r} \nabla \times \mathbf{E}(\mathbf{r},t) \right] \right\} \mathrm{d}S \end{aligned} \tag{4.8}$$

The surface integral in the 4.8 is related to the boundary conditions. It vanishes on PEC due to 4.2. For the *absorbing boundary condition* (ABC) we substitute 4.4 and get

$$\begin{aligned} &\iiint_V \mathbf{N}_i(\mathbf{r}) \cdot \left\{ \nabla \times \left[\frac{1}{\mu_0 \hat{\mu}_r} \nabla \times \mathbf{E}(\mathbf{r},t) \right] \right\} \mathrm{d}V = \\ &\quad \iiint_V \left[\nabla \times \mathbf{N}_i(\mathbf{r}) \right] \cdot \left[\frac{1}{\mu_0 \hat{\mu}_r} \nabla \times \mathbf{E}(\mathbf{r},t) \right] \mathrm{d}V \\ &\quad - \iint_{S_{ABC}} \mathbf{N}_i(\mathbf{r}) \cdot \left\{ Y_0 \frac{\partial}{\partial t} \left[\vec{n} \times \vec{n} \times \mathbf{E}(\mathbf{r},t) \right] \right\} \mathrm{d}S \\ &+ \iint_{S_{ABC}} \mathbf{N}_i(\mathbf{r}) \cdot \left\{ \vec{n} \times \left[\frac{1}{\mu_0 \hat{\mu}_r} \nabla \times \mathbf{E}_{inc}(\mathbf{r},t) \right] \right\} \mathrm{d}S \\ &+ \iint_{S_{ABC}} \mathbf{N}_i(\mathbf{r}) \cdot \left\{ Y_0 \frac{\partial}{\partial t} \left[\vec{n} \times \vec{n} \times \mathbf{E}_{inc}(\mathbf{r},t) \right] \right\} \mathrm{d}S \end{aligned} \tag{4.9}$$

We get more convenient form using vector identities:

$$\begin{aligned}
&\iiint_V \mathbf{N}_i(\mathbf{r}) \cdot \left\{ \nabla \times \left[\frac{1}{\mu_0 \hat{\mu}_r} \nabla \times \mathbf{E}(\mathbf{r},t) \right] \right\} \mathrm{d}V = \\
&\quad \iiint_V \left[\nabla \times \mathbf{N}_i(\mathbf{r}) \right] \cdot \left[\frac{1}{\mu_0 \hat{\mu}_r} \nabla \times \mathbf{E}(\mathbf{r},t) \right] \mathrm{d}V \\
&\quad + \iint_{S_{ABC}} Y_0 \frac{\partial}{\partial t} \left[\vec{n} \times \mathbf{N}_i(\mathbf{r}) \right] \cdot \left[\vec{n} \times \mathbf{E}(\mathbf{r},t) \right] \mathrm{d}S \\
&- \iint_{S_{ABC}} \left[\vec{n} \times \mathbf{N}_i(\mathbf{r}) \right] \cdot \left[\frac{1}{\mu_0 \hat{\mu}_r} \nabla \times \mathbf{E}_{inc}(\mathbf{r},t) \right] \mathrm{d}S \\
&\quad - \iint_{S_{ABC}} Y_0 \frac{\partial}{\partial t} \left[\vec{n} \times \mathbf{N}_i(\mathbf{r}) \right] \cdot \left[\vec{n} \times \mathbf{E}_{inc}(\mathbf{r},t) \right] \mathrm{d}S
\end{aligned} \tag{4.10}$$

Equation 4.10 is substituted back into the main system 4.6. Field quantities over given tetrahedron are approximated using

$$\mathbf{U}(\mathbf{r},t) \approx \sum_{j=1}^{6} \tilde{\mathbf{N}}_j(\mathbf{r})\, u_j(t) \tag{4.11}$$

where $\mathbf{U}(\mathbf{r},t)$ is approximated field quantity (continuous), $\mathbf{N}_j(\mathbf{r})$ is spatially-dependent basis function and $u_j(t)$ is time-dependent edge coefficient of the approximated quantity for j-th edge of given tetrahedron.

The system of equations before time discretization becomes:

$$\mathbf{S}\mathbf{e}\left(t\right)+\mathbf{T}^{\epsilon}\frac{\partial^{2}\mathbf{e}\left(t\right)}{\partial t^{2}}+\mathbf{T}^{\sigma}\frac{\partial\mathbf{e}\left(t\right)}{\partial t}+\mathbf{Q}\frac{\partial\mathbf{e}\left(t\right)}{\partial t}-\mathbf{P}\mathbf{e}_{inc}\left(t\right)-\mathbf{Q}\frac{\partial\mathbf{e}_{inc}\left(t\right)}{\partial t}+\mathbf{f}=0 \tag{4.12}$$

where

$$S_{i,j}=\iiint_{V}\left[\nabla\times\mathbf{N}_{i}\left(\mathbf{r}\right)\right]\cdot\left[\frac{1}{\mu_{0}\hat{\mu}_{r}}\nabla\times\mathbf{N}_{j}\left(\mathbf{r}\right)\right]\,\mathrm{d}V \tag{4.13a}$$

$$T_{i,j}^{\epsilon}=\iiint_{V}\mathbf{N}_{i}\left(\mathbf{r}\right)\cdot\epsilon_{0}\hat{\epsilon}_{r}\mathbf{N}_{j}\left(\mathbf{r}\right)\,\mathrm{d}V \tag{4.13b}$$

$$T_{i,j}^{\sigma}=\iiint_{V}\mathbf{N}_{i}\left(\mathbf{r}\right)\cdot\hat{\sigma}\mathbf{N}_{j}\left(\mathbf{r}\right)\,\mathrm{d}V \tag{4.13c}$$

$$P_{i,j}=\iint_{S_{ABC}}\left[\vec{n}\times\mathbf{N}_{i}\left(\mathbf{r}\right)\right]\cdot\left[\frac{1}{\mu_{0}\hat{\mu}_{r}}\nabla\times\mathbf{N}_{j}\left(\mathbf{r}\right)\right]\,\mathrm{d}S \tag{4.13d}$$

$$Q_{i,j}=\iint_{S_{ABC}}Y_{0}\left[\vec{n}\times\mathbf{N}_{i}\left(\mathbf{r}\right)\right]\cdot\left[\vec{n}\times\mathbf{N}_{j}\left(\mathbf{r}\right)\right]\,\mathrm{d}S \tag{4.13e}$$

$$f_{i}=\iiint_{V}\mathbf{N}_{i}\left(\mathbf{r}\right)\cdot\frac{\partial\mathbf{J}_{imp}\left(\mathbf{r},t\right)}{\partial t}\,\mathrm{d}V \tag{4.13f}$$

We employ Newmark method with $\beta=1/4$ and $\gamma=1/2$. The relevant substitutions are:

$$u(t)\approx\frac{1}{4}u^{n+1}+\frac{1}{2}u^{n}+\frac{1}{4}u^{n-1} \tag{4.14a}$$

$$\frac{\partial u(t)}{\partial t}\approx\frac{1}{2\Delta t}u^{n+1}-\frac{1}{2\Delta t}u^{n-1} \tag{4.14b}$$

$$\frac{\partial^{2}u(t)}{\partial t^{2}}\approx\frac{1}{\left(\Delta t\right)^{2}}u^{n+1}-\frac{2}{\left(\Delta t\right)^{2}}u^{n}+\frac{1}{\left(\Delta t\right)^{2}}u^{n-1} \tag{4.14c}$$

Applying time discretization 4.14 on 4.12 forms resulting system of linear equations

$$\begin{aligned}\left[\frac{1}{4}\mathbf{S}+\frac{1}{\left(\Delta t\right)^{2}}\mathbf{T}^{\epsilon}+\frac{1}{2\Delta t}\mathbf{T}^{\sigma}+\frac{1}{2\Delta t}\mathbf{Q}\right]\mathbf{e}^{\{n+1\}}&=\left[-\frac{1}{2}\mathbf{S}+\frac{2}{\left(\Delta t\right)^{2}}\mathbf{T}^{\epsilon}\right]\mathbf{e}^{\{n\}}\\&+\left[-\frac{1}{4}\mathbf{S}-\frac{1}{\left(\Delta t\right)^{2}}\mathbf{T}^{\epsilon}+\frac{1}{2\Delta t}\mathbf{T}^{\sigma}+\frac{1}{2\Delta t}\mathbf{Q}\right]\mathbf{e}^{\{n-1\}}\\&+\left[\frac{1}{4}\mathbf{P}+\frac{1}{2\Delta t}\mathbf{Q}\right]\mathbf{e}_{inc}^{\{n+1\}}+\left[\frac{1}{2}\mathbf{P}\right]\mathbf{e}_{inc}^{\{n\}}+\left[\frac{1}{4}\mathbf{P}-\frac{1}{2\Delta t}\mathbf{Q}\right]\mathbf{e}_{inc}^{\{n-1\}}\\&-\frac{1}{4}\mathbf{f}^{\{n+1\}}-\frac{1}{2}\mathbf{f}^{\{n\}}-\frac{1}{4}\mathbf{f}^{\{n-1\}}\end{aligned} \tag{4.15}$$

where $\left[\frac{1}{4}\mathbf{S}+\frac{1}{\left(\Delta t\right)^{2}}\mathbf{T}^{\epsilon}+\frac{1}{2\Delta t}\mathbf{T}^{\sigma}+\frac{1}{2\Delta t}\mathbf{Q}\right]$ is the coefficient matrix, vector $\mathbf{e}^{\{n+1\}}$ represents unknown field coefficients in the most advanced time step and the rest is right-hand vector formed by already known quantities.

Dispersive media

The solver can handle electrically dispersive materials. The following text presents a brief description of the modeling procedure. For more detail see [10]. The time-invariant permittivity tensor $\hat{\epsilon}_r$ in the Ampere law 4.1b has to be replaced by a Debye model representation. Amelet-HDF specification [34] defines a multipole Debye model as

$$\hat{\epsilon}\left(\omega\right) = \hat{\epsilon}_\infty + \left(\hat{\epsilon}_s - \hat{\epsilon}_\infty\right) \sum_{p=1}^{P} \frac{G_p}{1 + j\omega\tau_p} \quad \text{where} \quad \sum_{p=1}^{P} G_p = 1 \tag{4.16}$$

where $\hat{\epsilon}_\infty$ is permittivity at optical frequency, $\hat{\epsilon}_s$ is permittivity at zero frequency, G_p and τ_p are weight and characteristic relaxation time of p-th pole respectively. The modeling procedure [10] assumes

$$\overset{\leftrightarrow}{\epsilon}\left(\omega\right) = \overset{\leftrightarrow}{\epsilon}_\infty + \overset{\leftrightarrow}{\chi}_e\left(\omega\right) = \overset{\leftrightarrow}{\epsilon}_\infty + \sum_{p=1}^{N_e} \frac{\overset{\leftrightarrow}{a}_{e,p}}{j\omega + b_{e,p}} \tag{4.17}$$

where $\overset{\leftrightarrow}{\epsilon}_\infty$ is permittivity tensor at optical frequency and $\overset{\leftrightarrow}{a}$ and b are parameters of the material. One can see that we get the same result using

$$\overset{\leftrightarrow}{a}_{e,p} = \frac{G_p}{\tau_p}\left(\hat{\epsilon}_s - \hat{\epsilon}_\infty\right) \quad \text{and} \quad b_{e,p} = \frac{1}{\tau_p} \tag{4.18}$$

The arrow notation of a tensor was used in order to denote relation between user definition (Amelet-HDF) and the modeling approach[10]. Now, we return to the original tensor notation. Applying Debye model and neglecting source terms in 4.5, we obtain [10]

$$\nabla \times \left[\frac{1}{\mu_0\hat{\mu}_r}\nabla \times \mathbf{E}\left(\mathbf{r},t\right)\right] + \epsilon_0\hat{\epsilon}_\infty\frac{\partial^2\mathbf{E}\left(\mathbf{r},t\right)}{\partial t^2} + \epsilon_0\hat{\chi}_e\left(t\right) * \frac{\partial^2\mathbf{E}\left(\mathbf{r},t\right)}{\partial t^2} = 0 \quad \mathbf{r} \in V \tag{4.19}$$

Since the program stores ϵ_∞ in the same place as the original $\hat{\epsilon}_r$, the third term of 4.19 containing time convolution makes the only difference from non-dispersive case. Applying the steps described in [10] will result in the following changes. Matrix $\mathbf{T}_\epsilon$ from 4.13b will change into [10]

$$T^\epsilon_{i,j} = \iiint_V \mathbf{N}_i\left(\mathbf{r}\right) \cdot \epsilon_0\hat{\epsilon}_\infty\mathbf{N}_j\left(\mathbf{r}\right)\,\mathrm{d}V + \mathbf{\Phi}^0_{i,j} \tag{4.20}$$

where

$$\Phi^0_{i,j} = \sum_{p=1}^{P} G_p\epsilon_0\left(1 - e^{-\frac{\Delta t}{2\tau_p}}\right) \iiint_V \mathbf{N}_i\left(\mathbf{r}\right) \cdot \left(\hat{\epsilon}_s - \hat{\epsilon}_\infty\right)\mathbf{N}_j\left(\mathbf{r}\right)\,\mathrm{d}V \tag{4.21}$$

In addition, right hand of the system of equations 4.15 will get additional term [10]

$$-\frac{1}{\left(\Delta t\right)^2}\mathbf{\Psi}^{\{n\}}_e \tag{4.22}$$

where

$$\mathbf{\Psi}^{\{n\}}_e = \mathbf{\Phi}^{1/2}\left[\mathbf{e}^{\{n\}} - 2\mathbf{e}^{\{n-1\}} + \mathbf{e}^{\{n-2\}}\right] + \sum_{p=1}^{P} e^{-\frac{\Delta t}{\tau_p}}\mathbf{\Psi}^{\{n-1\}}_e \tag{4.23}$$

$$\Phi^{1/2}_{i,j} = \sum_{p=1}^{P} G_p\epsilon_0\left(1 - e^{-\frac{\Delta t}{2\tau_p}}\right)\left(e^{-\frac{\Delta t}{2\tau_p}}\right) \iiint_V \mathbf{N}_i\left(\mathbf{r}\right) \cdot \left(\hat{\epsilon}_s - \hat{\epsilon}_\infty\right)\mathbf{N}_j\left(\mathbf{r}\right)\,\mathrm{d}V \tag{4.24}$$

Thin wire approximation

Aircraft modeling involves the use of wires. Geometric proportions of the wires are usually much smaller in comparison with other parts of the model. Meshing the wire as a volumetric object would require a markable increase of mesh density and computational demands. Thin wire approximation allows to solve the problem without previously mentioned difficulties.

The BUTFE solver employs the thin wire approach published in [40]. A simplified case (considering only a straight wire) of this approach can be found in [10]. This thin wire approximation is based on a combination of telegrapher's equations and TDFE. It splits the original problem into a one-dimensional solution of the wire current and the three-dimensional solution of the electric field in the surrounding area. The final system of equations provides bi-directional coupling between the wire current and the electric field via coupling matrices. An advantage of this approach is that the nodal mesh of the wire can cross a tetrahedral mesh of the surrounding area. This is really useful because mesh generators usually have a problem with attaching a tetrahedral mesh to a curve.

Only a brief summary of the method will follow. A more precise explanation can be found in [10] and [40]. All equations in the chapter oriented to thin wire approximation of straight wires are originally taken from [10].

The wire with unit vector $\hat{s}$ and radius a is virtually surrounded by a cylindrical surface of radius r_0 (see figure 4.12). It is assumed that the following telegrapher's equations are satisfied in the neighborhood of the wire

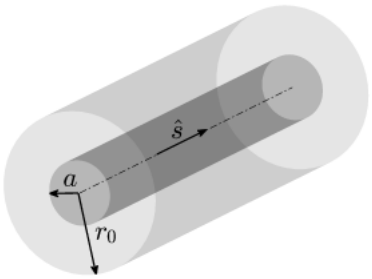


Fig. 4.12 *Wire surrounded by a virtual cylinder*

$$C_w \frac{\partial V}{\partial t} = -C_w \frac{\sigma}{\epsilon} V - \frac{\partial I}{\partial s} \tag{4.25a}$$

$$L_w \frac{\partial I}{\partial t} = -\frac{\partial V}{\partial s} + \left(\mathbf{E} \cdot \hat{s}|_{r=r_0} + V_{imp} - I R_w\right) \tag{4.25b}$$

where σ and ϵ are material parameters, V denotes voltage along the wire, I stands for wire current, R_w is possible resistive loading with unit of Ohm per meter, $\hat{s}$ denotes position-dependent unit vector of the wire and V_{imp} stands for possible impressed voltage source with unit of Volt per meter. Parameters C_w and L_w are in-cell capacitance and inductance

$$L_w = \frac{\mu}{2\pi} \ln \frac{r_0}{a}, \quad C_w = \frac{\epsilon\mu}{L_w} \tag{4.26}$$

where r_0 is radius of the cylindrical surface and a is the wire radius. Symbol μ is permeability of media. The telegrapher's equations 4.25 can be rewritten into a wave equation. With an assumption that the medium in the wire neighborhood is non-conducting, following finite element representation can be derived

$$\mathbf{T}^w \frac{\partial^2}{\partial t^2} \mathbf{i}(t) + \mathbf{R}^w \frac{\partial}{\partial t} \mathbf{i}(t) + \mathbf{S}^w \mathbf{i}(t) = \mathbf{f}^w(t) \tag{4.27}$$

where

$$T_{i,j}^{w} = L_w \int_s N_i^w(s) \cdot N_j^w(s)\, \mathrm{d}s \tag{4.28a}$$

$$R_{i,j}^{w} = R_w \int_s N_i^w(s) \cdot N_j^w(s)\, \mathrm{d}s \tag{4.28b}$$

$$S_{i,j}^{w} = \frac{L_w}{\mu\epsilon} \int_s \nabla N_i^w(s) \cdot \nabla N_j^w(s)\, \mathrm{d}s \tag{4.28c}$$

Symbol $\mathbf{i}(t)$ represents vector of nodal coefficients of the wire current, N^w denotes basis function of the 1D wire mesh and $\mathbf{f}^w$ is time-dependent excitation vector with elements

$$f_i^w(t) = \int_s N_i^w(s) \frac{\partial}{\partial t} \left(\mathbf{E}(\mathbf{r}, t) \cdot \hat{s}|_{r=r_0} + V^{imp}(s, t) \right) \mathrm{d}s \tag{4.29}$$

In order to achieve symmetric excitation between solution of the wire and surrounding area so that achieve stable formulation, a radial weighting function $g(r)$ satisfying

$$\int_{r \geq a} 2\pi r g(r)\, \mathrm{d}r = 1 \tag{4.30}$$

has been introduced:

$$g(r) = \begin{cases} 0 & : \quad r < a \\ \frac{1+\cos(\pi r/r_0)}{\pi\left(r_0^2 - a^2\right) - 2r_0^2/\pi(1+\cos(\pi a/r_0) + \pi a/r_0 \sin(\pi a/r_0))} & : \quad a \leq r \leq r_0 \\ 0 & : \quad r > r_0 \end{cases} \tag{4.31}$$

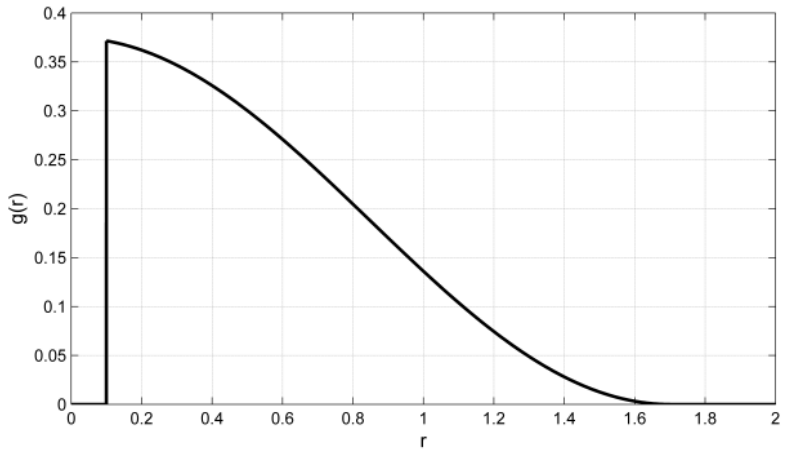


Fig. 4.13 *Radial weighting function $g(r)$ for $a = 0.1$ and $r_0 = 1.7$*

The excitation vector 4.29 can be now rewritten into

$$f_i^w(t) = \int_s N_i^w(s) \frac{\partial V_{imp}(s, t)}{\partial t} \mathrm{d}s + \sum_{j=0}^{N_{edge}} \frac{\partial e_j(t)}{\partial t} \iiint_V N_i^w(s)\, g(r)\, \mathbf{N}_j(\mathbf{r}) \cdot \hat{s}\, \mathrm{d}V \tag{4.32}$$

In order to make the excitation symmetric, current density is written as

$$\mathbf{J}_{imp}(\mathbf{r},t) = \sum_{i=1}^{N_{node}} N_i^w(s)\, i_i(t)\, g(r)\, \hat{s} \tag{4.33}$$

Excitation vector of the equation for the volumetric region will be given as

$$f_i(t) = -\sum_{i=1}^{N_{node}} \frac{\partial i_i(t)}{\partial t} \iiint_V N_i^w(s)\, g(r)\, \mathbf{N}_i(\mathbf{r}) \cdot \hat{s}\, \mathrm{d}V \tag{4.34}$$

The final system of equations (omitting ABC and dispersive media) can be now assembled as follows

$$\begin{bmatrix} \mathbf{T}^{\epsilon} & 0 \\ 0 & \mathbf{T}^{w} \end{bmatrix} \frac{\partial^2}{\partial t^2} \begin{bmatrix} \mathbf{e}(t) \\ \mathbf{i}(t) \end{bmatrix} + \begin{bmatrix} \mathbf{T}^{\sigma} & \mathbf{P}^{w\top} \\ -\mathbf{P}^{w} & \mathbf{R}^{w} \end{bmatrix} \frac{\partial}{\partial t} \begin{bmatrix} \mathbf{e}(t) \\ \mathbf{i}(t) \end{bmatrix} + \begin{bmatrix} \mathbf{S} & 0 \\ 0 & \mathbf{S}^{w} \end{bmatrix} \begin{bmatrix} \mathbf{e}(t) \\ \mathbf{i}(t) \end{bmatrix} = \begin{bmatrix} 0 \\ \tilde{\mathbf{f}}^{w} \end{bmatrix} \tag{4.35}$$

where the coupling matrix P^w and the remaining excitation vector $\tilde{\mathbf{f}}^w$ are defined as

$$P_{i,j}^w = \iiint_V N_i^w(s) \cdot g(r) \cdot \mathbf{N}_j(\mathbf{r}) \cdot \hat{s}\, \mathrm{d}V \tag{4.36a}$$

$$\tilde{f}_i^w = \int_s N_i^w(s) \frac{\partial V_{imp}(s,t)}{\partial t}\, \mathrm{d}s \tag{4.36b}$$

Magnetic field

The solver is computing only coefficients of the electric field. However the magnetic field can be obtained by taking advantage of Faraday law 4.1a which can be rewritten into

$$\mathbf{H}(\mathbf{r},t) = -\frac{1}{\mu_0 \hat{\mu}_r} \int_t \nabla \times \mathbf{E}(\mathbf{r},t)\ \mathrm{d}t \tag{4.37}$$

Since the electric field is approximated over the element as

$$\tilde{\mathbf{E}}(\mathbf{r},t) \approx \sum_{j=1}^{6} \tilde{\mathbf{N}}_j(\mathbf{r})\, e_j(t) \tag{4.38}$$

the magnetic field over an element can be obtained as follows

$$\tilde{\mathbf{H}}(\mathbf{r},t) \approx -\frac{1}{\mu_0 \hat{\mu}_r} \sum_{j=1}^{6} \nabla \times \tilde{\mathbf{N}}_j(\mathbf{r}) \int_t e_j(t)\ \mathrm{d}t \tag{4.39}$$

which requires only 18 new coefficients (6 edge-oriented shape functions × 3 Cartesian components) and additional storage for the time integral of the coefficients of the electric field.

Anisotropy

Despite the fact that BUTFE is not intended to handle anisotropic materials, it accepts material definition in the form of a full 3×3 tensor. It is rather preparation for an anisotropic PML [14, 15]. Such PML implementations usually require adding up to 26 new domains around the propagation media, where each domain absorbs an incoming wave in a given direction. All the domains behave in the same way. They differ only in the direction of the absorption (i.e. content of the material tensor).

Instead of defining fixed surrounding domains, an application of a rotation matrix has been introduced. The rotation matrix modifies the content of the original material tensor just as we would physically rotate the material inside the structure. Apart from the planned usage for PML (virtual rotating of the surrounding domains), the rotation can be user-defined through an Amelet-HDF file interface in case of different orientation of an anisotropic material with respect to the global coordinate system.

The following replacements have been made

$$\hat{\epsilon} \Rightarrow \mathbf{R}\left(\epsilon_0 \hat{\epsilon}_r\right)\mathbf{R}^{\top}, \quad \frac{1}{\hat{\mu}} \Rightarrow \mathbf{R}^{\top -1}\frac{1}{\mu_0 \hat{\mu}_r}\mathbf{R}^{-1}, \quad \hat{\sigma} \Rightarrow \mathbf{R}\hat{\sigma}\mathbf{R}^{\top} \tag{4.40}$$

where the original (relative) material tensors are

$$\hat{\epsilon}_r = \begin{pmatrix} \epsilon_r^{xx} & \epsilon_r^{xy} & \epsilon_r^{xz} \\ \epsilon_r^{yx} & \epsilon_r^{yy} & \epsilon_r^{yz} \\ \epsilon_r^{zx} & \epsilon_r^{zy} & \epsilon_r^{zz} \end{pmatrix}, \quad \hat{\mu}_r = \begin{pmatrix} \mu^{xx} & \mu^{xy} & \mu^{xz} \\ \mu^{yx} & \mu^{yy} & \mu^{yz} \\ \mu^{zx} & \mu^{zy} & \mu^{zz} \end{pmatrix}, \quad \hat{\sigma} = \begin{pmatrix} \sigma^{xx} & \sigma^{xy} & \sigma^{xz} \\ \sigma^{yx} & \sigma^{yy} & \sigma^{yz} \\ \sigma^{zx} & \sigma^{zy} & \sigma^{zz} \end{pmatrix} \tag{4.41}$$

and the rotation matrix is defined as

$$\mathbf{R} = \begin{bmatrix} X^x & X^y & X^z \\ Y^x & Y^y & Y^z \\ Z^x & Z^y & Z^z \end{bmatrix} \tag{4.42}$$

Here, the $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ represent coordinates of the orthonormal basis of a local coordinate system expressed in terms of the global coordinate system (x, y, z). Note that the default rotation matrix is the identity matrix (no rotation is applied).

Summary

This sub-chapter presents a mathematical formulation of the BUTFE solver considering all the approaches presented so far. It includes ABC, Debye media and thin-wire approximation of N wires. The resulting system of equations is as follows

$$\begin{aligned}\left[\frac{1}{(\Delta t)^2}\mathbf{A}+\frac{1}{2\Delta t}\mathbf{B}+\frac{1}{4}\mathbf{C}\right]\mathbf{v}^{\{n+1\}} &= \left[\frac{2}{(\Delta t)^2}\mathbf{A}-\frac{1}{2}\mathbf{C}\right]\mathbf{v}^{\{n\}} \\ &+\left[-\frac{1}{(\Delta t)^2}\mathbf{A}+\frac{1}{2\Delta t}\mathbf{B}-\frac{1}{4}\mathbf{C}\right]\mathbf{v}^{\{n-1\}} \\ +\left[\frac{1}{2\Delta t}\mathbf{D}+\frac{1}{4}\mathbf{E}\right]\mathbf{v}_{inc}^{\{n+1\}}+\left[\frac{1}{2}\mathbf{E}\right]\mathbf{v}_{inc}^{\{n\}}+\left[-\frac{1}{2\Delta t}\mathbf{D}+\frac{1}{4}\mathbf{E}\right]\mathbf{v}_{inc}^{\{n-1\}} & \\ -\frac{1}{4}\mathbf{f}^{\{n+1\}}-\frac{1}{2}\mathbf{f}^{\{n\}}-\frac{1}{4}\mathbf{f}^{\{n-1\}} & \\ +\frac{1}{4}\mathbf{g}^{\{n+1\}}+\frac{1}{2}\mathbf{g}^{\{n\}}+\frac{1}{4}\mathbf{g}^{\{n-1\}} & \\ -\frac{1}{(\Delta t)^2}\boldsymbol{\Psi}_e^{\{n\}} &\end{aligned} \tag{4.43a}$$

where

$$\mathbf{A}=\begin{bmatrix}\mathbf{T}^{\epsilon}+\boldsymbol{\Phi}^0 & 0 & \cdots & 0\\ 0 & \mathbf{T}_1^w & 0 & 0\\ \vdots & 0 & \ddots & 0\\ 0 & 0 & 0 & \mathbf{T}_N^w\end{bmatrix} \qquad \mathbf{D}=\begin{bmatrix}\mathbf{Q} & 0 & \cdots & 0\\ 0 & 0 & 0 & 0\\ \vdots & 0 & \ddots & 0\\ 0 & 0 & 0 & 0\end{bmatrix}$$

$$\mathbf{B}=\begin{bmatrix}\mathbf{T}^{\sigma}+\mathbf{Q} & \mathbf{P}_1^{w\top} & \cdots & \mathbf{P}_N^{w\top}\\ -\mathbf{P}_1^w & \mathbf{R}_1^w & 0 & 0\\ \vdots & 0 & \ddots & 0\\ -\mathbf{P}_N^w & 0 & 0 & \mathbf{R}_N^w\end{bmatrix} \qquad \mathbf{E}=\begin{bmatrix}\mathbf{P} & 0 & \cdots & 0\\ 0 & 0 & 0 & 0\\ \vdots & 0 & \ddots & 0\\ 0 & 0 & 0 & 0\end{bmatrix}$$

$$\mathbf{C}=\begin{bmatrix}\mathbf{S} & 0 & \cdots & 0\\ 0 & \mathbf{S}_1^w & 0 & 0\\ \vdots & 0 & \ddots & 0\\ 0 & 0 & 0 & \mathbf{S}_N^w\end{bmatrix} \tag{4.43b}$$

$$\boldsymbol{\Psi}_e^{\{n\}}=\boldsymbol{\Phi}^{1/2}\begin{bmatrix}\mathbf{e}^{\{n\}}-2\mathbf{e}^{\{n-1\}}+\mathbf{e}^{\{n-2\}}\\ 0\\ \vdots\\ 0\end{bmatrix}+\sum_{p=1}^{P}e^{-\frac{\Delta t}{\tau_p}}\boldsymbol{\Psi}_e^{\{n-1\}}$$

$$\mathbf{f}=\begin{bmatrix}\tilde{\mathbf{f}}\\ 0\\ \vdots\\ 0\end{bmatrix} \qquad \mathbf{g}=\begin{bmatrix}0\\ \tilde{\mathbf{f}}_1^w\\ \vdots\\ \tilde{\mathbf{f}}_N^w\end{bmatrix} \qquad \mathbf{v}=\begin{bmatrix}\mathbf{e}\\ \mathbf{i}_1\\ \vdots\\ \mathbf{i}_N\end{bmatrix} \qquad \mathbf{v}_{inc}=\begin{bmatrix}\mathbf{e}_{inc}\\ 0\\ \vdots\\ 0\end{bmatrix}$$

and

$$
\begin{aligned}
S_{i,j} &= \iiint_V \left[\nabla \times \mathbf{N}_i(\mathbf{r})\right] \cdot \left[\mathbf{R}^{\top -1} \frac{1}{\mu_0 \hat{\mu}_r} \mathbf{R}^{-1} \nabla \times \mathbf{N}_j(\mathbf{r})\right] \mathrm{d}V \\
T^{\epsilon}_{i,j} &= \iiint_V \mathbf{N}_i(\mathbf{r}) \cdot \mathbf{R}\left(\epsilon_0 \hat{\epsilon}_\infty\right) \mathbf{R}^{\top} \mathbf{N}_j(\mathbf{r}) \, \mathrm{d}V \\
T^{\sigma}_{i,j} &= \iiint_V \mathbf{N}_i(\mathbf{r}) \cdot \mathbf{R} \hat{\sigma} \mathbf{R}^{\top} \mathbf{N}_j(\mathbf{r}) \, \mathrm{d}V \\
P_{i,j} &= \iint_{S_{ABC}} \left[\vec{n} \times \mathbf{N}_i(\mathbf{r})\right] \cdot \left[\mathbf{R}^{\top -1} \frac{1}{\mu_0 \hat{\mu}_r} \mathbf{R}^{-1} \nabla \times \mathbf{N}_j(\mathbf{r})\right] \mathrm{d}S \\
Q_{i,j} &= \iint_{S_{ABC}} Y_0 \left[\vec{n} \times \mathbf{N}_i(\mathbf{r})\right] \cdot \left[\vec{n} \times \mathbf{N}_j(\mathbf{r})\right] \mathrm{d}S \\
\Phi^0_{i,j} &= \sum_{p=1}^{P} G_p \epsilon_0 \left(1 - e^{-\frac{\Delta t}{2\tau_p}}\right) \iiint_V \mathbf{N}_i(\mathbf{r}) \cdot \mathbf{R}\left(\hat{\epsilon}_{s,p} - \hat{\epsilon}_\infty\right) \mathbf{R}^{\top} \mathbf{N}_j(\mathbf{r}) \, \mathrm{d}V \\
\Phi^{1/2}_{i,j} &= \sum_{p=1}^{P} G_p \epsilon_0 \left(1 - e^{-\frac{\Delta t}{2\tau_p}}\right) \left(e^{-\frac{\Delta t}{2\tau_p}}\right) \iiint_V \mathbf{N}_i(\mathbf{r}) \cdot \mathbf{R}\left(\hat{\epsilon}_{s,p} - \hat{\epsilon}_\infty\right) \mathbf{R}^{\top} \mathbf{N}_j(\mathbf{r}) \, \mathrm{d}V \\
\tilde{f}_i &= \iiint_V \mathbf{N}_i(\mathbf{r}) \cdot \frac{\partial \mathbf{J}_{imp}(\mathbf{r}, t)}{\partial t} \, \mathrm{d}V \\
\left(\tilde{f}^w_n\right)_i &= \int_s \left(N^w_n\right)_i (s) \frac{\partial V_{n,imp}(s, t)}{\partial t} \, \mathrm{d}s
\end{aligned}
\tag{4.43c}
$$

Note that the first two rows of the final system of equations 4.43a handle the basic electric field behavior and its coupling with wire currents. The following row (containing $\mathbf{v}_{inc}$) takes care of the excitation defined by a boundary condition. Vector $\mathbf{f}$ represents current source and $\mathbf{g}$ stands for voltage source. Vectors $\mathbf{f}$ and $\mathbf{g}$ are set to zero by default (standard BUTFE code does not consider such sources). Matrix $\mathbf{\Phi}$ and vector $\mathbf{\Psi}$ are related to dispersive media.

The boundary excitation $\mathbf{E}_{inc}(\mathbf{r}, t)$ is approximated as

$$
\tilde{\mathbf{E}}_{inc}(\mathbf{r}, t) \approx \sum_{j=1}^{6} \mathbf{N}_j(\mathbf{r}) \left(e_{inc}(t)\right)_j \tag{4.44}
$$

which may cause some inaccuracy. However passing analytically computed results of terms $\nabla \times \mathbf{E}_{inc}(\mathbf{r}, t)$ and $\vec{n} \times \mathbf{E}_{inc}(\mathbf{r}, t)$ (see 4.10) as inputs of the solver would complicate implementation into the framework significantly.

Approximation of the magnetic field over an element is given by

$$
\tilde{\mathbf{H}}(\mathbf{r}, t) \approx -\mathbf{R}^{\top -1} \frac{1}{\mu_0 \hat{\mu}_r} \mathbf{R}^{-1} \sum_{j=1}^{6} \nabla \times \tilde{\mathbf{N}}_j(\mathbf{r}) \int_t e_j(t) \, \mathrm{d}t \tag{4.45}
$$

The formula contains in addition to the original one 4.39 rotation of the material tensor.

4.1.5 FEM approximation

Approximation of desired quantity is a key point of the finite element method. The integral of an unknown field quantity is replaced by a multiplication of unknown coefficients and integrals of the basis functions which are easy to determine.

The BUTFE module expresses the unknown field quantity using coefficients and basis functions. Various mathematical operations applied on basis functions lead to matrices of coefficients for separate tetrahedrons. A left-right multiplication by a coupling matrix forms matrices of coefficients related to global edges. The direction of the global edges is set from a node with a lower number to a node with a higher number.

Since the BUTFE solver employs 1D nodal elements for an approximation of wire currents and 3D edge elements for an approximation of the electric field, the description has been separated into two subchapters.

Aproximation in 1D

One-dimensional approximation is used for discretization of wires. Wires are discretized into a set of line segments. Each segment (element) is expressed in terms of simplex coordinates

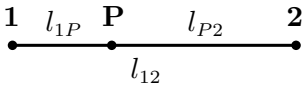


Fig. 4.14 *Line segment*

$$\xi_1(s) = \frac{l_{P2}}{l_{12}} \quad \xi_2(s) = \frac{l_{1P}}{l_{12}} \quad s \in [1,2] \tag{4.46}$$

where P is the point of interest on the line segment and l denotes the length of a line formed by given nodes (see fig. 4.14). It is clear that the i-th simplex coordinate takes value 1 at the i-th node, vanishes on the other one, $\sum_{i=1}^{2} \xi_i(s) = 1$ and $\xi_i(s) \in [0,1]$.

Shape function is a node-oriented scalar function defined as follows:

$$\tilde{N}_1^w(s) = \xi_1(s), \quad \tilde{N}_2^w(s) = \xi_2(s) \tag{4.47}$$

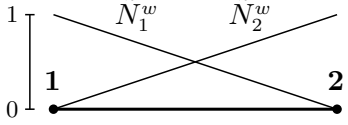


Fig. 4.15 *Nodal shape function*

Approximation in 3D

The structure is discretized into a set of tetrahedrons where each tetrahedron is expressed in terms of simplex coordinates. The local simplex coordinate system $\zeta_{1..4}$ satisfies for a given tetrahedron

$$\zeta_1(\mathbf{r}) = \frac{V_{P234}}{V_{1234}} \quad \zeta_2(\mathbf{r}) = \frac{V_{1P34}}{V_{1234}}$$
$$\zeta_3(\mathbf{r}) = \frac{V_{12P4}}{V_{1234}} \quad \zeta_4(\mathbf{r}) = \frac{V_{123P}}{V_{1234}} \quad \mathbf{r} \in V \tag{4.48}$$

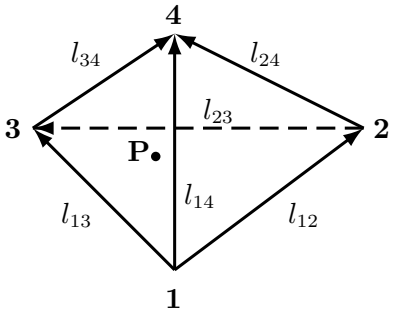


Fig. 4.16 *Tetrahedron*

where P is the point of interest inside the tetrahedron and V denotes the volume of a tetrahedron formed by given nodes (see fig. 4.16). It is clear that the i-th simplex coordinate takes value 1 at i-th node, vanishes on the opposite face, $\sum_{i=1}^{4} \zeta_i(\mathbf{r}) = 1$ and $\zeta_i(\mathbf{r}) \in [0,1]$.

The simplex coefficients $a_1..d_4$ preserve the following relation between the simplex and Cartesian system:

$$\zeta_i(\mathbf{r}) = a_i + b_i P^x + c_i P^y + d_i P^z \quad \mathbf{r} \in V \tag{4.49}$$

where P^x, P^y and P^z are the Cartesian components of the point of interest.

Shape function is edge-oriented vector function defined as follows:

$$\tilde{\mathbf{N}}_k(\mathbf{r}) = l_{i,j}\left[\zeta_i(\mathbf{r})\nabla\zeta_j(\mathbf{r}) - \zeta_j(\mathbf{r})\nabla\zeta_i(\mathbf{r})\right] \quad k = 1..6 \tag{4.50}$$

where $l_{i,j}$ is the length of the k-th edge going from the i-th towards the j-th node. Orientation of the local edges related to a given shape function and their node numbers are presented in figure 4.16 and table 4.1. Figure 4.17 shows the geometrical representation of a shape function. Arrows represent the shape function related to the edge marked by a solid line. One may see that the shape function takes on the remaining (dotted) edges only the perpendicular component (or zero value). A tangential component along any edge can be set only using the coefficient and shape function belonging to that edge. Since the relevant coefficient value is shared between adjacent elements, tangential continuity of the computed field is guaranteed.

Edge *k*	**Node *i***	**Node *j***
1	1	2
2	1	3
3	1	4
4	2	3
5	2	4
6	3	4

Tab. 4.1 *Edge numbering*

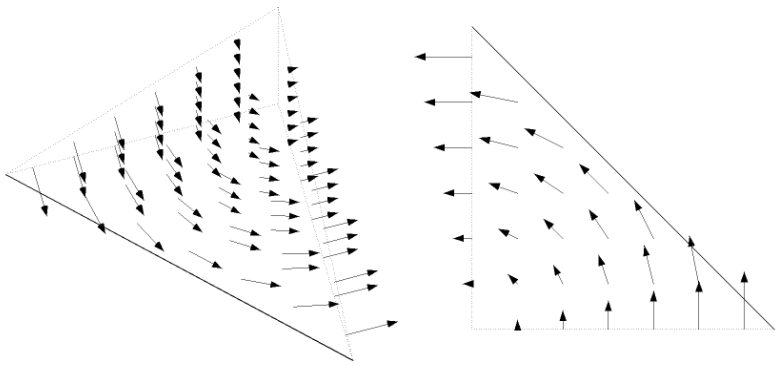


Fig. 4.17 *Shape function of the tetrahedron: 3D view (left), top view (right)*

For further information about obtaining simplex coefficients and computation with basis functions see [42].

4.1.6 MATLAB code

Although most of the thesis is related to the code written in C and the MATLAB code is rather old, the MATLAB code is easier and more transparent to describe the code developed in MATLAB when compared with the C code. The system of orientation of global edges, handling simplex coefficients, usage of coupling matrix and treatment of the main time domain loop remained more or less the same.

The former state of the module was described in the technical report for the project [43]. The code was divided into three parts. The first one solved the eigenmode analysis, the second one solved propagation of the EM wave in the frequency domain and the last one analyzed the structure in the time domain. A description of these codes will follow.

Eigenmode analysis

Parameters At the beginning it is necessary to set all constants and controlling variables. The variables `mode_no`, `accuracy` and `sigma` set the parameters of the MATLAB function *eigs*() that computes eigenvalues. The variable `field` determines whether the electric or magnetic field distribution is computed. The sequence number of the mode to display was given by the variable `disp_mode`.

```
c = 299792458;
e0 = 8.854187817e-12;
m0 = pi*4e-7;
mesh_path = 'GiD_mesh.h5';
mode_no = 100;
accuracy = 1e-10;
sigma = 20;
disp_mode = 1;
field = 'E';
```

Code 4.1 *Setting of the constants and parameters [43].*

Import of the mesh The mesh is imported from the Amelet-HDF file in the path defined by the variable `mesh_path`. MATLAB has implemented two methods of the accessing the HDF5 format: the "high-level" and the "low-level" one. The first one is more compact and easier to implement, the second one is similar to HDF5 C API [44]. Due to the lack of specific functions in the high-level access, the low-level one was used.

Code 4.2 shows opening of the input file and loading of the list of nodes. The file is opened as read-only using the default file access properties. The list of nodes is loaded from the dataset */mesh/gid/all/nodes*. Since the program GiD uses the path */mesh/gid/all*[8], all the paths in the mesh part of the code are set absolutely. Each identifier has to be closed using an appropriate closing function.

```
file_fid = H5F.open(mesh_path,'H5F_ACC_RDONLY','H5P_DEFAULT');

object_nodes_did = H5D.open(file_fid,'/mesh/gid/all/nodes');
NODES = H5D.read(object_nodes_did,'H5ML_DEFAULT','H5S_ALL',...
'H5S_ALL','H5P_DEFAULT')';
H5D.close(object_nodes_did);
```

Code 4.2 *Opening the HDF5 file, loading of the list of the nodes [43].*

[8] A newer version of the GiD plug-in uses */mesh/gid/unstructured*.

The list of elements is stored in two datasets `elementTypes` and `elementNodes`. These datasets are loaded in the same way as in the case of the nodes. For computation purposes, they are converted into matrices containing elements of the same type. The dataset `elementNodes` is divided into chunks with the size defined by the dataset `elementTypes` according to the Amelet-HDF documentation [34].

The tool looks for the following (unstructured) shape types: *bar2*, *tri3* and *tetra4* (*bar2* and *tri3* in the case of the 2D analysis). After loading all the elements, they have to be formed in groups in order to distinguish between individual parts of the mesh. The */mesh/gid/all/groupGroup* is not taken into account.

All main elements are stored in the matrix `ELEMENTS`. In the case of the 3D computation, the main elements are tetrahedrons. The number of rows equals the number of elements in the mesh.

Triangles (or eventually lines) are used only for the description of the boundary conditions. The program looks for PEC and PMC. If the `field` variable is set to 'E' (electric field), the program creates the list of PEC edges `EGred` in order to allow a reduction of the final system of equations.

The list of global numbered edges is created as follows:

- all possible combinations of the nodes according to the elements are created,
- they are sorted by node number,
- only unique combinations form the list of global numbered edges.

Thanks to the previous procedure, each global edge is oriented from the node with a smaller number to the node with a higher number. The list of PEC edges `EGred` is converted from the two-column matrix containing node numbers into a list

```
EDGES = unique(sort([ELEMENTS(:,1) ELEMENTS(:,2);
    ELEMENTS(:,1) ELEMENTS(:,3);
    ELEMENTS(:,1) ELEMENTS(:,4);
    ELEMENTS(:,2) ELEMENTS(:,3);
    ELEMENTS(:,2) ELEMENTS(:,4);
    ELEMENTS(:,3) ELEMENTS(:,4)],2),'rows');

[temp,EGred] = ismember(EGred,EDGES,'rows');

[temp,ind] = ismember(sort([ELEMENTS(:,1) ELEMENTS(:,2);
    ELEMENTS(:,1) ELEMENTS(:,3);
    ELEMENTS(:,1) ELEMENTS(:,4);
    ELEMENTS(:,2) ELEMENTS(:,3);
    ELEMENTS(:,2) ELEMENTS(:,4);
    ELEMENTS(:,3) ELEMENTS(:,4)],2),EDGES,'rows');

ELEMENTS(:,6:30) = zeros(size(ELEMENTS,1),25);
ELEMENTS(:,6:11) = reshape(ind,size(ELEMENTS,1),6);
```

Code 4.3 *Global edges numbering, the matrix of the elements [43].*

of indices connected to the list of the global edges `EDGES`. The remaining half of the Code 4.3 incorporates information about which edges it consists of. This will be stored in the matrix `ELEMENTS`.

Subsidiary variables This step creates matrices containing information about material properties. The properties are set in GiD through the name of a layer. The name syntax is as follows: `<real_name>#e=<er_x>;<er_y>;<er_z>#m=<mr_x>;<mr_y>;<mr_z>`, where:

`<real_name>` is the name of the layer,
`<er_x>` means the relative permittivity in a given direction and
`<mr_x>` the relative permeability in a given direction.

The resulting variable `materials` contains ID and 3 diagonal components of the permittivity and permeability tensor for each material. Subsidiary matrices `Er` and `Mr` store these material components (xx, yy and zz) for each element.

```
materials = [unique(ELEMENTS(:,5)) ...
zeros(numel(unique(ELEMENTS(:,5))),6)];
for k = 1:size(materials,1)
  temp = regexp(NAMES{materials(k)}, '#e=([i0-9.\-+]*);([i0-9.\-...
  +]*);([i0-9.\-+]*)','tokens');
  materials(k,2:4) = str2double(temp{:});
  temp = regexp(NAMES{materials(k)}, '#m=([i0-9.\-+]*);([i0-9.\-...
  +]*);([i0-9.\-+]*)','tokens');
  materials(k,5:7) = str2double(temp{:});
  Er(ELEMENTS(:,5)==materials(k,1),:) = ...
  repmat(materials(k,2:4),sum(ELEMENTS(:,5)==materials(k,1)),1);
  Mr(ELEMENTS(:,5)==materials(k,1),:) = ...
  repmat(materials(k,5:7),sum(ELEMENTS(:,5)==materials(k,1)),1);
end
```

Code 4.4 *Subsidiary variables `materials`, `Er` and `Mr` [43].*

In order to efficiently compute the rest of the subsidiary variables, the matrix `ELM` containing all the coordinates of each element is created. The tetrahedron's volumes (their sextuple's) are stored in the 12-th column of the matrix `ELEMENTS`. In the 2D case, doubles of the triangle's areas are stored in the 8-th column.

Matrix `l` stores lengths of the (global) edges. Vector `b1` contains the derivatives of the first simplex coordinate of all the elements along the x axis. The remaining coefficients `b2`...`d4` are derived similarly.

A list of non-reduced edges `EGnr` is derived from the list of all edges as the complement of the reduced edges `EGred`.

```
ELM = double([NODES(ELEMENTS(:,1),1:3) NODES(ELEMENTS(:,2),1:3) ...
NODES(ELEMENTS(:,3),1:3) NODES(ELEMENTS(:,4),1:3)]);
ELEMENTS(:,12) = ELM(:,1).*ELM(:,8).*ELM(:,6) - ...
ELM(:,1).*ELM(:,5).*ELM(:,9) + ELM(:,4).*ELM(:,2).*ELM(:,9) - ...
ELM(:,4).*ELM(:,8).*ELM(:,3) - ELM(:,7).*ELM(:,2).*ELM(:,6) + ...
ELM(:,7).*ELM(:,5).*ELM(:,3) + ELM(:,1).*ELM(:,5).*ELM(:,12) - ...
ELM(:,1).*ELM(:,11).*ELM(:,6) - ELM(:,4).*ELM(:,2).*ELM(:,12) + ...
ELM(:,4).*ELM(:,11).*ELM(:,3) + ELM(:,10).*ELM(:,2).*ELM(:,6) - ...
ELM(:,10).*ELM(:,5).*ELM(:,3) - ELM(:,1).*ELM(:,8).*ELM(:,12) + ...
ELM(:,1).*ELM(:,11).*ELM(:,9) + ELM(:,7).*ELM(:,2).*ELM(:,12) - ...
ELM(:,7).*ELM(:,11).*ELM(:,3) - ELM(:,10).*ELM(:,2).*ELM(:,9) + ...
ELM(:,10).*ELM(:,8).*ELM(:,3) + ELM(:,4).*ELM(:,8).*ELM(:,12) - ...
ELM(:,4).*ELM(:,11).*ELM(:,9) - ELM(:,7).*ELM(:,5).*ELM(:,12) + ...
ELM(:,7).*ELM(:,11).*ELM(:,6) + ELM(:,10).*ELM(:,5).*ELM(:,9) - ...
ELM(:,10).*ELM(:,8).*ELM(:,6);

l = sqrt([sum((ELM(:,4:6)-ELM(:,1:3)).^2,2) ...
sum((ELM(:,7:9)-ELM(:,1:3)).^2,2) ...
sum((ELM(:,10:12)-ELM(:,1:3)).^2,2) ...
sum((ELM(:,7:9)-ELM(:,4:6)).^2,2) ...
```

```
sum((ELM(:,10:12)-ELM(:,4:6)).^2,2) ...
sum((ELM(:,10:12)-ELM(:,7:9)).^2,2)];

b1 = (- (ELM(:,8).*ELM(:,12) - ELM(:,9).*ELM(:,11)) + ...
(ELM(:,5).*ELM(:,12) - ELM(:,6).*ELM(:,11)) - ...
(ELM(:,5).*ELM(:,9) - ELM(:,6).*ELM(:,8)))./(ELEMENTS(:,12));

EGnr = setdiff(1:EGn,EGred);
```

Code 4.5 *Subsidiary variables* `ELM`, `l` *and derivations of the simplex coordinates [43].*

Coupling matrix The coupling matrix is created in the Code 4.6. Vector `Cl` determines the local numbers(edges), vector `Cg` determines the global numbers and vector `Cv` contains values 1 or -1. The value 1 is written in the case of identical directions of the local and corresponding global edge, value -1 is written in the case of opposite directions. The size of the coupling matrix is given as (number_of_edges_per_element · `ELn`) × `EGn`.

```
Cl = 1:ELn*6;
Cg = reshape(ELEMENTS(:,6:11)',1,ELn*6);
Cv = reshape([-1+(ELEMENTS(:,1)<ELEMENTS(:,2))*2 ...
-1+(ELEMENTS(:,1)<ELEMENTS(:,3))*2 ...
-1+(ELEMENTS(:,1)<ELEMENTS(:,4))*2 ...
-1+(ELEMENTS(:,2)<ELEMENTS(:,3))*2 ...
-1+(ELEMENTS(:,2)<ELEMENTS(:,4))*2 ...
-1+(ELEMENTS(:,3)<ELEMENTS(:,4))*2]',1,ELn*6);
C = sparse(Cl,Cg,Cv,ELn*6,EGn);
```

Code 4.6 *Generation of the coupling matrix [43].*

Matrices of the coefficients In order to speed-up the run of the program, only the upper triangular part and the diagonal of the matrices of the coefficients are assembled. The lower triangular part is added at the end of the creation process.

Matrices `Se` and `Te` contain the coefficients for the final system of equations. Every row represents the upper triangular part with the diagonal of each square matrix originally of size 6 (3 for a 2D case). Code 4.7 shows the assembly of matrix `SE` characterizing the $\int_\Omega \nabla\times \vec{N}_i \cdot \nabla\times \vec{N}_j \, \mathrm{d}\Omega$. The code converts the coefficients represented by 21 values per each element into a symmetric diagonal matrix of coefficients `StE`. Coefficients are associated thanks to the coupling matrix `C` and the (PEC)boundary condition is applied. The resulting matrix `SE` is ready to be used in the system of linear equations.

```
r = [1 1 1 1 1 1 2 2 2 2 2 3 3 3 3 4 4 4 5 5 6];
s = [1 2 3 4 5 6 2 3 4 5 6 3 4 5 6 4 5 6 5 6 6];
temp = reshape(repmat(0:ELn-1,21,1),1,21*ELn)*6;
Xr = repmat(r,1,ELn)+temp;
Xs = repmat(s,1,ELn)+temp;

StE = reshape(Se',1,ELn*21);

StE = sparse(Xr,Xs,StE,ELn*6,ELn*6);

StE = StE+triu(StE,1)';

SE = C'*StE*C;

SE = SE(EGnr,EGnr);
```

Code 4.7 *Generation of the matrix of the coefficients [43].*

Computation Computation lies in solving the eigenvalue problem. The MATLAB built-in function *eigs()* is employed. It requires sparse input matrices (`SE`, `TE`) and allows to set up the number of requested eigenvalues `mode_no` around a value `sigma` with desired accuracy `accuracy`. The computed field coefficients are supplemented with the zero values at the PEC boundaries.

A list of critical frequencies is built. At the beginning, the list may include degenerated spurious solutions. The first valid frequency is detected by the difference in at least 6 digits in two consequent values. Vector `Ed` is used to display the field distribution given by `disp_mode`.

```
opts = struct('tol',accuracy,'disp',0);
[Eq,k] = eigs(SE,TE,mode_no,sigma,opts);
[k,temp] = sort(real(sqrt(diag(k))));
Eq = Eq(:,temp);
E = zeros(EGn,size(Eq,2));
E(EGnr,:) = Eq;

f_krit = (c/(2*pi))*k;

E(:,f_krit==0) = [];
f_krit(f_krit==0) = [];
sel = logical([0; abs(diff(f_krit))>1e6]);
Es = E(:,sel);
f_krits = f_krit(sel);
Ed = Es(:,disp_mode);
```

Code 4.8 *Computation of the electric field [43].*

Post-processing The values of the electric/magnetic field in the centroids of the tetrahedrons are computed using the edge coefficients as $E^x = \sum_{k=1}^{n} c_k N_k^x$.

Expression `E(ELEMENTS(:,6))` denotes computed coefficients c_1 for the first edge of each element. The front of this expression is multiplied by 1 or -1 according to the relative orientation of the local and relevant global edge. The rest represents the basis function N_k^x.

```
Ex = 0.25*(...
(-1+(ELEMENTS(:,1)<ELEMENTS(:,2))*2).*E(ELEMENTS(:,6)).*l(:,1).*...
(b2-b1)+...
(-1+(ELEMENTS(:,1)<ELEMENTS(:,3))*2).*E(ELEMENTS(:,7)).*l(:,2).*...
(b3-b1)+...
(-1+(ELEMENTS(:,1)<ELEMENTS(:,4))*2).*E(ELEMENTS(:,8)).*l(:,3).*...
(b4-b1)+...
(-1+(ELEMENTS(:,2)<ELEMENTS(:,3))*2).*E(ELEMENTS(:,9)).*l(:,4).*...
(b3-b2)+...
(-1+(ELEMENTS(:,2)<ELEMENTS(:,4))*2).*E(ELEMENTS(:,10)).*l(:,5).*...
(b4-b2)+...
(-1+(ELEMENTS(:,3)<ELEMENTS(:,4))*2).*E(ELEMENTS(:,11)).*l(:,6).*...
(b4-b3));
```

Code 4.9 *Computation of the field component Ex [43].*

The remaining field components are computed similarly. In the case of 2D computation, one may expect only three expressions related to three edges of a triangle.

Frequency domain propagation

Most of the steps of the program remain the same as described in the previous subchapter. The frequency of interest has to be defined. The desired frequency `f` is used for the computation of the free space wave number `k0`.

```
f = 3.2e8;
w2 = (2*pi*f)^2;
k2 = w2*e0*m0;
```

Code 4.10 *Definition of the frequency [43].*

Importing the mesh, subsidiary variables, computation and coupling matrix and FE matrix assembly can be done exactly in the same way as in the case of eigenmode analysis.

The excitation has to be defined in order to examine the frequency response of the structure. This is realized by setting the coefficients of appropriate global edges via the function *plot_geom()*. This function displays the whole mesh of the analyzed structure and allows to setup the excitation. The edges previously denoted as PEC are depicted with slashed lines and can not be used for the excitation. The user can then choose the edges to be excited simply by clicking on them once or twice to define their direction. This situation is depicted in figure 4.18.

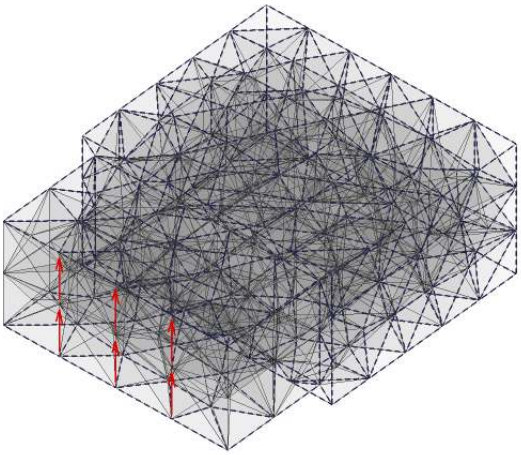


Fig. 4.18 *Excitation of the structure [43].*

The main difference is in the computation part of the code (see Code 4.11). The system of linear equations can be solved using the Gaussian elimination. The variable `EGexc` is a column vector containing coefficient values on the positions of the excitation edges.

```
LE = SE-k2*TE;
EGexc = plot_geom(NODES,EDGES,ELEMENTS,EGred);
EGexc(EGred) = [];
E = zeros(EGn,1);
E(EGnr,:) = LE\EGexc;
```

Code 4.11 *Computation of the field approximation* `E` *[43].*

Coefficients of the vector `EGexc` corresponding to the PEC walls have to be taken out to fit the system of equations. The resulting coefficients should be saved in appropriate positions (non-reduced edges) in the column vector `E`. Once the approximation coefficients are known, post-processing can be done in the same way as in the previous case.

Time domain analysis

This type of analysis computes the field approximation for the defined set of time steps. Although the formulation of the problem is derived in another domain than both the previously described programs, a lot of steps of the algorithm remains the same.

The time domain algorithm differs from the frequency domain one in the use of a time loop. Variables for the time domain loop are set up as seen in Code 4.12.

```
Step = 1000;
cdt = min(pdist(NODES))/sqrt(3);
dt = cdt/c;
t = linspace(dt, dt*Step,Step);
exc = (1-exp(-t*f)).*sin(2*pi*f*t);
Y = TE/(cdt^2) + SE/4;
Z = 2*TE/(cdt^2) - SE/2;
E = zeros(size(SE,1),3);
timeline = zeros(size(SE,1),Step);
EGexc(EGred) = [];
```

Code 4.12 *Setup variables for the time domain loop [43].*

The length of the loop is controlled by the user who sets the variable `Step`. The choice of `dt` is usually made according to the Courant-Friedrichs-Lewy condition. The vector `t` stores all multiples of parameter `dt` up to `Step*dt`.

The vector `exc` contains values of the excitation pulse in the time steps defined by vector `t`. Matrices `Y` and `Z` are used for sake of the readability and lucidity of the written code. The matrix `timeline` is allocated to store the column vectors `E` for every time step of the analysis. Finally, the coefficients belonging to PEC walls are removed from `EGexc`.

The time domain loop is depicted in Code 4.13. The first line saves the coefficients of the excitation pulse `exc` for the current time step `step` to the column vector `g`. In the next run, Gaussian elimination is done and the columns of matrix `E` containing information about the field approximation in last three time steps are shifted.

```
for step = 1:Step
  g = EGexc*exc(step);
  E(:,3) = Y \ (Z*E(:,2) - Y*E(:,1) - g);
  E(:,1) = E(:,2);
  E(:,2) = E(:,3);
  timeline(:,step) = E(:,3);
end
```

Code 4.13 *The time domain loop [43].*

Finally, the last achieved time step `E(:,3)` is stored in the matrix `timeline`. This matrix can be then used for depicting the change in the field distribution during the time dedicated for the analysis.

4.1.7 Results

A waveguide with dielectric obstacles

This test case compares the BUTFE solver with the time-domain solver of the commercial FEM-based software COMSOL Multiphysics©. The model is formed by a rectangular waveguide containing two dielectric obstacles. The waveguide with dimensions 22.86 x 10.16 x 100 mm (x, y, z) is depicted in figure 4.19.

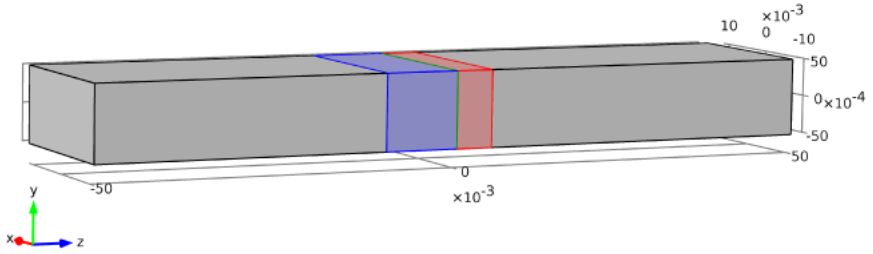


Fig. 4.19 *Waveguide geometry*

Walls are considered as perfectly conducting (PEC). The waveguide is excited on the left side (z = -0.05 m) and terminated with ABC on the right side (z = 0.05 m). The first obstacle (blue) has permittivity $\epsilon_r = 3$, it is 10 mm wide with center at z = -0.005 m. The second obstacle (red) has $\epsilon_r = 5$, width 5 mm and center at $z = 0.0025$ m.

The excitation pulse is set as follows:

$$E_y = \exp\left\{-\left[2 \cdot 10^9 (t - 1.2 \cdot 10^{-9})\right]^2\right\} \cos\left(2\pi 10^{10} t\right) \cos\left(\frac{\pi x}{0.02286}\right) (V/m) \qquad (4.51)$$

The excitation pulse is formed by three parts. The Gaussian pulse helps to fulfill the time condition of zero derivative of the field quantity at the beginning of the simulation. The second part performs a frequency shift into the operation band of the waveguide and the third one shapes the spatial distribution to fit boundary conditions of the port. Simulation time was set to 3.5 ns. See figure 4.20 for a graph of the excitation pulse.

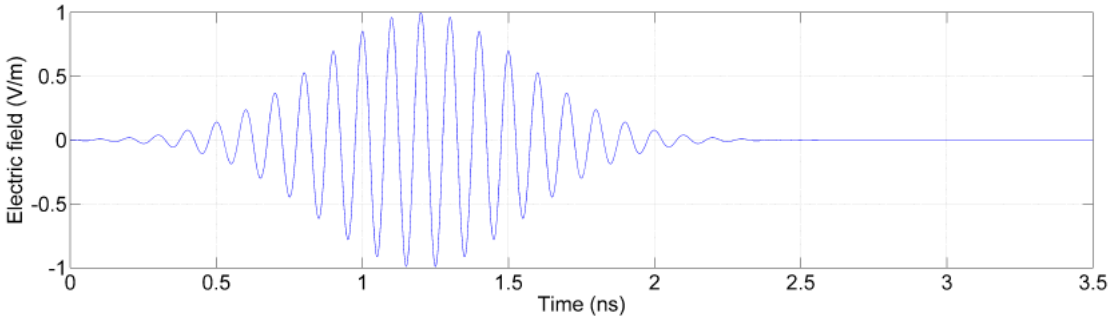


Fig. 4.20 *Excitation pulse (x = 0 m, z = -0.05 m)*

Comparison of the results in both the time and spatial domain would be demanding and thus only partial comparisons have been made.

The time domain comparison is performed for four discrete points in the middle of the structure ($x = 0$, $y = 0$). Two points are located in the middle of the obstacles and the other two are 5 mm in front of and after them (z_1 = -0.015 m, z_2 = -0.005 m, z_3 = 0.0025 m, z_4 = 0.01 m). See figure 4.21 for placement of the test points.

Fig. 4.21 *Test points*

Only the dominant y-component was observed. The simulation was performed three times in order to avoid inaccuracy caused by underestimation of the mesh density. The COMSOL provides some automatic meshing capability which sets mesh density with respect to the geometry. The user can set element size from *Extremely coarse* to *Extremely fine*. COMSOL Multiphysics© simulations were performed with the *Element size* option set to *Normal* (760 elements), *Finer* (2251 elements) and *Extremely fine* (46257 elements). The BUTFE solver simulations were performed with 55197, 104990 and 229702 elements. Differences between the amount of elements are given by the fact that COMSOL Multiphysics© solves magnetic vector potential and uses higher order approximation.

Figures 4.22 to 4.25 compare all 6 simulations for the given points. COMSOL simulations are in red, BUTFE simulations in blue. Table 4.2 gives a detailed description of the figures including computation times. Simulations were performed on a computer with Core2 Quad Q9450 processor (4 x 2.66 GHz) and 8 GB RAM.

Solver	**Line**	**Number of the elements (-)**	**Time steps (-)**	**Computation time (h:m:s)**
COMSOL	**red dotted**	760	3500	0:08:32
	red dash-dot	2251	3500	0:23:21
	red solid	46257	3500	8:08:43
BUTFE	**blue dotted**	55197	3500	0:07:34
	blue dash-dot	104990	5000	0:21:26
	blue solid	229705	5000	0:42:05

Tab. 4.2 *Performed simulations*

Influence of the time discretization was observed as well. Since the influence of spatial discretization was seen to be stronger, the number of time steps was reduced for dense meshes and set to a default value afterwards.

Additional windows show a detailed view in order to see differences between various simulations.

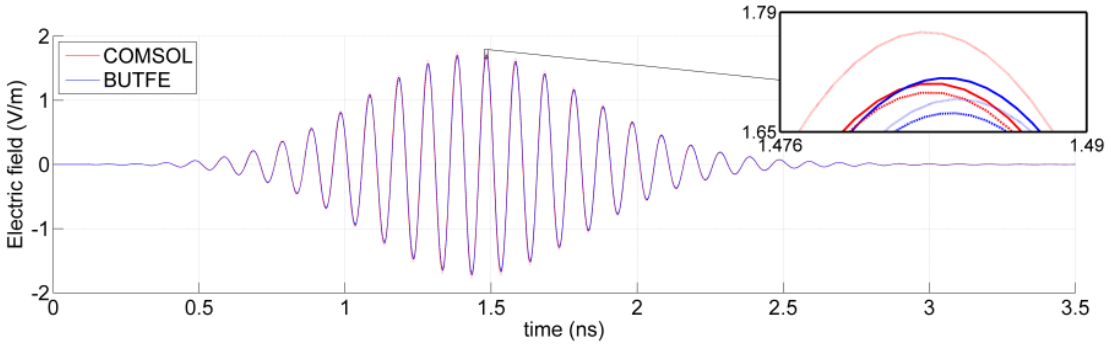


Fig. 4.22 *E_y at point [0, 0, -0.015] m*

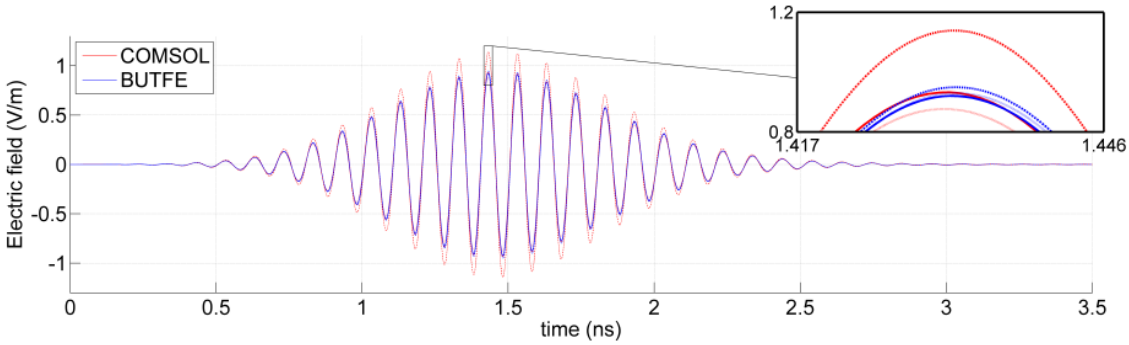


Fig. 4.23 *E_y at point [0, 0, -0.005] m*

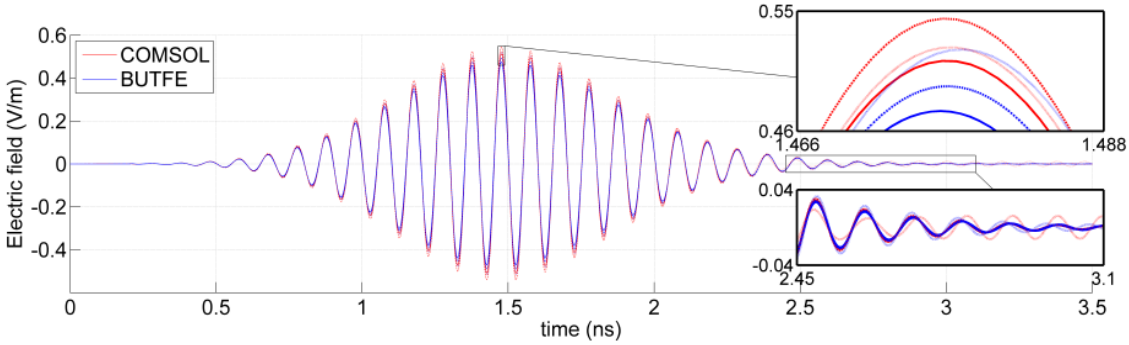


Fig. 4.24 *E_y at point [0, 0, 0.0025] m*

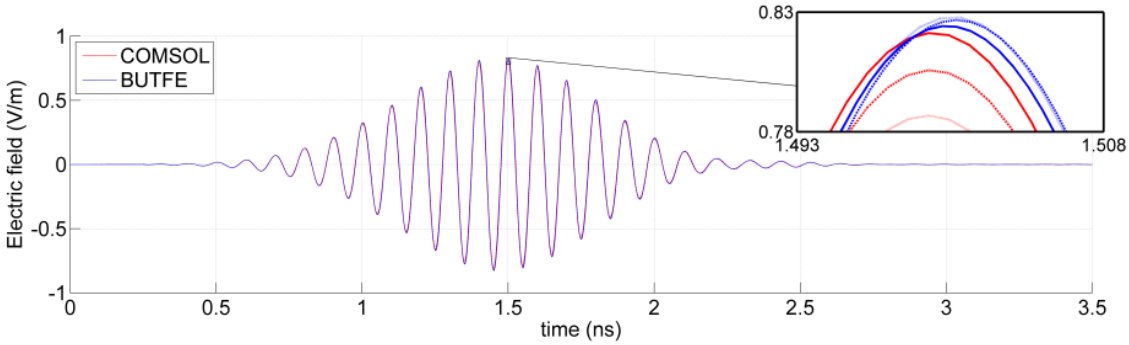


Fig. 4.25 *E_y at point [0, 0, 0.1] m*

Spatial distribution of the electric field at randomly given times is depicted in figures 4.26 and 4.27. In this way, we do not try to imitate an accurate comparison. The comparison gives us a clear picture about the electric field distribution inside the whole structure at a specific time ($t = 2.182$ ns).

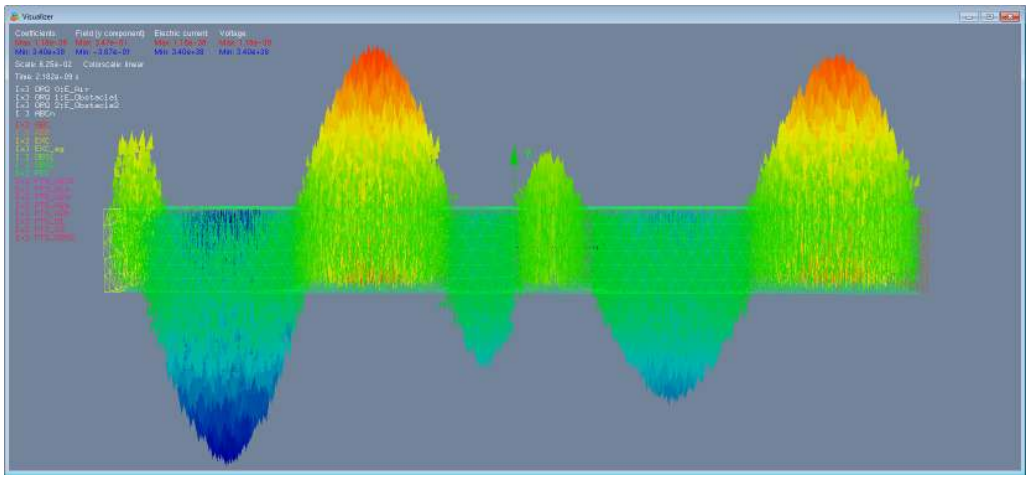


Fig. 4.26 *BUTFE: Electric field distribution at 2.182 ns*

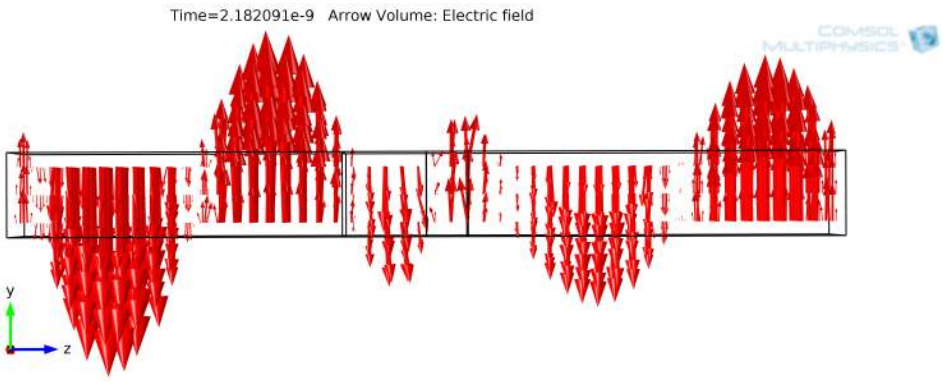


Fig. 4.27 *COMSOL: Electric field distribution at 2.182 ns*

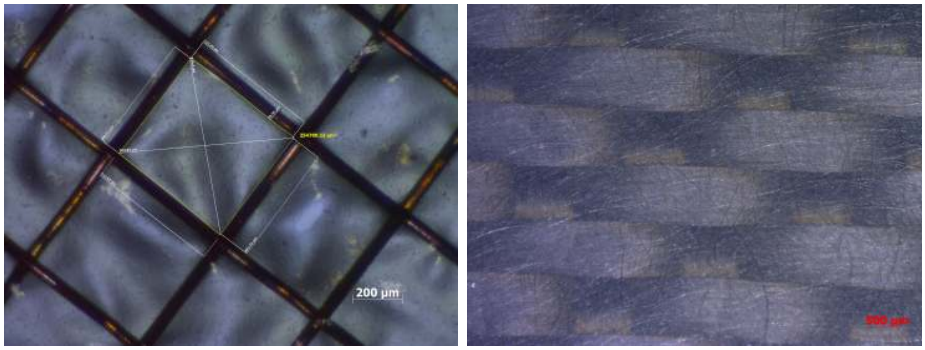
Shielding effectiveness of composite materials

This test case deals with modeling two composite materials: a *prepreg copper mesh* (PCM) and an *epoxy resin prepreg carbon fiber* (ERPCF). The observed physical quantity is frequency dependent *shielding effectiveness* (SE). The following text is a summary, details can be found in [48].

PCM is formed by a grid of copper wires. The wires have diameter 0.05 mm and relative angle 89 degrees. Spatial density is 20 wires per cm. The grid is embedded in a non-conductive epoxy resin having $\epsilon_r = 3.4$. Total thickness of the sample is 0.12 mm.

ERPCF is a woven structure with fiber orientation 0/90 degrees and thickness of 0.27 mm. Relative permittivity of the non-conductive epoxy resin is about 3.4.

Both materials are replaced by a homogeneous equivalent. Material properties of the equivalent (relative permittivity and conductivity) are provided by EMCLab of The Department of Astronautical, Electrical and Energy Engineering (DIAEE) at Sapienza University of Rome.



(a) *prepreg copper mesh* **(b)** *epoxy resin prepreg carbon fiber*

Fig. 4.28 *A view of the composite materials under test [48]*

The homogeneous equivalents are simulated in a rectangular vacuum-filled waveguide. A cross-section of 60 x 4 mm provides a cut-off frequency of 2.5 GHz. PML turned out to be unreliable, therefore the waveguide is designed to be long enough to solve the problem with unwanted reflections. The simulation is stopped before the reflections could reach the observation point.

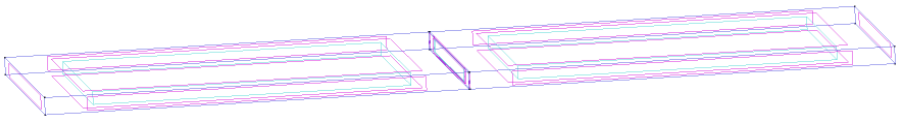


Fig. 4.29 *GiD model*

The simulation runs for 3001 time steps using $\Delta t = 0.217$ ps. The excitation signal is defined as

$$E = E_{max} \left\{ \left(\frac{t}{t_r} \right)^{pw} \exp \left[-pw \left(\frac{t}{t_r} - 1 \right) \right] \right\} \tag{4.52}$$

where $E_{max} = 1$ V/m, rise time $t_r = 5\Delta t$, power $pw = 2$.

SE is obtained using data from two simulations. The first one presents an empty waveguide without any material. The second one has the material placed in it. Considering observation point P in the middle of the cross-section, 1 mm behind the location of material under test, SE can be computed as follows

$$SE(f) = \frac{FFT\{E^*(P)\}}{FFT\{E(P)\}} \tag{4.53}$$

where $E^*(P)$ is electric field intensity observed at point P inside the empty waveguide and $E(P)$ represents electric field intensity observed at point P with the material placed in it. FFT stands for the Fast Fourier Transform.

Figure 4.30 indicates good agreement between simulated results and measurements.

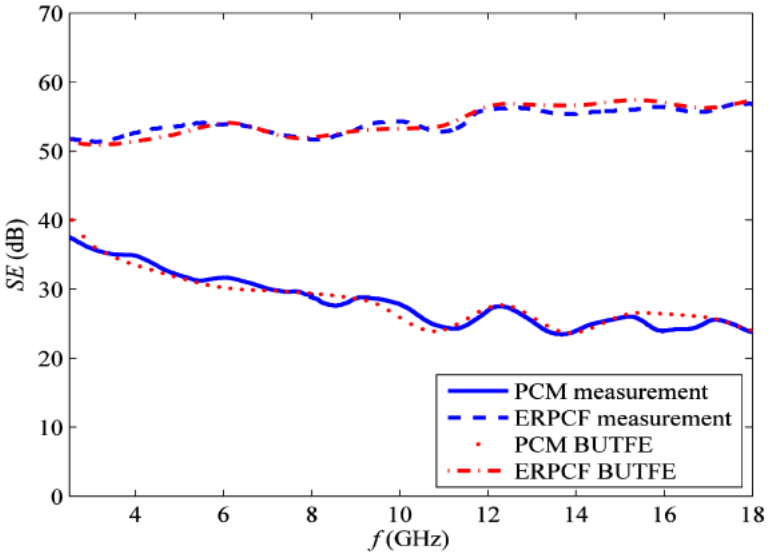
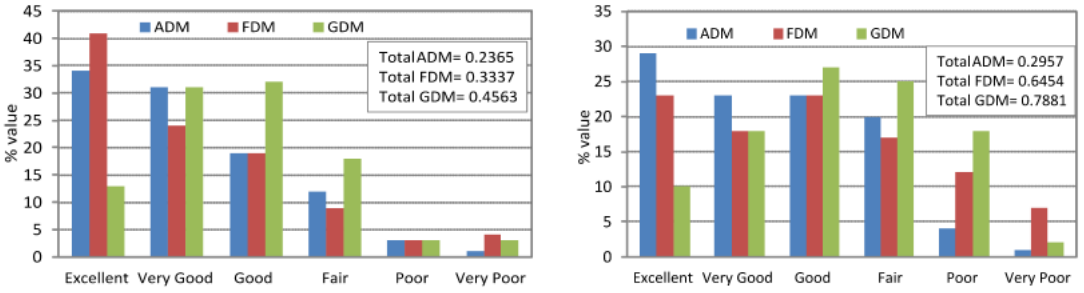


Fig. 4.30 *Direct comparison of the computed shielding effectiveness with the measurements [48]*

Figure 4.31 shows results of the FSV tool which provides statistical evaluation of the agreement.



(a) *prepreg copper mesh* **(b)** *epoxy resin prepreg carbon fiber*

Fig. 4.31 *Results of the FSV validation software [48]*

4.1.8 Parallel performance

The code has been rewritten to work in parallel in order to follow current trends increasing computational power. Unfortunately, the time domain solver is difficult to parallelize due to its iterative way of working (the current iteration requires data from the previous one). Because of this difficulty, entering and leaving the parallel segments arise more often and the gain caused by parallel processing may be comparable or even smaller than the overhead required for handling multiple threads. Saving output data also plays a role. This was found to be a significant bottleneck, especially in smaller simulations.

In order to get an overview of the efficiency of parallel computations, a test has been performed. The time required for solving a test simulation was observed on several computers. A number of 5001 time steps were performed employing the computational mesh containing 1.94 mil. tetrahedrons. Average memory utilization was about 5 GB.

Technical specifications of the machines involved in the test:

- **PowerEdge M610 win HT:** 2×Intel© Xeon© X5675 (3.06-3.46 GHz, 2×6 cores, 2×12 threads), 96 GB RAM, HDD 300GB SAS 10k 2,5″, Windows Server 2008 R2 Enterprise SP1 (64-bit)
- **PowerEdge M610 lnx HT:** 2×Intel© Xeon© X5675 (3.06-3.46 GHz, 2×6 cores, 2×12 threads), 96 GB RAM, HDD 300GB SAS 10k 2,5″, Linux CentOS 6.4 (64-bit)
- **PowerEdge M610 lnx:** 2×Intel© Xeon© X5675 (3.06-3.46 GHz, 2×6 cores, 2×6 threads), 96 GB RAM, HDD 300GB SAS 10k 2,5″, Linux CentOS 6.4 (64-bit)
- **bullx B510:** 2×Intel© Xeon© E5-2665 (2.4 GHz, 2×8 cores, 2×8 threads), 64 GB RAM, HDD 500GB SATA 7k2 2,5″, bullx Linux Server 6.3 (64-bit)
- **BL465c G6:** 2×Six-Core AMD Opteron™ 2435 (2.6 GHz, 2×6 cores, 2×6 threads), 32 GB RAM, HDD 300GB SAS 10k 2,5″, Linux CentOS 6.4 (64-bit)
- **BL465c G5:** 2×Quad-Core AMD Opteron™ 2384 (2.7 GHz, 2×4 cores, 2×4 threads), 16 GB RAM, HDD 300GB SAS 10k 2,5″, Linux CentOS 6.4 (64-bit)
- **PC HT:** 1 ×Intel© Core™ i7-2600 (3.4-3.8 GHz, 1×4 cores, 1×8 threads), 16 GB RAM, SSD Intel© 320 160 GB SATA, Windows 7 Professional SP1 (64-bit)
- **PC:** 1 ×Intel© Core™ i7-2600 (3.4-3.8 GHz, 1×4 cores, 1×4 threads), 16 GB RAM, SSD Intel© 320 160 GB SATA, Windows 7 Professional SP1 (64-bit)

Figure 4.32 shows average computation times for a given number of threads employed. Relative efficiency per thread has been computed to present efficiency of parallel computations while suppressing differences in tested hardware. It can be obtained by dividing computational speed per core in a multi-threaded operation by computational speed in a single-threaded operation

$$\text{efficiency per thread} = \frac{\frac{1}{t_n * n}}{\frac{1}{t_1}} = \frac{t_1}{t_n * n} \tag{4.54}$$

where t_n is average computational time achieved using n threads. See figure 4.33 for graphical representation of the efficiency per thread.

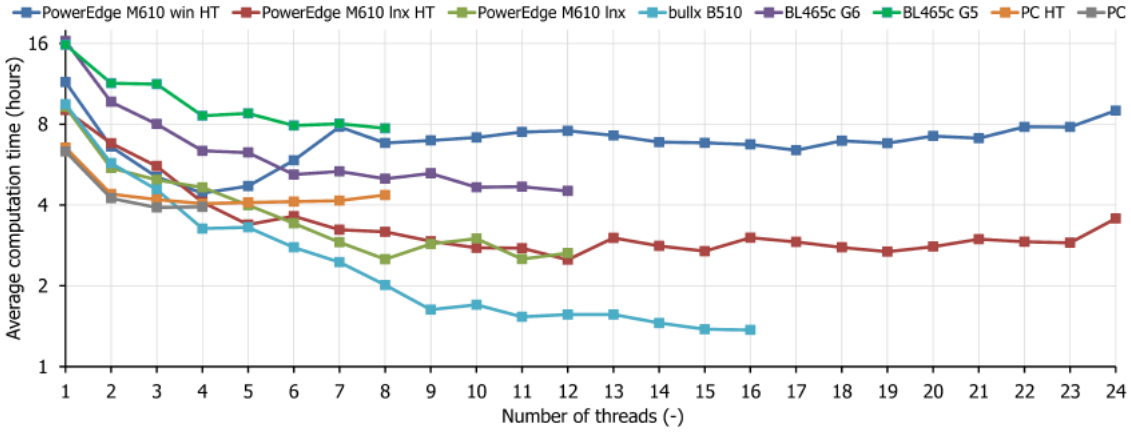


Fig. 4.32 *Average computation time of the BUTFE solver.*

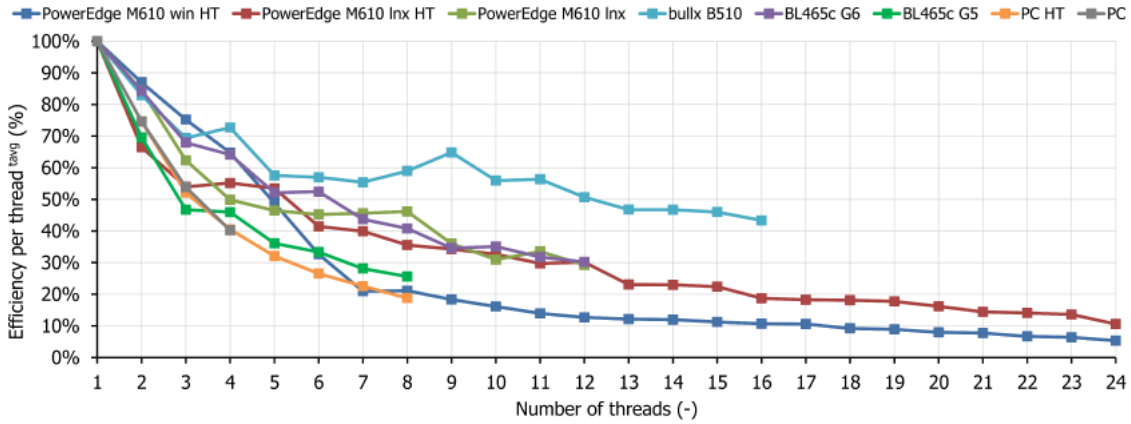


Fig. 4.33 *Efficiency per thread based on average computation time.*

One may note that figures 4.32 and 4.33 are not smooth enough. The values are averaged through at least 5 runs of the simulation performed on a machine with the same hardware configuration. The maximal number of averaged runs is given by the availability of computational resources. Simulation times vary significantly especially on blade servers (simulations may take up to twice as much time).

Increasing the number of tests would improve smoothness and credibility of the graphs, however, it would be very time consuming. The test should just give an overview of the efficiency of parallel computations.

Figures 4.34 and 4.35 show results of the test based on minimum computation time achieved for a given number of the threads. They may offer better information value in certain situations.

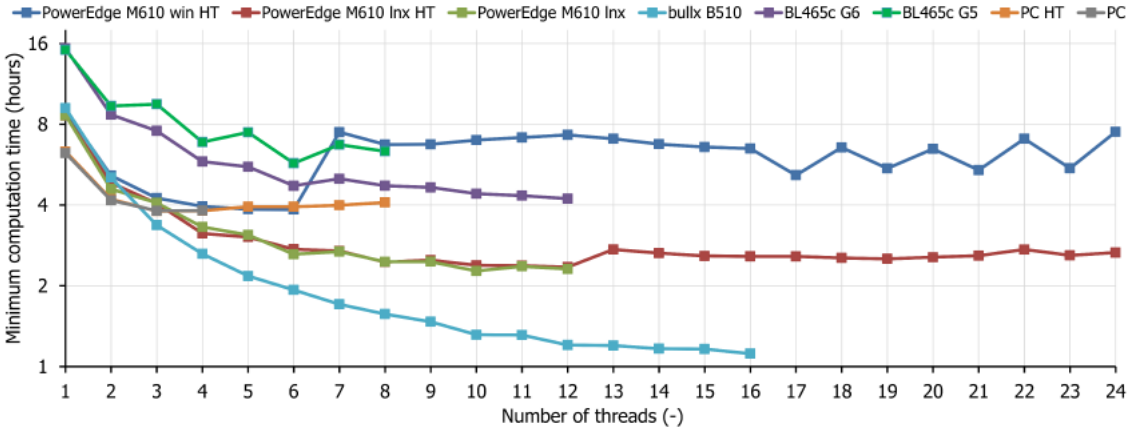


Fig. 4.34 *Minimum computation of the BUTFE solver.*

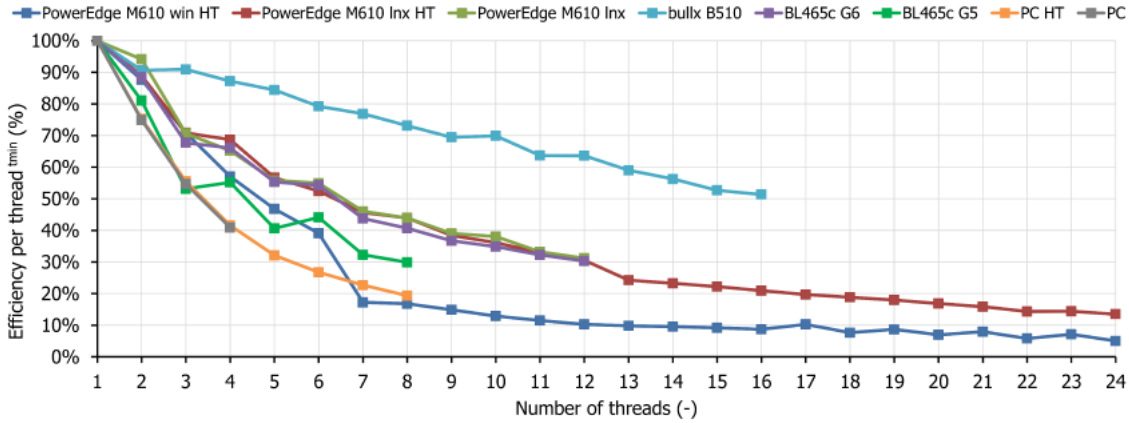


Fig. 4.35 *Efficiency per thread based on minimum computation time.*

One of the key reasons for performing this test was to determine the influence of the Intel© *Hyper-Threading technology* (HT). We can see that BUTFE cannot utilize it. Machines (PowerEdge M610 and PC) perform equally or even better when HT is turned off.

Gradual loss of efficiency per thread is caused by the inability of utilizing more threads at a time. The code contains OpenMP sections which divide the workload into a predefined number of portions. The computer is not fully utilized in the case of a higher number of available threads than the number of sections.

Operation systems can be compared on the PowerEdge M610. You may see similar performances at the beginning, however, simulations performed on Windows begin to slow down with more than four threads and finally take more than twice as much time as simulations performed on Linux. A reason for this behavior was out of scope of this work.

Note that newer processor series have better efficiency per thread than the older ones (BL465c G5 vs. BL465c G6, PowerEdge M610 lnx vs. bullx B510).

Note that the comparison may be influenced by Intel© Turbo Boost technology which automatically overclocks CPU cores based on utilization and power dissipation. This virtually degrades the efficiency per core because a single threaded simulation is computed

faster while running on an intensively overclocked CPU core.

While considering the gradual loss of efficiency per thread, one may try to run more different simulations, each of them using less threads in order to efficiently use given computational resources. Table 4.3 shows the results of this test for two hardware configurations. The test was considered as finished once the last simulation of the test configuration was finished. Each test configuration was performed at least five times. Measure of computational speed is based on the minimum of the five tests.

Machine	**Test configuration**	**Best time (h:m:s)**	**Computational speed (simulations/hour)**	**Speedup (%)**
PowerEdge M610 lnx HT	1 simulation × 12 threads	2:20:58	0.43	
	2 simulations × 6 threads	4:32:05	0.44	+4
	3 simulations × 4 threads	6:15:43	0.48	+13
	4 simulations × 3 threads	8:06:50	0.49	+16
	6 simulations × 2 threads	11:56:50	0.50	+18
	12 simulations × 1 thread	23:21:18	0.51	+21
bullx B510	1 simulation × 16 threads	1:07:04	0.89	
	2 simulations × 8 threads	2:01:03	0.99	+11
	3 simulations × 5 threads	2:55:20	1.03	+15
	4 simulations × 4 threads	3:47:55	1.05	+18
	5 simulations × 3 threads	4:35:35	1.09	+22
	8 simulations × 2 threads	7:02:04	1.14	+27

Tab. 4.3 *The concurrent simulations*

We can see that it is more efficient to run multiple simulations at once. Note that only the time of the slowest simulation was taken into account. The other simulations finished earlier. The efficiency of multiple simulations would be even better in the case of proper job management.

The test of 16 simultaneous single-threaded simulations cannot be performed on the bullx server because the memory requirements would exceed amount of available memory of the computational node.

4.1.9 Conclusion

Chapter 4.1 has been dedicated to the TDFE solver BUTFE and its excitation module BUTFE_EXC, including the description of the semantic location within the computational framework HIRF-SE.

Based on a *three-dimensional* (3D) nodal-based FEM code for solving modal analysis, BUTFE has evolved into a 3D edge-based TDFE solver with significant improvements in excitation, applicable material properties and boundary conditions.

Further development is facilitated thanks to a well-arranged definition of matrices of the final system 4.43. Each matrix is stored in a stack. Accompanying details contain information like which coupling matrices to multiply by (general/wire), which matrix generator to use (**S**/**T** in 1D, **S**/**T**/**P**/**Q** in 3D) or which temporal operation to apply (none/derivative/second derivative/integral/double integral). The program passes through the stack, automatically applies a given temporal scheme and puts the resulting matrices in the proper place in the system of equations.

BUTFE_EXC is a GUI application that allows to set excitation coefficients related to a given layer (mesh group) of the computational mesh. Excitation can be set in three ways: by formula, by formula considering a perforated slab (using PEMF) or waveguide excitation.

Considering the nature of the chapter content, most of the work presented here is rather of applied nature. The solver (module) is important from three main perspectives:

- **HIRF-SE framework:** The module offers results obtained by another method. Results obtained by various methods can give a better overview of the solved problem.
- **fundamental research:** The solver offers basic functionality which forms the basis for further research activities.
- **applied research:** The solver represents a good starting point for further applied industrial research.

Although the BUTFE solver can still be considered as underdeveloped (only basic order of approximation, first order ABC only, no PML), significant steps forward have been made. Future students of this faculty will be able to focus on more attractive topics of numerical analysis having a functional program core already available.

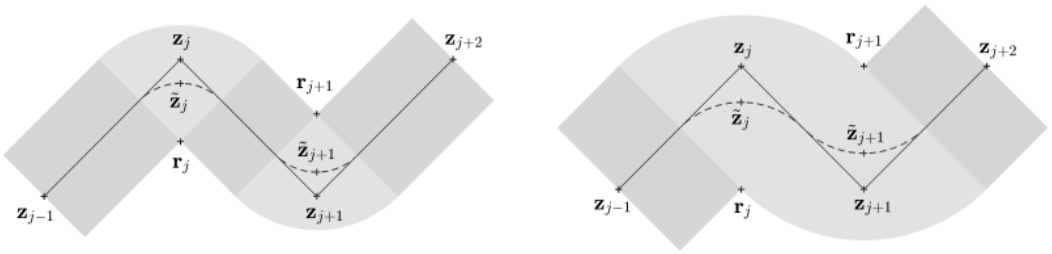
4.2 Bent wires

While the thin-wire approximation of straight wires was already explained in section 4.1.4, this chapter is going to focus on the problem of bent wires. Chapter 4.2.1 describes shortcomings of already published method and briefly introduces suggested improvements. Chapter 4.2.2 focuses on a detailed explanation of the proposed approach. Comparison of both approaches on several test cases can be found in chapter 4.2.4. Chapter 4.2.5 describes countermeasures required for successfully performing of the approach. Chapter 4.2.6 concludes this topic.

4.2.1 Proposed approach

The solution for arbitrarily oriented wires is given in [40]. Authors divide the wire into a set of straight parts and bends. While the straight parts follow the original line between subsequent nodes, bends require special treatment. In order to keep continuity between two subsequent straight parts, the centerline of the interpolation cylinder does not pass exactly through a given node.

Figure 4.36a shows this approach applied on two 90° bends. The wire segment is described by a mesh of four nodes **z**. One may see that the centerline of the interpolation cylinder matches the wire mesh in the case of the straight parts but goes rather through a virtual point $\tilde{\mathbf{z}}$ in the case of a bent part.



(a) $r_0 = 0.3\,\|\mathbf{z}_j\mathbf{z}_{j+1}\|$ **(b)** $r_0 = 0.5\,\|\mathbf{z}_j\mathbf{z}_{j+1}\|$

Fig. 4.36 *The interpolation cylinder proposed in [40]*

As authors admit, they assume that two neighboring sectors are not overlapping each other. Such an overlap can be caused by a large radius of the interpolation cylinder r_0 in the case of sharp bends and an inappropriate choice of discretization of the wire with respect to the surrounding area. Note that the radius r_0 is given by discretization of the surrounding area. Figure 4.36b shows the limit case when bent parts are touching each other. An increase of r_0 or a sharper bend causes overlaps of the wire segments.

It is usually difficult to discover the discretization issue because the approach solves each bend separately without any knowledge about position of the following bend and thus the possibility to detect possible overlaps. The modeler would probably have to start an iterative process of meshing and check of the continuity of the interpolation cylinder visually.

I have focused on inventing another approach which would handle the discretization issue by itself. While the previous approach focused on the straight parts and treated the bends to create a smooth cylindrical object, I have decided to do the opposite. I have arranged that the centerline of the bend parts is passing through nodal points. The straight parts are then treated to match the bends and create smooth cylinder (see figure 4.37).

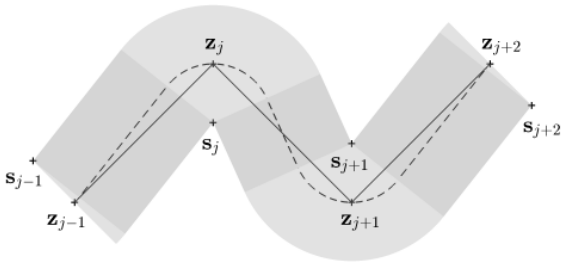
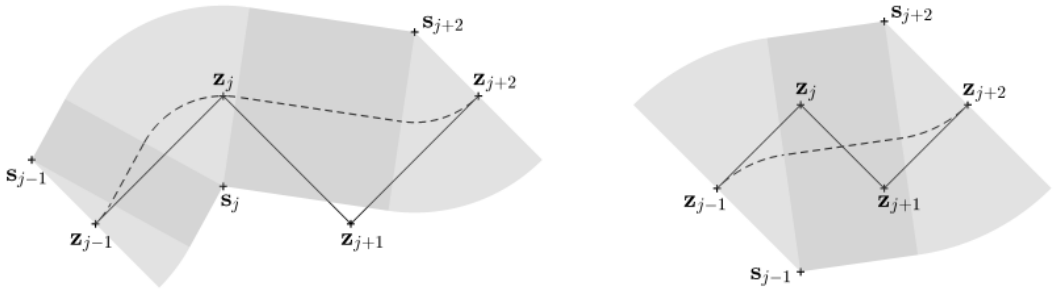


Fig. 4.37 *The interpolation cylinder, proposed apporach,* $r_0 = 0.3\,\|\mathbf{z}_j\mathbf{z}_{j+1}\|$

As it turned out, this approach is sensitive to a large r_0 as well. However, it is able to detect overlaps of the neighboring segments. The detection allows to skip one (fig. 4.38a) or more (fig. 4.38b) nodes causing the overlap(s).



(a) $r_0 = 0.5\,\|\mathbf{z}_j\mathbf{z}_{j+1}\|$ **(b)** $r_0 = \|\mathbf{z}_j\mathbf{z}_{j+1}\|$

Fig. 4.38 *The interpolation cylinder, proposed apporach*

4.2.2 Proposed approach in detail

As we can see in figure 4.39, the area between two consequent nodes is divided into three parts: first bend BI, straight part S and second bend BII.

The bends BI and BII seemed to be a single one in previous figures. However, the bends must be separated in order to follow an arbitrary wire in the 3D space.

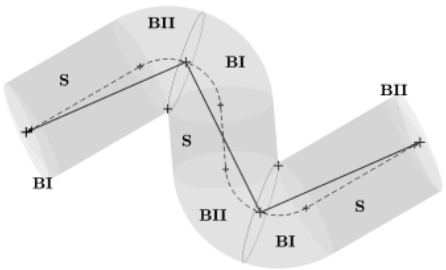


Fig. 4.39 *Division of the wire segments*

The initial phase of the description of the parts is similar to [40]. Some basic vectors have to be determined (see figure 4.40). The direction $\mathbf{v}$ of each wire segment and normal to the bend $\mathbf{n}$ for given node can be easily determined using nodal coordinates $\mathbf{z}$:

$$\mathbf{v}_j = \frac{\mathbf{z}_{j+1} - \mathbf{z}_j}{\|\mathbf{z}_{j+1} - \mathbf{z}_j\|}, \quad \mathbf{n}_j = \mathbf{v}_j \times \mathbf{v}_{j-1} \tag{4.55}$$

Vectors $\mathbf{t}_{b1}$ and $\mathbf{t}_{b2}$ are well known from [40]. Additional vector $\mathbf{t}_{b3}$ has been defined:

$$\mathbf{t}_{b1,j} = \mathbf{n}_j \times \mathbf{v}_{j-1}, \quad \mathbf{t}_{b2,j} = \mathbf{n}_j \times \mathbf{v}_j, \quad \mathbf{t}_{b3,j} = \frac{\mathbf{t}_{b1,j} + \mathbf{t}_{b2,j}}{\|\mathbf{t}_{b1,j} + \mathbf{t}_{b2,j}\|} \tag{4.56}$$

Auxiliary point $\mathbf{s}$ will help us to determine rotation angles of the bends in the future

$$\mathbf{s}_j = \mathbf{z}_j - r_0 \mathbf{t}_{b3,j} \tag{4.57}$$

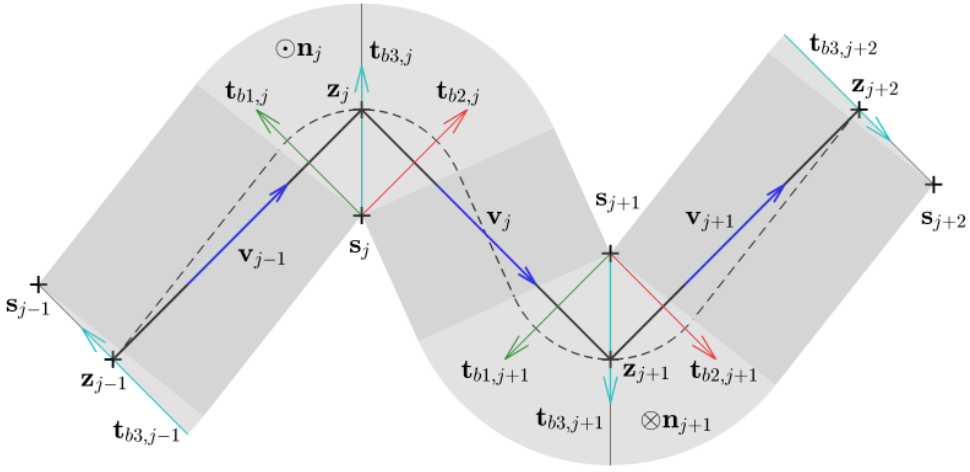


Fig. 4.40 *Determining basic vectors*

Normals of the circles (see 4.41) having their center in node $\mathbf{z}_j$ can be determined as

$$\mathbf{n}_{c1,j} = \mathbf{t}_{b3,j} \times \mathbf{n}_j \tag{4.58a}$$

$$\mathbf{n}_{c2,j} = \mathbf{t}_{b3,j+1} \times -\mathbf{n}_{j+1} \tag{4.58b}$$

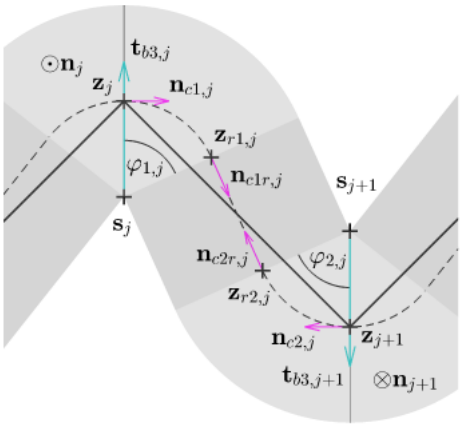
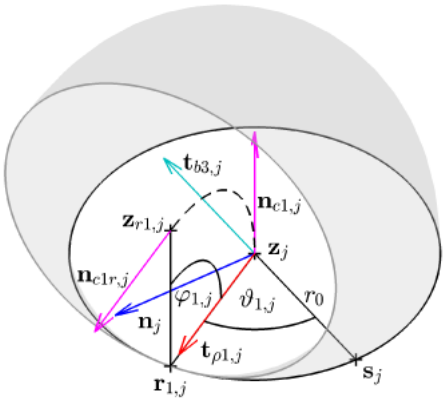
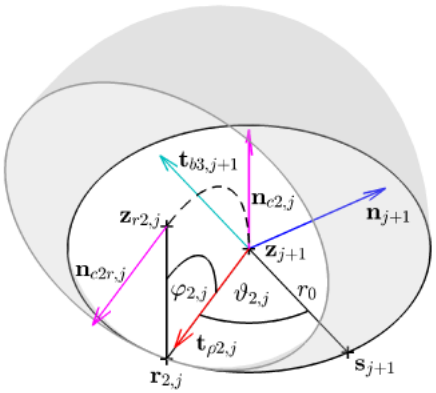


Fig. 4.41 *Detail of the wire segment*

Node $\mathbf{z}_{r1,j}$ and vector $\mathbf{n}_{c1r,j}$ are formed by rotation of $\mathbf{z}_j$ and $\mathbf{n}_{c1,j}$ by angles $\vartheta_{1,j}$ and $\varphi_{1,j}$. Node $\mathbf{z}_{r2,j}$ and vector $\mathbf{n}_{c2r,j}$ are formed by rotation of $\mathbf{z}_{j+1}$ and $\mathbf{n}_{c2,j}$ by angles $\vartheta_{2,j}$ and $\varphi_{2,j}$. The angles $\vartheta_{1,j}$, $\varphi_{1,j}$, $\vartheta_{2,j}$ and $\varphi_{2,j}$ determine shape of bends BI and BII and are not yet known. See figure 4.42 for details of the circular bends. Note the different orientation of ϑ for BI and BII. It follows rule of clockwise orientation of ϑ when looking from inner part S to given circle.



(a) *Detail of the bend part BI*



(b) *Detail of the bend part BII*

Fig. 4.42 *Details of the circular bends*

The aim is to find such $\vartheta_{1,2}$ and $\varphi_{1,2}$ that cause that $\mathbf{n}_{c1r,j}$ is pointing to $\mathbf{z}_{r2,j}$ while $\mathbf{n}_{c2r,j}$ is pointing to $\mathbf{z}_{r1,j}$. Vectors $\mathbf{n}_{c1r,j}$ and $\mathbf{n}_{c2r,j}$ become exactly opposite.

Determining of $\vartheta_{1,2}$ and $\varphi_{1,2}$ is difficult due to circular dependencies between ϑ, φ and $\mathbf{z}_r$. However, the angles can be found using an iterative approach where sufficient precision can be reached within a few iterations.

At first, the procedure will be explained generally, followed by detailed description including necessary math.

Let us focus on first bend BI. In order to get $\vartheta_{1,j}$ and $\varphi_{1,j}$, target location of $\mathbf{z}_{r2,j}$ should be known. Instead of $\mathbf{z}_{r2,j}$, we use $\mathbf{z}_{j+1}$ as an initial target location for $\mathbf{n}_{c1r,j}$ (we denote target location for $\mathbf{n}_{c1r,j}$ by $\mathbf{x}_{1,j}$). Now we can compute $\mathbf{z}_{r1,j}$ using (inaccurate) $\vartheta_{1,j}$, $\varphi_{1,j}$. Angles $\vartheta_{2,j}$ and $\varphi_{2,j}$ of BII are can be computed using (inaccurate) $\mathbf{z}_{r1,j}$. Resulting $\mathbf{z}_{r2,j}$ is used for next iteration where $\vartheta_{1,j}$ and $\varphi_{1,j}$ are getting closer to the wanted solution.

The iteration loop is as follows:

- **Iteration 1:**
  get $\vartheta_{1,j}$ and $\varphi_{1,j}$ with $\mathbf{x}_{1,j} = \mathbf{z}_{j+1}$, obtain $\mathbf{z}_{r1,j}$
  get $\vartheta_{2,j}$ and $\varphi_{2,j}$ with $\mathbf{x}_{2,j} = \mathbf{z}_{r1,j}$, obtain $\mathbf{z}_{r2,j}$
- **Iteration 2:**
  get more accurate $\vartheta_{1,j}$ and $\varphi_{1,j}$ with $\mathbf{x}_{1,j} = \mathbf{z}_{r2,j}$, obtain more accurate $\mathbf{z}_{r1,j}$
  get more accurate $\vartheta_{2,j}$ and $\varphi_{2,j}$ with $\mathbf{x}_{2,j} = \mathbf{z}_{r1,j}$, obtain more accurate $\mathbf{z}_{r2,j}$
- **Iteration 3:**
  ...

Setting $\mathbf{x}_{2,j} = \mathbf{z}_{r1,j}$ in the first iteration speeds up the process. Setting $\mathbf{x}_{2,j} = \mathbf{z}_j$ (in the first iteration) would allow parallel computation of angles $\vartheta_{1,j}$, $\varphi_{1,j}$ in one thread and $\vartheta_{2,j}$, $\varphi_{2,j}$ in the other one. Both the threads would use targets computed in the previous iteration.

The detailed description will follow. In order to get base point $\mathbf{r}$ of the rotation by ϑ, projection of the target $\mathbf{x}$ into the plane ρ of the circle must be known. The plane ρ is given as follows

$$\rho_{1,j}: \quad n^x_{c1,j}x + n^y_{c1,j}y + n^z_{c1,j}z - \mathbf{n}_{c1,j} \cdot \mathbf{z}_j = 0 \tag{4.59a}$$

$$\rho_{2,j}: \quad n^x_{c2,j}x + n^y_{c2,j}y + n^z_{c2,j}z - \mathbf{n}_{c2,j} \cdot \mathbf{z}_{j+1} = 0 \tag{4.59b}$$

We can get the projection $\mathbf{x}_\rho$ using a parametric expression of a line passing through $\mathbf{x}$ and perpendicular to the plane ρ

$$\mathbf{x}_{\rho 1,j} = \mathbf{x}_{1,j} + \frac{\mathbf{n}_{c1,j} \cdot (\mathbf{z}_j - \mathbf{x}_{1,j})}{\mathbf{n}_{c1,j} \cdot \mathbf{n}_{c1,j}} \mathbf{n}_{c1,j} \tag{4.60a}$$

$$\mathbf{x}_{\rho 2,j} = \mathbf{x}_{2,j} + \frac{\mathbf{n}_{c2,j} \cdot (\mathbf{z}_{j+1} - \mathbf{x}_{2,j})}{\mathbf{n}_{c2,j} \cdot \mathbf{n}_{c2,j}} \mathbf{n}_{c2,j} \tag{4.60b}$$

Vector $\mathbf{t}_\rho$ defines direction of $\mathbf{x}$ ($\mathbf{x}_\rho$) in the plane ρ of the circle

$$\mathbf{t}_{\rho 1,j} = \frac{\mathbf{x}_{\rho 1,j} - \mathbf{z}_j}{\|\mathbf{x}_{\rho 1,j} - \mathbf{z}_j\|}, \quad \mathbf{t}_{\rho 2,j} = \frac{\mathbf{x}_{\rho 2,j} - \mathbf{z}_{j+1}}{\|\mathbf{x}_{\rho 2,j} - \mathbf{z}_{j+1}\|} \tag{4.61}$$

Angle ϑ is measured from $-\mathbf{t}_{b3}$ in clockwise direction when looking from the straight part S (between BI and BII)

$$\vartheta_{1,j} = \arccos\left(\mathbf{t}_{\rho 1,j} \cdot -\mathbf{t}_{b3,j}\right), \quad \vartheta_{2,j} = \arccos\left(\mathbf{t}_{\rho 2,j} \cdot -\mathbf{t}_{b3,j+1}\right) \tag{4.62}$$

Since the arccos() in 4.62 returns the smallest angle between $\mathbf{t}_\rho$ and $-\mathbf{t}_{b3}$, the angle must be conditionally determined

$$\|\mathbf{x}_{\rho 1,j} - (\mathbf{z}_j + \mathbf{n}_j)\| > \|\mathbf{x}_{\rho 1,j} - (\mathbf{z}_j - \mathbf{n}_j)\| \tag{4.63a}$$

$$\implies \quad \vartheta_{1,j} = 2\pi - \arccos\left(\mathbf{t}_{\rho 1,j} \cdot -\mathbf{t}_{b3,j}\right)$$

$$\|\mathbf{x}_{\rho 2,j} - (\mathbf{z}_{j+1} - \mathbf{n}_{j+1})\| > \|\mathbf{x}_{\rho 2,j} - (\mathbf{z}_{j+1} + \mathbf{n}_{j+1})\| \tag{4.63b}$$

$$\implies \quad \vartheta_{2,j} = 2\pi - \arccos\left(\mathbf{t}_{\rho 2,j} \cdot -\mathbf{t}_{b3,j+1}\right)$$

Angle ϑ is determined by equation 4.63. Now, we are going to determine φ. A rotation matrix can be assembled for rotation by angle α in a plane defined by a normal $\mathbf{u}$ [41]

$$\mathbf{R}_{\{\mathbf{u};\alpha\}} = \begin{bmatrix} \cos\alpha + {u_x}^2\,(1-\cos\alpha) & u_x u_y\,(1-\cos\alpha) - u_z \sin\alpha & u_x u_z\,(1-\cos\alpha) + u_y \sin\alpha \\ u_y u_x\,(1-\cos\alpha) + u_z \sin\alpha & \cos\vartheta + {u_y}^2\,(1-\cos\alpha) & u_y u_z\,(1-\cos\alpha) - u_x \sin\alpha \\ u_z u_x\,(1-\cos\alpha) - u_y \sin\alpha & u_z u_y\,(1-\cos\alpha) + u_x \sin\alpha & \cos\alpha + {u_z}^2\,(1-\cos\alpha) \end{bmatrix} \tag{4.64}$$

Rotation bases $\mathbf{r}$ are points $\mathbf{s}$ rotated by ϑ

$$\mathbf{r}_{1,j} = \mathbf{s}_j \mathbf{R}_{\{\mathbf{n}_{c1,j};\vartheta_{1,j}\}} + \mathbf{z}_j \tag{4.65a}$$

$$\mathbf{r}_{2,j} = \mathbf{s}_{j+1} \mathbf{R}_{\{\mathbf{n}_{c2,j};\vartheta_{2,j}\}} + \mathbf{z}_{j+1} \tag{4.65b}$$

We can see that the simplified case in figure 4.39 has $\vartheta_1 = \vartheta_2 = 0$ and $\mathbf{r} = \mathbf{s}$. Equation 4.65 determines base $\mathbf{r}$ for the second rotation (by φ). Situation in the plane given by points $\mathbf{z}_j$, $\mathbf{r}_{1,j}$ and $\mathbf{x}_{1,j}$ is depicted in figure 4.43. Two auxiliary vectors are defined

$$\mathbf{t}_{c1a,j} = \frac{\mathbf{z}_j - \mathbf{r}_{1,j}}{\|\mathbf{z}_j - \mathbf{r}_{1,j}\|}, \qquad \mathbf{t}_{c1b,j} = \frac{\mathbf{x}_{1,j} - \mathbf{r}_{1,j}}{\|\mathbf{x}_{1,j} - \mathbf{r}_{1,j}\|} \tag{4.66a}$$

$$\mathbf{t}_{c2a,j} = \frac{\mathbf{z}_{j+1} - \mathbf{r}_{2,j}}{\|\mathbf{z}_{j+1} - \mathbf{r}_{2,j}\|}, \qquad \mathbf{t}_{c2b,j} = \frac{\mathbf{x}_{2,j} - \mathbf{r}_{2,j}}{\|\mathbf{x}_{2,j} - \mathbf{r}_{2,j}\|} \tag{4.66b}$$

Since the auxiliary vectors have unit length, we can determine angle γ between them as follows

$$\gamma_{1,j} = \arccos\left(\mathbf{t}_{c1a,j} \cdot \mathbf{t}_{c1b,j}\right) \tag{4.67a}$$

$$\gamma_{2,j} = \arccos\left(\mathbf{t}_{c2a,j} \cdot \mathbf{t}_{c2b,j}\right) \tag{4.67b}$$

The cosine law 4.67 returns the smallest angle between the two vectors. Angles greater than π radians must be treated. Such angles may arise in the case of intentional omission of some nodes. We have to use the following formula

$$\gamma_{1,j} = 2\pi - \arccos\left(\mathbf{t}_{c1a,j} \cdot \mathbf{t}_{c1b,j}\right) \tag{4.68}$$

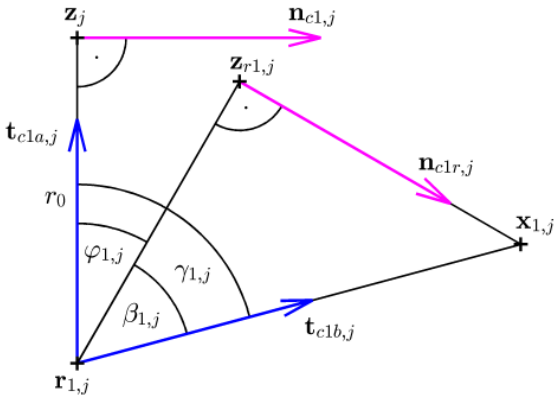


Fig. 4.43 *Bend in plane given by points* $\mathbf{z}_j$, $\mathbf{r}_{1,j}$ *and* $\mathbf{x}_{1,j}$

in case of

$$\frac{\mathbf{t}_{c1a,j} \times \mathbf{t}_{c1b,j}}{\|\mathbf{t}_{c1a,j} \times \mathbf{t}_{c1b,j}\|} = -\mathbf{t}_{c1a,j} \times \mathbf{n}_{c1,j} \tag{4.69}$$

and similarly for the BII

$$\gamma_{2,j} = 2\pi - \arccos\left(\mathbf{t}_{c2a,j} \cdot \mathbf{t}_{c2b,j}\right) \tag{4.70}$$

in case of

$$\frac{\mathbf{t}_{c2a,j} \times \mathbf{t}_{c2b,j}}{\|\mathbf{t}_{c2a,j} \times \mathbf{t}_{c2b,j}\|} = -\mathbf{t}_{c2a,j} \times \mathbf{n}_{c2,j} \tag{4.71}$$

Angle β can be easily determined using right triangle formed by nodes $\mathbf{z}_r$, $\mathbf{r}$ and $\mathbf{x}$. Distance between $\mathbf{z}_r$ and $\mathbf{r}$ is well known: it is r_0.

$$\beta_{1,j} = \arccos\left(\frac{r_0}{\|\mathbf{x}_{1,j} - \mathbf{r}_{1,j}\|}\right), \quad \beta_{2,j} = \arccos\left(\frac{r_0}{\|\mathbf{x}_{2,j} - \mathbf{r}_{2,j}\|}\right) \tag{4.72}$$

Angle φ can be determined using subtraction of β from γ

$$\varphi_{1,j} = \gamma_{1,j} - \beta_{1,j}, \quad \varphi_{2,j} = \gamma_{2,j} - \beta_{2,j} \tag{4.73}$$

Now we have obtained ϑ. We can rotate $\mathbf{z}$ around $\mathbf{r}$ by φ using the rotation matrix 4.64 to get target $\mathbf{z}_r$ for the next iteration.

$$\mathbf{z}_{r1,j} = (\mathbf{z}_j - \mathbf{r}_{1,j})\,\mathbf{R}_{\left\{\mathbf{n}_j\mathbf{R}_{\left\{\mathbf{n}_{c1,j};\vartheta_{1,j}\right\}};\varphi_{1,j}\right\}} + \mathbf{r}_{1,j} \tag{4.74a}$$

$$\mathbf{z}_{r2,j} = (\mathbf{z}_{j+1} - \mathbf{r}_{2,j})\,\mathbf{R}_{\left\{-\mathbf{n}_{j+1}\mathbf{R}_{\left\{\mathbf{n}_{c2,j};\vartheta_{2,j}\right\}};\varphi_{2,j}\right\}} + \mathbf{r}_{2,j} \tag{4.74b}$$

Rotated normal vectors of circles can be used for future testing of continuity of the parts of the cylinder. They can be computed as follows

$$\mathbf{n}_{cr1,j} = (\mathbf{n}_{c1,j} + \mathbf{z}_j - \mathbf{r}_{1,j})\,\mathbf{R}_{\left\{\mathbf{n}_j\mathbf{R}_{\left\{\mathbf{n}_{c1,j};\vartheta_{1,j}\right\}};\varphi_{1,j}\right\}} + \mathbf{r}_{1,j} - \mathbf{z}_{r1,j} \tag{4.75a}$$

$$\mathbf{n}_{cr2,j} = (\mathbf{n}_{c2,j} + \mathbf{z}_{j+1} - \mathbf{r}_{2,j})\,\mathbf{R}_{\left\{-\mathbf{n}_{j+1}\mathbf{R}_{\left\{\mathbf{n}_{c2,j};\vartheta_{2,j}\right\}};\varphi_{2,j}\right\}} + \mathbf{r}_{2,j} - \mathbf{z}_{r2,j} \tag{4.75b}$$

The rotated normals should be exactly opposite. The difference in the values (not talking about the sign) means that parts S and BII (or BI and S) are not properly sticked together.

The inner rotation in 4.74 and 4.75 defines normal $\mathbf{u}$ (see equation 4.64) of the outer rotation. In order to get the unit vector defining the rotation plane for φ, we have to rotate the normal vector of a given node by angle ϑ. We are using the same rotation matrix as in equation 4.65.

Part BI is represented by rotating the circle with center point $\mathbf{z}_j$ by angle $\varphi_{1,j}$ using the base point of rotation $\mathbf{r}_{1,j}$. This rotation is performed in a plane defined by the unit vector derived by rotation of $\mathbf{n}_j$ by $\vartheta_{1,j}$ in a plane defined by the normal $\mathbf{n}_{c1,j}$. The straight part S is formed by a cylinder of radius r_0 with axis going from $\mathbf{z}_{r1,j}$ to $\mathbf{z}_{r2,j}$. The part BII is represented by rotating the circle with center point $\mathbf{z}_{j+1}$ by angle $\varphi_{2,j}$ using the base point of rotation $\mathbf{r}_{2,j}$. This rotation is performed in a plane defined by the unit vector derived by rotation of $-\mathbf{n}_{j+1}$ by $\vartheta_{2,j}$ in a plane defined by the normal $\mathbf{n}_{c2,j}$. You may notice that formulations for both bends are the same except for the reversed normal $\mathbf{n}$.

The beginning and the end of the wire have a special rule for creating the normal $\mathbf{n}$ and unit vector $\mathbf{t}_{b3}$. These points are considered as a straight wire. Since vectors $\mathbf{v}$ of two subsequent segments are the same, we cannot use a cross product to get the normal of "the bend" (see equation 4.55). The normal can be obtained using cross product

$$\mathbf{n}_j = \frac{\mathbf{v} \times \mathbf{m}}{\|\mathbf{v} \times \mathbf{m}\|} \tag{4.76}$$

where $\mathbf{v}$ defines the direction of the segment (original $\mathbf{v}_{j-1}$ or $\mathbf{v}_j$) and $\mathbf{m}$ stands for any vector which is other than $\mathbf{v}$.

Considering figure 4.40, we assume normals

$$\mathbf{n}_{j-1} = \mathbf{n}_j, \quad \mathbf{n}_{j+2} = \mathbf{n}_{j+1} \tag{4.77}$$

and the unit vectors

$$\mathbf{t}_{b3,j-1} = \mathbf{t}_{b1,j}, \quad \mathbf{t}_{b3,j+2} = \mathbf{t}_{b2,j+1} \tag{4.78}$$

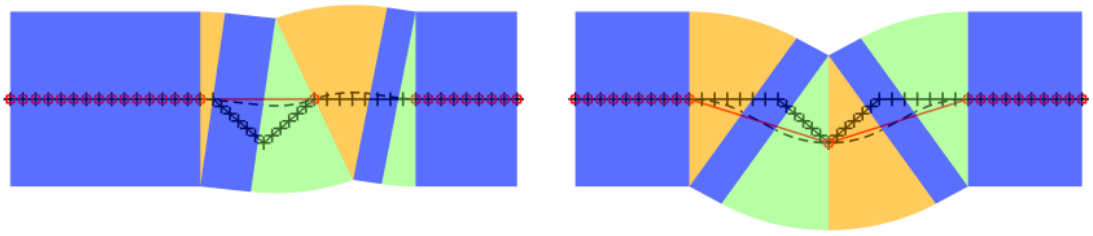
The advantage of this approach is that it solves a pair of two subsequent bends together. Unsolvable geometry produces imaginary angle values which are easy to detect. The user can be asked to increase the density of the tetrahedral mesh in proximity of the wire. A second option is to slightly modify the wire mesh in order to avoid intersections. This avoids user interaction, however it may lead to loss of accuracy. The following chapter shows a proposal of how to deal with the modification of the wire mesh.

4.2.3 Construction strategies

Chapter 4.2.2 described the construction of the cylindrical area between two consecutive nodes ($\mathbf{z}_j$ and $\mathbf{z}_{j+1}$) including feedback giving information whether a wire segment is feasible or not. We are going to present two strategies which use the feedback to make a subset of the original nodes. This subset contains non-conflicting nodes which form seamless cylindrical area.

The easiest strategy could be called *avalanche processing*. It means that we sequentially process segments from one end of the wire to the other. Considering unsolvable segment between $\mathbf{z}_j$ and $\mathbf{z}_{j+1}$, we skip $\mathbf{z}_{j+1}$ and try to connect $\mathbf{z}_j$ with $\mathbf{z}_{j+2}$. The original normals $\mathbf{n}_j$,$\mathbf{t}_{b3,j}$ and $\mathbf{n}_{j+2}$,$\mathbf{t}_{b3,j+2}$ keep influenced by $\mathbf{z}_{j+1}$ (see figure 4.38a). In case of unsolvable segment between $\mathbf{z}_j$ and $\mathbf{z}_{j+2}$, we continue on the next node until a solvable segment is found.

Another option is to use so-called *progressive refinement*. This strategy is more complicated but offers better approximation of the original shape. Nodes are processed in according to their importance where the most important node defines the sharpest bend of the wire. See figure 4.44 for comparison of these two strategies. The original mesh is denoted by black color, nodes of the final segments are marked by red squares. Centerline of the resulting cylindrical area is denoted by black dashed line. Part BI is colored in orange, BII in green and S is colored in blue.



(a) *Avalanche processing* **(b)** *Progressive refinement*

Fig. 4.44 *Results of two construction strategies*

A process of the *adaptive refinement* is shown on a simplified geometry (see fig. 4.45. Source mesh of the wire is formed by 11 nodes counted from the left to the right. It contains sharp bend angle at node 6 and two smoother bends at nodes 5 and 7. Each following paragraph is connected with a particular step of the process depicted in figure 4.45.

After determining all the necessary vectors and points we try to create a single wire segment between nodes 1 and 11.

Successful connection (marked by green line) of the endpoints implies fundamental feasibility of the problem. Nodes 1 and 11 are added into the subset of the non-conflicting nodes. Node 6 is determined as candidate for a new member of the subset. Nodes 1 and 11 are the closest nodes from the vicinity of the node 6.

Both connections 1-6 and 6-11 were successful. Node 5 is determined as candidate for a new node. Now, 1 and 6 are the closest nodes from the vicinity of the node 5.

Connection 1-5 is feasible but we are not able to connect nodes 5-6 (due to high radius of the cylinder). Thus, node 5 will not appear in the subset.

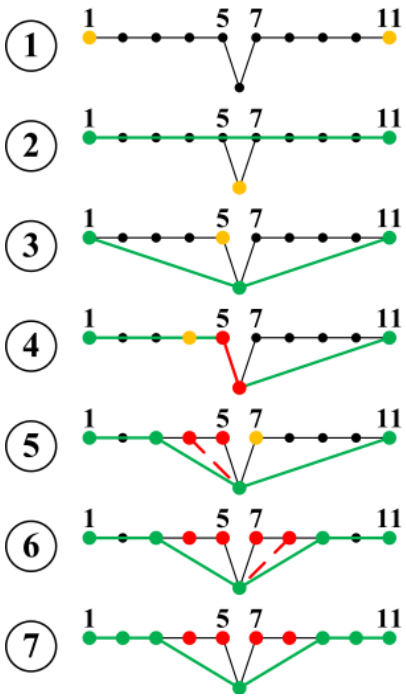


Fig. 4.45 *Construction strategy: example*

Node 4 is too far from node 6 to create a feasible segment and so it is omitted as well (1-4 OK but 4-6 failed). Finally, both checks 1-3 and 3-6 have succeeded. Node 3 is added into the subset, node 7 is determined as a new candidate.

Nodes 7 and 8 are omitted for the same reason as nodes 4 and 5. The wire is formed by nodes 1-3-6-9-11 so far.

The remaining checks 1-2/2-3 and 9-10/10-11 are both successful. The final subset is formed by nodes 1-2-3-6-9-10-11.

A general flowchart of the adaptive refinement strategy is depicted in figure 4.46.

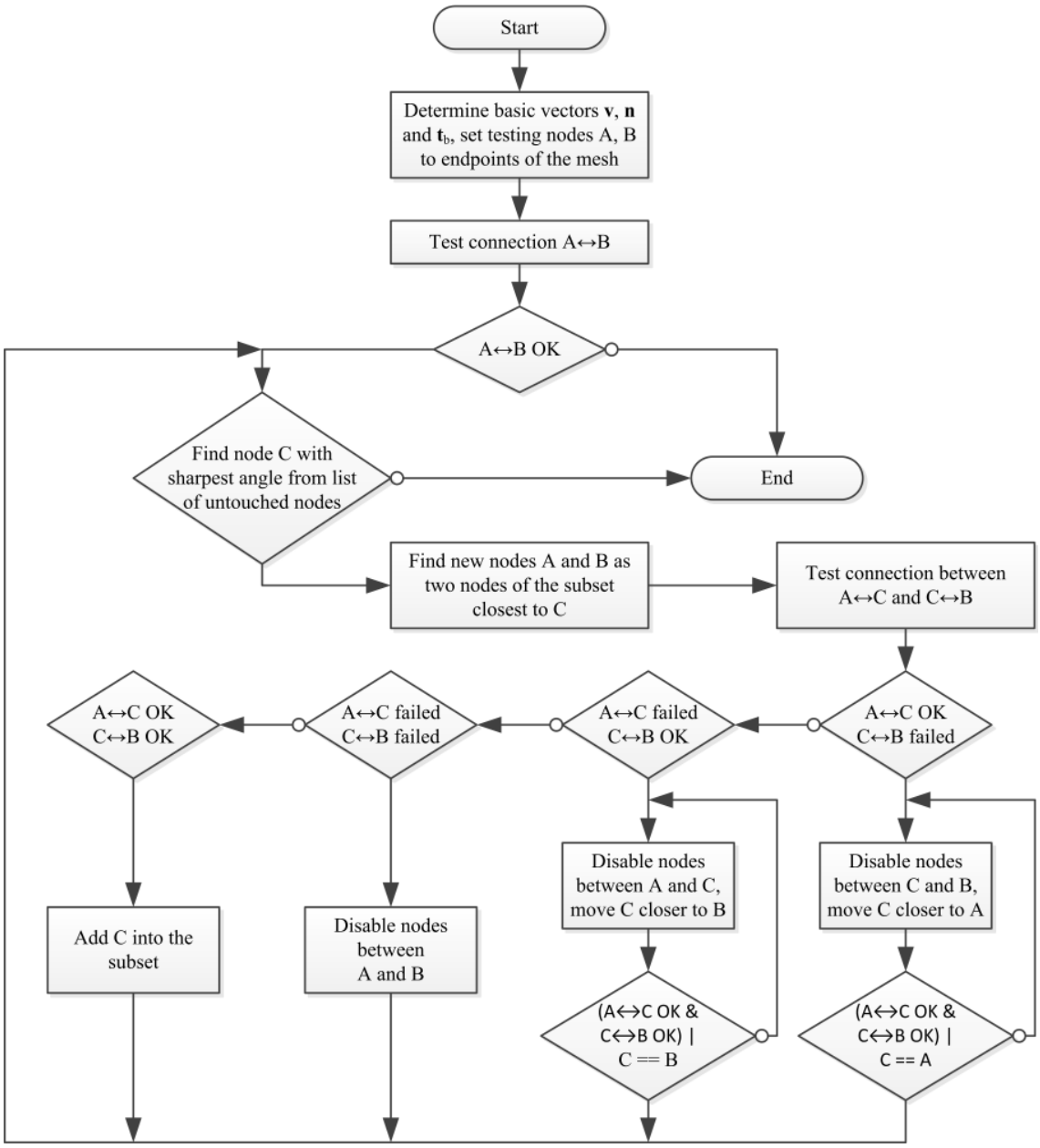


Fig. 4.46 *Progressive refinement construction strategy: flowchart*

4.2.4 Test cases

A comparison between Edelvik's and the proposed approach will be presented in the following chapter. The same geometry and radius are used for both approaches. The radius of the wire is 0.001 m, and the radius of the cylinder r_0 is 0.4 m.

The original nodes are marked by a black cross. Nodes forming the final subset are noted by a red square. The first node is highlighted by a filled square. Points inside the cylindrical area are interior points of the surrounding tetrahedral mesh. They cause non-zero contribution to the coupling between the system of equations of the wire and system of equations of the surrounding space.

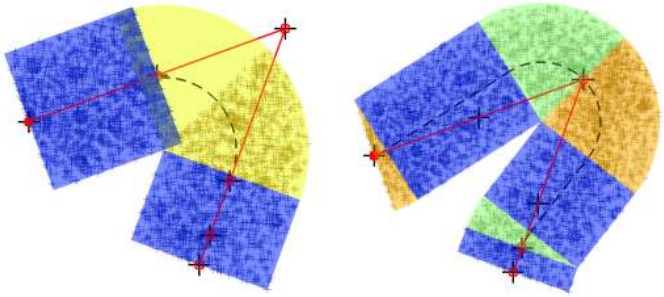
The blue area belongs to the straight area S, orange to BI and green to BII. The original approach does not separate bends. It only has a single bent part which is denoted by a yellow background. However, interior points providing the coupling can be divided into orange and green depending on what element they contribute.

Each part of the cylindrical area (denoted by a given background color) is supposed to be filled with dark points almost uniformly and the parts should not overlap each other. Otherwise, it would indicate some sort of error.

Bends are naturally three-dimensional, however the test bends are performed in a single plane for easier presentation of the results.

A 50 degree bend

First test case is a bend of 50 degrees (see figure 4.47). The original approach (fig. 4.47a) is not omitting any points. Due to a relatively sharp bend, the distance between $\mathbf{z}_{b1}$ and the third node is greater than the distance between the second and third node. The straight part and bend area of the second element (between nodes 2 and 3) are now overlapping with the straight part of the previous one. This causes various errors depending on implementation and presence of validity checks. We may see for example omitting some coupling points in the bend area.



(a) *Edelvik et al.* **(b)** *proposed approach*

Fig. 4.47 *A 50 degree bend*

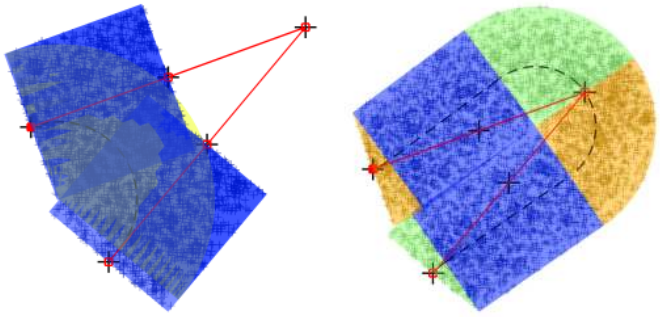
The proposed approach (4.47b) has omitted nodes 2 and 4. The cylinder contains no overlaps and approximates the wire mesh well. Following checks have been performed:

1.) 1-6 (OK)
2.) 1-3 & 3-6 (OK)
3.) 3-5 & 5-6 (OK)
4.) 1-2 & **2-3** (**FAILED**)
5.) **3-4** & 4-5 (**FAILED**)

A 30 degree bend

A bend of 30 degrees is depicted in figure 4.48. Looking at figure 4.48a, we can see that $\mathbf{r}_c$ is now even much further from third node. The wire is not affected by the electric field around node 3.

The proposed approach (4.48b) approximates the cylinder better, however you may notice slight overlap in the middle part of the figure. The procedure ensures non-overlapping parts BI, S and BII within a single wire segment. It does not check mutual overlapping of all segments. The following checks have been performed:



(a) *Edelvik et al.* **(b)** *proposed approach*

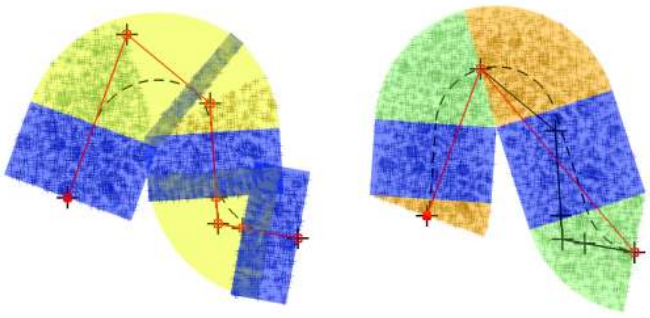
Fig. 4.48 *A 30 degree bend*

1.) 1-5 (OK)
2.) 1-3 & 3-5 (OK)
3.) **3-4** & 4-5 (**FAILED**)
4.) 1-2 & **2-3** (**FAILED**)

A general wire segment

A general wire is shown in figure 4.49. The original approach (fig. 4.49a) does not prevent overlaps. The points which are coupling the wire with surrounding medium do not form a continuous cylinder.

The proposed approach (fig. 4.49b) omits nodes 3 up to 6. This allows to obtain coupling points forming a continuous cylinder. On the other hand, the cylinder area does not follow the original mesh exactly. The following checks have been performed:



(a) *Edelvik et al.* **(b)** *proposed approach*

Fig. 4.49 *A general wire segment*

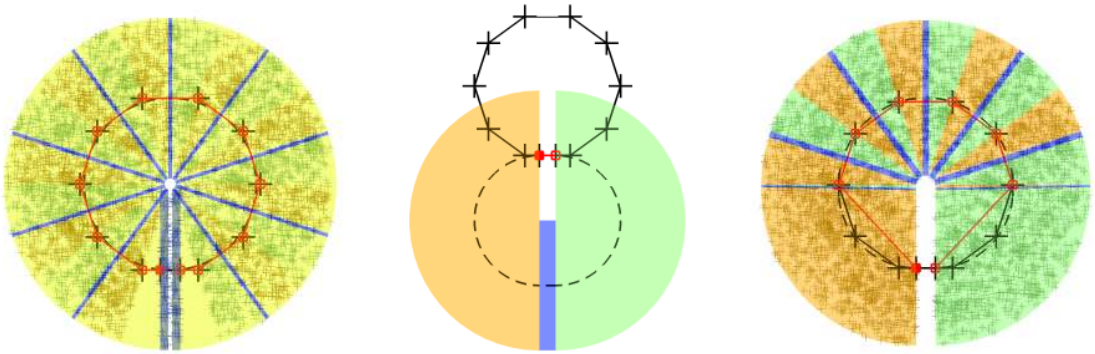
1.) 1-7 (OK)
2.) 1-2 & 2-7 (OK)
3.) 2-5 & **5-7** (**FAILED**)
4.) 2-4 & **4-7** (**FAILED**)
5.) **2-3** & 3-7 (**FAILED**)

A circular loop

A circular loop with a radius of 0.45 m is depicted in figure 4.50 (r_0 = 0.4 m). Note that the loop is not closed. It starts at node 1 (red filled square at the bottom), follows a clockwise direction and ends at node 12.

The original procedure (fig. 4.50a) performs well except the area near endpoints. Segments 1-2 and 11-12 are not long enough and thus the bend parts around nodes 2 and 11 are overlapping with straight parts near nodes 1 and 12 which are exceeding the wire area.

Figure 4.50b shows first iteration of the proposed iterative approach. Since there is no additional node in the final subset, path between nodes 1 and 12 is given randomly and may not follow the wire at all. It may be final state in case that no other successful connections are made (e.g. due to higher r_0). Anyway, the subset cannot be formed only by two nodes. Considering boundary conditions, there are no more unknowns left.



(a) *Edelvik et al.* **(b)** *first iteration of the proposed approach* **(c)** *proposed approach*

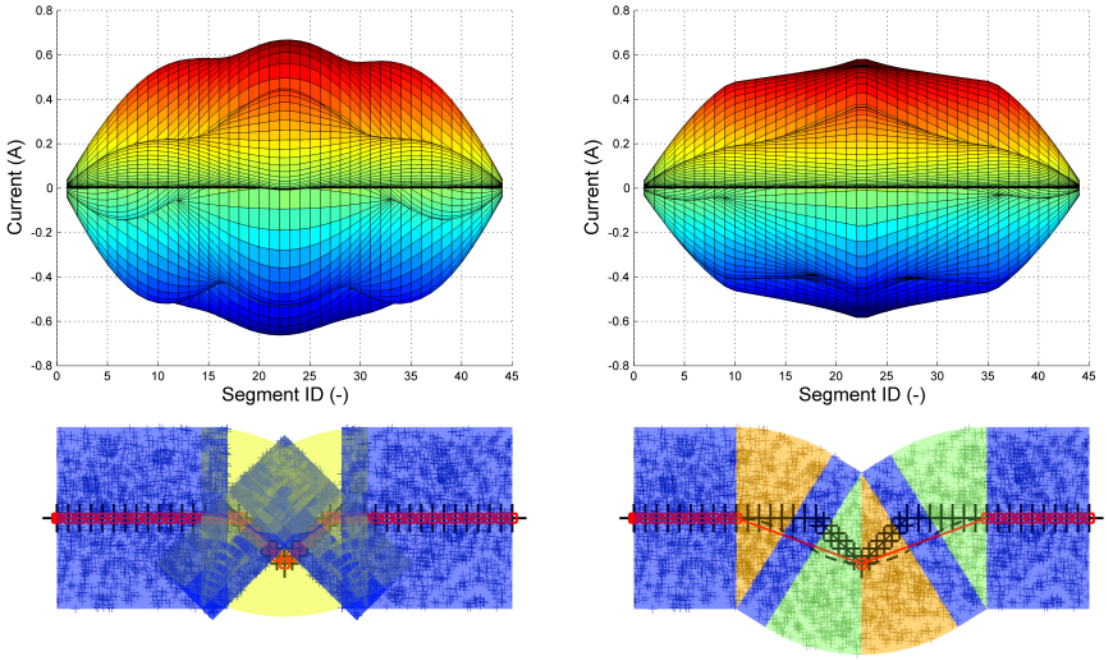
Fig. 4.50 *A circular loop*

Final state of this geometry is depicted in fig. 4.50c. All nodes could not be used but the centerline still follows the original mesh (which is important). The following checks have been performed:

1.) 1-12 (OK)
2.) 1-5 & 5-12 (OK)
3.) 5-8 & 8-12 (OK)
4.) 8-10 & **10-12** (**FAILED**)
5.) 8-9 & 9-12 (OK)
6.) **1-3** & 3-5 (**FAILED**)
7.) 1-4 & 4-5 (OK)
8.) 5-6 & 6-8 (OK)
9.) 6-7 & 7-8 (OK)

Straight wire with a cog

The last example was formerly used for explanation of the adaptive refinement strategy. It is a wire having a cog in the middle of it (see fig. 4.51). Original approach (fig. 4.51a) considers only coupling points given by straight parts of the wire. Bend parts are omitted due to errors in processing (bends are much larger than wire segments).



(a) *Edelvik et al.* **(b)** *proposed approach*

Fig. 4.51 *Wire with a cog in the middle*

Proposed approach (fig. 4.51b) omits many points near the cog to form continuous cylinder. Resulting cog given by nodes 10, 23 and 36 is much more obtuse than the original one formed by nodes 17, 23 and 29.

Influence of such replacement on the results can be significant. Figure 4.51 shows current distribution on that wire. Originally, it is a 3D figure with time axis that is perpendicular to the view. The original approach creates a ripple which is probably caused by overlapping parts in the middle. The proposed approach causes a little loss of accuracy between nodes 17, 23 and 29 caused by presence of only two elements considering linear approximation over them. This can be solved using higher order finite elements.

One may note that the current distribution in figure 4.51 begins with non-zero values. This happens because the resulting current distribution is related to edges of the wire mesh. Values are evaluated in the middle of the original wire segments.

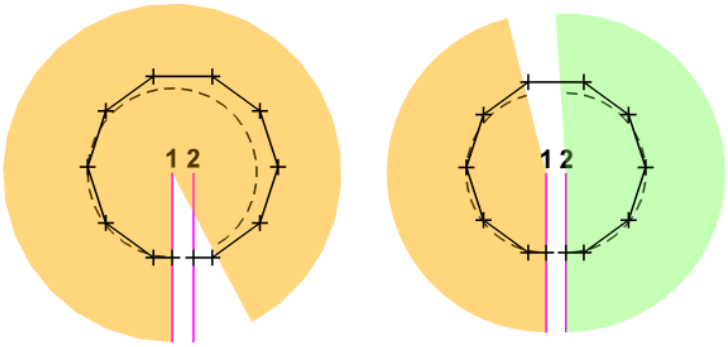
4.2.5 Countermeasures

The iterative process of searching for proper combination of angles φ (among others) converges relatively fast for segments having high length/r_0 ratio. Otherwise the angles may change significantly in each iteration. Particularly connection of segments having large angle between end circles (forming cross-section of the cylinder) may become tricky. This section features two countermeasures to avoid divergence of the iterative process in such case.

Avoiding large angles

Example of the problem with large angles can be shown on the circular loop test case in chapter 4.2.4, specifically figure 4.50b. It shows connection of two parallel circles, each heading away from the other one.

Let us to remind the connection process. After all the main parameters are determined (normals $\mathbf{n}$, $\mathbf{n}_c$, vectors $\mathbf{t}_b$, nodes $\mathbf{s}$ and angles ϑ), the iterative process consist in searching for proper combination of $\varphi_{1,j}$ and $\varphi_{2,j}$. It starts with determining $\varphi_{1,j}$: a rotation of $\mathbf{n}_{c1,j}$ that makes it pointing to $\mathbf{z}_{r2,j}$ (which is the same as $\mathbf{z}_{j+1}$ at the beginning). The $\varphi_{1,j}$ is determined as about 5.8 radians. You can see such rotation in figure 4.52a [9]. The end circles to be connected are colored in magenta. You can see that the first rotation has crossed the second circle. The second rotation cannot find proper angle and connection process of nodes 1 and 12 fails. Since this connection is essential, creation of whole cylindrical area fails consequently as well.



(a) *No countermeasure applied* **(b)** *After countermeasure*

Fig. 4.52 *Avoiding large angles*

The countermeasure consist in avoiding large change of the angles φ within each iteration. At first, $\varphi_{1,j}$ is determined as usual. If determining of $\varphi_{2,j}$ fails, $\varphi_{1,j}$ is reduced by averaging of the present and previous value. Previous value is set to zero at the beginning. This means division of $\varphi_{1,j}$ by two for this case. The smaller change of $\varphi_{1,j}$ allows to determine $\varphi_{2,j}$. Figure 4.52b shows the situation after first successful determination of $\varphi_{2,j}$. Following iterations will increase accuracy of the connection. The method can similarly correct $\varphi_{2,j}$ in case of unsuccessful determination of $\varphi_{1,j}$.

[9] Note that the rotation is performed clockwise (opposite to situation in fig. 4.50b) in order to effectively use image space.

Changing order

In certain situations it is better to start with determining $\varphi_{2,j}$ instead of $\varphi_{1,j}$. This modification lies in change of the order when none of the angles can be properly determined. Reversed order of determining φ was successfully applied in the circular loop test case in connection of nodes 5-12, 8-12 and 9-12.

4.2.6 Conclusion

The proposed approach described in chapter 4.2 aims to improve thin wire approximation of bent wires published in [40]. The improvement involves different procedure of creation of cylindrical area surrounding the wire. Core functionality of the thin-wire model remains the same.

Main advantage of the proposed approach lies in presence of the feedback. The approach allows to detect and prevent overlaps within a wire segment (between two consecutive nodes). The feedback can recommend the user to modify computation mesh or it can be used for an automatic mesh correction during build process of the cylindrical area.

Two strategies for the automatic mesh correction are suggested in chapter 4.2.3: *avalanche processing* and *progressive refinement*. Since the second method exhibited better performance, *avalanche processing* was no longer applied.

Performance of both the approaches ([40] and the proposed one) was compared on test cases in chapter 4.2.4. The figures show that the proposed approach creates much less overlaps. Overlaps may only appear when parts of two different segments are close enough (see 4.48b). Preventing this case would require comprehensive test of segment overlap with any other segment.

The method has been developed for automatic handling of situations where the original approach fails. While focusing on the continuity of the cylindrical area, it may not follow the original route exactly. Results of the simulation with cylindrical area lying elsewhere than the original wire would be considered as results for given wire in that case. This brings some loss of accuracy. However user may be warned about the inconsistency (by number of omitted nodes) and take reasonable measures.

Unfortunately, the loss of accuracy given by overlaps and errors (Edelvik et al.) can be hardly compared with the loss of accuracy given by a different route of the cylinder and omitted nodes (proposed approach). Current distribution cannot be compared due to unavailability of measurements or any validated software, more likely due to inability to ensure same conditions (boundary conditions and excitation).

4.3 File interface

This chapter is devoted to detailed description of the file interface between modules in the HIRF-SE framework. Chapters 4.3.1 and 4.3.2 describe connection of the modules BUTFE_EXC and BUTFE. Chapter 4.3.3 introduces Amelet-HDF C library acting as an input wrapper (used in the BUTFE module). It describes organization of the functions and their main purpose. Finally, chapter 4.3.4 concludes current state of the Amelet-HDF specification and the library.

4.3.1 Module BUTFE_EXC

Tree structure of the input and output files is depicted in figure 4.53. Each object is described by a path starting with "/". This symbol represents the file root, another "/" symbols indicate one step down in the tree structure (e.g. /mesh/mesh_butfe). The file contains groups marked by icon (for unfolded view). Each group can contain another group(s) or dataset(s) marked by for dataset containing numbers or for dataset containing strings. Objects with attributes are denoted by additional red "A" symbol.

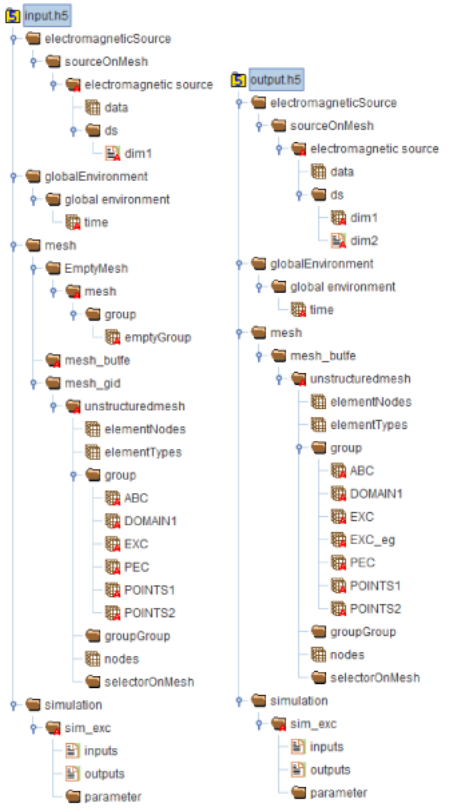


Fig. 4.53 *Input/output files of BUTFE_EXC.*

Entry point of the simulation is *input.h5:/simulation/sim_exc*. Here the dataset *inputs* contains path to all inputs: only the input mesh */mesh/mesh_gid* in this case. The dataset *outputs* contains paths to all requested outputs: */mesh/mesh_butfe*, */globalEnvironment/global environment* and */electromagneticSource/sourceOnMesh/electromagnetic source*.

Amelet-HDF file cannot contain an invalid path. Due to this rule, all the requested outputs are already presented in the input file, however they are empty (see */mesh/EmptyMesh* and */mesh/mesh_butfe*).

The output file looks similarly. It contains only outputs, so that *mesh_gid* is no longer present. Note that there is a new layer *EXC_eg* created by the excitation module from layer *EXC*. Categories */electromagneticSource* and */globalEnvironment* are filled by correct values of excitation coefficients and time samples based on the user definition of the excitation.

4.3.2 Module BUTFE

Figure 4.54 (left) shows a tree structure of the input file for the BUTFE (TDFE) module.

Entry point of the simulation is *input.h5:/simulation/sim*. Contents of the datasets *inputs* and *outputs* are shown in table 4.4.

Path */simulation/sim/parameter* is determined to contain parameters of the simulation. In this case, it contains three attributes: *maxThreads*, *preconditioner* and *solver*. The first one sets the maximal number of threads to be used for the computation. The default value (zero) sets the number of threads equal to the number of cores available in the system. Remaining attributes pick a combination from a list of preconditioners/solvers of the mathematical library *Lis* used for the computation [54].

Because the excitation coefficients */electromagneticSource/sourceOnMesh/electromagnetic source* are related to mesh of same name as the input mesh of the simulation */mesh/mesh_butfe*, the mesh related to the excitation coefficients has been renamed to */mesh/mesh_butfe_1*. Meshes */mesh/mesh_butfe* and */mesh/mesh_butfe_1* are identical.

User has defined four *output request* instances (see tab. 4.5). Each instance defines outputs of the simulation and contains *subject* (physical quantity to compute), *object* (where to compute that quantity) and *output* (storage for the result). Due to the rule related to in/valid paths mentioned before, the storage must be a known valid path within the file. In this case, the storage objects are situated in */floatingType* category. The floatingType objects are of type *data on mesh* (basically dependent on mesh). The framework creates a fictional mesh *EmptyMesh* for each *data on mesh* object. This ensures validity of paths within the input file.

Subjects of the *output request* instances are stored in the */label/output request* dataset. In this case, the dataset contains strings "electricField" and "magneticField" (see tab. 4.6).

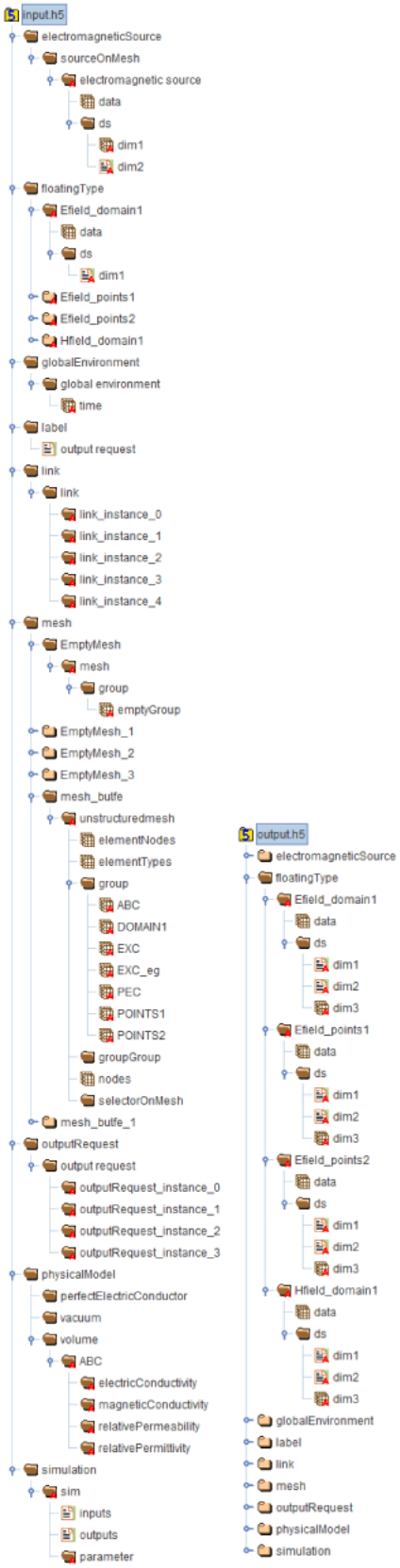


Fig. 4.54 *Input/output files of the BUTFE.*

0	*/outputRequest/output request*
1	*/mesh/mesh_butfe*
2	*/electromagneticSource/sourceOnMesh/electromagnetic source*
3	*/physicalModel/perfectElectricConductor*
4	*/physicalModel/vacuum*
5	*/globalEnvironment/global environment*
6	*/link/link*
7	*/physicalModel/volume/ABC*

(a) */simulation/sim/inputs*

0	*/floatingType/Efield_points1*
1	*/floatingType/Efield_points2*
2	*/floatingType/Hfield_domain1*
3	*/floatingType/Efield_domain1*

(b) */simulation/sim/outputs*

Tab. 4.4 *Simulation inputs/outputs*

Name	**Data**
outputRequest_instance_0	subject: */label/output request*
	subject_id: 0
	object: */mesh/mesh_butfe/unstructuredmesh/group/DOMAIN1*
	output: */floatingType/Efield_domain1*
outputRequest_instance_1	subject: */label/output request*
	subject_id: 0
	object: */mesh/mesh_butfe/unstructuredmesh/group/POINTS1*
	output: */floatingType/Efield_points1*
outputRequest_instance_2	subject: */label/output request*
	subject_id: 0
	object: */mesh/mesh_butfe/unstructuredmesh/group/POINTS2*
	output: */floatingType/Efield_points2*
outputRequest_instance_3	subject: */label/output request*
	subject_id: 1
	object: */mesh/mesh_butfe/unstructuredmesh/group/DOMAIN1*
	output: */floatingType/Hfield_domain1*

Tab. 4.5 *Output request instances*

0	electricField
1	magneticField

Tab. 4.6 *Labels in /label/output request*

The category */physicalModel* defines materials of the model. The *ABC* material is reserved for absorbing boundary condition. In normal situation, material parameters would be set via attributes of groups *electricConductivity*, *magneticConductivity*, *relativePermeability* and *relativePermittivity*.

Instances of the */link* category couple any two objects together. Each instance contains *subject* (material/object) and *object* (place). The instances are detailed in table 4.7. Note that material ABC has to be coupled with excitation layer *EXC* as well in order to achieve proper form of the equation system.

Name	**Data**
link_instance_0	subject: */electromagneticSource/sourceOnMesh/electromagnetic source*
	object: */mesh/mesh_butfe/unstructuredmesh/group/EXC_eg*
link_instance_1	subject: */physicalModel/volume/ABC*
	object: */mesh/mesh_butfe/unstructuredmesh/group/EXC*
link_instance_2	subject: */physicalModel/volume/ABC*
	object: */mesh/mesh_butfe/unstructuredmesh/group/ABC*
link_instance_3	subject: */physicalModel/perfectElectricConductor*
	object: */mesh/mesh_butfe/unstructuredmesh/group/PEC*
link_instance_4	subject: */physicalModel/vacuum*
	object: */mesh/mesh_butfe/unstructuredmesh/group/DOMAIN1*

Tab. 4.7 *Link instances*

The output file (see fig. 4.54, right) differs only in content of the */floatingType* category. The */floatingType* now contains results of the simulation.

The output file must contain only updated */floatingType*, */mesh* and */simulation* categories to allow the framework to read results back to its database. BUTFE is forming the output file by copying the input one and replacing the empty data structures in */floatingType* by the results. This allows to save both results and all input data for the simulation in single file. However, the framework reads only the three above mentioned categories.

Focusing on the objects in */floatingType* category, each of them contains three-dimensional dataset *data* containing the field values. Group *ds* includes three datasets containing description of particular dimension of the *data*. Dataset *dim1* stores path to given mesh group (e.g. */mesh/mesh_butfe/unstructuredmesh/group/DOMAIN1*), *dim2* describes components of the field ("x", "y", "z") and *dim3* contains time samples of the *data*.

4.3.3 Amelet-HDF C library

AxesSim company, a developer of the Amelet-HDF format, created a set of source codes that should help module developers to employ this new file format. However, it appeared that it contained several bugs and memory leaks. So the code was developed rather from scratch instead of using developer's one. Unique orientation of the new code has offered possibility to implement the code as a library and make it more universal so that other partners could benefit from it (although it did not cover all the possibilities of the Amelet-HDF format). The resulting code of the library was uploaded into the source repository of the Amelet-HDF website into a separate branch */svn/amelethdf-c/branches/vlse* [34].

The library provides all necessary functions for reading input files mentioned in chapter 4.3.2 and some extra reading functions which may be valuable for other participants of the project or people who decide to use Amelet-HDF. Since writing capabilities of the solver were pretty well covered by the HDF functions, there was no need to implement them into the library.

The code is divided into files so that each file covers particular category of the Amelet-HDF (except the general purpose ones). Each file contains functions for reading and printing data from given Amelet-HDF file. Functions for unallocating memory occupied by the read data are included too. Names of the library functions start with library prefix "AH5_" followed by "read_", "print_" or "free_" in order to suggest above-mentioned purpose.

Amelet-HDF contains several levels of the tree structure. For example, */outputRequest* category contains various number of groups. Each group contains various number of instances. The library contains functions for accessing each level individually. For example, the function *AH5_read_outputrequest()* reads whole */outputRequest* category. It reads number of groups and calls *AH5_read_ort_group()* to get contents of the instance. Each *AH5_read_ort_group()* reads number of instances and calls *AH5_read_ort_instance()* to get their contents. A programmer can use any of these functions to work on specific level.

The library includes following features of the Amelet-HDF specification:

- sources: *planeWave*, *sphericalWave*, *generator*, *dipole*, *antenna* and *sourceOnMesh*
- all kinds of floatingTypes: *singleInteger*, *singleReal*, *singleComplex*, *singleString*, *vector*, *rationalFunction*, *generalRationalFunction*, *rational*, *dataSet*, *arraySet*
- *mesh* (both types: *structured* and *unstructured*) including *selectorOnMesh* and *meshLink*
- *physicalModel*: *volume material*, *anisotropic*, *surface materials*, *interface*
- *exchange surface*, *external element*, *globalEnvironment*, *label*, *link*, *localizationSystem*, *outputRequest*

File name	Purpose
ah5.h	Includes all header files
ah5_attribute.c, ah5_attribute.h	Integer, float, complex and string attributes
ah5_c_emsource.c, ah5_c_emsource.h	Handles */electromagneticSource* category
ah5_c_exsurf.c, ah5_c_exsurf.h	Handles */exchangeSurface* category
ah5_c_extelt.c, ah5_c_extelt.h	Handles */externalElement* category
ah5_c_fltype.c, ah5_c_fltype.h	Handles */floatingType* category
ah5_c_globenv.c, ah5_c_globenv.h	Handles */globalEnvironment* category
ah5_c_label.c, ah5_c_label.h	Handles */label* category
ah5_c_link.c, ah5_c_link.h	Handles */link* category
ah5_c_locsys.c, ah5_c_locsys.h	Handles */localizationSystem* category
ah5_c_mesh.c, ah5_c_mesh.h	Handles */mesh* category
ah5_c_outreq.c, ah5_c_outreq.h	Handles */outputRequest* category
ah5_c_phmodel.c, ah5_c_phmodel.h	Handles */physicalModel* category
ah5_c_simulation.c, ah5_c_simulation.h	Handles */simulation* category
ah5_category.h	Defines constants (general purpose)
ah5_dataset.c, ah5_dataset.h	Handles datasets (general purpose)
ah5_general.c, ah5_general.h	Other functions (general purpose)

Tab. 4.8 *Files of the Amelet-HDF C library*

4.3.4 Conclusion

This chapter has been devoted to more detailed explanation of encapsulation of the BUTFE modules inside the HIRF-SE platform. Amelet-HDF has proven to be useful format for storing data of the electromagnetic simulations (although there is still some room for improvements). Many developers of EM software are using their own formats which brings compatibility issues.

It should be said that Amelet and especially all the HDF stuff behind may be difficult to understand and limiting when comparing to developing own data format. Overhead connected with the implementation of a new format may be significant, especially for smaller projects. This is exactly the case where the library presents main benefit. It makes the process of the new format understanding and implementation much quicker.

Original contribution of the work mentioned in this chapter consist in encapsulated design of the Amelet-HDF C library and its data structures.

Besides its original use in the BUTFE module, the library was used in modules ANN-Training and ANN-Estimation which were developed for solving postprocessing issues using artificial neural networks in the HIRF-SE framework. The library has recently taken place of the main C library in the Amelet-HDF project. AxesSim company (developer of the Amelet-HDF specification) has decided to incorporate the library in their software and still enhances its functionality.

4.4 Development tools

This chapter deals with description of tools that had to be created in order to simplify development and testing of the modules. Chapter 4.4.1 describes a comparison tool CMP that compares results with a reference. Chapter 4.4.2 is devoted to a mesh inspector/post-processor called Visualizer and finally 4.4.3 concludes this topic.

4.4.1 CMP

The BUTFE module employs OpenMP *application program interface* (API) that supports a shared-memory parallelism [56]. Development of such applications can be tricky. Apart from programming mistakes known from a single-threaded programs, the OpenMP programs can contain additional issues caused by broken thread safety. Some of the thread safety issues can be observed only once per tens of simulations. Performing such a number of simulations and checks manually would require large amount of time. However by using a comparison tool, one may run the simulations in a loop and check resulting comparison of all iterations at once.

The comparison tool examines all simulation outputs of an output Amelet-HDF file with outputs of a reference output file (obtained from a previous simulation declared to be valid). Since the BUTFE module employs an iterative solver, results may vary a little. Therefore a number of samples with relative deviation greater than some limit is observed. Looking at a sample output of the CMP tool (see figure 4.55), we can see that

```
Output #0: (tst-ref)/ref > 2.0% in 0 samples (0.0% of 102576000);
maxdiff[12702, 0, 1275]: ref=1.20796e-010 tst=1.20929e-010
Output #1: (tst-ref)/ref > 2.0% in 0 samples (0.0% of 230000);
maxdiff[0, 0, 0]: ref=0 tst=0
```

Fig. 4.55 *A sample output of the CMP tool.*

the comparison was performed on two output requests with cca 103 and 0.2 millions of samples, respectively. In the first output request, the CMP tool found a maximal deviation from the reference at position [12702, 0, 1275] which was still below the 2 % limit and thus negligible.

4.4.2 Visualizer

Almost all of the input data for FEM are closely connected with the computational mesh. The mesh is formed by a complex set of data which is not human readable without any visualization. The visualization is usually provided by a meshing software, which allows to check validity of the mesh right after its creation. However the mesh data format is converted several times before the computation (mainly due to the Amelet-HDF format).

The Visualizer shows the mesh data right before the computation. It shares its mesh import code with the BUTFE so it operates with same data as the solver. This helps avoiding potential errors caused by wrong mesh import and visualizing other variables related to the solver (global edges, outputs).

In fact, the Visualizer tool does the job of the module Paraview or post-processing mode of the module GiD mentioned in chapter 4.1.2. The reason for developing the in-house visualization tool was due to the fact that the GiD did not work at all with the BUTFE outputs and the Paraview module had usually crashed. The Visualizer is even better because it can work in the course of running simulation while the others would have to wait for the end of the simulation to post-process the output data.

The tool operates in three modes. The first one is mesh inspector only. It is called by the Visualizer executable with a name of an Amelet-HDF file as an input parameter.

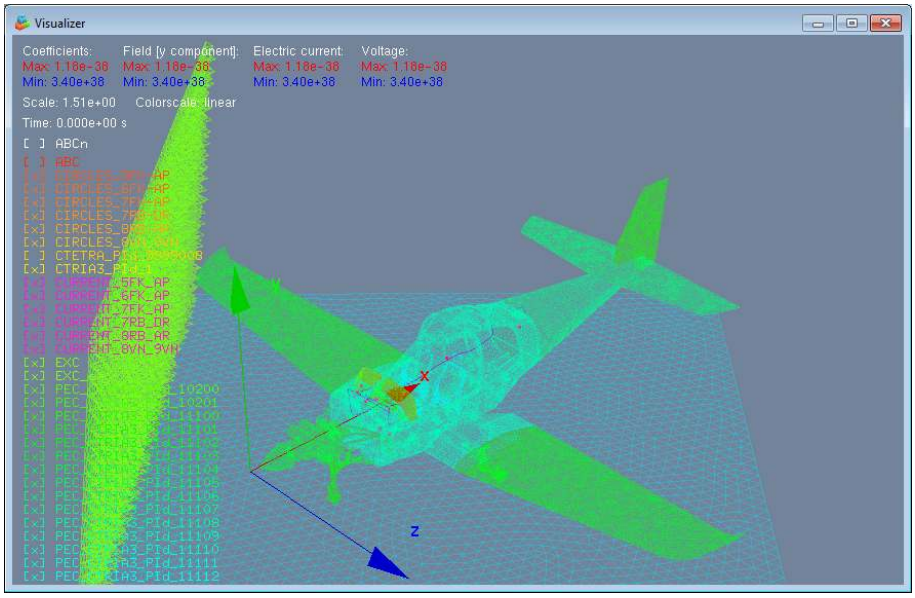


Fig. 4.56 *Visualizer: mesh inspector.*

The second mode is combined inspector/post-processor. The Visualizer executable is called with no parameters. After the execution, the visualization tool waits for the execution of the solver (BUTFE module). Once the solver is running, the visualization tool starts displaying the mesh and output requests of the simulation (only one simulation is expected to running at a time). If the simulation is already running, visualization starts immediately.

The data transfer between both the programs is carried out using *named pipes* [57]. Although this *inter-process communication* (IPC) is not "naturally cross-platform", it is relatively simple and applicable on both the Linux and Windows compilers used for the BUTFE module. The pipe data contain name of the input file of the solver to be read by the visualization tool[10]. The mesh data are stored in the system memory twice. Once for the solver and once for the visualization tool. It may look like a wasting of resources but on the other hand, both the tools must be able to work independently. Rest of the pipe data contains the post-processing data of the output requests for last computed time step. The BUTFE module can work independently considering a small overhead caused by the IPC (it sends the pipe data regardless of a receiving).

[10] The HDF permits that single HDF file can be read by multiple processes [58].

The third mode is post-processor only. It allows to open already saved output file containing results. Unlike the second mode, there is no connection with the solver and no simulation is running. The visualization tool allows to move in time back and forth within given data range of the file.

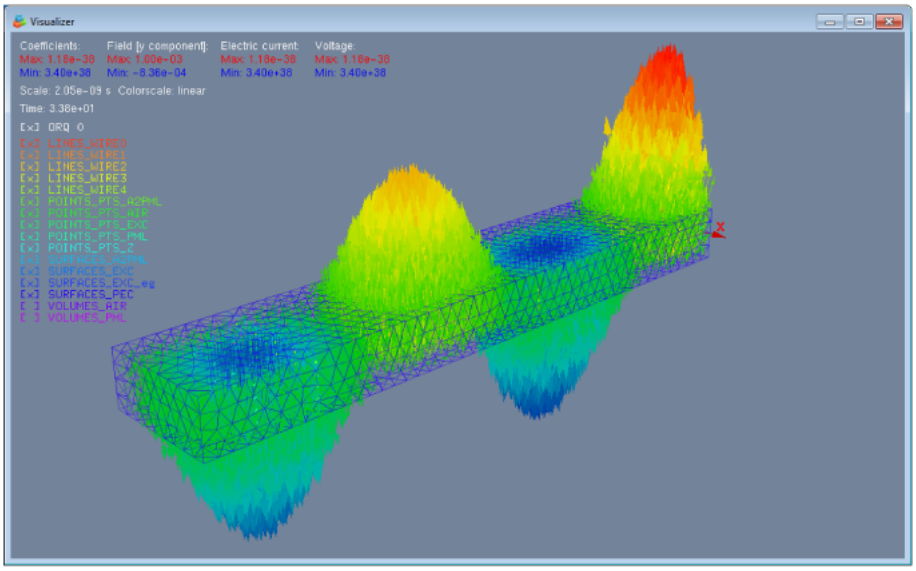


Fig. 4.57 *Visualizer: mesh inspector/post-processor.*

The tool allows to toggle visibility of elements and/or element numbers of particular mesh group, switch between filled and wireframe mode and set type of coloring of the field arrows. The above mentioned options are accessible through a menu by pressing right mouse button. Some of them are also accessible using a keyboard. For example pressing *H* and then *V* hides all volumetric elements in the structure. This approach works similarly for *H*iding *A*ll elements, *N*odes, *E*dges, *S*urfaces, *V*olumes and *F*ield (see figure 4.58).

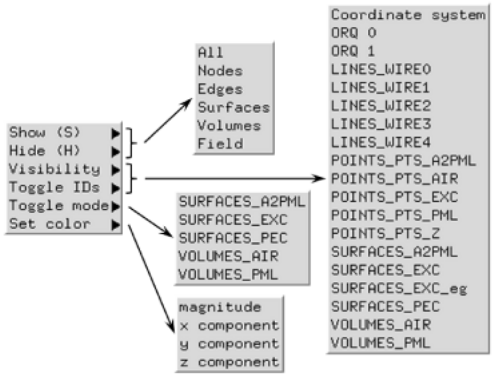


Fig. 4.58 *Visualizer: menu structure.*

Scale of the field arrows can be in/decreased by pressing Q/W. Pressing $T+\langle element\_number \rangle$ highlights particular element and its neighborhood. The neighborhood is formed by all edges, triangles, and tetrahedrons touching selected element. Important information about the element is displayed in the top right area of the window.

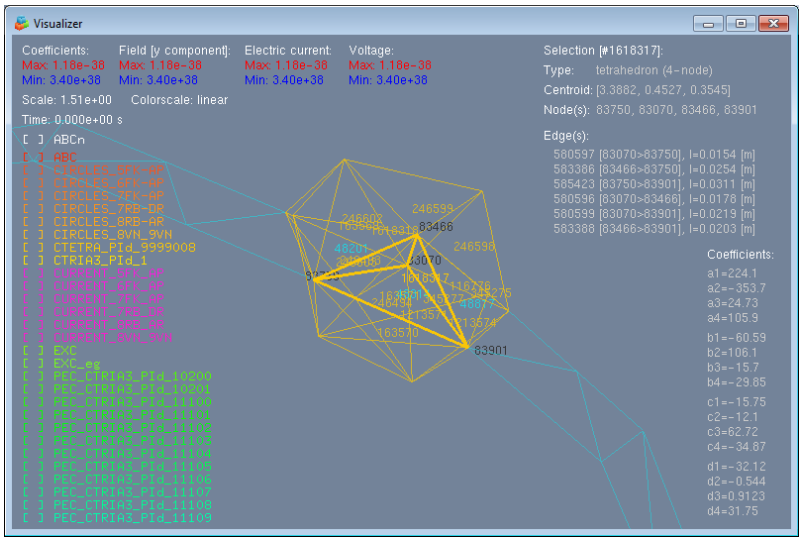


Fig. 4.59 *Visualizer: element inspection.*

Particular global edge of the mesh can be located using key *G*. This is important for the situation, when placement of the given edge coefficient is examined.

Complicated geometry contains lot of overlapped objects which make the inspection difficult. Key *X* cuts all currently not visible elements off. Area of interest can be selected using repetitive cutting, zooming and rotation of the geometry.

Full list of keyboard shortcuts of the tool is listed in table 4.9.

The visualization tool uses OpenGL [59] toolkit FreeGLUT [60]. Display lists, vertex arrays and vertex buffer objects are used in order to improve performance of the code [59].

Desired function	**Keyboard shortcut**
Show/hide all	S, A / H, A
Show/hide nodes	S, N / H, N
Show/hide edges	S, E / H, E
Show/hide surfaces	S, S / H, S
Show/hide volume objects	S, V / H, V
Toggle fullscreen mode	F
Increase/decrease field scale	Q / W
Focus on element	T+⟨element_number⟩
Focus on edge	T+⟨edge_number⟩
Toggle logarithmic scaling on the field	L
Time shift by -10 / -1 / +1 / +10 samples	U / I / O / P
Toggle camera rotation	R
Create a frustum	X

Tab. 4.9 *Visualizer: keyboard shortcuts*

4.4.3 Conclusion

The development process showed the need for some new tools to be designed simultaneously with the solver creation. They were not requested by the project, but they made the development more efficient [11].

The tool CMP became valuable in testing of parallel code. It can be used in the future for automatic comparison of two different approaches implemented in single module or comparison of two different modules giving same kind of results.

The Visualizer had proven its usefulness in many occasions. At the beginning, it was valuable tool for checking intersections between given volumetric mesh and objects inside (objects must be part of the volumetric mesh). The GiD mesher did not provide such an option. Further, some prevention of intersections was incorporated into the solver, however still without any option to see origin of the problem.

Visualization of the results of the ongoing simulation allows to observe evolution of the simulation process. User can stop the simulation in case of any problem or when all important phenomena have ended up earlier than expected.

Visualization of the results of the finished simulation allows to skip uninteresting phase of the simulation and slowly scroll through the interesting one.

[11] "Give me six hours to chop down a tree and I will spend the first four sharpening the axe." Abraham Lincoln [55]

5 CONCLUSION

This thesis deals with four main topics where three of them are of rather applied nature and one provides an improvement of a thin wire approximation in the FEM.

The applied nature of the work consist in development of various programs and libraries aiming to implement a TDFE solver into the HIRF-SE framework. This required to design unique development tools and proposed several innovative solutions. Development of the module was complicated due to the following problems:

- the HIRF-SE framework became available too late
- the need to rewrite the code into the C language while it was originally written in the MATLAB©
- lack of any excitation tool
- lack of any suitable post-processing tool
- problem with implementation of the time-domain PML into the solver

Since many of the HIRF-SE code developers joined the project with already finished programs, their main task was only to adapt their solver to the framework by adding support of the Amelet-HDF format. On the other hand, the BUTFE module has originated from scratch during the project lifetime. Therefore the Amelet-HDF format became native input/output format of the module. BUTFE required input data which should be generated by the HIRF-SE framework (which was not available at that time). This issue was solved by programming a substitute code for assembling input data for the solver.

The need to rewrite the code into the C language was probably caused by some misunderstanding in the requirements during initial phase of the project.

Definition of an excitation for the solver was expected to be handled by the framework itself. The solver had to be a commandline tool only and therefore it would be impossible (extremely user-unfriendly) to define excitations without any GUI. New module BUTFE_EXC was invented to deal with this problem.

The lack of suitable post-processing tool was solved by programming of the Visualization tool mentioned in chapter 4.4.2.

Problem with the PML remains unsolved. All previous attempts to implement the PML failed.

Despite all the mentioned complications, the module was succesfully implemented into the framework as an optional module. The Amelet-HDF library (chapter 4.3.3) recorded succes while it became an official C library of the Amelet-HDF project.

The improvement of the thin wire approximation (chapter 4.2) lies in different procedure of construction of the cylindrical area around the wire, core of the method remains the same. The proposed procedure can detect and prevent overlaps of neighboring wire segments. The feedback allows to implement automatic omission of certain nodes and creation of a geometry that approximates the original wire mesh most closely which leads to more accurate results provided by the numerical solver. The method cannot prevent overlaps caused by proximity of two or more nodes that are not strictly next to each other

in the wire mesh. However, occurrence of such situation is less probable than occurrence of overlaps caused by two consecutive nodes. A general algorithm that would avoid intersection caused by non-consecutive nodes could be subject of a future investigation.

Experiences were presented on the International Travelling Summer School on Microwaves and Lightwaves [45], International Conference on Electromagnetics in Advanced Applications [46] and on the COST workshop [47]. Integration and validation of the TDFE module was presented in a joint paper prepared with colleagues from University of Gdansk and Sapienza University of Rome [48].

BIBLIOGRAPHY

[1] RAINFORD EMC SYSTEMS LIMITED. Aircraft Chamber [online]. 1997 [cit. 2011-04-13]. Available at WWW: <http://www.rainfordemc.com/aircraft-chamber.html>.

[2] SADIKU, M.N.O.; PETERSON, A.F. A comparison of numerical methods for computing electromagnetic fields. In *Proceedings of the IEEE Southeast Conference*. 1990, p. 42-47.

[3] SADIKU, M.N.O. *Numerical Techniques in electromagnetics*. 2nd ed. [s.l.]: CRC Press, 2001. 743 pages. ISBN 0-8493-1395-3.

[4] COURANT, R. Variational methods for the solution of problems of equilibrium and vibrations. *Bulletin of the American Mathematical Society* 1943, vol. 49, p. 1-23.

[5] SILVESTER, P. P. Finite-element solution of homogeneous waveguide problems. *URSI Symposium on Electromagnetic Waves*. 1968, paper 115.

[6] SILVESTER, P. P., FERRARI, R. L. *Finite Elements for Electrical Engineers*. 3rd ed. Cambridge: Cambridge University Press, 1996. 516 pages. ISBN 0-5214-4505-1.

[7] SILVESTER, P.P.; PELOSI, G. *Finite Elements for Wave Electromagnetics: Methods and Techniques*. Piscataway, NJ: IEEE Press, 1994. 534 pages. ISBN 0-7803-1040-3.

[8] JIN, J.M. *The Finite Element Method in Electromagnetics*. 1st ed. New York: Wiley, 1993. 442 pages. ISBN 0-471-58627-7.

[9] JIN, J.M. *The Finite Element Method in Electromagnetics*. 2nd ed. New York: Wiley, 2002. 780 pages. ISBN 0-471-43818-9.

[10] JIN, J.M., Riley, D.J. *Finite Element Analysis of Antennas and Arrays*. Wiley, 2009. 435 pages. ISBN 0-470-40128-1.

[11] *List of finite element software packages* [online]. 11.12.2008, 6.4.2011 [cit. 2011-04-19]. Wikipedia, the free encyklopedia. Available at WWW: <http://en.wikipedia.org/wiki/List_of_finite_element_software_packages>.

[12] *Přehled open-source softwaru na bázi MKP* [online]. 27.6.2007 [cit. 2011-04-19]. <http://wood.mendelu.cz/cz/sections/SC/files/OS%20FEM.pdf>.

[13] BERENGER, J.P. A perfectly matched layer for the absorption of electromagnetic waves. *Journal of Computational Physics*. October 1994, vol. 114, no. 2, p. 185-200.

[14] SACKS, Z.S.; KINGSLAND, D.M.; LEE, R.; LEE, J.F. A perfectly matched anisotropic absorber for use as an absorbing boundary condition. *IEEE Transactions on Antennas and Propagation*. December 1995, vol. 43, no. 12, p. 1460

[15] MATHIS, V. An anisotropic perfectly matched layer-absorbing medium in finite element time domain method for Maxwell's equations. In *Proceedings of IEEE Antennas and Propagation Society International Symposium*. 1997, vol. 2, p. 680-683.

[16] TSAI, H.P.; WANG, Y.; ITOH, T. An unconditionally stable extended (USE) finite-element time-domain solution of active nonlinear microwave circuits using perfectly matched layers. *IEEE Transactions on Microwave Theory and Techniques*. Oct. 2002, vol. 50, no. 10, p. 2226–2232.

[17] JIAO, D.; JIN, J.M.; MICHIELSSEN, E.; RILEY, D.J. Time-domain finite-element simulation of three-dimensional scattering and radiation problems using perfectly matched layers. In *IEEE Transactions on Antennas and Propagation*. Feb 2003, vol. 51, no. 2, p. 296-305.

[18] RYLANDER, T.; JIN, J.M. Perfectly matched layer in three dimensions for the time-domain finite element method applied to radiation problems. In *IEEE Transactions on Antennas and Propagation*. April 2005, vol. 53, no. 4, p. 1489-1499.

[19] RYLANDER, T.; JIN J.M. Conformal perfectly matched layers for the time domain finite element method. *IEEE Antennas and Propagation Society International Symposium*. June 2003, vol.1, p. 698-701.

[20] LOU, Z.; CORREIA, D.; JIN, J.M. Second-Order Perfectly Matched Layers for the Time-Domain Finite-Element Method. *IEEE Transactions on Antennas and Propagation*. March 2007, vol. 55, no. 3, p. 1000-1004.

[21] YOON, S.; LEE, J.H.; WON, T. A novel advancing front meshing algorithm for 3D parallel FEM. *International Conference on Simulation of Semiconductor Processes and Devices*. 1999, p. 135-138.

[22] FIALKO, S. PARFES: A method for solving finite element linear equations on multi-core computers. *Advances in Engineering Software*. 2010, vol. 41, no. 12, p. 1256-1265. ISSN 0965-9978.

[23] CECKA, C.; LEW, A.; DARVE, E. Assembly of Finite Element Methods on Graphics Processors. International Journal for Numerical Methods in Engineering, 2000, p. 1-35. Available at WWW: <http://www.stanford.edu/ ccecka/research/FEMGPU/FEMGPU_Paper.pdf>.

[24] YE, Z.B.; DING, D.Z.; FAN, Z.H.; YANG, Y. Analysis of waveguides by use of the time-domain finite-element method solved by preconditioned iterative methods. *IET Microwaves, Antennas & Propagation*. February 2009, vol. 3, no. 1, p. 23-31.

[25] LEE, S.C.; LEE, J.F.; LEE, R. Hierarchical vector finite elements for analyzing waveguiding structures. *IEEE Transactions on Microwave Theory and Techniques*. August 2003, vol. 51, no. 8, p. 1897-1905.

[26] GIANNACOPOULOS, D.D. Optimal discretization-based load balancing for parallel adaptive finite-element electromagnetic analysis. *IEEE Transactions on Magnetics.* March 2004, vol. 40, no. 2, p. 977-980.

[27] SEHLSTEDT, N.; ZDUNEK, A.; RACHOWICZ, W. Application of an hp-adaptive FE method for computing electromagnetic scattering in the frequency domain. *IEEE/ACES International Conference on Wireless Communications and Applied Computational Electromagnetics, 2005.*. Honolulu(Hawai) : Hilton Hawaiian Village 3-7 April 2005, p. 682-685. ISBN 0-7803-9068-7

[28] COCCIOLI, R. ; ITOH, T. ; PELOSI, G. ; SILVESTER, P.P. Finite-element methods in microwaves: a selected bibliography. *IEEE Antennas and Propagation Magazine.* 1996, vol. 38, no. 6, p. 34-48.

[29] JIN, J.M.; LOU, Z.; LI, Y.J.; RILEY, N.W.; RILEY, D.J. Finite Element Analysis of Complex Antennas and Arrays. *IEEE Transactions on Antennas and Propagation.* Aug. 2008, vol. 56, no. 8, p. 2222-2240.

[30] TEIXEIRA, F.L. Time-Domain Finite-Difference and Finite-Element Methods for Maxwell Equations in Complex Media. *IEEE Transactions on Antennas and Propagation.* August 2008, vol. 56, no. 8, p. 2150-2166.

[31] LEE, J.F.; LEE, R.; CANGELLARIS, A. Time-domain finite-element methods. *IEEE Transactions on Antennas and Propagation.* 1997, vol. 45, no. 3, p. 430-442.

[32] *HIRF SE High Intensity Radiated Field Synthetic Environment* [online]. 2008 [cit. 2011-04-14]. Available at WWW: <http://www.hirf-se.eu>.

[33] *HIRF SE High Intensity Radiated Field Synthetic Environment* [online]. 2008 [cit. 2011-04-22]. Description — HIRF SE. Available at WWW: <http://www.hirf-se.eu/hirf/?q=node/9>.

[34] *amelet-hdf - Tool development for Amelet-HDF Specification. - Google Project Hosting* [online]. 2009 [cit. 2011-04-26]. Available at WWW: <http://code.google.com/p/amelet-hdf/>.

[35] *The HDF Group : Information, Support, and Software* [online]. 1988 [cit. 2011-04-26]. Available at WWW: <http://www.hdfgroup.org/>.

[36] ŠTUMPF, M.; DE HOOP, A.; LAGER, I. Closed-Form Time- Domain Expressions for the 2D Pulsed EM Field Radiated by an Array of Slot Antennas of Finite Width. In *Symposium Digest - 20th International URSI Symposium on Electromagnetic Theory.* Berlin: International Union of Radio Science, 2010. p. 786-789. ISBN: 978-1-4244-5154- 8.

[37] ŠEDĚNKA, V. *Synthesis of microwave resonators*. Brno: Brno University of Technology, Faculty of Electrical Engineering and Communication, 2008. 42 pages. Supervisor prof. Dr. Ing. Zbyněk Raida.

[38] ŠEDĚNKA, V.; RAIDA, Z. Critical Comparison of Multi-objective Optimization Methods: Genetic Algorithms versus Swarm Intelligence. *Radioengineering*. 2010, vol. 19, no. 3, p. 369-377. ISSN: 1210-2512.

[39] RAIDA, Z.; ŠKVOR, Z.; et al. Analýza mikrovlnných struktur v časové oblasti. Brno : VUTIUM, 2004. 232 pages. ISBN 80-214-2541-5

[40] EDELVIK, F.; LEDFELT, G.; LÖTSTEDT, P.; RILEY, D. J. An Unconditionally Stable Subcell Model for Arbitrary Oriented Thin Wires in the FETD Method. *IEEE Transactions on Antennas and Propagation*. 2003, vol. 51, no. 8, p. 1797-1805.

[41] Rotation matrix. *Wikipedia, the free encyclopedia* [online]. 2013-04-18 [cit. 2013-04-19]. Available at WWW: <http://en.wikipedia.org/wiki/Rotation_matrix>.

[42] SAVAGE, J.S.; PETERSON, A.F. Higher-order vector finite elements for tetrahedral cells. *IEEE Transactions on Microwave Theory and Techniques*. June 1996, vol. 44, no. 6, p. 874-879.

[43] CIGÁNEK, J.; KADLEC, P.; RAIDA, Z.; ŠEDĚNKA, V.; ŠTUMPF, M. *WP3 - Technical Note 14 - BUT - TDFE-PEMF (technical note for FP7 project no. 205294, HIRF SE)*. 2010. p. 1-115.

[44] *MathWorks - MATLAB and Simulink for Technical Computing : Summary of MATLAB HDF5 capabilities - MATLAB* [online]. 1994 [cit. 2011-04-26]. hdf5. Available at WWW: <http://www.mathworks.com/help/techdoc/ref/hdf5.html>.

[45] ŠEDĚNKA, V. Advanced finite element method. In *Proceedings of 20th International Travelling Summer School on Microwaves and Lightwaves*. 2010, p. 1-8.

[46] CIGÁNEK, J.; KADLEC, P.; RAIDA, Z.; ŠEDĚNKA, V. Finite Element Analysis of Composite Objects: Time Domain Versus Frequency Domain. In *2010 International Conference on Electromagnetics in Advanced Applications*. Sydney (Australia) : IEEE, 2010. p. 164-167. ISBN: 978-1-4244-7367- 0.

[47] KADLEC, P.; RAIDA, Z.; ŠEDĚNKA, V.; CIGÁNEK, J. Time domain finite element analysis of electrically large composite objects. In *8th Management Comitee business meeting and Working Groups Worshop*. 2010, p. 1-13.

[48] ŠEDĚNKA, V., CIGÁNEK, J., KADLEC, P., RAIDA, Z., WIKTOR, M., SARTO, M.S., GRECO, S. Time-Domain Finite Elements for Virtual Testing of Electromagnetic Compatibility. *Radioengineering*. 2013, vol. 22, no. 1, p. 309-317.

[49] *MathWorks - MATLAB and Simulink for Technical Computing : MATLAB Compiler - MATLAB* [online]. 2011 [cit. 2011-12-28]. MATLAB Compiler. Available at WWW: <http://www.mathworks.com/products/compiler/>.

[50] *MathWorks - MATLAB and Simulink for Technical Computing : Working with the MCR :: Deployment Process (MATLAB©Compiler^{TM})* [online]. 2011 [cit. 2011-12-28]. Working with the MCR. Available at WWW: <http://www.mathworks.com/help/toolbox/compiler/f12-999353.html>.

[51] *Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities* [online]. 2011 [cit. 2011-12-30]. Available at WWW: <http://geuz.org/gmsh/>.

[52] *www.gidhome.com* [online]. 2011 [cit. 2011-12-30]. What's GiD. Available at WWW: <http://gid.cimne.upc.es/>.

[53] *ParaView - Open Source Scientific Visualization* [online]. 2011 [cit. 2011-12-30]. ParaView. Available at WWW: <http://www.paraview.org/>.

[54] *SSI: Scalable Software Infrastructure for Scientific Computing: Lis: a Library of Iterative Solvers for Linear Systems* [online]. 2011 [cit. 2011-12-31]. Available at WWW: <http://www.ssisc.org/lis/index.en.html>.

[55] *Give me six hours to chop... at BrainyQuote* [online]. 2001 - 2012 [cit. 2012-05-01]. Available at WWW: <http://www.brainyquote.com/quotes/quotes/a/abrahamlin109275.html>.

[56] *OpenMP.org*: *The OpenMP® API specification for parallel programming* [online]. 2012 [cit. 2012-05-01]. Available at WWW: <http://www.openmp.org/>.

[57] *Named pipe - Wikipedia, the free encyclopedia*: *Named pipe* [online]. 2012 [cit. 2012-05-01]. Available at WWW: <http://en.wikipedia.org/wiki/Named_pipe>.

[58] *HDF5 FAQ – Questions About the Software* [online]. 2012 [cit. 2011-05-01]. Available at WWW: <http://www.hdfgroup.org/hdf5-quest.html#gconc1>.

[59] *OpenGL - The Industry Standard for High Performance Graphics* [online]. 2012 [cit. 2011-05-02]. Available at WWW: <http://www.opengl.org/>.

[60] *The freeglut Project :: About* [online]. 2012 [cit. 2011-05-02]. Available at WWW: <http://freeglut.sourceforge.net>.

CURRICULUM VITAE

Name: Vladimír Šeděnka
Born: August 1983
Contact: vlada.cr@volny.cz

Education

2003-2009	**Technical University of Brno / Department of Radio Electronics** Undergraduate study of Radio Electronics State exam passed in June 2009 Diploma thesis Synthesis of microwave resonators defended in June 2009
2009	**Technical University of Brno / Department of Radio Electronics** Ph.D. study of Radio Electronics State exam passed in May 2011

Experience

2009-2012	**HIRF-SE** developer of a finite-element time-domain solver

Courses

2010	**International Travelling Summer School on Microwaves and Lightwaves** Supélec, Metz (France)

Languages

Czech, English, German