POLITECNICO DI TORINO

Corso di Laurea Magistrale
in Ingegneria Energetica e Nucleare

Tesi di Laurea Magistrale

Object oriented modeling of LTS and HTS superconducting cables

Relatore
Prof. Laura Savoldi
Prof. Roberto Silvestro Poccia

Candidato
Daniele Placido

Anno accademico 2020/2021

Abstract

Superconducting (SC) cables and magnets in the past decades have enabled fundamental discoveries in the field of high-energy physic, amazing steps forward in the research on a clean energy based on nuclear fusion and a significant increase in the power transfer capability, as well as reduction of transmission loss and construction cost, for power cables. Different kinds of SC cables are available, according to the applications, based on different SC materials and different concept for their cooling (conduction-cooling, coolant bath of forced-flow). The SC cables for fusion, for instance, are mainly based on a thousand of thin strands, embedding the low-critical-temperature superconducting (LTS) material, twisted together and pulled inside a metal jacket (the so-called cable-in-conduit concept). The coolant, supercritical Helium at $4.5\ \mathrm{K}$ and $5\ \mathrm{bar}$ flows inside the porous matrix originated by the strands. The SC cables for power transmission typically adopt high-critical-temperature superconducting (HTS) tapes, stacked on a stabilizing former and cooled by a forced flow of liquid nitrogen at $77\ \mathrm{K}$. The cables are surrounded by the cryostat.

Focusing on forced-flow SC cables, which are crucial both for fusion applications and for power transmission, the availability of an appropriate, reliable and flexible modelling of the SC cables is of paramount importance. In the field of fusion cables, several numerical tools are well established for the analysis of the transients in fusion (LTS) cables, as the 4C code, developed at DENERG few years ago. In the field of power transmission cables, few numerical tools are available for the HTS cables, based on a very simplified modelling.

A single model, flexible enough to cope with both LTS and HTS cables for fusion applications and power transmission, respectively, is missing from the research arena. The aim of the present thesis is the conceptual development of a single tool for the analysis of thermal-hydraulic transients in SC cables, though from the very beginning to be capable to model LTS, as well as HTS cables. Euler-like sets of 1D equations should be used to model the fluid flow along the cable, for an arbitrary number of fluid regions in single phase. The fluid equations need to be coupled to 1D transient heat conduction equations, for the SC strands/tapes and for the stabilizer elements and jacket/insulation layers/cryostat. The spatial grid used to discretize the equations with a finite element scheme should account for non-uniform elements distribution along the cable, while the time-marching scheme should take advantage of an adaptive time stepping.

The model should be developed based on a user-friendly concept of a graphical interface that allows to easily define the cable topology, to choose the appropriate numerical solvers and run the simulations, checking the evolution of selected thermal-hydraulic variables during the code execution.

In perspective, the model needs to be extended to include also a lumped electrical model.

TABLE OF CONTENTS

List of symbols vii
List of acronyms xi
1 Introduction 1
1.1 Context 1
1.2 Overview of modeling of SC cables for fusion applications 4
1.3 Overview of modeling of SC cables for power applications 6
1.4 Aim and novelty of the work 10
1.5 Structure of the thesis 11
2 Mesoscale modelling of superconductors 13
2.1 Mathematical model 13
2.1.1 Fluid components equations 14
2.1.2 Solid components equations 15
2.1.3 Coupling of equations 17
2.1.4 The final set of PDEs 18
2.1.4.1 Case study equations 18
2.1.4.1.1 3P-HTS configuration 18
2.1.4.1.2 ITER LTS Toroidal Field Coil configuration 20
2.1.5 Summary 21
2.2 Numerical schemes 22
2.2.1 Spatial discretization with Finite Elements Method (FEM) 22
2.2.1.1 Mesh construction 24
2.2.2 Time discretization 27
2.2.3 Summary 29
3 Code description 31
3.1 The SC2 code organization in nutshell 31
3.1.1 SC2 architecture 31
3.1.2 SC2 input and output organization 37
3.1.3 Summary 41
3.2 Code kernel: Conductors, FluidComponents and SolidComponents classes 41

3.2.1 FluidComponents class 43
3.2.2 SolidComponents class 44
3.2.3 Conductors class 47
3.2.3.1 __init__ method 49
3.2.4 Summary 55
3.3 Simulation class: the steps through the solution 55
3.3.1 Fluid components initialization 57
3.3.2 Solid components initialization 60
3.3.3 BCs application and problem solution 61
3.3.4 Summary 61
3.4 A user-friendly GUI 62
3.4.1 Run… 63
3.4.2 …and check (real-time visualization of the results) 68
3.4.3 Summary 68
3.5 Ease of post processing 68
3.5.1 How the data are managed 68
3.5.2 Default post processing 78
3.5.3 Advanced post processing 80
3.5.4 Summary 84
4 SC2 Verification and Validation 85
4.1 Convergence analyses 85
4.1.1 Space convergence 87
4.1.2 Time convergence 88
4.1.3 Summary 90
4.2 Validation against the 4C code 90
4.2.1 Benchmark with the 4C code: an HTS power cable 91
4.2.2 Benchmark with the 4C code: an ITER LTS Toroidal Field Coil cable 99
4.2.3 Summary 109
4.3 SC2 versatility checks 110
4.3.1 Inner benchmark: INTIAL 111
4.3.2 Inner benchmark: BE vs CN 120

4.3.3 Inner benchmark: backflow 121
4.3.4 Inner benchmark: refined mesh 127
4.3.5 Summary 130
5 Conclusions and future perspectives 131
A Extended form of equations 135
B Matrix elements 141
B.1 General form of matrices and vectors 141
B.2 3P-HTS case study: matrices and vectors 148
B.3 ITER-TF case study: matrices and vectors 150
C Evaluation of the properties of the solid components 155
D Input data of the simulations 159
D.1 3P-HTS input data 159
D.2 ITER-TF input data 163
E Further details 169
E.1 How to compile input files 169
E.2 Interfaces and coupling between components 169
Bibliography 171

LIST OF SYMBOLS

Scalar quantities

Quantity	Symbol	Unit SI
hydraulic diameter	D_h	m
inverse characteristic time	$\tilde{F}$	$1/\mathrm{s}$
specific heat at constant pressure	c_p	$\mathrm{J/kg/K}$
specific heat at constant volume	c_v	$\mathrm{J/kg/K}$
localized pressure drop coefficient	$k_{f,loc}$	$-$
mass flow rate	$\dot{m}$	$\frac{\mathrm{kg}}{\mathrm{s}}$
energy source term	Λ_e	$\mathrm{W/m^3}$
momentum source term	Λ_v	$\mathrm{J/m^4}$
mass source term	Λ_ρ	$\mathrm{kg/(m^3\ s)}$
parameter to keep into account the kind of interface	λ_v	$-$
reference value	ξ_0	$-$
heat transfer coefficient	h	$\mathrm{W/(m^2\ K)}$
length of the j-th subinterval	Δx_j	$-$
energy variation	ΔE	J
mass variation	Δm	kg
pressure drop	Δp	Pa
time increment	Δt	s
cross section	Σ	$\mathrm{m^2}$
heat capacity	C	$\mathrm{J/(m^3\ K)}$
average fluid acceleration in fluid component object	F	$\mathrm{m/s^2}$
mass transport coefficient	K'	ms
momentum transport coefficient	K''	$\mathrm{m^2}$
energy transport coefficient	K'''	$\mathrm{m^3/s}$
conductor length	L	m
number of something (according to the subscript), with no subscript the number of inner nodes of the spatial discretization of the domain.	N	$-$
contact perimeter	P	m
linear heat source	Q	$\mathrm{W/m}$
total linear heat sources in nodal point	$Q1, Q2$	$\mathrm{W/m}$
temperature	T	K
speed of sound	c	$\mathrm{m/s}$
specific energy	e	$\mathrm{J/kg}$
friction factor	f	$-$
i-th iteration	i	$-$
thermal conductivity	k	$\mathrm{W/(m\ K)}$
pressure	p	Pa
time	t	s
velocity	v	$\mathrm{m/s}$

Quantity	Symbol	Unit SI
specific enthalpy	w	$\mathrm{J/kg}$
spatial coordinate	x	m
generic vector element	y	$-$
Gruneisen parameter	Φ	$-$
hydraulic coefficient of the fluid component	α	$1/(\mathrm{kg\ m})$
smoothing coefficient	δ	$-$
relative error	ε	$-$
θ-method coefficient	θ	$-$
generic solution value (velocities, pressures or temperatures)	ξ	$-$
density	ρ	$\mathrm{kg/m^3}$
generic property value of fluid or solid components	χ	$-$
partial derivative operation	∂	$-$

Functions and finite element subspace

Quantity	Symbol
finite-dimensional subspace	$\mathrm{V_h}$
j-th sub interval of the spatial discretization	I_j
generic function (e.g. material property)	$f(\dots)$
basis function of the finite dimensional subspace	ψ
function that belongs to the finite dimensional subspace	ω
value of the function at the nodal point	$\bar{\omega}$

Matrices and vectors

Quantity	Symbol
matrix of basis function of the finite dimensional subspace	Ψ
matrix of test functions of the finite dimensional subspace	Ω
advection matrix	A
stiffness matrix	A_{sys}
conductive matrix	K
mass matrix	M
source terms matrix	S
not null vector constituting the diagonals of the advection matrix	$\boldsymbol{A}$
not null vector constituting the diagonal of the conductive matric	$\boldsymbol{K}$
not null vector constituting the diagonal of the mass matric	$\boldsymbol{M}$
vector of the linear heat source	$\boldsymbol{Q}$
not null vector constituting the diagonal of the source term matrix	$\boldsymbol{S}$
vector of the temperature	$\boldsymbol{T}$
known term vector	$\boldsymbol{b}$
vector of the pressure	$\boldsymbol{p}$
vector of the source terms	$\boldsymbol{s}$
vector of unknowns	$\boldsymbol{u}$
vector of the velocity	$\boldsymbol{v}$
generic vector	$\boldsymbol{y}$

LIST OF SYMBOLS

Quantity	Symbol
error vector	$\boldsymbol{\varepsilon}$
generic vector of the solution (velocities, pressures or temperatures)	$\boldsymbol{\xi}$
generic vector of property of fluid or solid components	$\boldsymbol{\chi}$
null vector	$\mathbf{0}$
identity vector	$\mathbf{1}$

Subscript and superscripts

Symbol	Meaning
$\parallel$	hydraulic parallel
$\perp$	transversal direction
$4C$	values obtained from simulation with the 4C code
BE	Backward Euler
C	close
$CH1$	fluid component CHAN_1
$CH2$	fluid component CHAN_2
CN	Crank-Nicolson
$Gauss$	quantity evaluated in the midpoint (Gauss point) of the spatial discretization of the domain
$JK1$	jacket component Z_JACKET_1
$Joule$	heat due to Joule effect in the strand(s)
O	open
$SC2$	values obtained from simulation with the SC2 code
$ST1$	strand component STR_MIX_1
$Sim1$	first simulation to perform the inner benchmark, the reference one
$Sim2$	first simulation to perform the inner benchmark, the test one
T	temperature
a	left end of the refined region
ave	average
b	right end of the refined region
$back$	backward flow simulation (flow from right to left)
beg	begin (of the transient)
ch	fluid component (channel)
ca	generic fluid component (channel)
$coarse$	coarse region
$cond$	conductor
$conv$	space or time convergence analysis
end	end (of the transient)
eq	total number of mathematical equations to model the conductor
ext	external heat
fi	generic current-carrying solid component (strand)
$flow$	forward flow simulation (flow from left to right)
$global$	global error
i,j	indexes (e.g. of the generic node of the spatial discretization)
in	generic non-current-carrying solid component (jacket)

Symbol	Meaning
inl	inlet
$inner\ bench$	inner benchmark
$input$	input files
jk	non-current-carrying solid component (jacket)
l	left
$main$	main diagonal
mat	materials constituting the solid components
new	actual evaluated value
$nodal$	quantity evaluated in the nodes of the spatial discretization of the domain
old	previous evaluated value
out	outlet
$outer\ bench$	outer benchmark
p	pressure
r	right
ref	refined region or refined mesh
$scomp$	solid components (both current-carrying and non-current-carrying)
sd	spatial discretization
$smoot$	smoot transition from coarse to refined mesh (or from refined to coarse)
st	current-carrying solid component (strand)
sub	sub diagonal
sup	super diagonal
tc	thermal contact
tot	total number of equations to be solved after the spatial discretization
try	attempt value
uni	uniform mesh
v	velocity
ζ	generic time step

LIST OF ACRONYMS

Acronym	Meaning
0D	Zero-dimensional
1D	One-dimensional
2D	Two-dimensional
3D	Three-dimensional
3P-HTS	Three Phase Coaxial High Critical Temperature Superconducting
AC	Alternate Current
AM4	Adams Moulton method of fourth order
BC	Boundary Condition
BE	Backward Euler
CFD	Computational Fluid Dynamics
CFETR	China Fusion Engineering Test Reactor
CICC	Cable-In-Conduit Conductors
CN	Crank-Nicolson
CSV	Comma Separated Value
DC	Direct Current
DENERG	Dipartimento Energia
DTT	Divertor Tokamak Test
EPRI	Electric Power Research Institute
EPS	Encapsulated PostScript
EU-DEMO	European Demonstration Reactor
FD	Finite Differences
FDM	Finite Differences Method
FDTD	Finite Differences Time Domain
FEM	Finite Elements Method
GUI	Graphical User Interface
HELIAS	HELIcal Advanced Stellarator
HEP	High-Energy Physics
HTC	Heat Transfer Coefficient
HTS	High Critical Temperature Superconducting
IBVP	Initial Boundary Value Problem
ID	Identifier
ITER	International Thermonuclear Experimental Reactor
ITER-TF	ITER Low Critical Temperature Superconducting Toroidal Field Coil
JA-DEMO	Japanese Demonstration Reactor
JT-60SA	Japan Tokamak 60 Super Advanced
K-DEMO	Korean Demonstration Reactor
LTS	Low Critical Temperature Superconducting
MRI	Magnetic Resonance Imaging
NMR	Nuclear Magnetic Resonance
ODE	Ordinary Differential Equation
OOP	Object-Oriented Programming

Acronym	Meaning
PDE	Partial Differential Equation
PDEPE	Parabolic-Elliptic Partial Differential Equations
PPLP	Polypropylene Laminated Paper
SC	Superconducting
SHe	Supercritical Helium
SMES	Superconducting Magnetic Energy Storage
SPARC	Short Pulse Affordable Robust Compact
TFC	Toroidal Field Coils
TSV	Tab-Separated Value
UML	Unified Modeling Language
V&V	Verification and Validation
VE	Volume Element
VEM	Volume Element Model

CHAPTER 1

1 INTRODUCTION

Superconducting cables and magnets in the past decades have enabled fundamental discoveries in the field of high-energy physics [1], amazing steps forward in the research on a clean energy based on nuclear fusion [2] and a significant increase in the power transfer capability, as well as reduction of transmission loss and construction cost, for power cables [3]. Different kinds of SC cables are available, according to the applications, based on different SC materials and different concept for their cooling (conduction-cooling, coolant bath of forced-flow): in this thesis the focus is on forced-flow SC cables.

The background in which the present work is inserted is contextualized in the section 1.1 on the level of applications and in the sections 1.2 and 1.3 on that of the current tools available for the modeling. After outlining the objectives (see section 0), the structure of the thesis is summarized in section 1.5.

1.1 CONTEXT

The research on nuclear fusion as a possible technology for a CO_2-free power production, capable to reduce the many issues related to the nuclear fission plants [4], [5], is flourishing around the world, with magnetic confinement devices being designed or constructed, by both public enterprises and private companies, to address the physics and technological challenges that are still open. Despite the alternative configuration and design choices, all the large fusion machine recently entered in operation, under commissioning or design are superconductive.

On the side of the stellarator/heliotron configuration [6], the two world largest operating machines, namely Wendelstein 7-X [7]–[9] in Germany and the Large Helical Device [10], [11] in Japan, rely on superconducting coils employing low critical temperature superconducting material [12], cooled by Helium mainly in forced flow conditions. Future machines, such as that targeted by the public consortium EUROfusion (namely, the HELIcal Advanced Stellarator, so called HELIAS machine [13]) or by the private company Reinassance Fusion, will be designed taking advantage of the recent development in both LTS and high-critical temperature superconductors [14], respectively.

On the side of the tokamak configuration, while the ITER machine [15], [16] is under construction in France, the main parties collaborating for that huge nuclear fusion experiment are separately working on the next step toward a commercial use of fusion power [17]. The China Fusion Engineering Test Reactor (CFETR) [18], designed to bridge the fusion experiments between ITER and a nuclear fusion power station, addresses steady-state operation and tritium self-sustainment, the end of the conceptual design phase. Its design currently relies on both LTS and HTS cable-in-conduit conductors (CICC) [19]. The European DEMOnstration reactor (EU-DEMO), in the European roadmap to fusion [20], should go beyond ITER and shows for the first time that electricity can be generated from the fusion process. The EU-

1.1 CONTEXT

DEMO has just finished its pre-conceptual design phase, and its magnetic system is based on the use of CICC, with different variants still open including both LTS and hybrid LTS/HTS coils [21]. The Japanese DEMO (JA-DEMO) [22] is also designed as a superconducting machine, and, relying on the technical maturity of the LTS Nb_3Sn technology, it has adopted Nb_3Sn as the prime superconductor (SC) option, at least for the Toroidal Field Coils (TFC). The design of the magnet system of the Korean DEMO (K-DEMO) [23] is based on the use of well-established Nb_3Sn and NbTi CICC [24], while the American pilot power plant, ARC [25], bets on the development of fusion-class HTS magnets [26], with a demountable structure, that should be already adopted in the wingdings of the SPARC machine [27], coming first (and soon) in the American accelerated pathway to fusion energy. Even the "satellite" tokamaks, which should complement the physics advancement reachable through the deployment of ITER within a "broader approach" to fusion energy [28], such as the JT-60SA and the Italian Divertor Tokamak Test (DTT) facility are or will be fully superconductive. The JT-60SA, under commissioning in Japan [29], uses LTS CICC for all the coils [30], while for the DTT, HTS cables are being considered as an insert to the Central Solenoid [31].

The use of superconducting cables is not limited to the specific technology of nuclear fusion but covers other applications of strategic interest, such as the transport of power and the transportations. For these areas, technologies are based on HTS cables because they, having a high value of critical temperature can be cooled using less expensive fluids than He, which is mandatory for the proper functioning of LTS cables, thus reducing the cost [32]. In any case, the cost of the technology does not depend exclusively on the choice of the thermo-vector fluid and is nowadays comparable with that of Nb_3Sn cables as it is still immature [33]. A further advantage compared to LTS (in particular to Nb_3Sn) that makes them suitable for this applications, is the lower dependence of the critical current on strain in the compression region, behavior weakly influenced by small increases in temperature and magnetic field [34].

Compared to ordinary copper or aluminum cables, HTS cables have additional advantages that justify the renewed and growing interest in this technology. If on the one hand superconductivity allows to carry a power 3 to 5 times larger than the one transported by a conventional cable with the same or less losses [35], on the other hand this allows, with the same size, to carry a higher power minimizing the visual impact, land consumption and environmental footprint, to be associated also to the generation of less intense magnetic fields. In addition, they operate at lower voltages with benefits on structures that are less complex, smaller and therefore less expensive. Finally, the fault current limiting functionalities is embedded in the cables [36].

Two main families of HTS cables are nowadays available, known as first and second generations [37], [38], that differs from materials, manufacturing, characteristics and employs [39]. Another classification can be done according to the transported current, distinguishing between the alternate current (AC) and the direct current (DC). The formers are mainly used for connections in urban environments, the first real word application occurred in Georgia with the cables energization on January 05 2000 [40]. Another example is the permanent installation of the first long length transmission level voltage HTS power cable in the Long

Island Power Authority grid in 2007 [41]. AC HTS cables can also be employed to interconnect particularly energy intensive industrial applications by means of short length links [42].

DC cables are suitable for connections over long distances. Among the main applications of these cables, of considerable interest is the connection of isolated renewable energy sources with centrally located load centers [43], [44]. Thanks to their advantages over ordinary cables, DC HTS cables represent a possible solution to the problem of the growing demand for energy by metropolis (such as New York or Tokyo) which, at the same time, have saturated the underground metropolitan installations, overcoming in this way the grid bottlenecks [45]. The prospect of designing a compact and lightweight electric propulsion system, alongside the possibility of reduced installation costs, makes HTS DC cables attractive in the transportation sector as well [46], [47].

Figure 1.1-1 shows some typical examples of cables topology and configuration both of LTS and HTS type and for both fusion reactors and power transport applications.

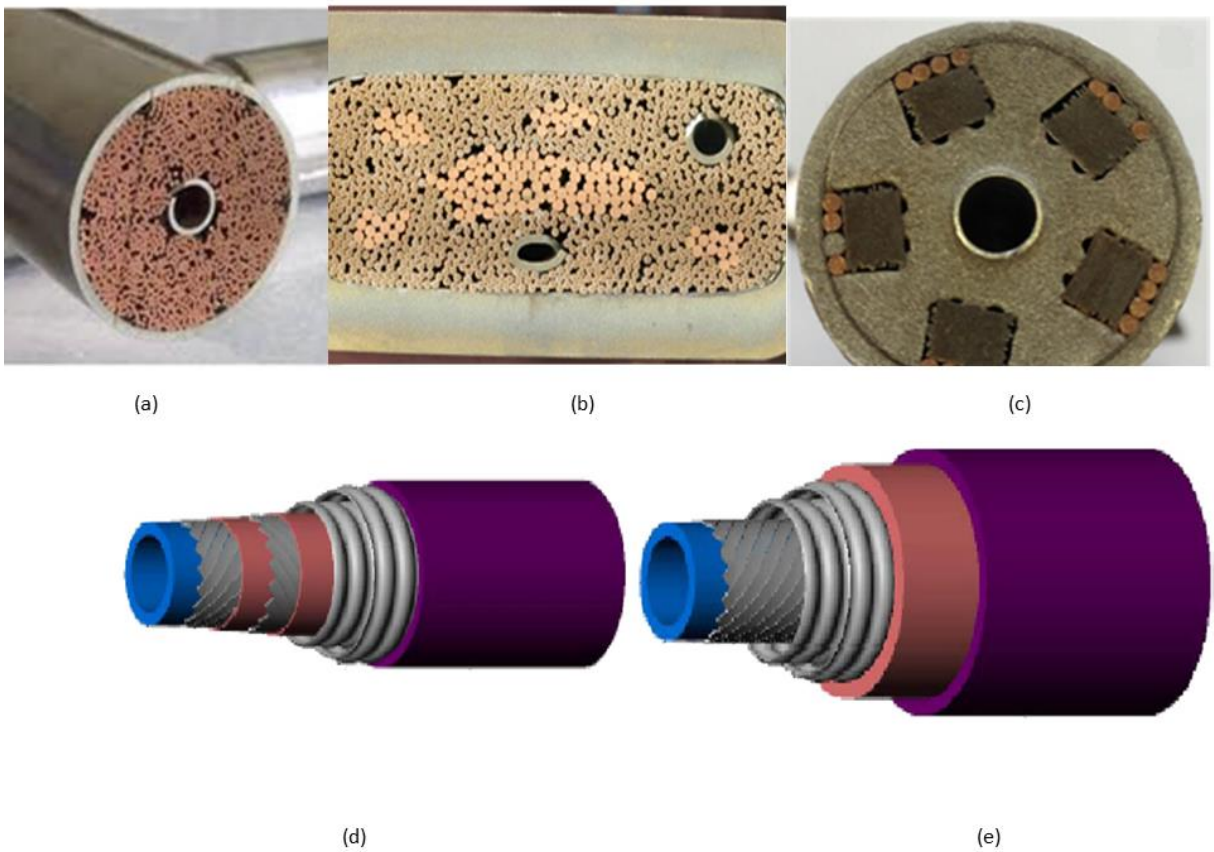


Figure 1.1-1 Examples of LTS and HTS cables for fusion and power transport applications: (a) ITER Toroidal Field coil LTS two region CICC [courtesy of ENEA]; (b) ENEA EU-DEMO LTS [courtesy of ENEA], (c) ENEA HTS [courtesy of ENEA]; (d) HTS cold dielectric AC [32]; (e) HTS warm dielectric DC [32].

1.2 OVERVIEW OF MODELING OF SC CABLES FOR FUSION APPLICATIONS

Being the use of forced-flow SC cables so relevant both for fusion applications and for power transmission, the availability of an appropriate, reliable and flexible modeling of the forced-flow SC cables is of paramount importance. In the field of fusion cables, several numerical tools are well established for the analysis of the transients in LTS cables, and namely the 4C code [48], the THEA/SUPERMAGNET Suite [49], and the VINCENTA/VENICIA suite [50].

Among the numerical tools mentioned above, the 4C code, which is proprietary and not available for commercial use, is largely the most validated among the ones quoted above [51]. It is able to perform steady and transient Thermal, Hydraulic and Electric analyses of forced-flow SC cables and magnets. It consists of a multi-conductor model [52] for the simulation of thermal-hydraulic transients in SC winding packs wound with ITER-like 2-channels CICCs or 1-channel CICCs, for which one-dimensional (1D) mass, momentum and energy conservation, in the non-conservative variables velocity pressure and temperature, are solved for the coolant (Supercritical He, SHe) in the different cooling channels, while transient heat diffusion is solved separately for the strands and the jacket. The possibility of a slow variation of coolant and solid cross sections along the cables was introduced several years ago in the code to account for the peculiar topology of joints [53]. A lumped-parameter model for the current distribution is embedded, that can account for the current repartition among the different conductive elements of the cable, but not for any current diffusion along the cable length. The heat conduction through bulky metal structures that embed the winding pack is also accounted for in the 4C code, but it becomes relevant only in presence of, e.g., the casing of a tokamak Toroidal Field Coils, bearing very high Lorentz forces. The cryogenic circuit that feed the magnets can also be modeled through a devoted library of cryogenic components, developed in the object-oriented language Modelica [54], [55]. The 4C model for the SC cables, developed in Fortran 77/Fortran 90, uses 1st order Finite Elements Method (FEM) for the spatial discretization with an adaptive grid, while an implicit (Backward Euler, BE) or semi-implicit scheme (Crank-Nicholson, CN) for the time marching, with an accuracy up to the 2nd order and the possibility to adapt the time stepping to capture steep variation in the cable transients. The code is based on a RUN-THEN-CHECK paradigm: the transient driver is pre-set in the simulation setup and cannot be modified during the code execution. The code allows to save the time evolution of selected variables (SHe velocity and pressure, solid and fluid temperatures, …) at selected locations, and the spatial profiles of the entire solution at given times during the transients; the post-processing is typically performed using a different software (MATLAB, Excel, …). The code has been originally developed for LTS cables, however a model for HTS cables (H4C, [56]) has been recently added to the 4C code family, which allows easily to model HTS macro-strands such as those shown in *Figure 1.1-1*, and includes a distributed-parameter model for the current distribution. However the H4C, which is currently under validation [31], is not suited for ITER-like LTS cables as it includes a simplified treatment for the thermal-hydraulic coupling between different cooling channels.

The commercial code THEA/SUPERMAGNET is able to perform steady and transient thermal, hydraulic and electric analyses of forced-flow SC cables. The model solves 1D mass, momentum and energy conservation for the coolant (SHe) in the non-conservative variables

velocity pressure and temperature, and energy conservation for the solid elements along each cable, and current diffusion and distribution along the cable, with distributed parameters for both thermal-hydraulic and electric models. An arbitrary number of thermal, hydraulic and electric components can be mutually coupled on the cable cross-section, with different possible materials forming the cable (SC, stabilizer, insulator, …) and their cross section can also vary along the cable to account for joints. An ancillary lumped model of the cryogenic circuit connected to the cable/magnet is also available. The model solver uses finite elements in space, with an adaptive grid, and an adaptive multi-step time marching scheme, with an accuracy up to the 3rd order. Although a post-processor is available within the tool, no interactive simulations are possible. The code structure is declared as "open", upon payment of the license fee, and in fact it has already been applied to the modeling/design of HTS magnets or inserts in the fusion field [57]. Note, however, that the applicability of models that cannot account for transversal gradients across the HTS strand cross section are controversial, see [58], and for that a different approach could be needed [56].

VINCENTA/VENECIA is another commercial package, aimed at the transient thermal-hydraulic simulation of large SC magnet system and accounting for several possible coolants simultaneously: helium, in the different states – superfluid, supercritical – 2-phase homogeneous mixture, but also nitrogen, hydrogen, oxygen, neon and water. The code is applicable to a wide range of devices including not only fusion devices, for which a validation of the code against experimental results is provided [59], but also magnet systems for NMR and MRI and superconducting motors, generators and SMES. It is based on a modular structure, with an individual set of algebraic, differential equations and equations in partial derivatives describing each component of the system (SC cables, pumps, valves and heat exchangers, and the like). Fluid flows are modeled using a 1D approximation, solving conservation laws in the non-conservative variables velocity, pressure and enthalpy, and they can be connected each-other and to two-dimensional (2D) models of the solid elements. The spatial discretization of the derivatives is performed through Finite Differences Method (FDM) with accuracy up to 5th order, while a semi-explicit splitting-up scheme for parabolic partial differential equations is implemented for the time marching. As also in 4C, when different conductors are modeled, each conductor can have a different meshing to better capture regions where the gradients of the drivers/solutions could be steep. Real-time monitoring of the results is included in the software, with also a Graphical User Interface (GUI) allowing for the selection of the task directory, checkout of input file and connection between cable elements, visualization of 2D mesh for the solids, launch of simulation, selection and plotting of results, but still within the Run-THEN-check paradigm. The programming language of VINCENTA/VENECIA is inferred from the website to be FORTRAN.

Although few benchmarks are available on couples of the above-mentioned tools [60], the different codes have never been applied to the same test case and rigorously benchmarked.

The main features of the codes are highlighted and compared in Table 1.2-1.

Table 1.2-1 Comparison of the main features of 4C, THEA/SUPERMAGNET and VINCENTA/VENICIA codes.

Features	4C	THEA/SUPERMAGNET	VINCENTA/VENICIA
Use	Proprietary	Need to pay a licence	Commercial
Program language	Fortran 77/90	Fortran	Fortran
Geometry	ITER-like CICCs up to 2 channels	Arbitrary	Wide range of geometries
Multi fluids	No (SHe)	No (SHe)	Yes (He, N2, H2, O2, Ne, H2O)
LTS	Yes	Yes	Yes
HTS	Yes	Yes	Yes
Spatial discretization scheme	1st order FEM	FEM	5th order FDM
Time integration scheme	Adaptive BE or CN	3rd order adaptive multi-step	semi-explicit splitting-up
GUI	No	No	Yes
Post processing	Performed with external tools	Inner post processor	Inner pre and post processor
Validation	Yes	Yes	Yes
Paradigm	Run then check	Run then check	Run then check

1.3 OVERVIEW OF MODELING OF SC CABLES FOR POWER APPLICATIONS

The landscape of modeling HTS cables for power transport appears very distant from that previously described in section 1.2 for LTS and HTS cables used for plasma magnetic confinement. This is due to the different designs and peculiarities of the latter category of cables, some of them are shown in Figure 1.3-1. In fact, the bibliographic search carried out has shown that currently there is no tool analogous to 4C, THEA/SUPOERMAGNET or VINCENTA/VENICIA that can be used for thermal-fluid dynamic and electromagnetic modeling of generic HTS cable for power applications. In the literature it is possible to find numerous examples of ad hoc modeling developed to solve specific problems, most of them in stationary condition, using commercial software or developing home-made algorithms, but a robust

comprehensive model in terms of thermal-hydraulic and electrical transient models is still missing.

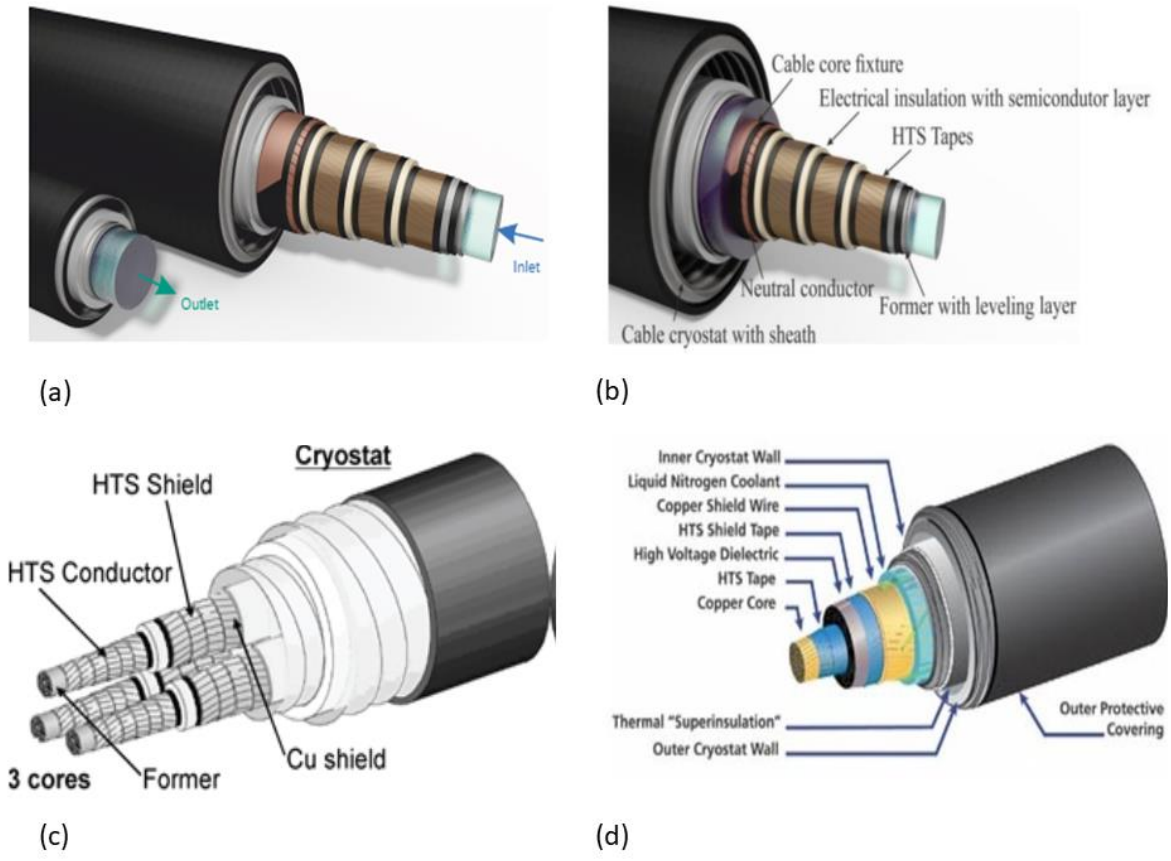


Figure 1.3-1 Collection HTS cables design for power transport both in single-phase and in three-phase configurations. (a) three-phase concentric cable with external return of liquid nitrogen [61]; (b) three-phase cable described in [62], (c) schematic of the three in-one cable concept from [63]; (d) design of a single-phase cable with single path for the coolant [courtesy of Nexans].

The first 1D model for compressible fluids involving the solution of Euler-like equations in nonconservative form dates back to the late 1970s [61]. The chronological evolution of mathematical and numerical models developed to simulate SC cables is contained in a section of [62]. Here an overview, though not exhaustive, of the models developed since 2010 to simulate the behavior of these cables is presented.

The Volume Element Model is an often-used discretization method, which consists in the decomposition of the domain into Volume Elements (VE). A model based on this formulation is proposed in [63] for HTS DC cables. The computational domain is divided into nine layers, each modeled with a different VE to which energy conservation applies. In the energy equations, terms that account for the heat transfer by conduction, convection and radiation are suitably considered, to keep into account different operational, environmental and design conditions. The proposed model is two-dimensional (2D) since the discretization affects both the axial and radial directions. Regarding the thermodynamic properties of the cooling fluid

(helium), they are calculated as a function of temperature and pressure, except for the thermal conductivity and viscosity which are considered constant. As far as the constitutive relations are of concern, the Dittus-Boelter correlation is adopted to evaluate the heat transfer coefficients. The model is applicable for both stationary and transient studies. In the former case, the system of equations is solved by the Newton-Raphson method, in the latter the time integration of the system of ordinary differential equations (ODEs) is performed according to a fourth order Runge-Kutta numerical scheme.

The model described above is applied in [64] and improved in [65] where the equations are integrated with a model for fluids and solids heat equations based on three dimensionless groups, namely time, temperature and pressure; then dimensionless variables such as specific heat, heat transfer coefficient, thermal conductivity, mass, mass flow rate, scale time and global heat transferhave been defined. The idea is also extended to the pumping power equation, introduced in [64], and to the heat generation equation due to current transmission, evaluated starting from the Ohm and Wiedemann-Franz laws.

VEM is also exploited in [66] to model the cable designed by the Electric Power Research Institute (EPRI) [67]. The paper is relevant since the system of differential equations resulting from the application of the model is solved using the flexible parabolic-elliptic partial differential equations (PDEPE) solver integrated in MATLAB [68]. The model keeps into account conductive, convective and radiative heat transfer; the only heat source comes from the surrounding environment in superconducting conditions.

However, when the transition to the normal state occurs in the conducting layer, inner power generation is considered.

The model is further developed by introducing coupling with current, for which a finite difference time-domain (FDTD) based analysis is used [69]. Specifically, the coupling of the thermal and electrical problems is at the level of the heat source term representing the Joule effect losses. A bidirectional coupling follows: on the one hand the Joule effect has an impact on the temperature of the superconductor, on the other hand the temperature influences its thermal resistivity characterized by a strongly nonlinear behavior.

In order to analyze the behavior in a failure situation of the 30 m HTS cable used in the Sumimoto Electric Industry experiment at Kumatori [70], a 1D finite difference (FD) based modeling is proposed in [71] in which the time-dependent heat equations in solids are discussed and the contributions of heat fluxes are evaluated by applying Fourier's law. The heat transfer coefficients are obtained using Dittus-Boelter correlation. At each time instant, the properties of the coolant (liquid nitrogen) are calculated using the GASPAK software package [72]. A zero-dimensional (0D) model for calculating the pressure drop of the cooling circuit is also proposed in the article.

Shabagin developed an alternative model to those considered so far for modeling three-phase AC HTS cables [73], which is tested with the geometry and operating parameters of the cable used in the AmpaCity project [74]. The calculation of electrical dissipations due to the use of AC is based on the elliptic Norris equation. The thermal model considers the radial and the axial direction. Along the radial coordinate the stationary heat equation in cylindrical

coordinates is adopted for solid layers; boundary conditions (BCs) take into account both convection at the surface and thermal contact between solid layers of different thermal conductivity. With this choice of equations and boundary conditions, the 2D and symmetrical temperature field on each conductor cross section is described. In the axial direction, the steady state 1D fluid temperature variation is deduced starting from the differential energy balance of the fluid flow in an infinitesimal volume of length dz. The roughness of the tube is considered in a corrected heat transfer coefficient, evaluated from correlations, and different equations are obtained for the supply and for the return flow. The 2D solid temperature radial distribution and the 1D axial coolant temperature distribution are linked by the wall temperatures; the result is a 3D steady state model. As far as the hydraulic model is considered, the pressure drops are evaluated with Darcy-Weisbach equation and friction factor comes from Karman-Nikuradse correlation. The solution algorithm is programmed in MATLAB.

The same model is adopted in [75] where the equations are discretized using the FD method, and in [76] where four different cooling options are compared. In the latter reference the roughness of the surface is neglected, thus the friction factor is evaluated from the Colebrook equation.

The three phase coaxial HTS cables described in [77] is modeled with a Finite Element Methods (FEM) that takes into account the heat transfer between the polypropylene laminated paper (PPLP) and the super conductor layers combined with analytical model to evaluate the temperature distribution when the coolant circulation occurs inside the cable. The equivalent thermal resistance is exploited to compute the overall thermal conductivity of the superconducting layer and the PPLP merged in only one equivalent solid. The same kind of cable is modeled with a 1D ODE network analysis in Sinda/Fluint commercial tool [78], after that the friction factor to evaluate the pressure drop is evaluated with Computational Fluid Dynamics (CFD) and compared with the available empirical correlations [79].

Within the European Project BEST PATHS [80] the Italian company RSE s.p.a. has developed a new mathematical model for the thermo-fluid dynamic simulation, able to describe the behavior of different cryogenic fluids in forced convection inside the cryostat of superconducting cables [81]. The model is 1D axial and derives from the manipulation of the equations of conservation of mass, momentum and energy. Many simplifying assumptions are considered: uniform cross section, constant material properties on the cross section, steady state regime, one-dimensional motion field, internal volumetric power generation is a function only of the axial coordinate, heat transfer by conduction between adjacent fluid volumes is neglected. The strength of this model is that it can be applied to all coolant of interest, since the thermophysical properties of these fluids are calculated as a function of temperature and density with the same analytical formulas, whose coefficients are a function of the chosen fluid. The final result is a set of two steady-state and nonlinear ordinary differential equations. The shear stress is computed with the Fanning friction factor calculated with Katheder correlation.

So far, the emphasis has been on dynamic thermo-fluid modeling of HTS power transport cables. A similar kaleidoscope of solutions exists for their electromagnetic modeling. A review of the evolution of these models is, for example, in [82].

In reference [83], the importance of modeling is emphasized in order to improve the design and performance of HTS cables. A simplified transient condition model, implemented through PSCAD/EMTDC software [84], is proposed for a multilayer coaxial HTS system with the addition of a copper former, used as a protection in case of overheating during a fault to bypass the transient current. The solution is obtained by exploiting the software libraries that however limit to eight the maximum number of layers available for cable discretization. Cables that require a description with a larger number of layers must be traced back to a simplified model with no more than eight layers. The presence of the copper former requires this simplification: the electrical resistances of the HTS cable are kept constant and equal to the maximum ones, which is a reasonable assumption during transient conditions due to the quench phenomenon, while those of the successive layers of copper vary to take into account the temperature increase due to the loss of the superconductive state of the cable.

Finally, methods for design optimization of second-generation HTS cables and applied to both single-phase and three-phase coaxial cables are proposed in [85]. The goal is to find the uniform current distribution between the conductor layers: the electrical circuit model of the cable, described as an electrical parallel of several branches, namely one for each layer of superconductive tapes, and a 3D finite element model are combined. The authors also show that manufacturing imprecision have non-negligible effects on the current distribution within the layers in multilayer HTS cables.

Hence the need to develop a reliable and robust software, including both thermo-fluid dynamic and electromagnetic models, which allows to use the most suitable coolant and able to model any geometric and topological configuration of HTS cables for power transport.

1.4 AIM AND NOVELTY OF THE WORK

The aim of the thesis is to develop the embryo of a new software dedicated to the modeling of superconducting cables at the mesoscale level, focusing on thermal-fluid dynamic aspects.

The design of the tool builds on the experience gained over twenty years of developing the codes already mentioned, in particular the solutions adopted by 4C, while introducing innovative ideas.

In the first place, it undertakes the road of the open source [86], taking advantage of a diffusion of the code and a wide use that allows to test several configurations, putting in evidence not only the virtues but above all the problems still not resolved or identified.

The second paradigm shift is the use of Object-Oriented Programming (OOP), which allows high flexibility in modeling different cables designs and topologies. In addition, the implemented material libraries ensure that the code is natively designed to model both LTS

and HTS conductors. Because of this versatility, the code is suitable to model the cables crafted for both fusion and power transmission, the main areas of interest of the technology.

The new concept of the Run-AND-check, opposed to the Run-THEN-check, is introduced which allows to keep under control the outcome of the simulation in real time and change the driver while running. User can interact with the simulation thanks to a user-friendly GUI.

Last but not least, an auxiliary tool is developed, also equipped with a graphical interface, to perform some advanced post processing of the data, above all the benchmarks for the validation.

The code is validated against 4C by considering two case studies characterized by different topologies and different operating temperatures. The first one is a power transport conductor of the HTS type [77], the second one is the cable used for the realization of toroidal magnets in ITER and belongs to the LTS category [87]. All these analyses are carried out with the developed auxiliary tool, which is in turn tested.

1.5 STRUCTURE OF THE THESIS

The mathematical model for mesoscale modeling of superconducting cables from a thermo-fluid dynamic perspective is described in depth in chapter 2, along with the methods adopted for numerical discretization in space and time. The equations are first presented in their general form and then detailed for the two case studies considered.

Chapter 3 is devoted entirely to the description of the code. There, the object-oriented approach used is detailed, highlighting the interactions and the class hierarchy, some of which are extensively discussed to underline the different strategies implemented with respect to 4C code. The idea behind Run-AND-check and the potential of the GUI is also shown. Finally, the data analysis performed implicitly by the code and the further analyses enabled by the auxiliary tool are discussed.

Analysis of the results of simulations performed for spatial and time convergence studies of the code, validation against 4C, and numerous inner benchmarks are the subject of chapter 4.

Conclusions and prospects for the future can be found in chapter 5.

CHAPTER 2

2 MESOSCALE MODELLING OF SUPERCONDUCTORS

The modeling of superconductors magnet is a multi-physical and multi-scale problem. To deal with the different time and spatial scales, the complex structure of the cable and the tricky interaction between the cable constituents, three different modeling level scales are adopted: *macroscale*, *mesoscale* and *microscale*. The first and the latter concern respectively the whole set of winding packs (i.e. a magnet) [88]–[90] and each cable basic components, such as extremely detailed regions of the channel to evaluate friction factors and/or heat transfer coefficients [91]–[93]. The mesoscale is at conductor (or at most at winding pack) level and it is considered in this work. Indeed, this approach is suitable both for the conductors used to make the magnetic field in fusion applications and to the ones used to transport power.

This chapter is organized as follows: the first section deals with the mathematical model used to describe the cables according to the mesoscale modeling, that will lead to a not linear system of partial differential equations (PDEs) in time and space, whose space and time numerical discretization are proposed in the second section.

2.1 MATHEMATICAL MODEL

The main aim of this section is to describe the mathematical model used to simulate the behavior of the superconducting cables, both from the coolant and solid structures point of view. A single conductor is considered in the following discussion. The developed tool (SC2 code) is thought to be extremely flexible and capable to model several thermodynamics configurations, so a quite general form of the equations is presented here.

The first assumption, of the set of hypotheses applied for that mathematical model, is that the cables can be modeled with a one-dimensional model since their length or longitudinal dimension is from three up to more than five orders of magnitude larger than the transverse ones (height and width respectively), therefore a single value is representative of the properties throughout the cross section. Cables are composed of two macro regions, namely the fluid and the solid one that are mathematically described separately. In general, a conductor can be described with N_{ch} channels, N_{st} *current-carrying solids* also called strands and N_{jk} jackets, global notation for *non-current-carrying solids*. It is assumed that the coolant behaves as a compressible fluid and that each channel is described by its own set of independent Euler-like equations that can be coupled by means of suitable source terms. Each solid component is described with a 1D time dependent heat transfer equation and their boundaries are considered adiabatic. Conductor cross section is assumed to be constant along its length and the potential contribute is neglected in the definition of the coolant specific energy.

2.1.1 FLUID COMPONENTS EQUATIONS

The generic channel is modeled by a set of three hyperbolic equations that stand for the conservation of mass, momentum and energy; their general conservative form is proposed in the following equation:

$$\begin{cases} \dfrac{\partial \rho}{\partial t} + \dfrac{\partial \rho v}{\partial x} = \Lambda_\rho \\ \dfrac{\partial \rho v}{\partial t} + \dfrac{\partial \rho v^2}{\partial x} + \dfrac{\partial p}{\partial x} = \Lambda_v \\ \dfrac{\partial \rho e}{\partial t} + \dfrac{\partial \rho e v}{\partial x} + \dfrac{\partial p v}{\partial x} = \Lambda_e \end{cases} \qquad (2.1\text{-}1)$$

Solving this system allows to directly obtain the fluid density, velocity and specific energy. However, its solution requires, at each time step, an iterative algorithm since some terms exhibit an implicit dependence on pressure and temperature, that are not the outcomes of the solution. Consequently, computational cost and execution time increase. Pressure gradient, heat transfer and conductive terms, for example, are expressed in an implicit form of these quantities and they cannot be neglected in the study of the transient. For this reason, a new form of the system was deduced [94] expressing the conservation equations as function of the non-conservative set of variables velocity, pressure and temperature, resulting in the corresponding equations written below:

$$\begin{cases} \dfrac{\partial v}{\partial t} + v\dfrac{\partial v}{\partial x} + \dfrac{1}{\rho}\dfrac{\partial p}{\partial x} = \dfrac{1}{\rho}\left(\Lambda_v - v\Lambda_\rho\right) \\ \dfrac{\partial p}{\partial t} + \rho c^2 \dfrac{\partial v}{\partial x} + v\dfrac{\partial p}{\partial x} = \Phi\left[\Lambda_e - v\Lambda_v - \left(w - \dfrac{v^2}{2} - \dfrac{c^2}{\Phi}\right)\Lambda_\rho\right] \\ \dfrac{\partial T}{\partial t} + \Phi T\dfrac{\partial v}{\partial x} + v\dfrac{\partial T}{\partial x} = \dfrac{1}{\rho c_v}\left[\Lambda_e - v\Lambda_v - \left(w - \dfrac{v^2}{2} - \Phi c_v T\right)\Lambda_\rho\right] \end{cases} \qquad (2.1\text{-}2)$$

For the physical meaning of the symbols refer to the list of symbols; some characteristics of this set of equations will be put in evidence. It is a set of first order partial differential equations in time and space; since the only spatial coordinate is the x that is the coordinate along the cable length, the equation is 1D in space. The first equation is called the *velocity equation*, the second the *pressure equation* and the third the *temperature equation*, they are coupled since in the left-hand side in all the equations the velocity appears, while the pressure is common to the first two equations; other coupling terms can be found in the right-hand side which collects the source terms. This set of PDEs is also not linear and the non-linearity is due to the $v\frac{\partial}{\partial x}$ addendum in all the equations (the typical advective term), as well as to the coefficients, since the thermophysical properties (ρ, c and Φ) are function of both pressure and temperature and these functions itself can be not linear. Moreover, the product $T\frac{\partial v}{\partial x}$ is of the same kind of the advective one. Further examples of non-linearity are visible in the right-hand side (one for all the $\frac{v^2}{2}$ in the energy transfer trinomials).

As already said, the right-hand side groups all the source terms which are constructed combining the Λ whose definition can be found in appendix A and will be recalled later in this discussion; they respectively represents the mass source (Λ_ρ), the momentum source (Λ_v) and the energy source (Λ_e). The momentum source definition involves friction factor while heat transfer coefficients are necessary to compute the energy one. These so-called *transport coefficients* are evaluated with some constitutive relations that are obtained experimentally or numerically exploiting microscale CFD simulation campaigns.

To practically solve each set of equations the initial condition and closure relations should be provided; the first allows a proper initialization of the numerical problem while for each channel different closure sets of BCs are allowed, provided the inlet temperature is assigned, together with one inlet and one outlet condition either on pressure or velocity.

Before addressing the solid equations, it is essential to explain the relevance of the hypothesis that each channel is modeled with its own independent set of Euler-like equations.

If the channels are isolated, i.e. there is no transfer of mass, momentum and energy among the channels, or if the channels are at the most in thermal contact (only energy exchange), velocity, pressure and temperature of each channel are weakly influenced by the corresponding variables of the others, so to each channel must correspond a set of non-conservative variables (v, p and T). This is true also in the case in which the channels are in hydraulic parallel, that means that there is a fraction of their contact perimeter that is open and through which occurs transfer not only of energy, but also of mass and momentum. Due to possible different hydraulic characteristics, the fluid velocity in the channels may be different even though they are subject to the same pressure drop, so at least channels velocities must be different in the set of equations that describe them. As far as pressures and temperatures are concerned, to understand why also these two variables need to be specific of the channel, the different time scales that characterize some phenomena came into play. There may be situations in which, the time scale in which the information of the change of pressure and temperature in the channel is propagated, is of a different magnitude from that in which it is transmitted from one channel to another. Not taking this phenomenon into account can lead to completely incorrect or non-physical results. A typical example is quench. The need for different sets of pressure and temperature values for different channels has been well established in literature [95], [96].

2.1.2 SOLID COMPONENTS EQUATIONS

Solid components are modeled with the cartesian transient 1D heat equation.

Although the left-hand side of the equation is the same for strands and jacket, the right-hand side differs in the coupling terms with both jackets and strands respectively, thus both the general equations for the generic strand fi and jacket in are indicated below:

2.1 MATHEMATICAL MODEL

$$\Sigma_{fi}\rho_{fi}c_{p,fi}\frac{\partial T_{fi}}{\partial t} - \Sigma_{fi}\frac{\partial}{\partial x}\left(k_{fi}\frac{\partial T_{fi}}{\partial x}\right) = \sum_{st=1}^{N_{ch}} P_{fi,st}h_{fi,st}\left(T_{st} - T_{fi}\right) + \sum_{st\neq fi}^{N_{st}} P_{fi,st}h_{fi,st}\left(T_{st} - T_{fi}\right) + \sum_{jk=1}^{N_{jk}} P_{fi,jk}h_{fi,jk}\left(T_{jk} - T_{fi}\right) + Q_{fi,Joule} + Q_{fi,ext} \qquad (2.1\text{-}3)$$

$$\Sigma_{in}\rho_{in}c_{p,in}\frac{\partial T_{in}}{\partial t} - \Sigma_{in}\frac{\partial}{\partial x}\left(k_{in}\frac{\partial T_{in}}{\partial x}\right) = \sum_{ch=1}^{N_{ch}} P_{in,ch}h_{in,ch}\left(T_{ch} - T_{in}\right) + \sum_{st=1}^{N_{st}} P_{in,st}h_{in,st}\left(T_{st} - T_{in}\right) + \sum_{jk\neq in}^{N_{jk}} P_{in,jk}h_{in,jk}\left(T_{jk} - T_{in}\right) + Q_{in,Joule} + Q_{in,ext} \qquad (2.1\text{-}4)$$

Cross section, density, specific heat and thermal conductivity are computed keeping into account that both strands and jackets are, in general, multi materials components. For instance, strands can be made of superconducting filaments in a matrix of stabilizer, typically copper, while jackets can be made by stainless steel and glass-epoxy insulating. In appendix C the problem is addressed with more detail.

The main features of these equations are their parabolic nature, the inherent non-linearity ascribed to the products $\rho c\frac{\partial T}{\partial t}$ and $k\frac{\partial T}{\partial x}$ since thermophysical properties are function of temperature (themselves are generally not linear) and the coupling terms with other solids and/or channels that build up part of the right-hand side, together with the drivers. To solve each solid component equation an initial condition should be given, besides one inlet and one outlet boundary conditions need to be applied. Thanks to the above hypothesis the homogeneous Neumann (adiabatic) BCs can be easily used.

Before deeper analyze the global system of equations, it is worthily to take a closer look to the drivers.

As can be seen from the above equations, there are two kind of heating source: the power generated by the Joule effect (Q_{Joule}) and the external heating (Q_{ext}). The former is due to the loss of superconductivity in strands or to the fact that some fraction of the current may be transported by some non-current-carrying solids in off-normal operating conditions. The latter is linear power externally introduced in the strands and it is the relevant driver for a couple of reasons: in the first place while studying superconducting cable transient they are always induced by an external heating of solid components (either strands or jackets), secondly in the present model the current module is not available yet, so the only possibility to induce a transient is by means of the external heating.

2.1.3 COUPLING OF EQUATIONS

So far, the equations that models mathematically fluid and solid components were described separately. Since in the right-hand side of equations just reported there are coupling terms between fluid and solid components, they cannot be solved individually; in other words, all the above-mentioned equations must be solved as a single set of PDEs. The number of equations making up the system increases proportionally to the detail with which the conductor is discretized into its basic components, according to the following law:

$$N_{eq} = 3N_{ch} + N_{st} + N_{jk} \tag{2.1-5}$$

There are six possibilities for the coupling that are managed by means of a coupling matrix:

1. channel-channel.
2. channel-strand.
3. channel-jacket.
4. strand-strand.
5. strand-jacket (or jacket-strand).
6. jacket-jacket.

To clarify the first three possibilities, the general expression of the source terms (Λ_ρ, Λ_v and Λ_e) for the ca channel should be recalled from appendix A, where further information can be found.

$$\Lambda_\rho^{ca} = \frac{\sum_{ch\neq ca}^{N_{ch}} K'_{ca,ch}\,(p_{ch} - p_{ca})}{L\Sigma_{ca}} \tag{2.1-6}$$

$$\Lambda_v^{ca} = \frac{\sum_{ch\neq ca}^{N_{ch}} K''_{ca,ch}\,(p_{ch} - p_{ca})}{L\Sigma_{ca}} - \rho_{ca}F_{ca} \tag{2.1-7}$$

$$\begin{aligned}\Lambda_e^{ca} = {} & \frac{\sum_{ch\neq ca}^{N_{ch}} K'''_{ca,ch}\,(p_{ch} - p_{ca})}{L\Sigma_{ca}} + \frac{\sum_{ch\neq ca}^{N_{ch}} \left(P^o_{ca,ch}h^o_{ca,ch} + P^c_{ca,ch}h^c_{ca,ch}\right)(T_{ch} - T_{ca})}{\Sigma_{ca}} \\ & + \frac{\sum_{st=1}^{N_{st}} P_{ca,st}h_{ca,st}\,(T_{st} - T_{ca})}{\Sigma_{ca}} + \frac{\sum_{jk=1}^{N_{jk}} P_{ca,jk}h_{ca,jk}\left(T_{jk} - T_{ca}\right)}{\Sigma_{ca}}\end{aligned} \tag{2.1-8}$$

The first addendum of each source term is evaluated with the open contact perimeter between the channels, hidden in the terms $K'_{ca,ch}$, $K''_{ca,ch}$ and $K'''_{ca,ch}$ as can be seen in appendix A. Two possibilities arise, the first is that the channels in contact are in hydraulic parallel which means that their open perimeter fraction is larger than zero, so their equations are fully coupled; the second considers that there is only thermal contact between channels. In this last case, the open fraction of the contact perimeter is zero, thus only the energy source couples the channels thanks to the addendum:

$$\frac{\sum_{ch\neq ca}^{N_{ch}} P^C_{ca,ch}h^C_{ca,ch}\,(T_{ch} - T_{ca})}{\Sigma_{ca}} \tag{2.1-9}$$

The last two contributions to Λ_e account also for the coupling between channel-strand and channel-jacket if the contact perimeters are not null; these are analogous to the first addendum of the right-hand side in equations (2.1-3) and (2.1-4).

There are three kinds of coupling among solids, two homogeneous (strand-strand and jacket-jacket) and one heterogeneous or hybrid (strand-jacket or jacket-strand), that appears in both the general equations for strands and jackets. The homogeneous kind is specific of the equation, thus the coupling between strands is described in the strand equation, analogously coupling between jackets can be found in the jacket equation. In any case, the three contributions to the right-hand sides will actually have an impact only if the contact perimeter is larger than zero, otherwise they count for nothing.

2.1.4 THE FINAL SET OF PDES

Once defined how the equations are coupled, it is convenient to rewrite them in a matrix form that allows to handle them more easily. To this purpose some notation should be introduced: $\boldsymbol{u}$ and $\boldsymbol{s}$ are respectively the unknowns and source vectors while M, A, K and S are the square matrices of coefficients. Their general definition can be found in appendix B.1. The elements of the last matrix came from the right-hand side of the PDEs since there are terms that includes the unknowns, so $\boldsymbol{s}$ is built only by the Joule and external linear power sources of solid components. The matrix form of the set is:

$$M\frac{\partial \boldsymbol{u}}{\partial t} + A\frac{\partial \boldsymbol{u}}{\partial x} + \frac{\partial}{\partial x}\left(K\frac{\partial \boldsymbol{u}}{\partial x}\right) + S\boldsymbol{u} = \boldsymbol{s} \qquad \textit{(2.1-10)}$$

This set cannot be classified according to the typical PDE classification since it combines both the hyperbolic and parabolic features.

What follows is a paragraph that shows what form takes the set of equations in the two case studies considered in this thesis. The corresponding matrices for their matrix formulation can be found in appendix B.2 and B.3.

2.1.4.1 CASE STUDY EQUATIONS

Chapter 4 is devoted to the analysis of the simulation results obtained with the SC2 code for two different conductors, the three-phase coaxial HTS cable, also called 3P-HTS, designed by Lee [77] and the ITER LTS Toroidal Field Coil (shortly ITER-TF) cable [87]. Here, to put in practice what discussed above, the extended, full set of equations is provided for both the configurations.

2.1.4.1.1 3P-HTS CONFIGURATION

The first conductor configuration considered in this work is the three-phase coaxial HTS superconductor whose topology is depicted in *Figure 2.1-1*. The inner cylinder is made by a multi material structure composed by the copper former surrounded by several concentric layers of HTS and insulating material made by PPLP; the coolant flows through the hollow region between this structure and the cryostat. From the mesoscale analysis point of view, this cable can be discretized using three basic components, one fluid and two solids; indeed, the multi-

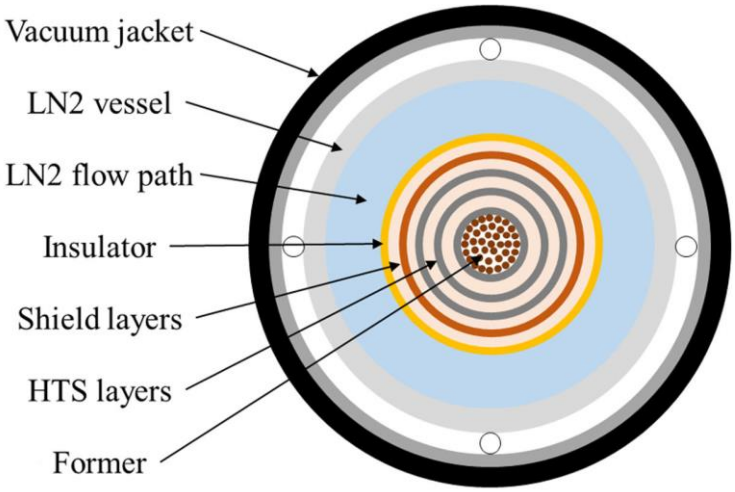


Figure 2.1-1 Cross section of the three phase coaxial HTS cable [79].

material region can be thought as a single current-carrying solid (subscript $ST1$), while the cryostat belongs to the non-current-carrying category, indicated as $JK1$. The cable is cooled with liquid helium and the channel subscript is $CH1$. The total number of equations is given by (2.1-5) and in this case the system is made by only five equations.

The coupling between the components is deduced from *Figure 2.1-1*: coolant flows between the jacket and the strand and there is only thermal contact; there is no contact among the solid components.

On the basis of the previous considerations, the following system of equations can be obtained:

$$\begin{cases} \dfrac{\partial v_{CH1}}{\partial t} + v_{CH1}\dfrac{\partial v_{CH1}}{\partial x} + \dfrac{1}{\rho_{CH1}}\dfrac{\partial p_{CH1}}{\partial x} = \dfrac{\Lambda_v^{CH1}}{\rho_{CH1}} \\ \dfrac{\partial p_{CH1}}{\partial t} + \rho_{CH1}c_{CH1}^2\dfrac{\partial v_{CH1}}{\partial x} + v_{CH1}\dfrac{\partial p_{CH1}}{\partial x} = \Phi_{CH1}[\Lambda_e^{CH1} - v_{CH1}\Lambda_v^{CH1}] \\ \dfrac{\partial T_{CH1}}{\partial t} + \Phi_{CH1}T_{CH1}\dfrac{\partial v_{CH1}}{\partial x} + v_{CH1}\dfrac{\partial T_{CH1}}{\partial x} = \dfrac{1}{\rho_{CH1}c_{v,CH1}}[\Lambda_e^{CH1} - v_{CH1}\Lambda_v^{CH1}] \\ \Sigma_{ST1}\rho_{ST1}c_{ST1}\dfrac{\partial T_{ST1}}{\partial t} - \Sigma_{ST1}\dfrac{\partial}{\partial x}\left(k_{ST1}\dfrac{\partial T_{ST1}}{\partial x}\right) = P_{CH1,ST1}h_{CH1,ST1}(T_{CH1} - T_{ST1}) + Q_{ST1,ext} \\ \Sigma_{JK1}\rho_{JK1}c_{JK1}\dfrac{\partial T_{JK1}}{\partial t} - \Sigma_{JK1}\dfrac{\partial}{\partial x}\left(k_{JK1}\dfrac{\partial T_{JK1}}{\partial x}\right) = P_{CH1,JK1}h_{CH1,JK1}\left(T_{CH1} - T_{JK1}\right) + Q_{JK1,ext} \end{cases} \quad (2.1\text{-}11)$$

Being:

$$\Lambda_\rho^{CH1} = 0 \quad (2.1\text{-}12)$$

$$\Lambda_v^{CH1} = -\rho_{CH1}F_{CH1} \quad (2.1\text{-}13)$$

$$\Lambda_e^{CH1} = \frac{P_{CH1,ST1} h_{CH1,ST1} (T_{ST1} - T_{CH1})}{\Sigma_{CH1}} + \frac{P_{CH1,JK1} h_{CH1,JK1} (T_{JK1} - T_{CH1})}{\Sigma_{CH1}} \quad (2.1\text{-}14)$$

The source terms of the 3P-HTS case study do not have the coupling addendum between channels since only one channel is considered. Moreover, it is taken into account that the current cannot be modeled considering only the external heating as driver in the solid component equations.

2.1.4.1.2 ITER LTS TOROIDAL FIELD COIL CONFIGURATION

ITER-TF topology is shown in the following Figure 2.1-2. It is the typical design of a two regions cable-in-conduit conductor made by a circular jacket, six petals of superconducting strands wrapped around a central helical spiral that delimits the hole, the central channel, from the bundle, the annular region of interstices in petals through which the coolant flows. The simplest discretization of this cable dictates the use of two fluid components that respectively models the hole (subscript $CH1$) and the bundle (subscript $CH2$) and two solid components, one for the strand ($ST1$) to model all the six petals and one for the jacket ($JK1$).

According to equation (2.1-5) the total number of equations in this configuration is eight, six for the fluid components and one for each solid component.

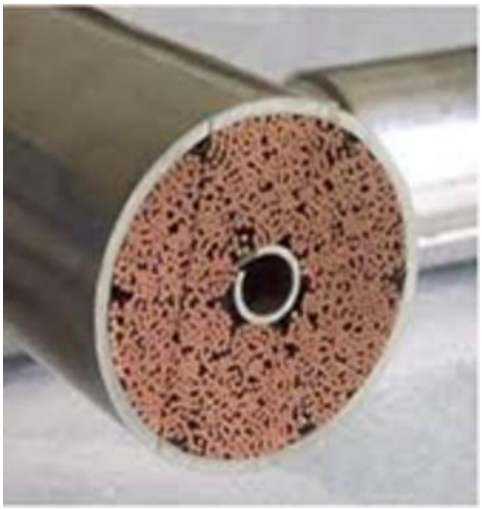


Figure 2.1-2 ITER LTS Toroidal Field Coil configuration [courtesy of ENEA]

As far as couplings are of concern, from the Figure 2.1-2 it can be seen that hole and bundle are in hydraulic parallel thanks to the helical spiral, the bundle is also in contact with both the solid components, finally there is thermal contact between strand and jacket. This configuration is more complicated than the 3P-HTS one. Since the current is not modeled yet by the code, only the external heating power is considered as driver in the solid component equations.

The full system of PDEs is written as follows:

$$
\begin{cases}
\dfrac{\partial v_{CH1}}{\partial t} + v_{CH1}\dfrac{\partial v_{CH1}}{\partial x} + \dfrac{1}{\rho_{CH1}}\dfrac{\partial p_{CH1}}{\partial x} = \dfrac{1}{\rho_{CH1}}\left(\Lambda_v^{CH1} - v_{CH1}\Lambda_\rho^{CH1}\right) \\
\dfrac{\partial v_{CH2}}{\partial t} + v_{CH2}\dfrac{\partial v_{CH2}}{\partial x} + \dfrac{1}{\rho_{CH2}}\dfrac{\partial p_{CH2}}{\partial x} = \dfrac{1}{\rho_{CH2}}\left(\Lambda_v^{CH2} - v_{CH2}\Lambda_\rho^{CH2}\right) \\
\dfrac{\partial p_{CH1}}{\partial t} + \rho_{CH1}c_{CH1}^2\dfrac{\partial v_{CH1}}{\partial x} + v_{CH1}\dfrac{\partial p_{CH1}}{\partial x} = \Phi_{CH1}\left[\Lambda_e^{CH1} - v_{CH1}\Lambda_v^{CH1} - \left(w_{CH1} - \dfrac{v_{CH1}^2}{2} - \dfrac{c_{CH1}^2}{\Phi_{CH1}}\right)\Lambda_\rho^{CH1}\right] \\
\dfrac{\partial p_{CH2}}{\partial t} + \rho_{CH2}c_{CH2}^2\dfrac{\partial v_{CH2}}{\partial x} + v_{CH2}\dfrac{\partial p_{CH2}}{\partial x} = \Phi_{CH2}\left[\Lambda_e^{CH2} - v_{CH2}\Lambda_v^{CH2} - \left(w_{CH2} - \dfrac{v_{CH2}^2}{2} - \dfrac{c_{CH2}^2}{\Phi_{CH2}}\right)\Lambda_\rho^{CH2}\right] \\
\dfrac{\partial T_{CH1}}{\partial t} + \Phi_{CH1}T_{CH1}\dfrac{\partial v_{CH1}}{\partial x} + v_{CH1}\dfrac{\partial T_{CH1}}{\partial x} = \dfrac{1}{\rho_{CH1}c_{v,CH1}}\left[\Lambda_e^{CH1} - v_{CH1}\Lambda_v^{CH1} - \left(w_{CH1} - \dfrac{v_{CH1}^2}{2} - \Phi_{CH1}c_{v,CH1}T_{CH1}\right)\Lambda_\rho^{CH1}\right] \\
\dfrac{\partial T_{CH2}}{\partial t} + \Phi_{CH2}T_{CH2}\dfrac{\partial v_{CH2}}{\partial x} + v_{CH2}\dfrac{\partial T_{CH2}}{\partial x} = \dfrac{1}{\rho_{CH2}c_{v,CH2}}\left[\Lambda_e^{CH2} - v_{CH2}\Lambda_v^{CH2} - \left(w_{CH2} - \dfrac{v_{CH2}^2}{2} - \Phi_{CH2}c_{v,CH2}T_{CH2}\right)\Lambda_\rho^{CH2}\right] \\
\Sigma_{ST1}\rho_{ST1}c_{ST1}\dfrac{\partial T_{ST1}}{\partial t} - \Sigma_{ST1}\dfrac{\partial}{\partial x}\left(k_{ST1}\dfrac{\partial T_{ST1}}{\partial x}\right) = P_{CH2,ST1}h_{CH2,ST1}(T_{CH2} - T_{ST1}) + P_{ST1,JK1}h_{ST1,JK1}(T_{JK1} - T_{ST1}) + Q_{ST1,ext} \\
\Sigma_{JK1}\rho_{JK1}c_{JK1}\dfrac{\partial T_{JK1}}{\partial t} - \Sigma_{JK1}\dfrac{\partial}{\partial x}\left(k_{JK1}\dfrac{\partial T_{JK1}}{\partial x}\right) = P_{CH2,JK1}h_{CH2,JK1}(T_{CH2} - T_{JK1}) + P_{ST1,JK1}h_{ST1,JK1}(T_{ST1} - T_{JK1}) + Q_{JK1,ext}
\end{cases}
\tag{2.1-15}
$$

Where the source terms for hole and bundle assume these forms:

$$\Lambda_\rho^{CH1} = \frac{K'_{CH1,CH2}(p_{CH2} - p_{CH1})}{L\Sigma_{CH1}} \tag{2.1-16}$$

$$\Lambda_v^{CH1} = \frac{K''_{CH1,CH2}(p_{CH2} - p_{CH1})}{L\Sigma_{CH1}} - \rho_{CH1}F_{CH1} \tag{2.1-17}$$

$$\Lambda_e^{CH1} = \frac{K'''_{CH1,CH2}(p_{CH2} - p_{CH1})}{L\Sigma_{CH1}} + \frac{\left(P^o_{CH1,CH2}h^o_{CH1,CH2} + P^c_{CH1,CH2}h^c_{CH1,CH2}\right)(T_{CH2} - T_{CH1})}{\Sigma_{CH1}} \tag{2.1-18}$$

$$\Lambda_\rho^{CH2} = \frac{K'_{CH1,CH2}(p_{CH1} - p_{CH2})}{L\Sigma_{CH2}} \tag{2.1-19}$$

$$\Lambda_v^{CH2} = \frac{K''_{CH1,CH2}(p_{CH1} - p_{CH2})}{L\Sigma_{CH2}} - \rho_{CH2}F_{CH2} \tag{2.1-20}$$

$$\Lambda_e^{CH2} = \frac{K'''_{CH1,CH2}(p_{CH1} - p_{CH2})}{L\Sigma_{CH2}} + \frac{\left(P^o_{CH1,CH2}h^o_{CH1,CH2} + P^c_{CH1,CH2}h^c_{CH1,CH2}\right)(T_{CH1} - T_{CH2})}{\Sigma_{CH2}} + \frac{P_{CH2,ST1}h_{CH2,ST1}(T_{ST1} - T_{CH2})}{\Sigma_{CH2}} + \frac{P_{CH2,JK1}h_{CH2,JK1}\left(T_{JK1} - T_{CH2}\right)}{\Sigma_{CH2}} \tag{2.1-21}$$

2.1.5 SUMMARY

The mathematical model for the mesoscale analysis of superconductors has been discussed in this section. The most important hypotheses of the model are the 1D approximation and the fact that each channel is described by its own independent set of Euler-like PDEs for inviscid compressible fluids (equation (2.1-2)), eventually coupled with the other channels. Solid components are treated with the cartesian 1D transient heat transfer equation considering that boundaries are adiabatic. Conductor cross section does not change along the

conductor length and the potential energy term is neglected in the coolant specific internal energy evaluation.

The total number of equations is given by equation (2.1-5), their general form is shown together with a detailed explanation of their main features. A focus on the drivers, which belong to the right-hand side of solid components equations, is provided.

Particular attention is given to the coupling of the equations practically managed with a coupling matrix and ruled by the contact perimeter between channels, channels and solid and between solids.

The general system of partial differential equations can be more easily handled when it is rewritten in its matrix form (equation (2.1-10)). The section ends with two examples of sets of equations whose solution will be deeply developed in chapter 4.

Next section explains how to solve numerically the mathematical model, to be implemented in an object-oriented fashion

2.2 NUMERICAL SCHEMES

Equation (2.1-10) derived in the previous section is the matrix form of a coupled not linear system of PDEs which is first order in time and second order in space. Having acknowledged the impossibility of analytically solving the system in question, the solution of the problem is approached numerically. As far as the spatial discretization is concerned, the choice falls on the Finite Elements Methods, already adopted in the 4C and Supermagnet Code (for Gandalf). The time discretization is performed with an implicit method (such as Backward Euler or Crank-Nicolson) due to the stiffness of the problem; indeed, implicit methods in general benefit of a better stability properties compared to the explicit ones.

The adopted procedure is known as the Galerkin method (actually it is a class of methods) to solve the initial boundary value problem (IBVP) and it consist of a first discretization in space, that reduces the complexity of the problem converting it from a system of PDEs to a system of ordinary differential equations, and a subsequently integration in time.

The remainder of this section shed the light on the topic providing guidelines of the selected methods for the solution.

FEM to solve PDEs is described in [97], a short description of the Galerkin methods can be found in [98] while [99] has a chapter devoted to the solution of ODEs.

2.2.1 SPATIAL DISCRETIZATION WITH FINITE ELEMENTS METHOD (FEM)

The cable length (computational domain $[0, L]$) is discretized with $N + 2$ nodal points $0 = x_0 < x_1 \dots < x_N < x_{N+1} = L$. This partition results into $N + 1$ subintervals $I_j = \left(x_{j-1}, x_j\right)$ of length $\Delta x_j = x_j - x_{j-1}\ \forall\ j = 1, \dots N + 1$.

The *finite-dimensional subspace* V_h is defined as the set of functions $\omega = \omega(x)$ such that $\omega \in \mathrm{V_h}$: $\omega \in \mathrm{C}^1\ \forall\ I_j$ and $\omega(0) = \omega(\mathrm{L}) = 0$. To construct these functions, the values at the

nodal points can be exploited $\bar{\omega}_i = \omega(x_j)$, together with the *basis functions* of V_h, the so-called hat functions, that are defined in this way:

$$\psi_j \in V_h, j = 1, \dots, N: \tag{2.2-1}$$

$$\psi_j(x_i) = \begin{cases} 1 \; if \; i = j \\ 0 \; if \; i \neq j \; i, j = 1, \dots, N \end{cases} \tag{2.2-2}$$

The generic function $\omega \in V_h$ can be expressed in a unique way as a linear combination of the above defined basis functions, i.e.:

$$\omega(x) = \sum_{i=1}^{N} \bar{\omega}_i \psi_{\,i}(x), \forall x \in [0, L] \tag{2.2-3}$$

Going back to the equation (2.1-10), the next step in the spatial discretization with FEM is its reformulations in the weak form according to a suitable matrix Ω of test functions $\omega \in V_h$:

$$\int_0^L \Omega M \frac{\partial \boldsymbol{u}}{\partial t} dx + \int_0^L \Omega A \frac{\partial \boldsymbol{u}}{\partial x} dx + \int_0^L \Omega \frac{\partial}{\partial x}\left(K \frac{\partial \boldsymbol{u}}{\partial x}\right) dx + \int_0^L \Omega S \boldsymbol{u} dx = \int_0^L \Omega \boldsymbol{s} \, dx \tag{2.2-4}$$

that can be rewritten in a more manageable way applying the Green theorem to the conduction integral and remembering that $\omega(0) = \omega(L) = 0$:

$$\int_0^L \Omega \frac{\partial}{\partial x}\left(K \frac{\partial \boldsymbol{u}}{\partial x}\right) dx = -\int_0^L \frac{d\Omega}{dx} * K \frac{\partial \boldsymbol{u}}{\partial x} dx \tag{2.2-5}$$

This will allow to construct an approximation of $\boldsymbol{u}$ as a linear combination of the hat functions $\psi_j \in V_h$ that are only piecewise C^1:

$$\int_0^L \Omega M \frac{\partial \boldsymbol{u}}{\partial t} dx + \int_0^L \Omega A \frac{\partial \boldsymbol{u}}{\partial x} dx - \int_0^L \frac{d\Omega}{dx} * K \frac{\partial \boldsymbol{u}}{\partial x} dx + \int_0^L \Omega S \boldsymbol{u} dx = \int_0^L \Omega \boldsymbol{s} \, dx \tag{2.2-6}$$

It must be premised that, to deal with numerical oscillations when the dominant terms are the advective rather than the conductive ones, some elements on the K matrix are modified adding an artificial diffusion as prescribed by the *upwind* method.

The approximation $\tilde{\boldsymbol{u}}$ of the exact solution ($\boldsymbol{u}$) takes the following form:

$$\tilde{\boldsymbol{u}}(x, t) = \sum_{j=1}^{N} \Psi_j \, \bar{\boldsymbol{u}}_j(t), \forall x \in [0, L] \tag{2.2-7}$$

if $\bar{\mathbf{u}}_j$ is the vector of the values of the nodal approximation of $\boldsymbol{u}$.

Choosing $\omega = \psi_i$, or in other words $\Omega = \Psi_i$, and introducing the above decomposition in equation (2.2-6) the following system is obtained:

$$\int_0^L \Psi_i M \sum_{j=1}^{N} \Psi_j \frac{d\,\overline{\boldsymbol{u}}_j}{dt} dx + \int_0^L \Psi_i A \sum_{j=1}^{N} \frac{d\Psi_j}{dx}\,\overline{\boldsymbol{u}}_j\, dx - \int_0^L \frac{d\Psi_i}{dx} * K \sum_{j=1}^{N} \frac{d\Psi_j}{dx}\,\overline{\boldsymbol{u}}_j\, dx + \int_0^L \Psi_i S \sum_{j=1}^{N} \Psi_j\,\overline{\boldsymbol{u}}_j\, dx = \int_0^L \Psi_i \boldsymbol{s}\, dx \;\; \forall i = 1, \dots, N \qquad (2.2\text{-}8)$$

As can be seen in appendix B.1 the matrices M, A, K and S are square matrices of dimension $N_{tot,eq}$x$N_{tot,eq}$ while the above equation is a system of N equations. Each equation of the system is itself a small system of $N_{tot,eq}$ equations, so to each nodal point are associated $N_{tot,eq}$ unknowns; in other words the whole dimension of the above system is given by:

$$N_{tot} = N_{eq}(N+2) \qquad (2.2\text{-}9)$$

since to the nodes corresponding to $i = 0$ and $i = N + 1$ are related N_{eq} closure equations (boundary conditions).

Equation (2.2-8) can be written in a new matrix form:

$$M_{sd} \frac{d\boldsymbol{u_{sd}}}{dt} + (A_{sd} + K_{sd} + S_{sd})\boldsymbol{u_{sd}} = \boldsymbol{s_{sd}} \qquad (2.2\text{-}10)$$

That is a system of N_{tot} ordinary differential equations in time to which suitable initial conditions must be applied.

2.2.1.1 MESH CONSTRUCTION

The above dissertation is general because it does not refer to a specific kind of mesh generation. Indeed, there are several possibilities and this section is devoted to illustrate the ones implemented in the SC2 code, that is equipped with both uniform and not uniform meshes. Recall that the computational domain $[0, L]$ is partitioned in $N + 1$ sub-intervals $I_j = (x_{j-1}, x_j)$ of length $\Delta x_j = x_j - x_{j-1}\ \forall\, j = 1, \dots, N+1$; by definition $\Delta x = \max(\Delta x_j)$ is a measure of how fine is the partition.

The *uniform mesh* is obtained imposing that the partition is uniform in the computational domain, in other words, it means that the spatial discretization parameter is constant and its value is given by:

$$\Delta x_j = \frac{L}{N+1} \;\forall\, j = 1, \dots N+1 \qquad (2.2\text{-}11)$$

In this case the above defined measure coincides with the generic Δx_j so for simplicity the spatial discretization parameter is called Δx. The uniform mesh is the easiest alternative and it has the virtue of being inexpensive from a computational point of view, but it is convenient only for those problems where the drivers are smooth or does not change sharply in space. To deal with these other situations more sophisticated meshes should be adopted.

The non-uniform mesh is a class of meshes that are characterized by one or more regions with a finer discretization parameter and others with coarser ones. They are suitable to

discretize the space when the location of the gradients of the drivers are well known. The extension of the refined regions is tuned to cover these steep variations while the smoother ones are discretized with a larger discretization parameter. This larger flexibility with respect to the uniform mesh is balanced by a not negligible computational cost for its generation. SC2 code is equipped with the *single region non-uniform mesh* characterized by only one refined zone; it is generated guaranteeing that the number of nodal points is kept constant and equal to $N + 2$.

The refined region $\left[x_{ref,a}, x_{ref,b}\right]$ with $x_{ref,a} \in [0, L)$ and $x_{ref,b} \in (0, L]$ of length:

$$L_{ref} = x_{ref,b} - x_{ref,a} \tag{2.2-12}$$

is uniformly discretized with $N_{ref} < N + 2$ nodal points, so the refined spatial discretization paranmeter is given by:

$$\Delta x_{ref} = \frac{L_{ref}}{N_{ref} - 1} \tag{2.2-13}$$

The remaining $N_{coarse} = N + 2 - N_{ref}$ nodes are used to build the two coarse regions on the left and on the right of the refined one, and they are distributed according to their length by means of a weighted average:

$$N_{coarse,l} = \frac{x_{ref,a} - 0}{L - L_{ref}} N_{coarse} \tag{2.2-14}$$

$$N_{coarse,r} = \frac{L - x_{ref,b}}{L - L_{ref}} N_{coarse} \tag{2.2-15}$$

The transition from the left coarse region to the refined one and from the refined region to the right coarse region should not be too sharp to avoid numerical instability, therefore to gradually decreasing/increasing the discretization parameter respectively a smoothing coefficient δ is introduced and exploited into two distinct while loops, both defined with the following condition:

$$\frac{\Delta x_{try}}{\Delta x_{smooth,new}} > \delta \tag{2.2-16}$$

The numerator of the inequality (2.2-16) is the value of the uniform discretization parameter obtained if the remaining length of the coarse region is uniformly discretized with the nodes still available. If after i_l iterations the left coarse region is the interval $[0, x_l]$ with $x_l \leq x_{ref,a}$ and the remaining nodes are $N_{coarse,l} - i_l$, results:

$$\Delta x_{try,l} = \frac{x_l - 0}{N_{coarse,l} - i_l - 1} \tag{2.2-17}$$

while the denominator is defined as:

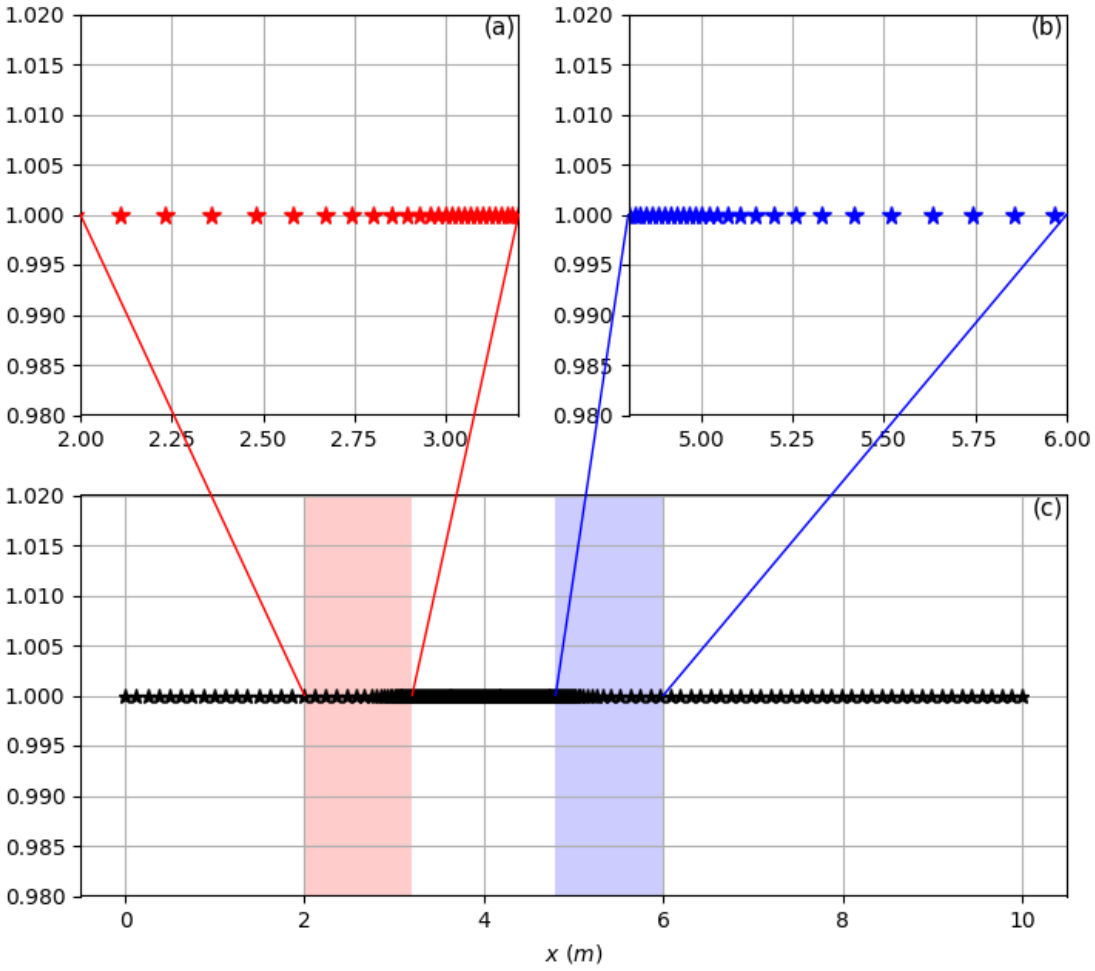


Figure 2.2-1 Not uniform, asymmetric mesh in a computational domain of $10\ m$ *with one refined region in* $[3{,}5]\ m$ *and two coarse regions. The total number of elements is 200, the refined region is discretized using 120 elements, while* $\delta = 1.2$*. (a) zoom of the smooth transition from coarse left to refined region; (b) zoom of the smooth transition from refined to the right coarse region; (c) whole mesh. Y axes is meaningless.*

$$\Delta x_{smooth,new,l} = \Delta x_{smooth,old,l}\delta = \Delta x_{ref}\delta^{i_l} \tag{2.2-18}$$

If equation (2.2-16) is verified $\Delta x_{smooth,new,l}$ is used as discretization parameter and a new iteration takes place setting:

$$x_l = x_l - \Delta x_{smooth,new,l} \tag{2.2-19}$$

Analogously if the right coarse region corresponds to the interval $[x_r, L]$ with $x_r \geq x_{ref,b}$ after i_r iterations the remaining nodes are $N_{coarse,r} - i_r$, thus:

$$\Delta x_{try,r} = \frac{L - x_r}{N_{coarse,r} - i_r - 1} \tag{2.2-20}$$

$$\Delta x_{smooth,new,r} = \Delta x_{smooth,old,r}\delta = \Delta x_{ref}\delta^{i_r} \tag{2.2-21}$$

A new iteration is needed if the ratio is larger than the smoothing coefficient and the new value of the coordinates is computed exploit $\Delta x_{smooth,new,r}$ as discretization parameter:

$$x_r = x_r + \Delta x_{smooth,new,r} \tag{2.2-22}$$

When the above inequality is not verified anymore, the remaining fractions of the coarse regions are uniformly discretized with the residual nodes. In general, the refined zone is not centered in the domain resulting with different lengths of the coarse regions leading to $i_l \neq i_r$ and, finally, $\Delta x_l \neq \Delta x_r$.

The borderline cases of a refined region that starts form one of the edge of the domain are kept into account by the algorithm; in these cases either $x_{ref,a} = 0$ or $x_{ref,b} = L$ and only one coarse region should be discretized with all the N_{coarse} nodes.

An example of single refined region non-uniform mesh is shown in Figure 2.2-1.

2.2.2 TIME DISCRETIZATION

The set of equations (2.2-10) results from the application of the FEM and therefore it is coupled, moreover the total number of equations is proportional to both the number of basic components adopted to model the conductor and to the number of nodes used to discretize the conductor length. The only reasonable strategy to solve this problem is to apply a numerical method to integrate the ODEs, that is the second step of the Galerkin procedure.

Since the matrix $(A_{sd} + K_{sd} + S_{sd})$ may be bad conditioned the problem should be treated as stiff and explicit methods (as Forward Euler) are not suitable in this case because they may require an excessively small time-step to solve it accurately. Based on the previous considerations, implicit methods are the only alternative: Backward Euler and Crank-Nicolson are considered, which can be seen as special cases of the more general θ*-method*, introduced below, with $\theta \in [0,1]$.

Let denote with the superscript ζ the variables at time t and with $\zeta + 1$ the variables at time $t + \Delta t$; specifically, $\boldsymbol{u}_{\boldsymbol{sd}}^{\zeta}$ and $\boldsymbol{u}_{\boldsymbol{sd}}^{\zeta+1}$ are respectively an approximation of $\boldsymbol{u}_{\boldsymbol{sd}}$ at time t and $t + \Delta t$. The θ-method yields:

$$M_{sd}^{\zeta+1}\frac{\boldsymbol{u}_{\boldsymbol{sd}}^{\zeta+1} - \boldsymbol{u}_{\boldsymbol{sd}}^{\zeta}}{\Delta t} + (1-\theta)\left(A_{sd}^{\zeta} + K_{sd}^{\zeta} + S_{sd}^{\zeta}\right)\boldsymbol{u}_{\boldsymbol{sd}}^{\zeta} + \theta\left(A_{sd}^{\zeta+1} + K_{sd}^{\zeta+1} + S_{sd}^{\zeta+1}\right)\boldsymbol{u}_{\boldsymbol{sd}}^{\zeta+1} = (1-\theta)\boldsymbol{s}_{\boldsymbol{sd}}^{\zeta} + \theta\boldsymbol{s}_{\boldsymbol{sd}}^{\zeta+1} \tag{2.2-23}$$

Reshaping the above equation returns:

$$\left[\frac{M_{sd}^{\zeta+1}}{\Delta t} + \theta\left(A_{sd}^{\zeta+1} + K_{sd}^{\zeta+1} + S_{sd}^{\zeta+1}\right)\right]\boldsymbol{u}_{\boldsymbol{sd}}^{\zeta+1} = \left[\frac{M_{sd}^{\zeta+1}}{\Delta t} - (1-\theta)\left(A_{sd}^{\zeta} + K_{sd}^{\zeta} + S_{sd}^{\zeta}\right)\right]\boldsymbol{u}_{\boldsymbol{sd}}^{\zeta} + (1-\theta)\boldsymbol{s}_{\boldsymbol{sd}}^{\zeta} + \theta\boldsymbol{s}_{\boldsymbol{sd}}^{\zeta+1} \tag{2.2-24}$$

For $\theta = 1$ it is the Backward Euler method while to $\theta = \frac{1}{2}$ corresponds the Crank-Nicolson numerical scheme.

In the equation (2.2-24) information at both time steps ζ and $\zeta + 1$ are required. As far as the external sources vector approximation $\boldsymbol{s_{sd}}$ is considered, there is no problem since they are well known and can be explicitly evaluated at each new time step. The same cannot be said for the matrices since their elements depend on the solution at the new time step because the problem is not linear. In order to avoid iterations at each time step to get the solution, that are not avoidable if bisection or Newton methods are applied to solve the not linear system of equations, a linearization of the system is necessary exploiting the *frozen coefficients* concept. This means that the solution obtained at the previous time step is used to evaluate the coefficients of the matrices, at the new time step; consequently, the solution of the system is more accurate the smaller the adopted time step. In practice it is assumed that:

$$\left[\frac{M_{sd}^{\zeta+1}}{\Delta t} + \theta\left(A_{sd}^{\zeta+1} + K_{sd}^{\zeta+1} + S_{sd}^{\zeta+1}\right)\right]\boldsymbol{u}_{\boldsymbol{sd}}^{\zeta+1} \approx \left[\frac{M_{sd}^{\zeta}}{\Delta t} + \theta\left(A_{sd}^{\zeta} + K_{sd}^{\zeta} + S_{sd}^{\zeta}\right)\right]\boldsymbol{u}_{\boldsymbol{sd}}^{\zeta+1} \tag{2.2-25}$$

$$\left[\frac{M_{sd}^{\zeta+1}}{\Delta t} - (1-\theta)\left(A_{sd}^{\zeta} + K_{sd}^{\zeta} + S_{sd}^{\zeta}\right)\right]\boldsymbol{u}_{\boldsymbol{sd}}^{\zeta} \approx \left[\frac{M_{sd}^{\zeta}}{\Delta t} - (1-\theta)\left(A_{sd}^{\zeta} + K_{sd}^{\zeta} + S_{sd}^{\zeta}\right)\right]\boldsymbol{u}_{\boldsymbol{sd}}^{\zeta} \tag{2.2-26}$$

The linearized system to be solved is:

$$\left[\frac{M_{sd}^{\zeta}}{\Delta t} + \theta\left(A_{sd}^{\zeta} + K_{sd}^{\zeta} + S_{sd}^{\zeta}\right)\right]\boldsymbol{u}_{\boldsymbol{sd}}^{\zeta+1} = \left[\frac{M_{sd}^{\zeta}}{\Delta t} - (1-\theta)\left(A_{sd}^{\zeta} + K_{sd}^{\zeta} + S_{sd}^{\zeta}\right)\right]\boldsymbol{u}_{\boldsymbol{sd}}^{\zeta} + (1-\theta)\boldsymbol{s}_{\boldsymbol{sd}}^{\zeta} + \theta\boldsymbol{s}_{\boldsymbol{sd}}^{\zeta+1} \tag{2.2-27}$$

and it can be rewritten in the compact form:

$$A_{sys}\boldsymbol{u}_{\boldsymbol{sd}}^{\zeta+1} = \boldsymbol{b} \tag{2.2-28}$$

Being:

$$A_{sys} = \left[\frac{M_{sd}^{\zeta}}{\Delta t} + \theta\left(A_{sd}^{\zeta} + K_{sd}^{\zeta} + S_{sd}^{\zeta}\right)\right] \tag{2.2-29}$$

$$\boldsymbol{b} = \left[\frac{M_{sd}^{\zeta}}{\Delta t} - (1-\theta)\left(A_{sd}^{\zeta} + K_{sd}^{\zeta} + S_{sd}^{\zeta}\right)\right]\boldsymbol{u}_{\boldsymbol{sd}}^{\zeta} + (1-\theta)\boldsymbol{s}_{\boldsymbol{sd}}^{\zeta} + \theta\boldsymbol{s}_{\boldsymbol{sd}}^{\zeta+1} \tag{2.2-30}$$

The θ-method is a class of single step methods, in the sense that to compute the numerical solution at the time step $\zeta + 1$ only the information at the previous time step (ζ) is required. Its order of convergence is at most 2 for Crank-Nicolson (1 for Backward Euler).

Regardless the numerical scheme adopted to numerically integrate the system of ODEs (2.2-28), the outcome is a linear system of algebraic equations characterized by a banded matrix A_{sys}. It is solved by applying a band solver that is computationally cheaper than method of *elimination of Gauss*, being the number of operations lower or equal to $c\ N_{tot}$ with c an integer number evaluated as a function of the bandwidth of matrix A_{sys}.

2.2.3 SUMMARY

The mesoscale mathematical modelling of the cable yields a not linear coupled system of parabolic partial differential equations that, that is solved numerically. The Galerkin method is applied to the IBVP that prescribed first a discretization in space leading to a system of ODEs that are subsequently numerically integrated.

The spatial discretization is performed according to the FEM recipe that would lead to a global convergence in space between the first and the order, due to the upwind terms. Mesh construction can be done with both uniform and not uniform with single refined region strategies, that are suitable for different kinds of drivers: the former is generally cheap and performs well if there are smooth or slowly changing drivers, the latter is developed to deal with a priori known spatial gradients and its computational cost may be not negligible.

Two possibilities are foreseen to the numerical integration providing solutions that have different order of convergence in time, namely the single step BE and CN methods are implemented resulting in first and second order numerical schemes respectively, both particular cases of the θ-method.

To deal with the not linearities the frozen coefficients concept is applied whose accuracy increases as the time step decreases.

The banded system of linear algebraic equations that is obtained from the application of the Galerkin procedure is solved with an algorithm for generic band matrix without pivoting.

CHAPTER 3

3 CODE DESCRIPTION

As mentioned in the introductory chapter 1, this work is to be considered as the preliminary step for the development of a new software for mesoscale modelling of superconducting cables. The previous chapter 2 focused on the numerical methods adopted for the spatial and temporal discretizations used to solve the problem. In this chapter the focus is on the structure and organization of the software with the aim of highlighting the main features and potential, which include:

- an object-oriented design exploiting class;
- a flexible topology since the object-oriented design allows to build the conductor as a combination of an arbitrary number of current-carrying (strands) and non-current-carrying (jackets) solid components, that can be characterized by different materials, and fluid channels;
- a Graphical User Interface to run the simulation and that implements the Run-AND-check philosophy;
- an automatic basic post processing of the simulation results;
- an external tool for the advanced post processing of the outcomes.

Section 3.1 provides an overview of the code architecture and the input files and the management of the output ones, specifically an in-depth look at class design that build up the conductor is offered in section 3.2. The chapter proceeds whit section 0 by analyzing the classes used to manage the different phases of the simulation, with emphasis on initialization and the application of boundary conditions. Section 3.4 discusses the main features of the graphical interface and the implementation of the real-time check of the simulation results. Finally, the intrinsic processing of the results is shown, as well as the possibilities offered by the external tool for more refined analyzes (0).

3.1 THE SC2 CODE ORGANIZATION IN NUTSHELL

The content of this section is twofold, the first subsection provides general description of the code architecture while the second focuses on the input and output organization.

3.1.1 SC2 ARCHITECTURE

The SC2 code is designed to model forced flow superconducting cables, alternatively called conductors, which in turn are composed by several basic components, in an object-oriented fashion. Although in the current code state only one conductor is addressed by the model, the code architecture allows easily to extend the modeling to several conductors simultaneously. It is completely written in Python 3 [100] and it relies on few libraries:

- *os* that allows to interact with the operating system;

- *warning* to prompt warning messages to the user;
- *numpy* [101] and *scipy* [102] that are heavily used to deal with numerical schemes;
- *pandas* [103] is exploited to manage the output files as data frames and to manipulate them easily;
- *openpyxl* [104] that allows to deal with the input files that are provided in the form of Excel spreadsheets;
- *matplotlib* [105] is a powerful library to make plots;
- *tkinter* is chosen to build the graphical user interface.

Usually, extended and complex Python codes are organized into several modules, each one accomplishing a specific task; the SC2 code makes no exception.

All SC2 modules are collected into the root directory SCMagnetCode and can be distinguished among primary and secondary (or ancillary) modules. Being an Object-Oriented Program, ten of the eleven primary modules deal with *class*, used to construct *objects*.

Before going on, it is worthily to define what are class and object in Python. Define a class is a way to bound data and functionality together to create data-structure; in other words, it is the blueprint for how something should be defined according to the user. When a class is defined, it is important to keep in mind that it does not contain any kind of data, moreover it creates a new type of object so that new instances of that type can be made, that is objects are constructed from class. Each class instance can have *attributes* attached to it for maintaining its state and can also have *methods* (the functions defined inside the class) that allow to modify its state. As a final remark, it should be reminded that Python class supports *inheritance*, a feature that is exploited in the code.

Once the differences among class and object are clarified, a short description of the primary modules used in SC2 code is provided, highlighting the relationship between classes. They are listed in Table 3.1-1 that summarizes the nomenclature adopted in this thesis for the classes and their objects. To avoid ambiguity, in the remainder of this work a color scheme is introduced to distinguish classes from objects, attributes from methods and methods functions:

- class names are written in purple;
- object names in green;
- class methods in blue;
- class attributes in dark blue;
- functions are marked in dark red;
- finally, dictionaries kesy as in **bold.**

To further improve the clarity of the exposition, a font code is also introduced to distinguish between files such as the modules or the input files and folders. The formers are written in Arial, the latter are in Cambria.

Table 3.1-1 List of the main modules that build up the SC2 code, together with the adopted nomenclature for the classes and objects. Modules and classes chare the same name. Notice that the Simulation_starter module is not associated to a class, and therefore to any objects.

Module name	Class name	Object name
Simulation_starter		
SC2_GUI	SC2_GUI	gui
Simulations	Simulations	simulation (rarely sim)
Conductors	Conductors	CONDUCTOR
FluidComponents	FluidComponents	CHAN (rarely fluidcomponent)
SolidComponents	SolidComponents	solidcomponent
Jacket	Jacket	Z_JACKET
Strands	Strands	strand
MixSCStabilizer	MixSCStabilizer	STR_MIX
SuperConductor	SuperConductor	STR_SC
Stabilizer	Stabilizer	STR_STAB

The Simulations_starter.py module makes an instance of the class SC2_GUI that generates both the *root* and the *main* windows of the GUI; in particular, the *mainloop* of the main window rules the whole code. While choosing the directories that groups the input files, user also makes an instance of the class Simulations, defined in module Simulations.py. This class will be further investigated in a next section, for the time being it is enough to know that it allows to make instances of class Conductors and that its methods allow to execute all the simulation steps; its objects store global information about the simulation. The class Simulations has the same name of the module in which it is defined. This is not by chance since this naming schemes avoids ambiguity and simplify the code architecture.

Going forward with the top-down approach, the next module to be discussed is Conductors.py that defines the homonym class Conductors. This class is a container of the five classes that are used to build the cable. One of the most important method of a class is the inner __init__ method that is automatically called when a class is instanced to create an object. When this is done for the class Conductors, the conductor object type is build invoking its inner constructor method __init__. Among the numerous actions performed by this method, probably the most important is the instantiation of the classes that correspond to cable basic components like FluidComponents, Jacket, MixSCStabilizer, SuperConductor and Stabilizer. These objects are stored in suitable Python *dictionaries* as *lists* that are attributes of the conductor object, so they are always available, together with both their attributes and methods, when the conductor object is instantiated. Indeed, since magnets can be composed of several cables, class Simulations allows to instantiate more than one conductor object at time and they are stored in a Python list as well; therefore it is not necessary to instantiate it whenever a conductor object is required: it will be selected from the list_of_Conductors attribute of class Simulations and it will bring all the instantiated basic components object that actually build the cable. Furthermore, Conductors methods allow to initialize the cables, get its topology and the kind of interfaces between each component and evaluate both thermophysical and electromagnetic properties of materials, together with the transport

properties such as friction factor for fluid components and heat transfer coefficients between fluid components, fluid and solid components and between solid components.

The above discussion concerns three of the ten classes used in the code. It was said that of the remaining seven, five are devoted to the cable basic components, thus two remain undefined, namely classes SolidComponents and Strands. At this point, it is worthily that the reader is aware of how these seven classes interacts and are organized; to this purpose, class hierarchy is represented in Figure 3.1-1: it is based on the different typology of materials and their role in the actual cable. The first main subdivision is among FluidComponents and SolidComponents; the former class is dedicated to model the coolant and its instances define the fluidcomponent python objects (more often called CHAN as it will be clear later on); the latter class deals with the solid components that constitutes the cable.

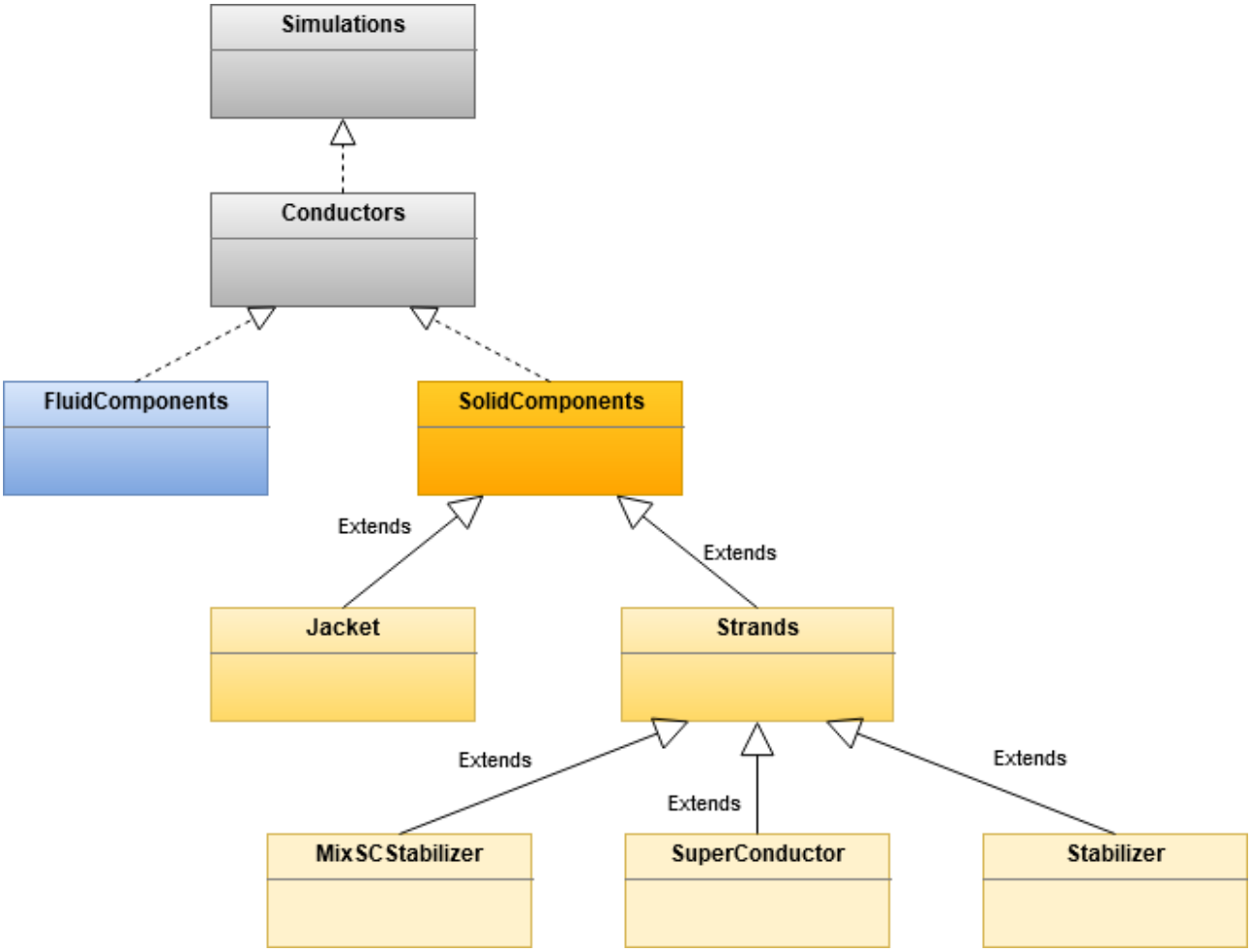


Figure 3.1-1 UML diagram of the relationships between the classes.

Since solid materials in a cable can be roughly distinguished into jacket and strand, exploiting the powerful concept of Python class inheritance, the classes Jacket and Strands are both build as child class of the class SolidComponents. This is useful since all the attributes and methods that are defined in the parent class (SolidComponents in this case) are inherited by the child classes (Jacket and Strands) so that in the child classes are defined only attributes

and methods that are not in common. As far as Jacket class is concerned, its Z_JACKET objects correspond to a basic conductor component, as well as fluidcomponent objects. To extend the SC2 flexibility, three (instead of only one) kinds of strand objects can be instantiated, each descending from a specific child class of the parent class Strands and constituting the third and last kind of basic component of the cable. Again, all the common features will be defined in the parent class Strands, while specific attributes and methods are detailed in the child class. The kinds of basic strands that can be defined are:

1. STR_MIX object that allows to model a strand that is made of both superconductor and stabilizer materials;
2. STR_SC object to introduce pure superconducting strand;
3. STR_STAB object to which correspond a stabilizing strand.

There are no restrictions on the use of this basic components, i.e. the user can adopt a single kind of basic strand or implement and combine the three of them to better discretize the cable topology, according to the level of detail required by the analysis.

Secondary modules are grouped into two subfolders called respectively Properties_of_materials and UtilityFunctions. The former, as its name says, is the subfolder that includes all the materials properties. On the one hand, for each fluid material both a module with functions to evaluate thermophysical properties and a subfolder with tables from which these functions interpolate can be found; on the other hand, for each solid material all the function that describes the thermophysical and electromagnetic (if any) properties are collected in a module. The self-explanatory name of the module identifies the material to which it refers. For instance, module named Nb3Sn_properties.py contain all the functions related to the properties of Nb_3Sn.

In subfolder UtilityFunctions there are functional modules that achieves specific and quite different purposes such as read the input files or load the fluid tables, initialize both fluid and solid components, perform the march in time on the algorithm, write the output files and make plots. A short description of these modules can be found in the following Table 3.1-2, they are explained in detail in subsequent sections.

3.1 THE SC2 CODE ORGANIZATION IN NUTSHELL

Table 3.1-2 List and description of the functional modules in subfolder UtilityFunctions.

Module name	Description
Auxiliary_functions.py	Stores functions that allow to read the auxiliary input files, load the fluid tables and perform the binary search in pressure and temperature to compute, by interpolation, fluid thermophysical properties.
Gen_Flow.py	Functions that allow to evaluate the initialization of fluid components variables (velocity, pressure and temperature) according to the value of flag INTIAL, the conductor topology and the kind of interface between fluid components objects.
InitializationFunctions.py	The function that reads and loads the main input files and the one that builds the spatial discretization of the cable are both in this module.
Output.py	Collects several functions that writes the output files of both solution spatial distribution and time evolution in Tab Separated Values (TSV) format using the pandas library.
Plots.py	Groups all the functions that allow to make the default plot of both variables' spatial distribution and time evolution, together with their initialization.
SolidComponents_temperature_initialization.py	Functions that allow to evaluate the initialization of solid components temperature according to the value of flag INTIAL, the conductor topology and the kind of interface between fluid and solid components and between solids.
Transient_solution_functions.py	This is the most important and complex of all these modules. It manages the adaptive time step evaluation, the coefficient matrix and known term vector assembly according to the Finite Element method and the solution of the resulting linear system of equation at each time step, together with the application of BCs according to flag INTIAL.

3.1.2 SC2 INPUT AND OUTPUT ORGANIZATION

Before starting the simulation, user must provide input data to set the simulation features, build the conductor and its basic components, construct the grid, impose the drivers, define cable topology and interfaces and, last but not least, compile the diagnostic to save the output.

This procedure is carried out with the aid of suitable input files in the form of Excel spreadsheets (extension .xlsx) that must be compiled before each run. User can find them in the subfolder called Description_of_Components and they can be further organized into subfolders, each containing a complete set of input files. The set is composed by *main* input files that are necessary for the simulation and must be filled-in at each simulation and *auxiliary* input files. The latter are thought to increase driver input flexibility as external heat, operation current and magnetic field, to assign time dependent boundary conditions or to assign strands strain according to a complicated function of both time and space. Typically, they are read if the value of some flags is negative. Since their peculiarity, they will not be further detailed in this section that concentrates on the main input files. Generally speaking, the information provided with them refers to three well defined levels which are, from top to bottom, simulation level, cable level and basic components level, according to the fact that the input files are loaded into the code as attributes of objects simulation, CONDUCTOR or each one of the required basic components. Typically, Python built-in dictionary data type is exploited to store the information carried by this input files. For a simulation with a single cable there seven main input files are required, specifically one input file at simulation level, four at conductor level and two at basic components level. Auxiliary input files always belong to the last level and their number is variable. Since the modelling of more cables is foreseen by SC2, if more than one cable is to be modeled, user should provide as many input files of the third level as the number of the defined cables, therefore the total number of input file is proportional to that number and can be written in this way:

$$N_{input} = 1 + 3 + (2 + 1)N_{cond} \qquad (3.1\text{-}1)$$

The reason why the number of cables in the above formula is multiplied by $(2 + 1)$ and not by 2 is that file conductor_couplig.xlsx, even though belongs to the second level, has a completely different structure with respect to all the other input files, so a new file of this type must be compiled for each cable. Indeed, while the others are organized by columns, each one identifying a different cable or, within the same cable, different fluid and solid components, each sheet of conductor_couplig.xlsx workbook holds matrix. Main input file compilation is illustrated in appendix E.1, here all of them are listed and concisely depicted:

1. Transitory_Input.xlsx: it is at simulation level, so its input values are saved in a simulation attribute, dictionary transient_input, and it allows to set global information about the simulation. Being the master of all the input files, it is the first file read when the code is launched. Table 3.1-3 lists the input data that user can set compiling it.

Table 3.1-3 List of the input data to be provided in the input file Transitory_Input.xlsx: details about units, the type and its meaning are given for each input data.

Variable name	Unit SI	Variable type	Meaning
SIMULATION	–	string	simulation name
MAGNET	–	string	Input file name that defines conductors
TEND	s	float	Simulation end time
IADAPTIME	–	integer	Flag for the time adaptivity
STPMIN	s	float	Minimum time step
STPMAX	s	float	Maximum time step
TIMEREF	s	float	Time when the most refined step is adopted
TAUREF	s	float	Time duration of the most refined grid

2. conductor_definition.xlsx: it is the file that contains all the information about the cable(s) to be simulated, so it is at conductor level. It can be divided into two parts, the first with the name to all the other input files (both main and auxiliary) to be read; the second with the input data for the cable(s). This file split is put in evidence by the following Table 3.1-4 and Table 3.1-5; two dictionaries, attributes of CONDUCTOR object, stores these inputs namely file_input and dict_input.

Table 3.1-4 List of the input data to be provided in the input file conductor_definition.xlsx. Details about units, the type and its meaning are given for each input data: variables related to the name of all the other input files to be loaded in order to perform the simulation.

Variable name	Unit SI	Variable type	Meaning
EXTERNAL_ALPTHAB	–	string	auxiliary file name to get magnetic field gradient
EXTERNAL_BFIELD	–	string	auxiliary file name to get magnetic field
EXTERAL_CURRENT	–	string	auxiliary file name to get operating current
EXTERNAL_FLOW	–	string	auxiliary file name to get flow input variables
EXTERNAL_HEAT	–	string	auxiliary file name to get external heating
EXTERNAL_STRAIN	–	string	auxiliary file name to get strand strain
GRID_DEFINITION	–	string	main file name to define conductor grid features
OPERATION	–	string	main file name to set conductor components operation properties
OUTPUT	–	string	main file name to define times to save solution spatial distribution and spatial coordinates to save solution time evolution
STRUCTURE_COUPLING	–	string	main file name to define conductor topology and evaluate heat transfer coefficient
STRUCTURE_ELEMENTS	–	string	main file name to set conductor components input data

Table 3.1-5 List of the input data to be provided in the input file conductor_definition.xlsx. Details about units, the type and its meaning are given for each input data: input data that characterizes the conductor.

Variable name	Unit SI	Variable type	Meaning
XLENGTH	m	float	conductor length
IOPFUN	–	integer	flag that defines current space and time dependence
IOP0_TOT	A	float	total initial operating current transported by the cable
ISJOINT	–	integer	flag to define the presence of the joints
XJBEG	m	float	beginning of the heated zone by joule effect in the inlet joint, if any
XJBEIN	m	float	end of the heated zone by joule effect in the inlet joint, if any
XJBEOUT	m	float	beginning of the heated zone by joule effect in the outlet joint, if any
XJENOUT	m	float	end of the heated zone by joule effect in the outlet joint, if any
MAXNOD	–	integer	maximum number of nodes for conductor spatial discretization
METHOD	–	integer	flag to define the numerical solution method for the system of ODE in time
UPWIND	–	integer	flag that switches on the upwind discretization in all the fluid equations according to the chosen METHOD

3. conductor_grid.xlsx: whit this file user specifies the kind of grid (uniform, refined, adaptive) and parameters to be adopted for the spatial discretization, listed in Table 3.1-6 below. This input file is also at cable level and the CONDUCTOR dictionary attribute created to store its information is called dict_discretization.

Table 3.1-6 List of the input data to be provided in the input file conductor_grid.xlsx. Details about units, the type and its meaning are given for each input data.

Variable name	Unit SI	Variable type	Meaning
NELEMS	–	integer	number of elements of the mesh
ITYMSH	–	integer	flag to define the mesh property
NELREF	–	integer	number of spatial elements in the refined zone
XBREFI	m	float	starting point of the refined zone
XEREFI	m	float	end point of the refined zone
SIZMIN	m	float	minimum spatial mesh size, if refined
SIZMAX	m	float	maximum spatial mesh size, if refined
DXINCRE	–	float	size increase ratio for the spatial mesh, from initially refined zone outwards

4. conductor_input.xlsx: this file is at components level, therefore the workbook is composed by five sheets, one for each kind of basic components that can be used to

build the cable. User must compile only the sheets that needs to initialize the cable basic components used in the simulation. It is devoted to the definition of the main features of the different conductor components, e.g. geometry, fluids and solid materials. Each basic component object has a dictionary attribute called dict_input to store these parameters.

5. conductor_operation.xlsx: this workbook inherits the same structure of file conductor_input.xlsx and it is the second main input file that belongs to the lowest level. With this file, user can define flow initial conditions and operating parameters (drivers) for the cable solid components such as magnetic field, current and external heating. Attribute dict_operation is a dictionary that holds these values defined for each of the five basic component objects.
6. After the assignment of the input data for the conductor components, the conductor topology and the interface parameters like the heat transfer coefficients between different components are fully characterized in the eight sheets of file conductor_coupling.xlsx. Its information is saved in dedicated CONDUCTOR dictionaries attributes in form of matrices, that share the same name with the sheets.
7. conductor_diagnostic.xlsx: is the last of the main input files and belongs to the cable level. It is used for the output analysis, since in its two sheets, called Space and Time respectively, user can specify the times at which the solution spatial distribution should be saved and set the spatial location of sensors to get variables time evolution for each cable. The times and spatial coordinates are stored in two different arrays that are attributes of CONDUCTOR object, namely Space_save and Time_save.

The last part of this subsection explains shortly how the output of the simulation is organized, since this problem is also managed by the code. The output of the simulation is both in the form of data files with the extension .tsv and of vector figures in the .eps format saved in Simulations_results subfolder. Thanks to the GUI, user can group together several simulations with similar (but not exactly the same) input data in subfolders inside Simulations_results, therefore this subfolder will contain an arbitrary number of sub-subfolders, which are further divided according to the solution method, each of them related to a specific simulation. The structure of this last folder is the object of this concluding part. A first distinction is among the data files and the figures: the formers are grouped in Output folder while the latter in the Figures folder. The inner structure of this folders is the same: there are four folders to distinguish between initialization, spatial distributions, time evolutions and final solution, respectively called Initialization, Space, Time and Solution. Inside each of these folders there are as many folders as the number of the initialized conductors in the simulation whose name correspond to the conductor identifier. As far as Output path is concerned, each conductor output data files are stored within the corresponding folder, while figures are grouped according to the conductor components in the homonym folder of the Figures path.

3.1.3 SUMMARY

To sum up, the SC2 code is a Python 3 object-oriented program thought to model thermal behavior of the superconducting cables, tested to model a single cable (for the time being). Its core is organized into eleven primary modules; of them only the launcher one, Simulation_Starter.py, does not define a class, while the others ten define likewise classes. Being a code equipped with a graphical user interface, one of them is dedicated to this feature and it is called SC2_GUI.py. Class Simulation manages both the conductor objects definition and the steps of the algorithm, while the others build up the cable, which is constructed combining only five kinds of basic conductor components objects, namely CHAN, Z_JACKET and one or an arbitrary combination of the three strands kinds (STR_MIX, STR_SC and STR_STAB). This simple structure is achieved exploiting python class inheritance on the classes that defines the solid components objects.

The secondary modules are grouped into two subfolders. Properties_of_materials collects all the fluid and solid material thermophysical and electromagnetic properties, while UtilityFunctions is where modules with different goals can be found.

The input files, in the form of excel worksheets, are stored in the subfolder Description_of_Components and can be divided into two groups: main input files to be compiled by the user for each new simulation, and other auxiliary ones available to extend input flexibility.

This section ends with a short but exhaustive description of the output organization into subfolders within the Simulations_results directory, as allowed by the GUI.

3.2 CODE KERNEL: CONDUCTORS, FLUIDCOMPONENTS AND SOLIDCOMPONENTS CLASSES

Having described the entire architecture of the code in the previous section, attention is paid to the classes that contribute to building the cable, proposing a description of the main attributes and methods which characterize them.

FluidComponents and SolidComponents classes have most of their attributes in common: the same kind of information is stored in a variable that has the same name regardless of the class. What distinguishes the two classes are their methods and the inheritance that is only a feature of SolidComponents class. Class Conductors has many attributes, but the same rationale is used also for this case: the same kind of information is stored in attributes with the same name. In this way it is easy to access the required variables and this is not error prone since each attribute belongs to a different object, even if they share the same name. The common attributes of these classes are listed and shortly described in Table 3.2-1.

It is a good practice to define all the attributes of a class in its __init__ method, so all the above-mentioned attributes (and the not mentioned ones) are at least declared in the constructor method of the class to which they belong.

3.2 CODE KERNEL: CONDUCTORS, FLUIDCOMPONENTS AND SOLIDCOMPONENTS CLASSES

For each class the attribute NAME is already defined in Table 3.1-1, while the ID construction is explained in section 5E.1.

The class Conductors does not have the dict_operation attribute, which is replaced by file_input, that keeps track of the input files to be red and loaded.

Table 3.2-1 List of the attributes in common between the Conductors, FluidComponents and SolidComponents classes. The class to which they belong and a brief description are given.

Attribute	Conductors	Fluid-Components	Solid-Components	Description
NAME	X	X	X	Name of the object, it is always the same for objects belonging to the same class.
ID	X	X	X	Object identifier, it changes for each object of the same class and allows to uniquely identify the object. It came from the input files.
dict_input	X	X	X	Dictionary to load the input data.
dict_operation		X	X	Dictionary to load the operation parameters.
dict_node_pt	X	X	X	Dictionary to save thermophysical and/or electromagnetic properties values in the nodal points.
dict_Gauss_pt	X	X	X	Dictionary to save thermophysical and/or electromagnetic properties values in the Gauss points.

The last two attributes of Table 3.2-1 not only store different quantities but are also differently organized according that they belong to Conductors or FluidComponents and SolidComponents. Generally speaking, dict_node_pt and dict_Gauss_pt contain the same properties that are evaluated in two different locations and in two different ways. In the former dictionary the properties are evaluated in the node of the spatial discretization; the latter stores the same quantities evaluated in the Gauss point which is the mid-point of each interval of the partition of the domain. The evaluation of the properties in the Gauss points requires a minimum amount of information in the nodal points.

The initialization or the solution of the linear system of equations, provides at each time step this minimum data set that is the spatial distribution of the unknown variables in the

nodal points, namely velocity, pressure and temperature for each fluid component and temperature for each solid component.

This are the arguments of the functions used to evaluate all the other properties in the nodal points.

$$\boldsymbol{\chi}_{nodal} = f(\boldsymbol{\xi}_{nodal}, \dots) \tag{3.2-1}$$

To evaluate the properties in the Gauss points, two paths can be followed. The simplified one is to take the average value in each interval of the partition of the domain:

$$\chi_{i,Gauss} = \frac{\chi_{i,nodal} + \chi_{i+1,nodal}}{2} \; i = 0, \dots, N \tag{3.2-2}$$

The second implies to evaluate only the solution in the Gauss points:

$$\xi_{i,Gauss} = \frac{\xi_{i,nodal} + \xi_{i+1,nodal}}{2} \; i = 0, \dots, N \tag{3.2-3}$$

and then use this information as arguments of the functions to evaluate the other properties in the Gauss points:

$$\boldsymbol{\chi}_{Gauss} = f(\boldsymbol{\xi}_{Gauss}, \dots) \tag{3.2-4}$$

The first strategy is currently adopted in the 4C; if dealt with the array smart notation it will allow to reduce the computational cost of the algorithm, however it is less accurate with respect to the last one implemented in SC2 code.

In any case, the $\boldsymbol{\chi}_{Gauss}$ and $\boldsymbol{\xi}_{Gauss}$ vectors have one less element than the nodal ones.

In the following three subsections, some details on the last two attributes of Table 3.2-1 are supplied, then the methods of the class are explained. Whenever it is possible, the methods are developed using the *array-smart* notation; this applies to the whole code in general.

3.2.1 FLUIDCOMPONENTS CLASS

As the name prefix suggest, dict_node_pt and dict_Gauss_pt are two dictionaries. Inasmuch attributes of CHAN objects, their keys store several fluid properties, as well as the transport properties like the friction factor and the steady state heat transfer coefficients. To each dictionary key corresponds a *numpy array* of length $N + 2$. The list of dictionary keys and their meaning is given in Table 3.2-2.

After that the main attributes of the class are described, it is worthily that the reader is aware of the method that it makes available to the user.

As mentioned, the most important method of a class is its inner __init__ method that is the constructor of the object when the class is instantiated. Fluid components class provides several other methods. Eval_fluid_comp_properties and Eval_properties allow to evaluate the CHAN properties both in nodal or Gauss points according to the value of the *keyword* (or

named) *argument* "Where" while the kind of fluid can be specified with the keyword argument "Fluid", a flag that selects the correct tables.

Compute_denisty_and_mass_flow_rate is a method introduced to easily evaluate, at each time step, the density and the mass flow rate (both at the inlet and at the outlet) in the nodal points to be saved together with the solution at the user required time step. As before, the kind of fluid is specified by the named argument "Fluid".

Friction factor is computed with different correlations as function of Reynolds and of the channel geometry in Friction method that calls the Newton_f and User_Friction methods; the latter allows to employ a user defined correlation for the friction factor. This is one of the methods that contains the constitutive relations introduced in section 2.1.1.

Finally, Compute_velocity and Compute_mass_flow are invoked in Gen_Flow.py module to compute the flow initialization, if inlet and outlet pressure are provided together with inlet temperature.

Table 3.2-2 List of dictionaries dict_node_pt and dict_Gauss_pt keys and their description.

Key	Description
velocity	Coolant velocity spatial distribution
pressure	Coolant pressure spatial distribution
temperature	Coolant temperature spatial distribution
density	Coolant density spatial distribution
enthalpy	Coolant enthalpy spatial distribution
Gruneisen	Coolant Gruneisen spatial distribution
sound_speed	Coolant speed of sound spatial distribution
spec_heat_p	Coolant specific heat at constant pressure spatial distribution
spec_heat_v	Coolant specific heat at constant volume spatial distribution
ther_cond	Coolant thermal conductivity spatial distribution
viscosity	Coolant dynamic viscosity spatial distribution
Reynolds	Coolant Reynolds dimensionless number spatial distribution
Prandtl	Coolant Prandtl dimensionless number spatial distribution
mass_flow_rate	Coolant mass flow rate spatial distribution
htc_steady	Coolant steady state heat transfer coefficient spatial distribution
friction	Coolant friction factor spatial distribution

3.2.2 SOLIDCOMPONENTS CLASS

In this section not only the parent class SolidComponents but also all their child and grandchild classes are considered, to detail what arguments and methods are in common thanks to the inheritance properties and what are specific of the class. The top-down approach is followed; the class hierarchy is shown in Figure 3.1-1.

All the attributes reported in Table 3.2-2 are listed in class SolidComponents but they are declared in the __init__ method of the basic solid components classes: Jacket, MixSCStabilizer, SuperConductor and Stabilizer. Since there is no need to build a solidcomponent object, this

parent class does not have the __init__ method, nevertheless there are several other methods used to evaluate properties and set the drivers.

Eval_sol_comp_properties and Eval_properties are a couple of methods that allow to evaluate some thermophysical properties like density, specific heat, thermal conductivity and electrical resistivity. The kind of object and its composition are taken into account with suitable flags and if-else branches; the average properties for the composite objects are computed as explained in appendix C. These methods can be used to evaluate the properties both in nodal and Gauss points thanks to the keyword argument "Where". The functions for the material are imported from modules stored in Properties_of_material subfolder.

Method Get_I initializes the electrical current in each solid component object. Even though the electrical module is not developed yet, this method is already foreseen in this class. The same can be said for Get_B_field that computes the magnetic field in each solid component both in nodal and Gauss points exploiting the "Where" argument.

Get_Q is thought to compute the external heating according to the value of the flag IQFUN, the named argument "Method" keeps into account the method chosen to integrate the system of ODEs. If IQFUN is larger than zero, this method calls Q0_where that determines the square wave in time and space for the external heating according to the input parameters in conductor_operation.xlsx workbook.

The last two methods JHTFLX_new_0 and Set_energy_counters allow to initialize to zero the vectors that represents the Joule heating and the external and Joule heating energies, respectively. These initializations are performed according to the selected solution method with the keyword "Method".

Child class Strands does not add other attributes to the ones already established in class SolidComponents and, as the previous, it does not have the __init__ method since there is no need to build strand objects. Its methods are inherited by the three strand kinds and are briefly described below.

Get_alphaB is the one devoted to the evaluation of the magnetic field gradient in the strands and at the time being it is not used since there is no current, however it is already integrated in the SC2 code; as usual the properties can be evaluated in nodal or in Gauss points by specifying the keyword "Where".

Methods Get_superconductor_critical_prop and Eval_critical_properties evaluate the superconductor critical properties. They can distinguish both the location, nodal or mid-point thanks to the named argument "Where", and the kind of superconductor: available choices are NbTi, Nb_3Sn and RE123. The correct functions are imported from modules NbTi_properties.py, Nb3Sn_properties.py and RE123_properties.py collected in subfolder Properties_of_materials. Since Nb_3Sn critical properties are function of the strain in the strand, the method Get_EPS allows to determine it according to the value of the flag IEPS in the input file conductor_operations.xlsx. Again, both the evaluation in nodal and in Gauss points are allowed exploiting the "Where" keyword.

Classes Jacket, MixSCStabilizer, SuperConductor and Stabilizer only have the __init__ method where the inherited attributes from the parent class SolidComponents are (at least) declared when the class is instantiated. Although the attributes dict_nodal_pt and dict_Gauss_pt are the same in these classes, as a consequence of the different methods that are used to construct them, their keys can be very different according to the object to which they belong; this is clarified in the following Table 3.2-3.

Table 3.2-3 List of dictionaries dict_node_pt and dict_Gauss_pt keys together with their description. Note that not all the keys are available for the dictionaries, according to the class these attributes belong.

Key	Jacket	MixSC-Stabilizer	Super-Conductor	Stabilizer	Description
alpha_B		X	X	X	Solid component gradient of the magnetic field spatial distribution
B_field	X	X	X	X	Solid component magnetic field spatial distribution
density	X	X	X	X	Solid component density spatial distribution
el_resist	X	X	X	X	Solid component electrical resistivity spatial distribution
Epsilon		X	X		Solid component strain spatial distribution
J_critical		X	X		Solid component critical current density spatial distribution
spec_heat_p	X	X	X	X	Solid component specific heat at constant pressure spatial distribution
T_critical		X	X		Solid component critical temperature spatial distribution
T_cur_sharing		X	X		Solid component current sharing temperature spatial distribution
T_cur_sharing_min		X	X		Solid component minimum current sharing temperature spatial distribution
temperature	X	X	X	X	Solid component temperature spatial distribution
ther_cond	X	X	X	X	Solid component thermal conductivity spatial distribution

Only two of the four classes, MixSCStabilizer and SuperConductor, have the full set of keys since they need both thermophysical and electromagnetic properties to be described, while

the Z_JACKET objects are completely described with only the former ones plus the magnetic field. STR_STAB, in addition to the thermophysical properties, also requires the magnetic field and its gradient but not the set of critical properties since it is not a superconductor. To be picky, the STR_MIX and STR_SC actually have the whole set of keys only if the critical properties of the superconductor material are function of the strain, as in the case of Nb_3Sn, otherwise also in this case the last key is missing from the dictionaries.

Dictionary dict_nodal_pt has also the keys for the drivers that are not shared with dict_Gauss_pt, these are EXTFLX, JHTFLX, EEXT and EJHT. The first two are summed together as follows to build the key Q1 and Q2 in the dict_Gauss_pt that are used to construct the load contribution to the known term vector:

$$Q1_i = EXTFLX_i + JHTFLX_i \; i = 0, \dots, N \tag{3.2-5}$$

$$Q2_i = EXTFLX_{i+1} + JHTFLX_{i+1} \; i = 0, \dots, N \tag{3.2-6}$$

3.2.3 CONDUCTORS CLASS

This class is quite complicated since it manages all the other classes discussed above. Many other attributes are associated to its objects and in this section only the most important are considered. Two of them are the already introduced dictionaries dict_node_pt and dict_Gauss_pt, the others are dict_discretization, dict_Step, dict_obj_inventory and dict_topology. The reader should have noticed that all the above attributes are Python dictionaries: actually, this Python built-in data type is extremely powerful and useful to organize data and information that belongs to the same kind, and easily access them calling the corresponding key. In this way the code is more readable, especially if the keys are self-explanatory.

If dict_node_pt and dict_Gauss_pt belong to CONDUCTOR objects, they are used to store the heat transfer coefficients between fluid components, fluid and solid components and between solid components. They are nested dictionaries organized as shown below:

```
# htc dummy dictionary and its sub-dictionary declaration
dict_dummy["HTC"] = dict()
dict_dummy["HTC"]["ch_ch"] = dict()
dict_dummy["HTC"]["ch_ch"]["Open"] = dict()
dict_dummy["HTC"]["ch_ch"]["Close"] = dict()
dict_dummy["HTC"]["ch_sol"] = dict()
dict_dummy["HTC"]["sol_sol"] = dict()
```

where dict_dummy is the generic dictionary dict_node_pt or dict_Gauss_pt. The main key **HTC** is associated with a dictionary that stores all the values of the heat transfer coefficients. This dictionary has three keys (**ch_ch**, **ch_sol** and **sol_sol**) to keep into account the three possible kinds of interfaces, to which are associated other dictionaries. The last two, **ch_sol** and **sol_sol**,

have keys that identifies the objects that constitute the interface. The convention is that the keys are constructed joining the objects ID alphabetically with an underscore, for example if CHAN_1 and STR_MIX_1 constitute an interface the key name is **CHAN_1_STR_MIX_1** and this key will belong to dictionary **ch_sol** since it is between a fluid and a solid component. The value of this key is an array of heat transfer coefficients evaluated calling Get_transp_coeff and Eval_transp_coeff methods of the class according to the information stored in input file conductor_couplign.xlsx; they can distinguish if the evaluation should be done in the nodal rather than in the Gauss points thanks to the keyword "Where". These methods and specifically Eval_transp_coeff implements the second kind of constitutive relations discussed in section 2.1.1.

Dictionary **ch_ch** is different from the previous two since it keeps into account that channels can have both open or close contact perimeter that lead to different values of the heat transfer coefficients, therefore another dictionary level is added with the keys Open and Close. Both dictionaries have the same structure described for **ch_so**l and **sol_sol**.

Attribute dict_Gauss_pt stores also the transport coefficients, K', K'' and K''' that are deeply depicted in appendix A, so together with the **HTC** key three other keys (**K1**, **K2** and **K3**) are added, to which are associated likewise dictionaries to store these values. Each of these dictionaries has a set of interface keys of the kind **CHAN_i_CHAN_j** with $i > j$ with values the transport coefficient array. These coefficients can be evaluated only for fluid components with a not null value of the open perimeter fraction and their evaluation is performed with the properties in the Gauss points.

Dictionary dict_discretization has three keys **Grid_input**, **N_nod** and **xcoord**. The first is a dictionary itself that stores the input data loaded from file conductor_grid.xlsx for the i-th CONDUCTOR, the second has the number of nodes of the spatial discretization, since the number of elements, that is the number of intervals, is provided in input, while the third is the numpy array of the spatial discretization. This single array is used for all the cable components whenever it is necessary.

Another dictionary, dict_Step, is devoted to store two arrays essentials for the solution of the linear system of equations. The first is associated to the key **SYSVAR** and it is the array of the initialization at the initial time step and the array of the solution at each subsequent time steps; the second key is called **SYSLOD** and it considers the contribution of the external source of heating in the solid components. Both are used to build the known term vector, but they can be thought as matrices with fixed number of rows (equal to N_{tot}) and a variable number of columns that is function of the method used to numerically integrate the linear system of equations.

The last two mentioned attributes dict_obj_inventory and dict_topology have a well-organized and articulated structure that will be addressed in the following subsection. It is time to take a closer look to the methods of the class. The conductor constructor is discussed in the next section since it is quite complicated; as a matter of fact, it invokes other methods that in turn rely on other methods and all of them will be somehow explained in that section. For the time being the focus is on the methods that are not called by __init__.

The method Initialization accomplishes several tasks: it builds the conductor mesh, initializes the fluid and solid components variables, initializes current, magnetic field and drivers for each solid component, constructs and initializes dict_Step attribute and saves the computed thermophysical and electromagnetic properties in suitable files. Last but not least, it starts the real time plot of selected variables, that will be discussed lather on. Several of these operations are performed invoking functions from modules in subfolder Utility_Function or other class methods like Operating_conditions, that evaluate at each time steps solid components current, magnetic field and its gradient together with the external drivers calling the methods discussed in section 3.2.2. This method also invokes CONDUCTOR Eval_Gauss_point that on the one hand evaluate both fluid and solid components thermophysical and electromagnetic properties (if any) in the Gauss points, taking advantage of the components methods presented above; on the other hand invokes Get_transp_coeff with the "Where" keyword set to Gauss to compute the heat transfer coefficients in the mid-points.

The Post_processing method is similar to the Operating_conditions one but it allows to evaluate the quantities in the nodal points, moreover it invokes functions Save_space_convergence and Save_time_converence to store data useful if space and/or time convergence analysis should be performed.

3.2.3.1 __INIT__ METHOD

This method can be roughly subdivided into four main actions:

1) read and load the content of the primary input files;
2) instantiate the cable components according to the information obtained with the previous step;
3) get the conductor topology;
4) define all the other attributes.

Here the second and third points of the list are detailed.

In order to have an exhaustive picture of the conductor components, the attribute dict_obj_inventory is introduced that, as its name suggests, is a Python dictionary that keeps track of the inventory of the defined objects, grouping them in several categories (see Figure 3.2-1). Each of these categories are the keys of this dictionary to which a nested dictionary is associated; key names are **FluidComponents**, **MixSCStabilizer**, **Stabilizer**, **SuperConductor**, **Jacket**, **Strands**, **SolidComponents** and **Conductor_components**. Please notice that, apart from the last one, all the keys coincide with the name of the classes that defines the cable components objects. The first five dictionaries have three keys:

1. **Name**: string value that corresponds to the kind of the object, it will be assigned to the NAME attribute of the class and used as the root of the object identifier;
2. **Number**: integer value that represents the total number of instantiated objects of the kind specified in Name;
3. **Objects**: Python list that stores all the instantiated objects of the corresponding class.

3.2 CODE KERNEL: CONDUCTORS, FLUIDCOMPONENTS AND SOLIDCOMPONENTS CLASSES

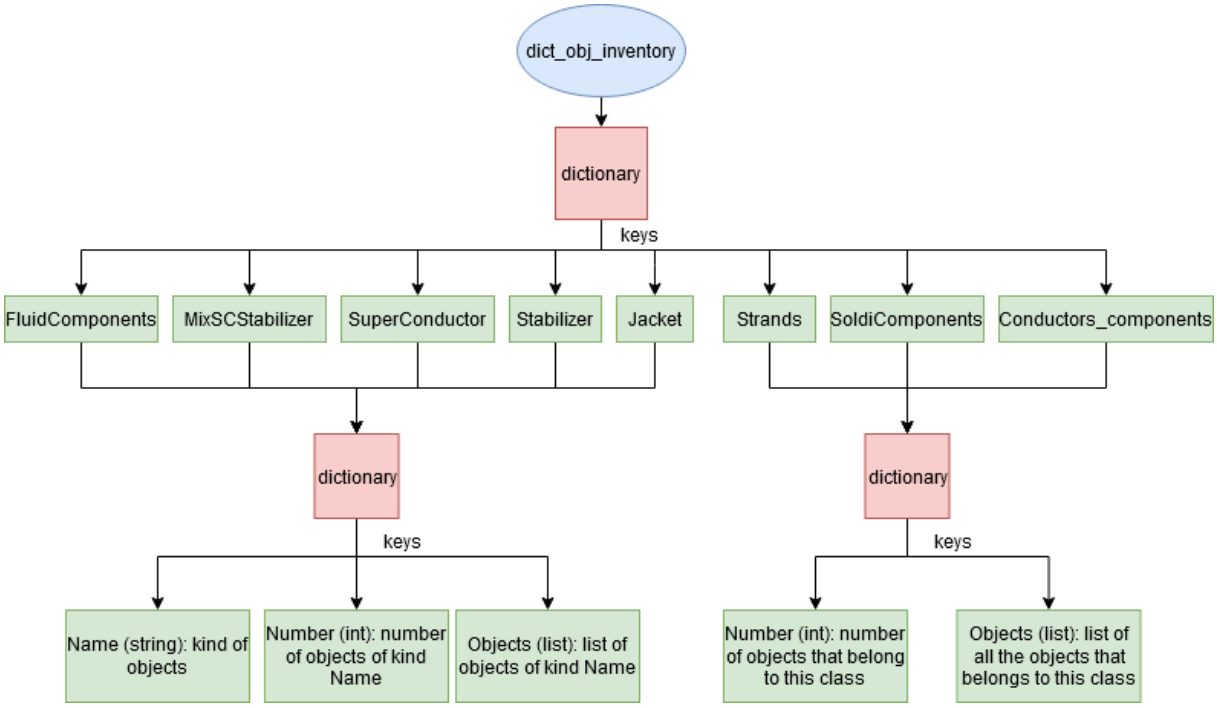


Figure 3.2-1 Nested structure of the attribute dict_obj_inventory of class Conductors.

If no objects of a basic kind are instantiated Number is equal to zero, the Objects list is empty while the Name key shows the object name in any case.

The dict_obj_inventory is declared as a dictionary in the __init__ method and its structure is built within the method Conductor_components_instance that instantiates all the conductor components defined by the user in primary input files conductor_input.xlsx and conductor_operation.xlsx. For each sheet in these files, after the check that the number of defined objects and their identifiers are the same, the corresponding class is instantiated to create the objects, that are stored in the dictionaries above described. An error is raised if the sheet name is not equal to one of the foreseen five possibilities. Outside the loop, the lists of the dictionaries **Strands**, **SolidComponents** and **Conductor_components** are sorted alphabetically and the total number of objects that belongs to each key is computed. The flow chart of the method is proposed in *Figure 3.2-2*.

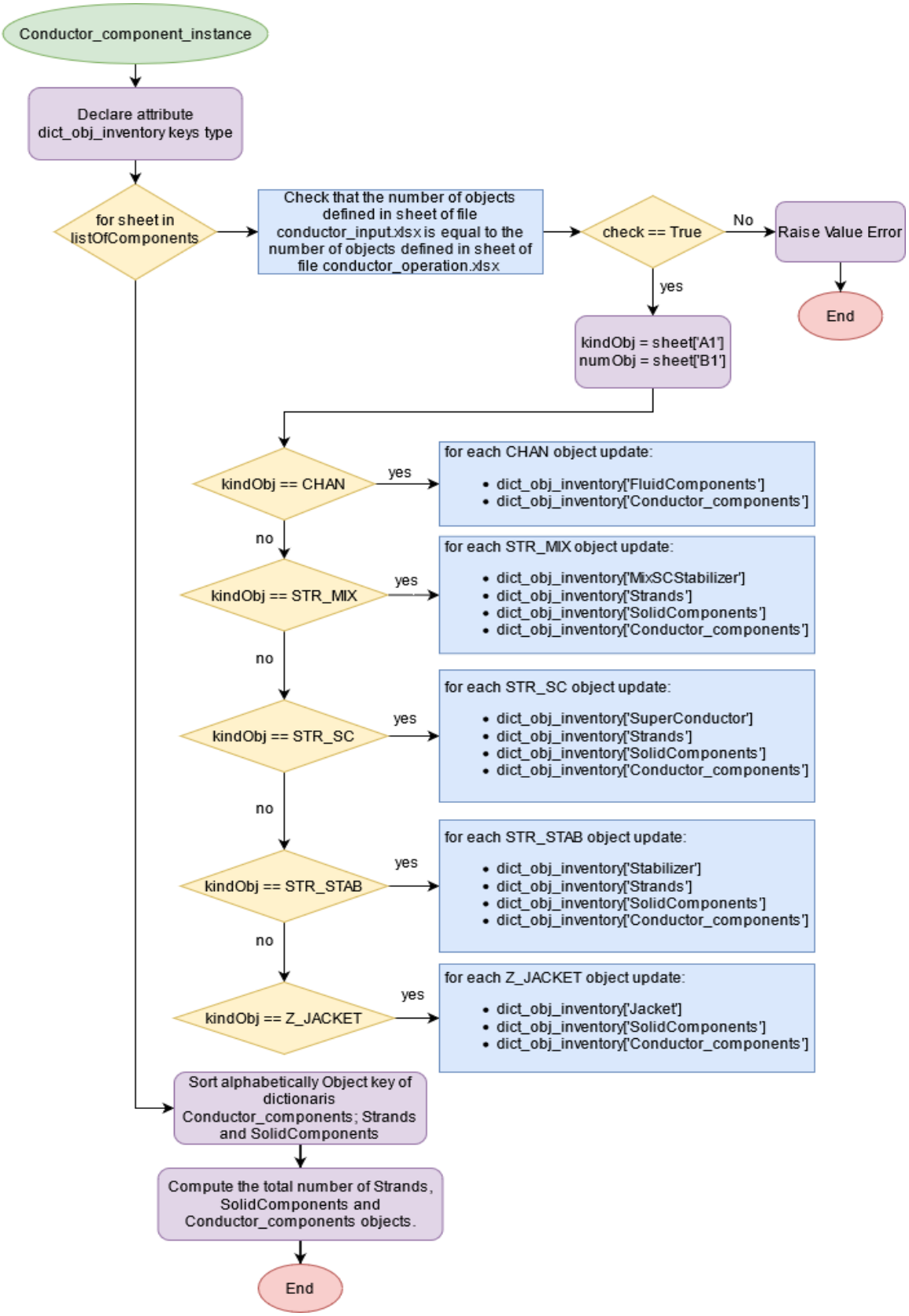


Figure 3.2-2 Flow chart of method Conductor_component_instance of class Conductors, invoked during the conductor definition by the __init__ method.

The topology of the conductor is fully described in the attribute dict_topology that is another example of structured nested dictionaries. Analogously to dict_obj_inventory, it is declared as a dictionary in __init__ method while its structure is developed in method Get_conductor_topology. To understand the architecture of this dictionary, it is crucial to explain the different kinds of interface foreseen in the SC2 code between conductor components. As already discussed about the dict_node_pt or the dict_Gauss_pt attributes of CONDUCTOR objects, there are three physically meaningful interfaces, namely interface between fluid components, interface between fluid and solid components and interfaces between solid components. From the design point of view, the last two possibilities can only be due to thermal contact between two or more components. From the fluid components side there are two physical relevant alternatives, and namely the hydraulic parallel and the thermal contact; the former means that the fluid components do not only exchange energy through the close fraction of the contact perimeter, but they also exchange mass, momentum and energy through the open fraction. These two possibilities modify significantly the coupling between the fluid equations as described in section 2.1.3, and have an impact on flow initialization and boundary conditions application that will be discussed in section 0. The main issue related to the hydraulic parallel configuration is that the properties of the fluid in one channel, especially the pressure, cannot be considered independently from the properties in the other linked channels; moreover, the transitive property must be guaranteed in order to not make serious mistakes. According to this property, if there are three channels with identifiers CHAN_1, CHAN_2 and CHAN_3 and if CHAN_1 is in hydraulic parallel with CHAN_2 and CHAN_2 is in hydraulic parallel with CHAN_3, then CHAN_1 and CHAN_3 are also in hydraulic parallel and they constitute a group of channels in hydraulic parallel. This is currently kept into account in dict_topology. The possible interfaces for fluid components are clarified in Figure 3.2-3.

There is another possibility left that is not achievable in practice and not physically relevant but can be useful form the computational point of view when debugging, that is the absence of interfaces between components. When this condition is applied to the fluid components the resulting channels that are not in contact are called standalone channels; analogously for the solid components. It is also allowed to remove the interfaces between fluid and solid components. A standalone channel that is not in thermal nor in hydraulic parallel with other channels is a condition that may occur so only the standalone channels are modeled in SC2 code.

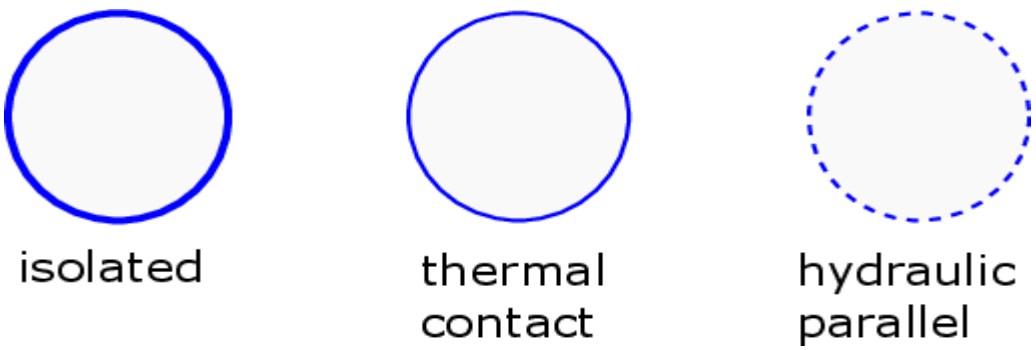


Figure 3.2-3 Graphic representation of the three kind of interfaces between fluid components.

Taking into account these observations, it is easier to introduce and describe the structure of attribute dict_topology declared as follows:

```
# nested dictionaries declaration
self.dict_topology["ch_ch"] = dict()
self.dict_topology["ch_ch"]["Hydraulic_parallel"] = dict()
self.dict_topology["ch_ch"]["Thermal_contact"] = dict()
self.dict_topology["Stand_alone"] = dict()
self.dict_topology["ch_sol"] = dict()
self.dict_topology["sol_sol"] = dict()
```

The four possibilities are the main keys of the dictionary to which other dictionaries are associated as values, except for the Standalone_channels that is a list of the CHAN objects that are isolated from other channels.

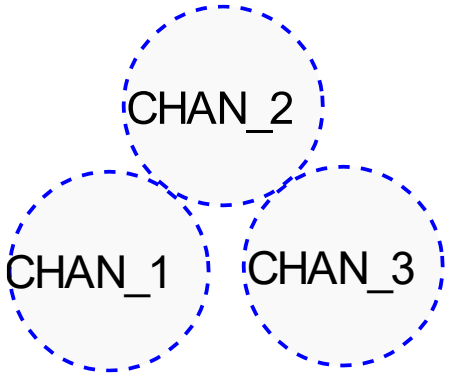
The first key is split into two dictionaries: **Hydraulic_parallel** for fluid components in hydraulic parallel, including the ones that are in parallel due to the transient properties; **Thermal_contact** that groups the channels that are only in thermal contact. The third level is a set of dictionaries that are identified by the reference component, i.e. the first component in alphabetical order that constitutes the interfaces. This is the last dictionary level; its keys are:

- **ID** of the components that contributes to the interface: the corresponding value is the interface identifier, a list of strings that are obtained joining alphabetically with an underscore the components identifiers;
- **Actual_number**: integer value that represent the current number of components that constitute the interface; it is used in the dictionary construction.
- **Number**: integer, the total number of the components belonging to the interface;
- **Group**: list of all the objects that build the interfaces, its length must be equal to the value in key **Number**. In the case of fluid components in hydraulic parallel, the list keeps into account also the channels that are not directly in contact but indeed they are thanks to the transitive properties.

A set of methods of class Conductors are devoted to gather cable topology exploiting only the information in contact_perimeter_flag and in open_perimeter_fraction sheets of file conductor_coupling.xlsx, the latter is used to determine if two channels are directly in hydraulic parallel or only in thermal contact. The content of this input file is described in appendix E.2.

When the object is instantiated, __init__ method invokes the Get_conductor_topology method. Interfaces between solid components and interfaces between fluid and solid components are quite straightforward to obtain since they are provided as input in sheet contact_perimetre_flag. The complicated part is the one relative to the interfaces between fluid components, since basing only on the value of the flag does not allow to get all the channels in hydraulic parallel, because the ones for which the transitive properties should be

applied are not explicitly specified. This problem can be overcome looking at the rows and columns index of the channels submatrix. To explain the idea, recall the previous example of the three channels above. The configuration and the corresponding submatrix are shown in Figure 3.2-4.



(a)

	CHAN_1	CHAN_2	CHAN_3
CHAN_1	0	1	0
CHAN_2		0	1
CHAN_3			0

	CHAN_1	CHAN_2	CHAN_3
CHAN_1	0	0.2	0
CHAN_2		0	0.3
CHAN_3			0

(b)

Figure 3.2-4 (a) fluid components configuration with CHAN_1 and CHAN_2 and CHAN_2 and CHAN_3 in direct hydraulic parallel, therefore by the transitive property also CHAN_1 and CHAN_3 are in hydraulic parallel;(b) top right matrix is the fluid components submatrix in sheet contact_perimeter_flag while the bottom right is the corresponding submatrix in sheet open_perimeter_fraction. Notice the conditional formatting in the cell, green cells should be filled, yellow cells should not, finally red cells are protected and cannot be edited.

If Get_conductor_topology treats this configuration as does for the interfaces between the solid components, the transitive properties would not be applied since the results is that CHAN_1 and CHAN_2 are in hydraulic parallel as well as CHAN_2 and CHAN_3 but not CHAN_1 and CHAN_3 because the value of the flag is 0; however, they are in hydraulic parallel. To get this, the method does not limit to read the values of the flags, but it interprets the hidden information looking at the rows and columns index. The flag associated to the interface CHAN_1_CHAN_2 is in (1,2) while the one associated to the interface CHAN_2_CHAN_3 is in (2,3). The column index (2) of the first is equal to the row index (2) of the second; subordinate methods to Get_conductor_topology catch this equality and establish that also CHAN_1 and CHAN_3 even if they do not constitute an interface, are in hydraulic parallel. This applies only if both the open perimeter fractions are larger than zero.

In general, this search is applied to all the pair of channels that are in direct hydraulic parallel to find all the others that are indirectly related to them, constituting a group of fluid components in hydraulic parallel. Practically this is done with method Get_hydraulic_parallel that in turn calls Search_on_ind_col and Search_on_ind_row. After that all the channels in hydraulic parallel are found, Get_conductor_topology identifies the ones in thermal contact invoking Get_thermal_contact_channels. It can occur that a channel is both in hydraulic parallel with some channels and in thermal contact with others; in this case, the method knows that the hydraulic parallel condition is stronger than the thermal contact one, so in the flow initialization it is treated like a channel that belong to a group of fluid components in hydraulic parallel. Finally, the Find_Standalone_channels method is used to collect all the

channels that are not in hydraulic parallel nor in thermal contact with other fluid components (if any).

The other interfaces are dealt directly by method Get_conductor_topology.

3.2.4 SUMMARY

This section describes with some level of detail the main attributes and methods of the classes that, altogether, participate in the mesoscale modeling of the cable. One of the basic philosophies adopted is that the same kind of information is stored in attributes that have the same name, and the same basic structure, while belonging to different objects. Unlike FluidComponents and SolidComponents classes that have a small amount of attributes, class Conductors has several attributes, many of them are well structured nested dictionaries. Among them the most important are dict_object_inventory and dict_topology. The section also describes the methods of the classes; emphasis is given to the __init__ method of the Conductors class that allows to initialize all the other objects and get their topology.

3.3 SIMULATION CLASS: THE STEPS THROUGH THE SOLUTION

The Simulation class handles all the steps of the simulation at the higher level and it is integrated with the graphical user interface that will be described in the next section. The essential flow chart of the code is shown in *Figure 3.3-1* and ultimately it coincides with the method of this class.

The constructor method __init__ takes care of reading the first of the main input file, Transitory_Input.xlsx calling the external function Read_input_file. The carried information are saved in the attribute transient_input,a python dictionary data type. Two other important tasks of this method are:

- build the base path, necessary to read all the input files including Transitory_Input.xlsx.
- declare the list of conductors in the corresponding attributes list_of_Conducutors , set equal to python list.

Simulation has two distinct methods that perform, respectively, the instance of class Conductor and the CONDUCTOR object initialization; the reason behind is that when more than one cable has to be modeled their initialization is subordinated to their interface. Method Conductor_instance, as its name suggests, is devoted creating the objects. The input file conductor_definition.xlsx is read at this point and a loop on the total number of conductors in cell B1 is performed to make an instance of the class Conductor that calls the __init__ method of the class deeply discussed in section 3.2.3.1; then the objects are appended to the list_of_Conductors attribute. After the loop, auxiliary function Load_fluid_tables from module Auxiliary_functions.py is invoked to load the proper fluid properties according to the fluids used in the simulation. In this way the tables are loaded only once and stored in the dictionary dict_fluid_tables, an attribute of the class Simulation. Finally, the sheet CONDUCTOR_COUPLING of workbook conductor_definition.xlsx is red and saved. This sheet was not discussed until now: it contains a matrix that describes the interfaces between the

3.3 SIMULATION CLASS: THE STEPS THROUGH THE SOLUTION

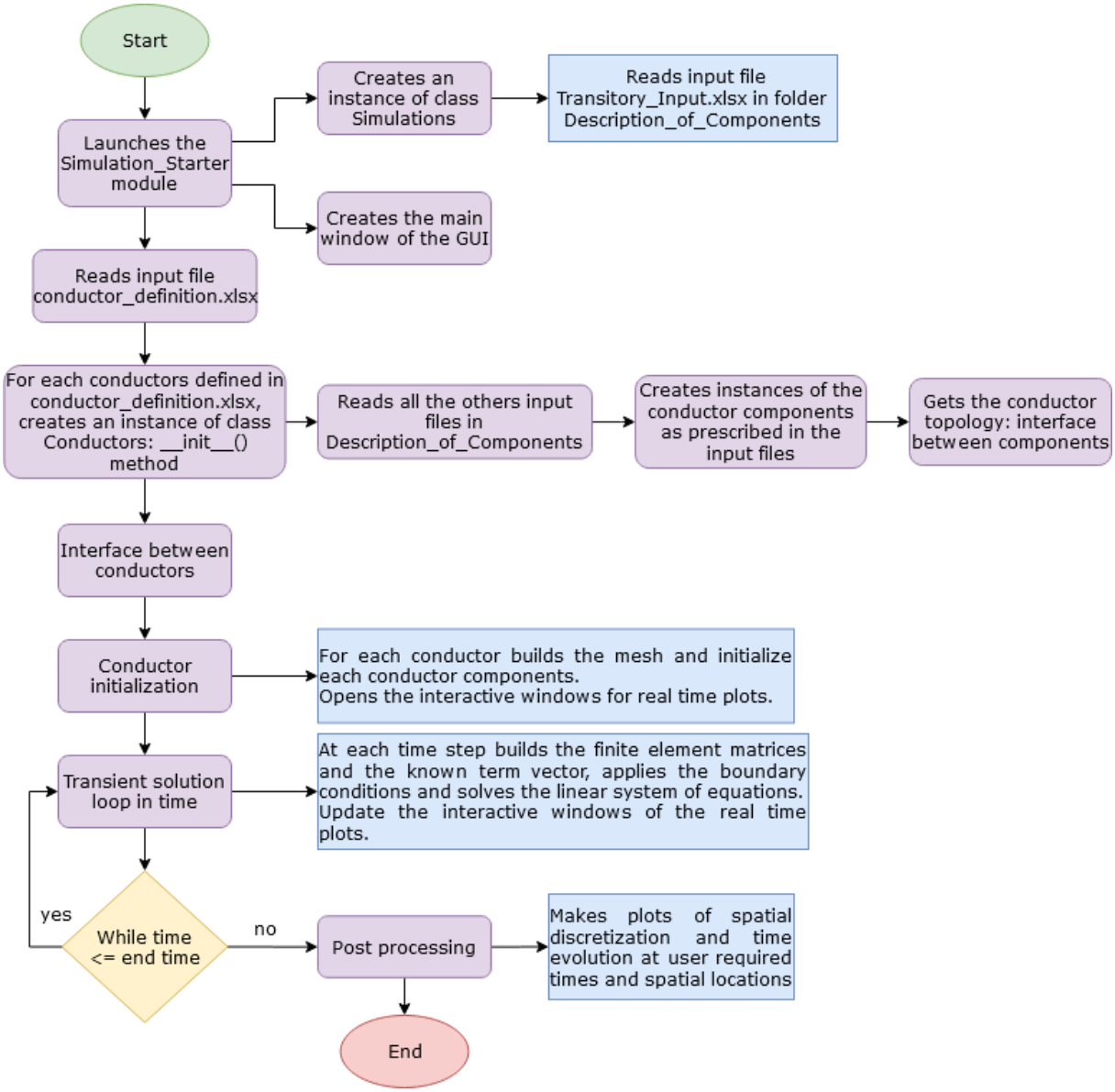


Figure 3.3-1 Essential flow chart of the SC2 code.

conductors in the same way of file conductor_coupling.xlsx; however, since in this thesis a single cable is considered, it has no effects.

For each instantiated object method, Conductor_initialization invokes the CONDUCTOR.Initialization method that, in turn, initializes its basic components. This method was shortly discussed in section 3.2, at this point it is worthily that the reader is aware of how the fluid and solid components initialization is evaluated, because it is one of the most important steps of the whole procedure. The former is managed by the module Gen_Flow.py, the latter by the module SolidComponents_initialization.py. Further details are give in following sections 3.3.1 and 3.3.2, while the problem solution is addressed in section 3.3.3.

3.3.1 FLUID COMPONENTS INITIALIZATION

There are several possibilities for the fluid components initialization that are ruled by the flag INTIAL, as an example the algorithms used to perform the initialization when the flag value is $|1|$, $|2|$, $|5|$, are explained below. The first important consideration is that for positive value of the flag, data came from the main input files, while if is negative they are loaded from ancillary input files. Table 3.3-1 below summarizes the initialized properties for the different value of INTIAL.

Alongside the flag value, another factor affecting fluid initialization is cable topology; indeed, for the same value of INTIAL, different algorithms are implemented according that the fluid components are in hydraulic parallel or not. It is worthy to mention that if the channels are not in hydraulic parallel, they can be initialized with different values of INTIAL, however if they belong to the same group of channels in hydraulic parallel, the same initial data must be provided for all of them, that means the same absolute value of the flag must be provided in the CHAN sheet of workbook conductor_input.xlsx. Function Check_INTIAL_values guarantees that this initialization is correctly done by the user.

Table 3.3-1 Effects on the fluid components initialization and on the application of the boundary conditions according to three possible values of the flag INTIAL, as far as fluid components are considered.

INTIAL	Initialization	Boundary condtions	Notes
± 1	p_{inl}	p_{inl}	
	T_{inl}	T_{inl}	Used as BC if $v_{inl} > 0$
	p_{out}	p_{out}	
	T_{out}	T_{out}	Used as BC if $v_{out} < 0$
± 2	$\dot{m}_{inl}$	v_{inl}	v_{inl} from $\dot{m}_{inl}$
	p_{inl}	T_{inl}	Used as BC if $v_{inl} > 0$
	T_{inl}	p_{out}	p_{out} evaluated in flow initialization
	T_{out}	T_{out}	Used as BC if $v_{out} < 0$
± 5	$\dot{m}_{inl}$	v_{inl}	v_{inl} from $\dot{m}_{inl}$
	T_{inl}	T_{inl}	Used as BC if $v_{inl} > 0$
	p_{out}	p_{out}	
	T_{out}	T_{out}	Used as BC if $v_{out} < 0$

The easiest initialization occurs when INTIAL is equal to ± 1 and the CHAN objects are not in hydraulic parallel. In this case all the objects can be initialized independently from the others. Since both the inlet and outlet pressure are known, the average pressure of the channel can be evaluated and used to compute the properties, density and dynamic viscosity, at the average pressure and inlet temperature. They are passed to the FluidComponents class method Comupute_velocity that computes the velocity reversing the equation of the hydraulic characteristic:

$$\Delta p = \frac{2 f \rho L v^2}{D_h} \qquad (3.3\text{-}1)$$

Moreover, this method returns the friction factor evaluated as function of Reynolds and geometry with the method Friction. The initial mass flow rate is determined with sign according to the relation of order among the inlet and outlet pressure.

If the objects are in hydraulic parallel it must be taken into account that both inlet and outlet pressure instantly equalize to the average value. Therefore, the average inlet and outlet pressure are evaluated and with these the average pressure of the channels:

$$p_{inl,ave} = \frac{\sum_i^{N_{ch,\parallel}} p_{inl,i}}{N_{ch,\parallel}} \tag{3.3-2}$$

$$p_{out,ave} = \frac{\sum_i^{N_{ch,\parallel}} p_{out,i}}{N_{ch,\parallel}} \tag{3.3-3}$$

$$p_{ave} = \frac{p_{inl,ave} + p_{out,ave}}{2} \tag{3.3-4}$$

Density and dynamic viscosities are evaluated at p_{ave} and $T_{inl,i}$ and with these parameters method Comupute_velocity is invoked to get the velocity and the friction factor of each channel; then the inlet mass flow rate is evaluated with the correct sign. Another important difference is that if $p_{inl,ave}$ and $p_{out,ave}$ differs from the values in input, the average values are used when BCs are applied.

Initialization performed with INITIAL ± 2 or ± 5 is not that much different when the channel is only in thermal contact or it is isolated. Since in both cases the inlet mass flow rate is given, the goal here is to find the value of the missing pressure, respectively the outlet pressure in the first case and the inlet pressure in the second one. The algorithm described in detail is the one used if flag is $|2|$, then the changes to deal with the other values are discussed.

An initial guess on the pressure drop is evaluated using the inlet properties exploiting the hydraulic characteristic equation (3.3-1) with the correct sign according to the inlet mass flow rate that can be positive or negative to simulate a back flow. At this point an iterative procedure starts, the outlet pressure is evaluated as:

$$p_{out} = p_{inl} - \Delta p_{old} \tag{3.3-5}$$

and it is exploited to evaluate the average pressure at which the properties (density, dynamic viscosity, Reynolds and friction factor) are recomputed to evaluate the new value of the pressure drop. The iterations end when the relative error is lower than the tolerance or the maximum number of iterations is reached.

If INTIAL is equal to ± 5 the initial guess on the pressure drop is evaluated at p_{out} and T_{inl} and in the iterative procedure the inlet pressure rather than the outlet one is computed at each iteration as:

$$p_{inl} = p_{out} + \Delta p \tag{3.3-6}$$

These evaluated values are also used as boundary conditions.

If the fluid components are in hydraulic parallel, the above iterative procedure can no longer be applied because the mass flow rate repartition according to the channel characteristic must be considered; as a matter of fact, it is not true in general, that the inlet mass flow rate provided by the user for the channels is the actual value. SC2 code computes them guaranteeing the respect of the physics and of the mass balance.

To compute the pressure-drop, the hydraulic characteristic of each channel is determined and it is approximated with a parable having vertex in the origin of the Cartesian reference frame (Δp, $\dot{m}$):

$$\Delta p_i = \Delta p = \alpha_i \dot{m}_{inl,i}^2 \;\; i = 1, \dots, N_{ch,\parallel} \tag{3.3-7}$$

with:

$$\alpha_i = \frac{2Lf_i}{D_{h,i}\Sigma_i^2 \rho_i} \;\; i = 1, \dots, N_{ch,\parallel} \tag{3.3-8}$$

This approximation is based on the hypothesis that the friction factor is weakly dependent from the velocity, and it is the more accurate the more turbulent the motion is, as can be seen from a Moody diagram.

If flag is equal to $|2|$ the average inlet pressure is evaluated to keep into account that in this case the pressure instantly equalize and this is exploited to compute the properties of each channel at $p_{inl,ave}$ and $T_{inl,i}$; else if INTIAL is $|5|$ the average outlet pressure is determined to get the fluid components properties at $p_{out,ave}$and $T_{inl,i}$.

At this point, the algorithm is the same in both cases and the α_i can be evaluated. The total inlet mass flow rate is given by:

$$\dot{m}_{inl,\parallel} = \sum_{i=1}^{N_{ch,\parallel}} \dot{m}_{inl,i} \tag{3.3-9}$$

Since the pressure drop is the same for all the channels of the group, results that:

$$\dot{m}_{inl,i} = \sqrt{\frac{\Delta p}{\alpha_i}} \tag{3.3-10}$$

Substituting in the equation of the total mass flow rate and isolating the Δp yields:

$$\Delta p = \left(\frac{\dot{m}_{inl,\parallel}}{\sum_{i=1}^{N_{ch,\parallel}} \alpha_i^{-\frac{1}{2}}} \right)^2 \tag{3.3-11}$$

Known the pressure drop of the channels, the real mass flow rate distribution can be evaluated exploiting equation (3.3-1) keeping into account the actual flow direction, then the check on the mass conservation is performed.

Finally, the missing pressure value is evaluated:

$$p_{out} = p_{inl,ave} - \Delta p \; if \; INTIAL = \pm 2 \tag{3.3-12}$$

$$p_{inl} = p_{out,ave} + \Delta p \; if \; INTIAL = \pm 5 \tag{3.3-13}$$

As usual this value is used also when the boundary conditions must be applied.

3.3.2 SOLID COMPONENTS INITIALIZATION

Flag INTIAL applies also to solid components, the allowed values are 0 and ± 1. In the first case solid components temperature spatial distribution is initialized basing on the initial fluid components temperature distribution. Two possibilities are contemplated:

1) solid components are in thermal contact with fluid components.
2) solid components are not in thermal contact with fluid components.

In the former case, the initial temperature spatial distribution is evaluated as the average weighted on the contact perimeters of the channels in contact with the solid components:

$$T_{scomp,ini} = \frac{\sum_{ch}^{N_{ch,tc}} P_{ch,scomp} T_{ch,ini}}{\sum_{ch}^{N_{ch,tc}} P_{ch,scomp}} \tag{3.3-14}$$

while in the latter it is assumed to be equal to the minimum temperature spatial distribution among the ones of the fluid components.

User can impose the initial temperature spatial distribution of the solid components setting the value of the flag equal to $|1|$: as usual positive value means that the values came from the main input file, negative value implies that the temperature initialization is loaded from a secondary input file. This last possibility allows more flexibility on the initialization; indeed, for INTIAL $+1$ the temperature is a linear distribution between inlet and outlet temperature given in input. This initialization can be done for all the solid components independently from the conductor topology, the thermal imbalance is reinstated during the transient evolution since the energy balance (3.3-15) must be guaranteed. The thermal energy deposited in the solid components at the initial time is disposed by the coolant flow, as a result the solid components temperature decrease while the outlet temperature of the coolant increase and their value is halfway from the initial temperature and the coolant inlet temperature.

$$\sum_{scomp=1}^{N_{scomp}} \int_0^L dx \, \Sigma_{scomp} \left[\left(\rho_{scomp} c_{p,scomp} T_{scomp} \right)_{t_{beg}} - \left(\rho_{scomp} c_{p,scomp} T_{scomp} \right)_{t_{end}} \right] = \sum_{ch=1}^{N_{ch}} \int_{t_{beg}}^{t_{end}} dt \, [(\dot{m}_{ch} \, w_{ch})_0 - (\dot{m}_{ch} \, w_{ch})_L] \tag{3.3-15}$$

The final task of function initialization is to call function Plot_properties with the keyword What set to "Initialization", that plots the fluid and solid components properties spatial distributions.

3.3.3 BCs APPLICATION AND PROBLEM SOLUTION

The problem solution is addressed at global level by the method Conductor_solution, where is placed the while loop in time. All the functions needed to achieve this goal are collected in module Transient_solution_functions.py. Until the end time of the simulation is reached a new time step for each conductor is evaluated with function Get_time_step, that will allow to use an adaptive time step. Then, for each conductor, the properties and external drivers are evaluated at the Gauss points calling the method Operation_conditions of class Conductor; these quantities are necessary to build the matrix elements and the known term vector of the system of equations. This is done in function Step, that keeps into account the selected method to integrate the ODE system with the named key "Method". As stated in the end of section 2.2.2, the final coefficient matrix of the system A_{sys} is a banded matrix and only the band, i.e. the not null diagonals, are evaluated and stored to decrease the computational time and save memory. Function Step non only limits to the matrix and known term vector construction, it also applies the BCs and solves the linear system of equations.

As far as the application of the boundary condition is concern, solid components are considered adiabatic, while for fluid components it is again ruled by flag INTIAL; it allows different sets of boundary conditions, provided the inlet temperature is assigned, together with one inlet and one outlet condition either on pressure or velocity. In case of a back flow, negative outlet velocity, the outlet temperature is also imposed. Table 3.3-1 also shows the values imposed on the known term vector for different flag values; on the coefficient matrix the column corresponding to the parameter is zeroed out apart from the row index that identifies the main diagonal, that is set equal to one.

The resulting linear system of equation is solved exploiting a method for banded matrices without pivoting. First, A_{sys} is reduced to a non-singular system of equations, then the backward substitution method is applied to get the solution. The solution array is subdivided into smaller arrays that collect the spatial distribution at the current time step of the unknowns, namely velocity, pressure and temperature for the fluid components and the temperature for the solid components that are used as initialization for a new computation.

During the loop, some spatial distributions and time evolutions are saved in suitable files. At the end of the transient, method Conductor_post_processing further elaborates these data making plots of the solution, spatial distributions at user required time steps and time evolutions at given spatial locations, as is deeply discussed in section 0.

3.3.4 SUMMARY

Class Simulation direct from the higher possible level all the steps of the simulation. Its constructor method reads the first of the main input files and saves its content in an attribute. Among this information the name of the next main input file to be red is found and thanks to it the instantiation of class Conductors can be done within method Conductor_instantiation.

After that, all the user defined cables are instantiated, they are initialized invoking method Conductor_initialization. In this section attention was paid in describing the effects of the conductor topology and the given input values, according to INTIAL flag value, on the fluid components initialization, as well as the possibility foreseen for the solid component initializations.

The while loop to solve the transient can be found in method Conductor_solution and at each new time step it calls the Step function, one of the most important functions of all the code that builds the coefficient matrix of the system and the known term vector exploiting as much as possible the array smart notation, considering the numerical scheme chosen to march in time, applies the BC and solves the resulting linear system of equations.

Finally, a rough post processing of the data is performed calling Conductor_post_processing method.

3.4 A USER-FRIENDLY GUI

SC2 graphical user interface is developed in module SC2_GUI. It is written exploiting the python standard library tkinter and it is based on the python class concept. Even if it is still in an embryonic phase of the realization, it allows user to easily manage the essentials actions to run a simulation. It consists of two parts, on the one hand the main window to deal with the simulation, on the other hand the real time plots of selected variables.

The GUI main window consists of a menu bar with five so called cascades as shown in *Figure 3.4-1*, whose options are briefly explained in Table 3.4-1.

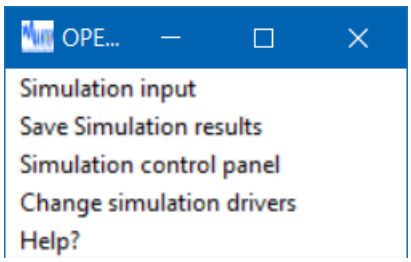


Figure 3.4-1 SC2 Graphical User Interface main window; menu bar composed by five cascades.

To each option of the cascade correspond one or more class methods that actually make the selected actions. As an example, Load_input_file opens the window that allows to select the folder where are stored the input file for the simulation; Create_directories and Open_existing_directories respectively open the window to create a new file or an existing folder in which save the simulation results; finally Run_simulation is the method that collects the cited Simulation class methods that manages all the steps of the simulation.

Table 3.4-1 List of all the available options in the GUI distinguishing from the already and not yet implemented. The table also provides a succinct description of the options.

Cascade	Option	Implemented	Description
Simulation input	Load input data	yes	Allows user to select the directory where the input flies for the simulation are saved.
Save Simulation results	Create main directory	yes	Allows user to create a new main folder within directory Simulation_results in which to save the results of a simulation set.
	Open main directory	yes	Allows user to save the result of a simulation in an already existent main folder within directory Simulation_results.
Simulation control panel	Run	yes	When clicked the simulation begins.
	Pause	no	If clicked the simulation pauses.
	Continue	no	Allows to continue the simulation once it was paused.
	Stop	no	Arrest the simulation but does not close the main window
	Close	yes	Closes the simulation main window killing the simulation if in progress.
Simulation drivers	Change drivers	no	Allows to change the drivers of the simulation while in progress to correct some unexpected behavior.
Help?	User guide: input files	yes	Link to a pdf file that collects detailed information on the input file compilation.

3.4.1 RUN…

To give a sample of the ease of interaction with the interface, this section consists of a short tutorial on how to run a simulation, provided the input files are correctly filled by the user.

The first step is to open the code with a text editor and run the Simulation_starter.py module or execute it from a terminal. As mentioned in section 3.1 this module makes an instance of class SC2_GUI that results in the pop up of the GUI main window (see *Figure 3.4-1*).

3.4 A USER-FRIENDLY GUI

At this point there are only two available options, the “Load simulation input data” in cascade “Simulation input” and the help guide for the input files compilation. To load the input files, click on the cascade “Simulation input” and select the only available options as shown in *Figure 3.4-2*.

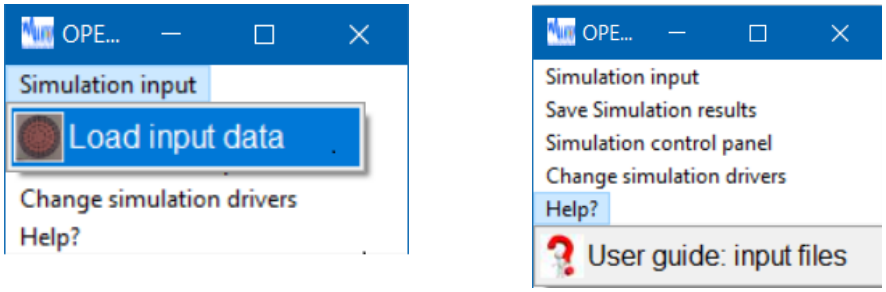


Figure 3.4-2 GUI menu bar cascades options available at the beginning of the simulation: (left) command “Load input data” allows to read and load all the user defined input files; (right) “User guide: input files” gives support in the input file compilation.

A window will pop up and it lists the folders within directory Description_of_components; users should select the folder where the compiled input files for the current simulation will be saved, for example test_ITER_TF_heating as in the following *Figure 3.4-3*.

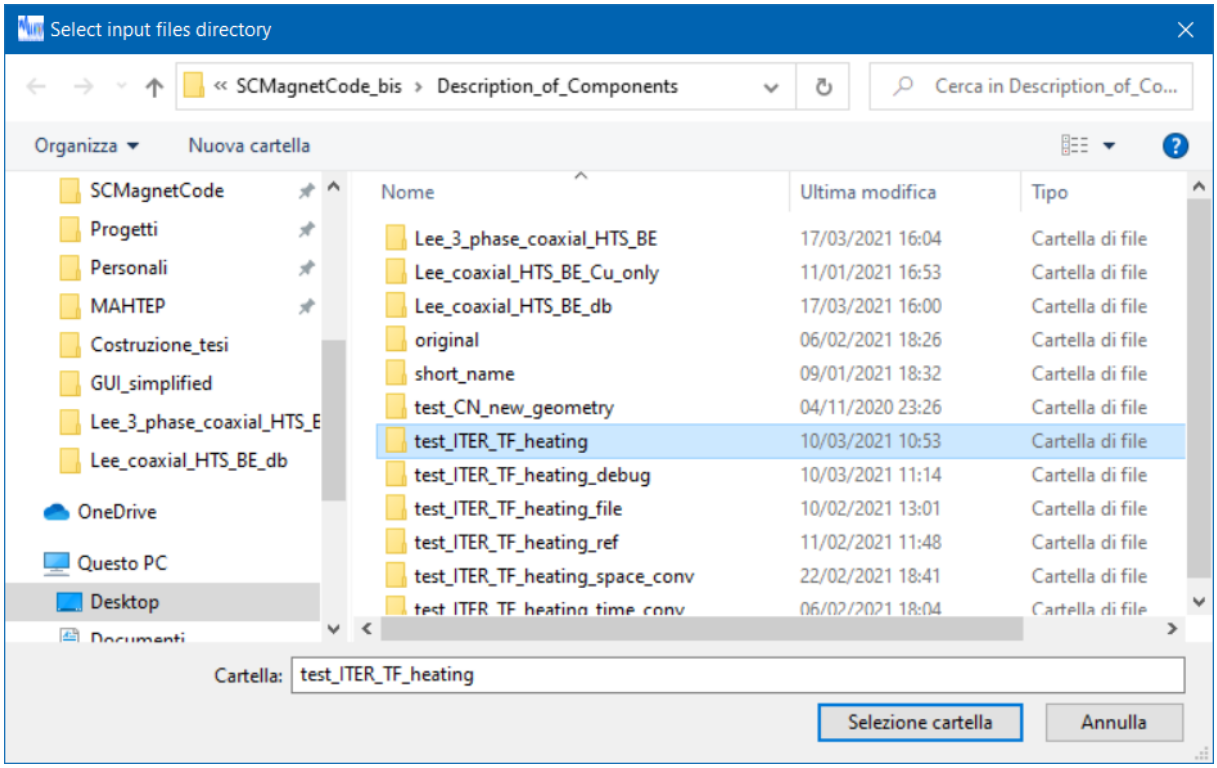


*Figure 3.4-3 Dialog windows that pop ups when the command “Load input data” is selected by the user. It allows to select from the folder that collects the different sets of input files (*Description_of_Components*) the one edited for the current simulation.*

Once the folder is selected the option Load simulation input data in cascade "Simulation input" is no longer available (Figure 3.4-4), however both the options of the second cascade are now clickable.

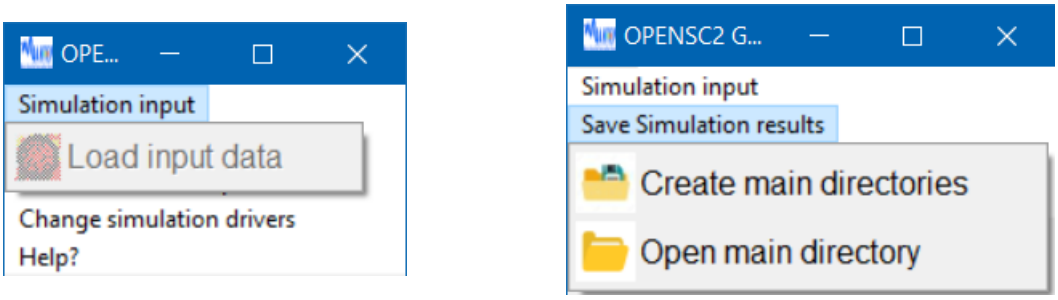


Figure 3.4-4 After the selection of the input folder: (left) command "Load input data" is no longer available; (right) possible choices for saving the output, new main directory or already existing main directory.

The user could decide to save the simulation results in a new main folder of the Simulation_results directory or to store them in an already existing main folder. In the former case, the option "Create simulation main directory" can be selected, in the latter the command "Open simulation main directory" should be considered. and Figure 3.4-6 shows the windows that will be opened whether either alternative is selected.

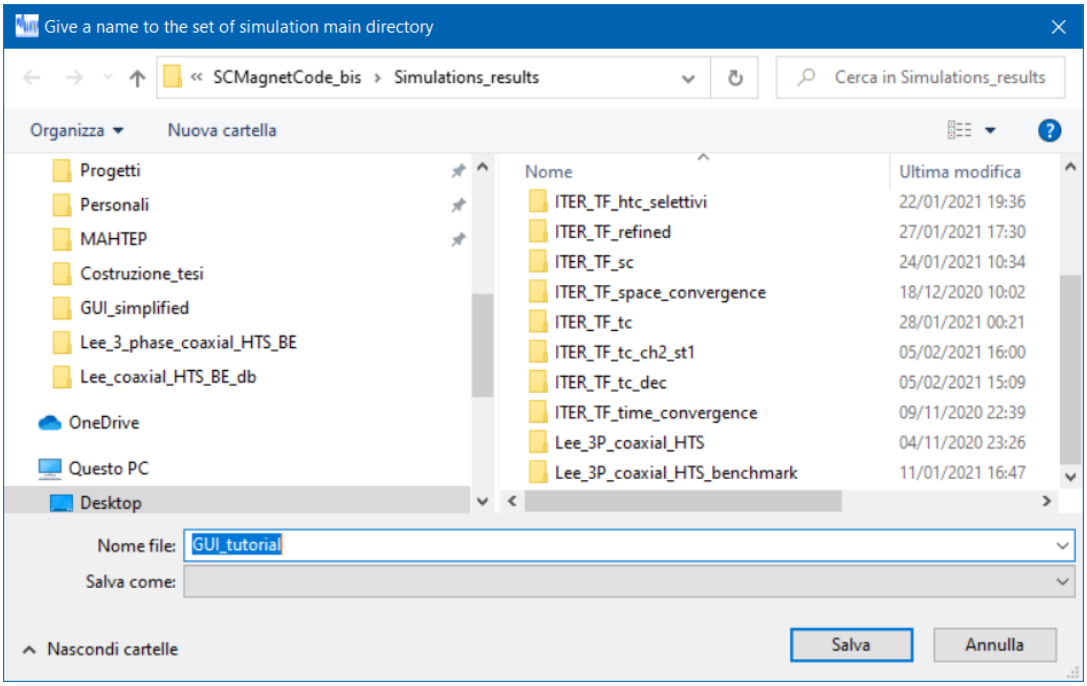


*Figure 3.4-5 Dialog windows that pop ups when the command "Create main directory" is selected by the user. It allows to create a new main directory where to save the outcome of the simulations (both .tsv files and .eps figures) directly in the folder that gathers all the outcomes (*Simulation_results*).*

3.4 A USER-FRIENDLY GUI

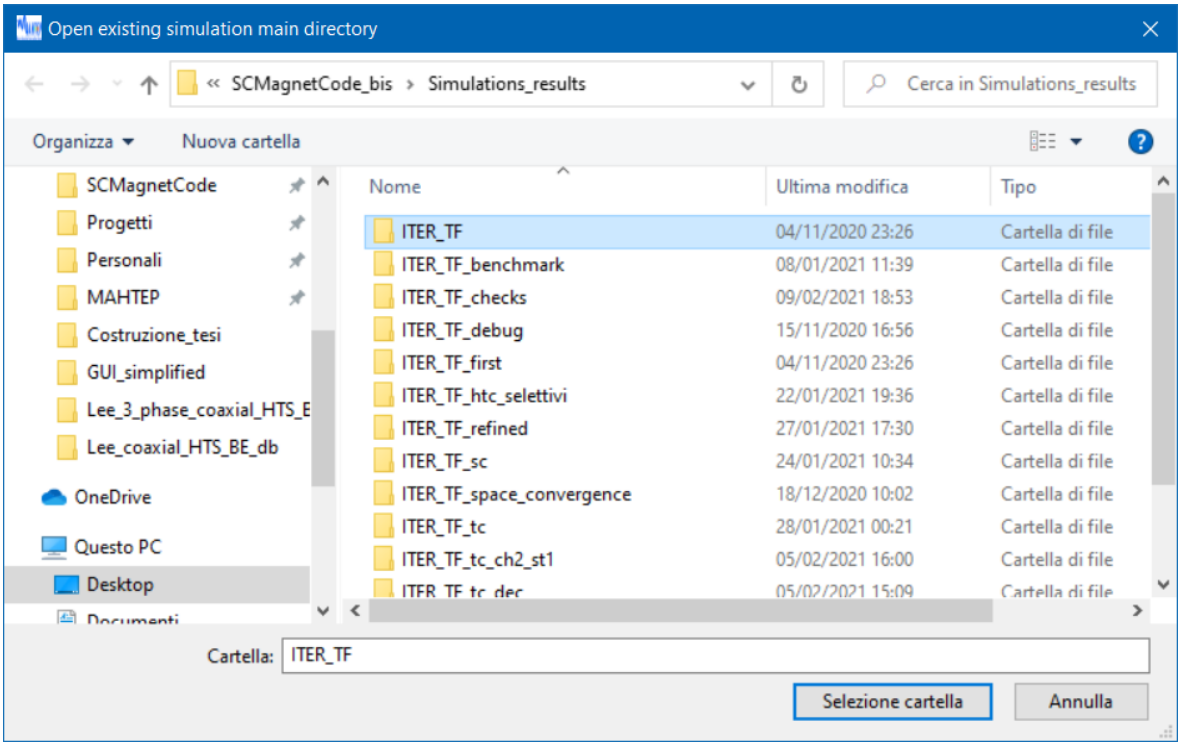


*Figure 3.4-6 Dialog windows that pop ups when the command "Open main directory" is selected by the user. It allows to open an already existent main directory where to save the outcome of the simulations (both .tsv files and .eps figures) directly in the folder that gathers all the outcomes (*Simulation_results*).*

The first choice allows to create a new folder inside directory Simulation_results as shown in ; the second choice opens a window that shows the list of already created folders in the directory.

In this tutorial the first approach is selected, the new folder where the outcome of the simulation will be saved is called GUI_tutorial. Now, since the code knows where to store the simulation results, commands "Create simulation main directory" and "Open simulation main directory" are no longer available in the cascade "Save Simulation results". User should consider the third cascade and press the only clickable option ("Run") to launch the simulation as shown in Figure 3.4-8.

An information box, that inform the user that the simulation is launched, will open; click "Ok" to close the box (Figure 3.4-7). At the end of the simulation and of the default post processing another window similar to the previous one informs that the whole procedure has been completed successfully; again, click the "Ok" button to close the window.

To close the GUI, select the command "Close" in the "Simulation control panel" cascade as shown in *Figure 3.4-9*, the only one still available except the help guide at this point. Whit this last action the simulation session ends.

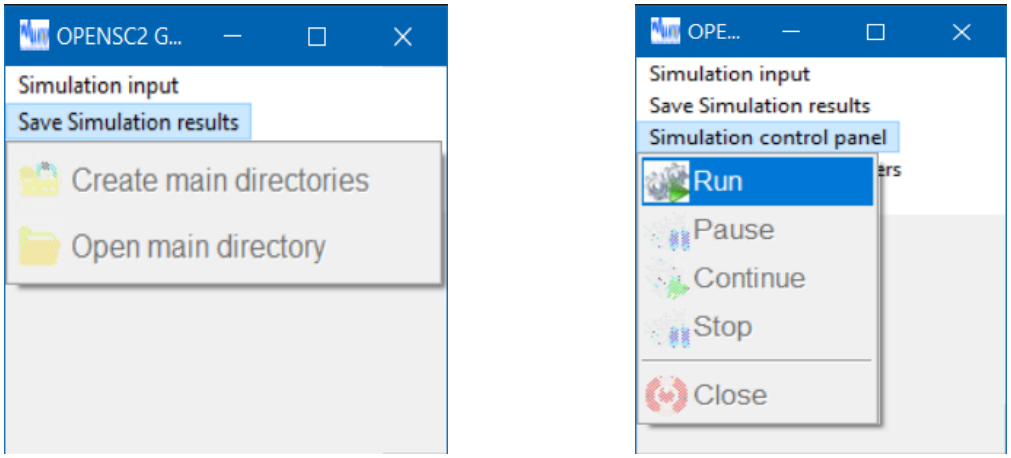


Figure 3.4-8 After deciding where to save the simulation results: (left) both commands in "Save Simulation results" cascade are no longer available; (right) command "Run" in the "Simulation control panel" to start the simulation.

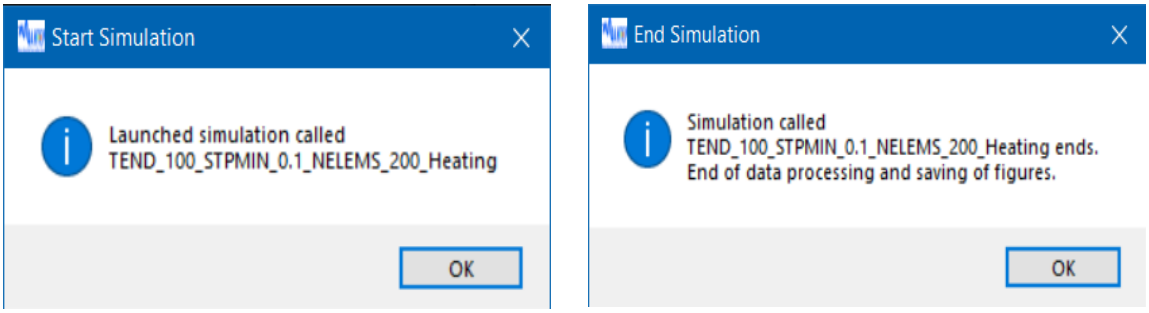


Figure 3.4-7 Information box: (left) after launching the simulation, a window informs of the start of the simulation showing its name, clicking the Ok button the simulation begins; (right) at the end of the simulation a windows shows up confirming the correct execution of the simulation and the default post-processing of the data.

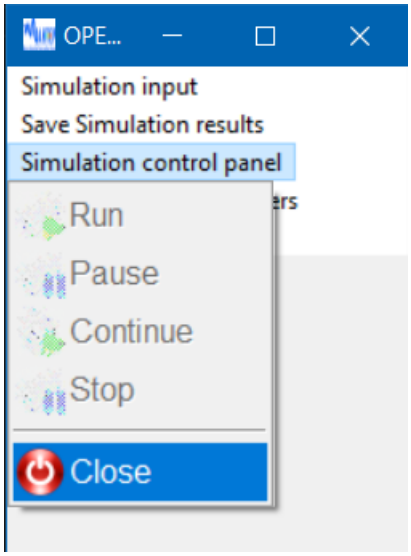


Figure 3.4-9 To close the simulation session, select the command "Close" form the "Simulation control panel" cascade.

3.4.2 …AND CHECK (REAL-TIME VISUALIZATION OF THE RESULTS)

One of the innovative features of the code is the *Run-AND-check* paradigm compared to the Run-THEN-check used in other code [48]–[50]. This new concept is achieved with the combination of two options in the GUI: the possibility of change the driver as the simulation progresses and the real time visualization of the results. As cited at the beginning of section 3.4, the former is not yet available, however the latter is already reality. After that, the command "Run" is clicked to start the simulation and the information window is closed, a group of figure appears that shows the time evolution of the maximum temperatures of fluid components as well as their inlet and outlet mass flow rate, and solid components maximum temperature. All these plots update at each new time step, so they allow to have an idea on what is going on during the simulation, for instance if there are some unexpected oscillations or if odd values of temperatures are obtained during the transient. Exploiting the information provided by these figures, the user should be able to evaluate if change the drivers to correct the simulation outcome.

3.4.3 SUMMARY

The graphical interface of the code, although in an embryonic state, allows the user to execute the basic instructions for launching the simulation and to keep its progress under control, thanks to the real-time graphs of some relevant parameters such as maximum temperatures and inlet and outlet mass flow rates.

It assumes the form of an interactive menu bar composed of five cascades for a total of ten commands, not all yet available. The path to launch a simulation is straightforward and it is described in section 3.4.1

3.5 EASE OF POST PROCESSING

The outcome of a generic simulation is a bunch of data that must be stored, organized and eventually further elaborated to get the desired results. This digression extends and completes what was introduced in section 0 about the folder organization. Specifically, the first section clarifies how the output of a single simulation and of a set of simulations is grouped and exhibited to user; the second and the last sections deal with the data postprocessing, respectively describing the default and the advanced post processing main features.

3.5.1 HOW THE DATA ARE MANAGED

The processing of the output of the simulation is twofold according that it is used to produce basic plots or to perform further analyzes. In this thesis, the first is referred as *default post processing* while the latter as *advanced post processing* and they will be addressed in the following two sections respectively; the current one explains how the data are organized and stored by the SC2 code.

The above cited classification is not the only possible one; the program is able to distinguish if a data is saved by default or because it is asked by the user, as well as the kind of data saved (if it is a spatial distribution or a time evolution of the considered properties).

Before diving deeper into these details, the main input file conductor_diagnostic.xlsx is further explained. As mentioned in section 0, the workbook is composed by sheets Space and Time that respectively allow user to set, for each defined cable, times (sheet Space) and coordinates (sheet Times) at which spatial distributions and time evolutions are saved. An arbitrary number of independent values can be provided by user for both the sheets; moreover, it is not mandatory although recommended, that the values are sorted. Initial and final spatial distributions together with inlet and outlet time evolutions are automatically saved by the code, so user cannot enter these sensors. The limit case of empty sheets is foreseen and, in this case, only the initialization and the spatial distribution at the end of the simulation are saved; analogously the inlet and outlet time evolutions are stored by default.

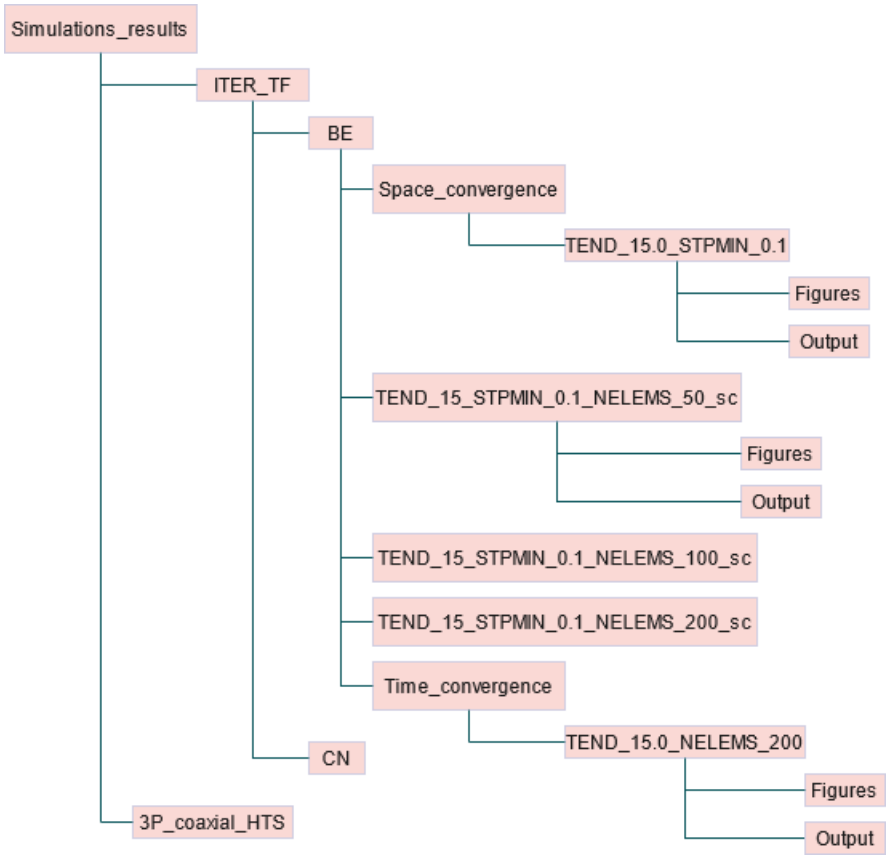


Figure 3.5-1 Tree of the folder Simulation_result.

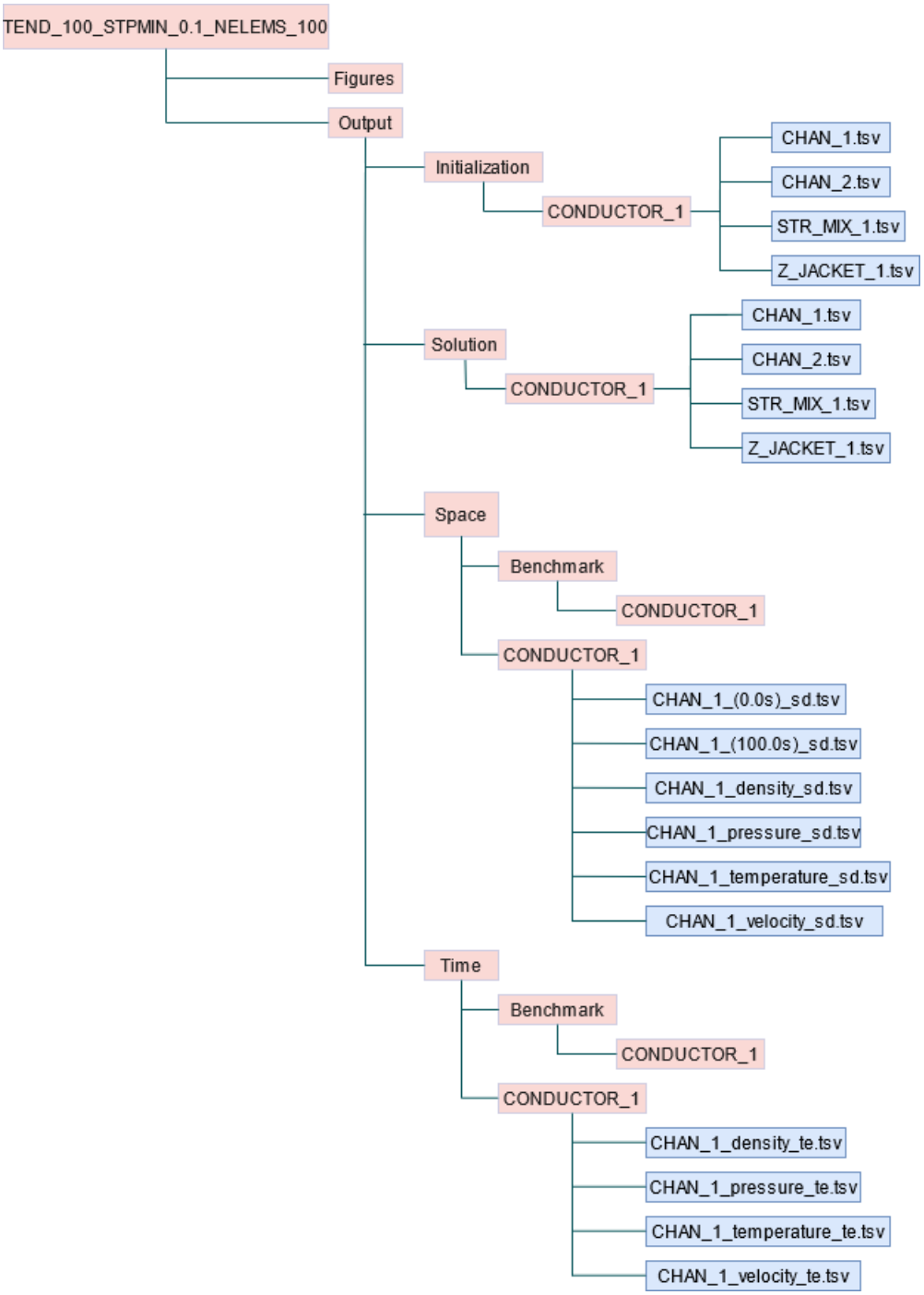


Figure 3.5-2 Tree of the Output subfolder. Notice the recurrent inner structure of the tree.

The general structure of the directory Simulations_results is shown in Figure 3.5-1. For the time being, let's focus on a folder that collects the outcome of a single simulation, such as TEND_100_STPMIN_0.1_NELEMS_100 in Figure 3.5-2. The Output sub-directory groups four folders: Initialization, Solution, Space and Time. Two of them are devoted to store the data required with conductor_diagnostic.xlsx, respectively Space for the spatial distributions and Time for the time evolutions. Both these folders share the same organization, as shown in Figure 3.5-2: they collect a subfolder for each defined cable, named with the conductor ID,

and the Benchmark subfolder that will be discussed later. The Tab-Separated Value (TSV, extension .tsv) format is chosen to save the data since it is supported by Pandas and uses a not ambiguous separator as opposed to the Comma Separated Value (CSV) extension, however it is not the smarter solution as will be discussed in the final chapter 5. In general, all files are equipped with a header that uniquely identify their content; they are created only if the basic conductor component object of a specific kind is defined to model the cable.

As far as spatial distribution is considered, the saving procedure consists of two steps:

1. for each time in input, SC2 code calls the function Save_simulation_space that writes files that collects the conductor spatial discretization and the relevant properties; Table 3.5-1 reports for each basic component the adopted naming scheme and the properties stored. These data are saved during the transient solution.
2. At the end of the transient a single file for each relevant property is generated in which the spatial distribution at all the given times is saved. This reshape is automatically performed by the code thanks to function Reorganize_spatial_distribution, exploiting pandas DataFrame features and allows a more manageable data processing. Further details can be obtained from Table 3.5-1.

A single file called xcoord.tsv collects the conductor spatial discretization at all the required times since, if the mesh is adaptive, it may change from one time step to another.

Time evolutions are saved in files that are organized similarly to the reorganized file of the spatial distribution described above. For each component and for its relevant property, a file that collects the time evolutions at the given spatial coordinates together with the time steps is created by function Save_simulation_time that writes and updates files as shown in Table 3.5-2.

3.5 EASE OF POST PROCESSING

Table 3.5-1 Naming scheme and content of the files saved to store the spatial distributions of variables at the user defined time steps. Both formats are detailed.

Object	Name scheme	Example	Content
CHAN	CHAN.ID_([t]s)_sd.tsv	CHAN_1_(5.0s)_sd.tsv	Spatial discretization, channel velocity, pressure, temperature and density spatial distributions at user required time = t (ex 5.0 s).
	CHAN.ID_density_sd.tsv	CHAN_1_density_sd.tsv	Channel density spatial distributions at all the user required time steps.
	CHAN.ID_pressure_sd.tsv	CHAN_1_pressure_sd.tsv	Channel pressure spatial distributions at all the user required time steps.
	CHAN.ID_temperature_sd.tsv	CHAN_1_temperature_sd.tsv	Channel temperature spatial distributions at all the user required time steps.
	CHAN.ID_velocity_sd.tsv	CHAN_1_velocity_sd.tsv	Channel velocity spatial distributions at all the user required time steps.
STR_MIX	STR_MIX.ID_([t]s)_sd.tsv	STR_MIX_1_(5.0s)_sd.tsv	Spatial discretization and mix strand temperature spatial distribution at user required time = t (ex 5.0 s).
	STR_MIX.ID_temperature_sd.tsv	STR_MIX_1_temperature_sd.tsv	Mix strand temperature spatial distributions at all the user required time steps.
STR_SC	STR_SC.ID_([t]s)_sd.tsv	STR_SC_1_(5.0s)_sd.tsv	Spatial discretization and superconductor strand temperature spatial distribution at user required time = t (ex 5.0 s).
	STR_SC.ID_temperature_sd.tsv	STR_SC_1_temperature_sd.tsv	Superconductor strand temperature spatial distributions at all the user required time steps.
STR_STAB	STR_STAB.ID_([t]s)_sd.tsv	STR_STAB_1_(5.0s)_sd.tsv	Spatial discretization and stabilizer strand temperature spatial distribution at user required time = t (ex 5.0 s).
	STR_STAB.ID_temperature_sd.tsv	STR_STAB_1_temperature_sd.tsv	Stabilizer strand temperature spatial distributions at all the user required time steps.
Z_JACKET	Z_JACKET.ID_([t]s)_sd.tsv	Z_JACKET_1_(5.0s)_sd.tsv	Spatial discretization and jacket temperature spatial distribution at user required time = t (ex 5.0 s).
	Z_JACKET.ID_temperature_sd.tsv	Z_JACKET_1_temperature_sd.tsv	Jacket temperature spatial distributions at all the user required time steps.

Table 3.5-2 Naming scheme and content of the files saved to store the time evolutions of variables at user defined spatial coordinates.

Object	Name scheme	Example	Content
CHAN	CHAN.ID_density_te.tsv	CHAN_1_density_te.tsv	Time steps and channel density time evolutions at all user required spatial locations.
	CHAN.ID_inlet_outlet_te.tsv	CHAN_1_inlet_outlet_te.tsv	Time steps and channel inlet and outlet velocity, pressure, temperature density and mass flow rate time evolutions.
	CHAN.ID_pressure_te.tsv	CHAN_1_pressure_te.tsv	Time steps and channel pressure time evolutions at all user required spatial locations.
	CHAN.ID_temperature_te.tsv	CHAN._1_temperature_te.tsv	Time steps and channel temperature time evolutions at all user required spatial locations.
	CHAN.ID_velocity_te.tsv	CHAN_1_velocity_te.tsv	Time steps and channel velocity time evolutions at all user required spatial locations.
STR_MIX	STR_MIX.ID_B_field_te.tsv	STR_MIX_1_B_field_te.tsv	Time steps and mix strand magnetic field time evolutions at all user required spatial locations.
	STR_MIX.ID_T_cur_sharing_te.tsv	STR_MIX_1_T_cur_sharing_te.tsv	Time steps and mix strand current sharing temperature time evolutions at all user required spatial locations.
	STR_MIX.ID_temperature_te.tsv	STR_MIX_1_temperature_te.tsv	Time steps and mix strand temperature time evolutions at all user required spatial locations.
STR_SC	STR_SC.ID_B_field_te.tsv	STR_SC_1_B_field_te.tsv	Time steps and superconductor strand magnetic field time evolutions at all user required spatial locations.
	STR_SC.ID_T_cur_sharing_te.tsv	STR_SC_1_T_cur_sharing_te.tsv	Time steps and superconductor strand current sharing temperature time evolutions at all user required spatial locations.
	STR_SC.ID_temperature_te.tsv	STR_SC_1_temperature_te.tsv	Time steps and superconductor strand temperature time evolutions at all user required spatial locations.
STR_STAB	STR_STAB.ID_B_field_te.tsv	STR_STAB_1_B_field_te.tsv	Time steps and stabilizer strand magnetic field time evolutions at all user required spatial locations.

Object	Name scheme	Example	Content
	STR_STAB.ID_temperature_te.tsv	STR_STAB_1_temperature_te.tsv	Time steps and stabilizer strand temperature time evolutions at all user required spatial locations.
Z_JACKET	Z_JACKET.ID_temperature_te.tsv	Z_JACKET_1_temperature_te.tsv	Time steps and jacket temperature time evolutions at all user required spatial locations.

From Table 3.5-2 results that files named CHAN.ID_inlet_outlet_te.tsv collect channels inlet and outlet velocity, pressure, temperature, density and mass flow rate. This is another example of default data saving and it is contemplated to better handle channels inlet and outlet properties.

Both the Space and Time folders contain the subfolder Benchmark. In this directory the user should save the files of the spatial distributions and the time evolutions he/she gets from other codes in order to perform the outer benchmark. Typically, there is only one more level that coincides with the folders named with the CONDUCTOR.ID where all the useful files should be saved, as can be seen from Figure 3.5-2.

The other two Output sub-directories, Initialization and Solution, are the locations where the code saves the simulation initialization and the final solution respectively. These two folders have the same organization: they collect the same number of folders as the number of defined cables, named with the CONDUCTOR.ID, in which files with the first and last obtained spatial distributions are stored respectively: indeed, the properties stored in these files are the same, the difference being the time at which they are evaluated, namely $t = t_{beg} = 0\ s$ in folder Initialization and $t = t_{end}\ s$ in folder Solution. The file content is detailed in Table 3.5-3 while an example of tree is again in Figure 3.5-2. The function in charge of this is Save_properties.

To certify the accuracy of the solution obtained with a simulation, typically space and time convergence analyzes should be accomplished, and in this case, several simulations are run that only differs for a single input value apart from all the others. Specifically, the space convergence is performed changing the number of elements of the spatial discretization while for the time convergence some orders of magnitude of the time step are explored. For these reasons, both analyzes define a *set of simulations*. Inside the folder that represents the chosen method for the ODE system solution (Figure 3.5-1), there are collected all the simulations performed with that method together with the folders Space_convergence and Time_convergence. These are created by default by SC2 code and they are devoted to save and organize the data useful for space and time convergence analyzes, respectively. They share a similar structure that will be described here; while reading, refer to that expand the tree of these folders.

Table 3.5-3 Naming scheme and content of the files that stores the initial and the final spatial distributions of the variables.

Object	Name scheme	Example	Content
CHAN	CHAN.ID.tsv	CHAN_1.tsv	Conductor spatial discretization and channel initial/final spatial distributions of velocity, pressure, temperature, density, enthalpy, entropy, thermal conductivity, Grunaisen, speed of sound, specific heat at both constant pressure and volume, Reynolds and Prandtl dimensionless number together with mass flow rate.
STR_MIX	STR_MIX.ID.tsv	STR_MIX_1.tsv	Conductor spatial discretization and mix strand initial/final magnetic field, temperature and current sharing temperature spatial distributions.
STR_SC	STR_SC.ID.tsv	STR_SC_1.tsv	Conductor spatial discretization and superconductor strand initial/final magnetic field, temperature and current sharing temperature spatial distributions.
STR_STAB	STR_STAB.ID.tsv	STR_STAB_1.tsv	Conductor spatial discretization and stabilizer strand initial/final magnetic field and temperature spatial distributions.
Z_JACKET	Z_JACKET.ID.tsv	Z_JACKET_1.tsv	Conductor spatial discretization and jacket initial/final temperature spatial distribution.

3.5 EASE OF POST PROCESSING

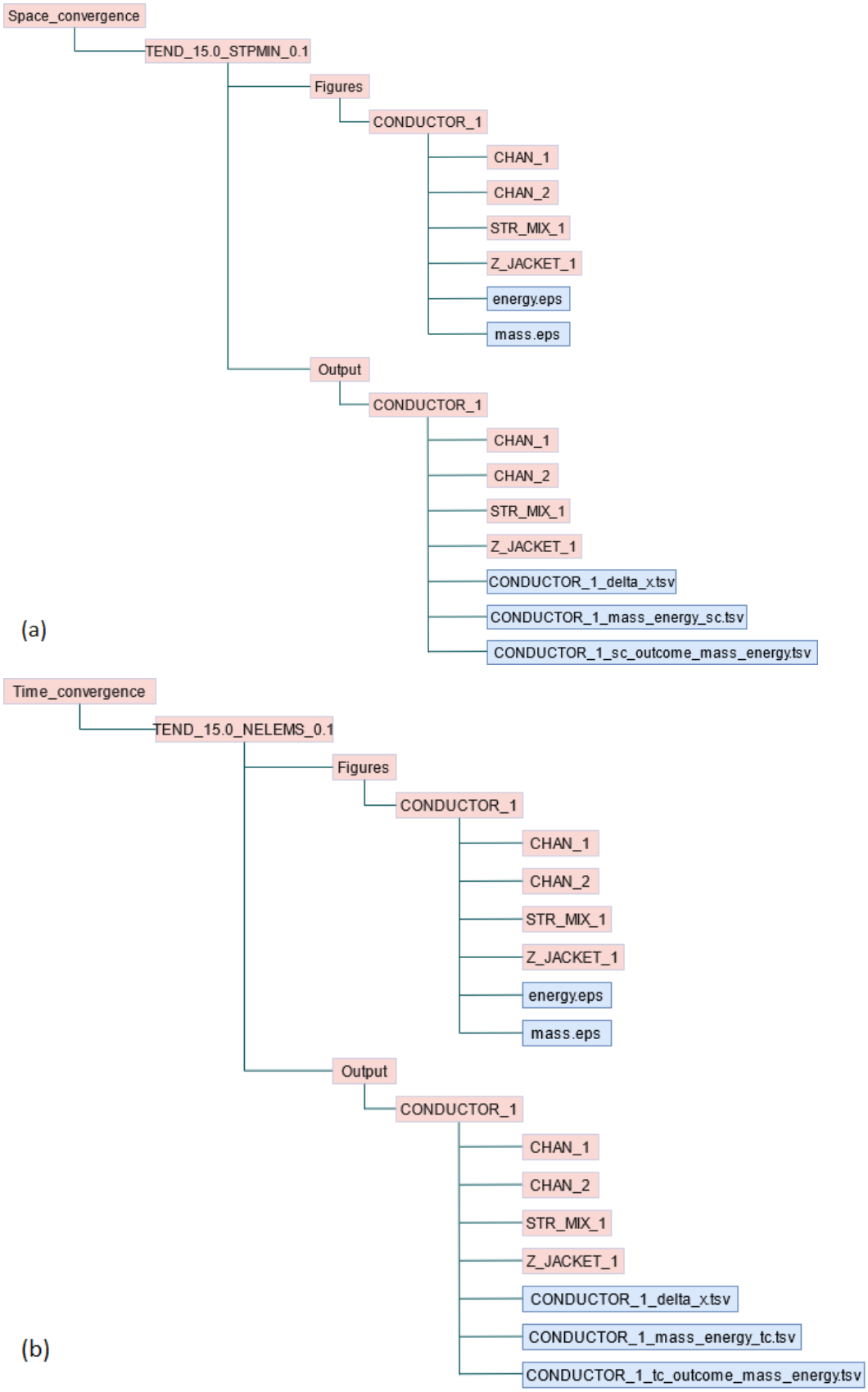


Figure 3.5-3 Tree of the convergence analyzes subfolders: (a) Space_convergence; (b) Time_convergece. The structure of the trees is symmetric.

The folder Space_convergence groups folders that are related to the set of simulations used to perform the space convergence study. Each folder name is built combining the end time of the simulation and the minimum value of the time step, as shown in . Within this folder there are the well-known Output and Figures sub-folders that are used to save the outputs of the simulations and the plots. Beyond the usual folders, the tree shows three files with the extension tsv. The first file, CONDUCTOR.ID_delta_x.tsv, stores both the number of elements and the discretization parameters used to make the space convergence analysis; the second is called CONDUCTOR.ID_mass_energy_sc.tsv and here are saved the values of the global mass and energy balances on the conductor, together with the number of elements and the discretization parameters. These balances are performed at the end of the simulation invoking the method Mass_energy_balance of class Conductors:

$$\Delta m = \Delta t \left(\sum_{i}^{N_{ch}} \dot{m}_{i,inl} - \sum_{i}^{N_{ch}} \dot{m}_{i,out} \right) \tag{3.5-1}$$

$$\Delta E = \Delta t \left[\sum_{i}^{N_{ch}} \dot{m}_{i,inl} \left(w_{i,inl} + \frac{v_{i,inl}^2}{2} \right) - \sum_{i}^{N_{ch}} \dot{m}_{i,out} \left(w_{i,out} + \frac{v_{i,out}^2}{2} \right) \right] \tag{3.5-2}$$

In the last equation the contribution of the potential energy (gz) is neglected.

Inside the other folders the spatial distribution of the solution at the end of the simulation is saved. The naming scheme and the content of the files is shown in Table 3.5-4.

Table 3.5-4 Naming scheme and content of the files saved to store the solution spatial distribution at the end of the simulation to perform the space convergence analysis.

Object	Name scheme	Example	Content
CHAN	CHAN.ID_(N+1).tsv	CHAN_1_(100).tsv	Channel velocity, pressure and temperature spatial distributions at TEND.
STR_MIX	STR_MIX.ID_(N+1).tsv	STR_MIX_1_(100).tsv	Mix strand temperature spatial distribution at TEND
STR_SC	STR_SC.ID_(N+1).tsv	STR_SC_1_(100).tsv	Superconductor strand temperature spatial distribution at TEND
STR_STAB	STR_STAB.ID_(N+1).tsv	STR_STAB_1_(100).tsv	Stabilizer strand temperature spatial distribution at TEND
Z-JACKET	Z_JACKET.ID_(N+1).tsv	Z_JACKET_1_(100).tsv	Jacket temperature spatial distribution at TEND

The architecture of folder Time_convergence is dual with respect to folder Space_convergence. The first difference is that the subfolder names are obtained joining the end time of the simulation and the number of elements used to perform the spatial discretization. The inner structure is very similar, starting from the presence of Output and

Figures folders. In the former, each cable has its own folder that is organized as described above, furthermore there are the three files analogous to the previous case. File CONDUCTOR.ID_mass_energy_tc.tsv stores the time steps used for the analysis and the values of mass and energy balance evaluated with the different time steps according to equations (3.5-1) and (3.5-2) at the end of the simulations. The time steps used to perform the analysis are stored in file CONDUCTOR.ID_delta_t.tsv while for each conductor basic components the values are organized as explained in the following Table 3.5-5.

Both folders Space_convergence and Time_convergence are supervised by function Save_convergence_data, that is able to deal with the similarities and the differences that characterize space and time convergence analyzes. This function does not create files called CONDUCTOR.ID_sc_outcome_mass_energy.tsv and CONDUCTOR.ID_tc_outcome_mass_energy.tsv that will be addressed in section 3.5.3.

Table 3.5-5 Naming scheme and content of the files saved to store the spatial distribution of the solution at the end of the simulation to perform the time convergence analysis.

Object	Name scheme	Example	Content
CHAN	CHAN.ID_(STPMINs).tsv	CHAN_1_(0.1s).tsv	Channel velocity, pressure and temperature spatial distributions at TEND.
STR_MIX	STR_MIX.ID_(STPMINs).tsv	STR_MIX_1_(0.1s).tsv	Mix strand temperature spatial distribution at TEND.
STR_SC	STR_SC.ID_(STPMINs).tsv	STR_SC_1_(0.1s).tsv	Superconductor strand temperature spatial distribution at TEND.
STR_STAB	STR_STAB.ID_(STPMINs).tsv	STR_STAB_1_(0.1s).tsv	Stabilizer strand temperature spatial distribution at TEND.
Z-JACKET	Z_JACKET.ID_(STPMINs).tsv	Z_JACKET_1_(0.1s).tsv	Jacket temperature spatial distribution at TEND.

3.5.2 DEFAULT POST PROCESSING

The default post processing consists of creating plots of the properties stored for the single simulation, organized as described above. These are stored in the Figures subfolder that has the same structure of the Output one. Indeed, also in this directory there are the four folders Initialization, Solution, Space and Time that respectively collects the figures of the initialization, the solution, the spatial distribution and the time evolution. The basic structure of all these folders coincides with the group of folders named with the basic components ID collected within the CONDUCTOR.ID folder, as shown in Figure 3.5-4. The outer benchmark outcomes are saved in sub-folder Benchmark of Space and Time. The figure name

corresponds to the plotted properties and the selected format is the vectorial Encapsulated PostScript (EPS).

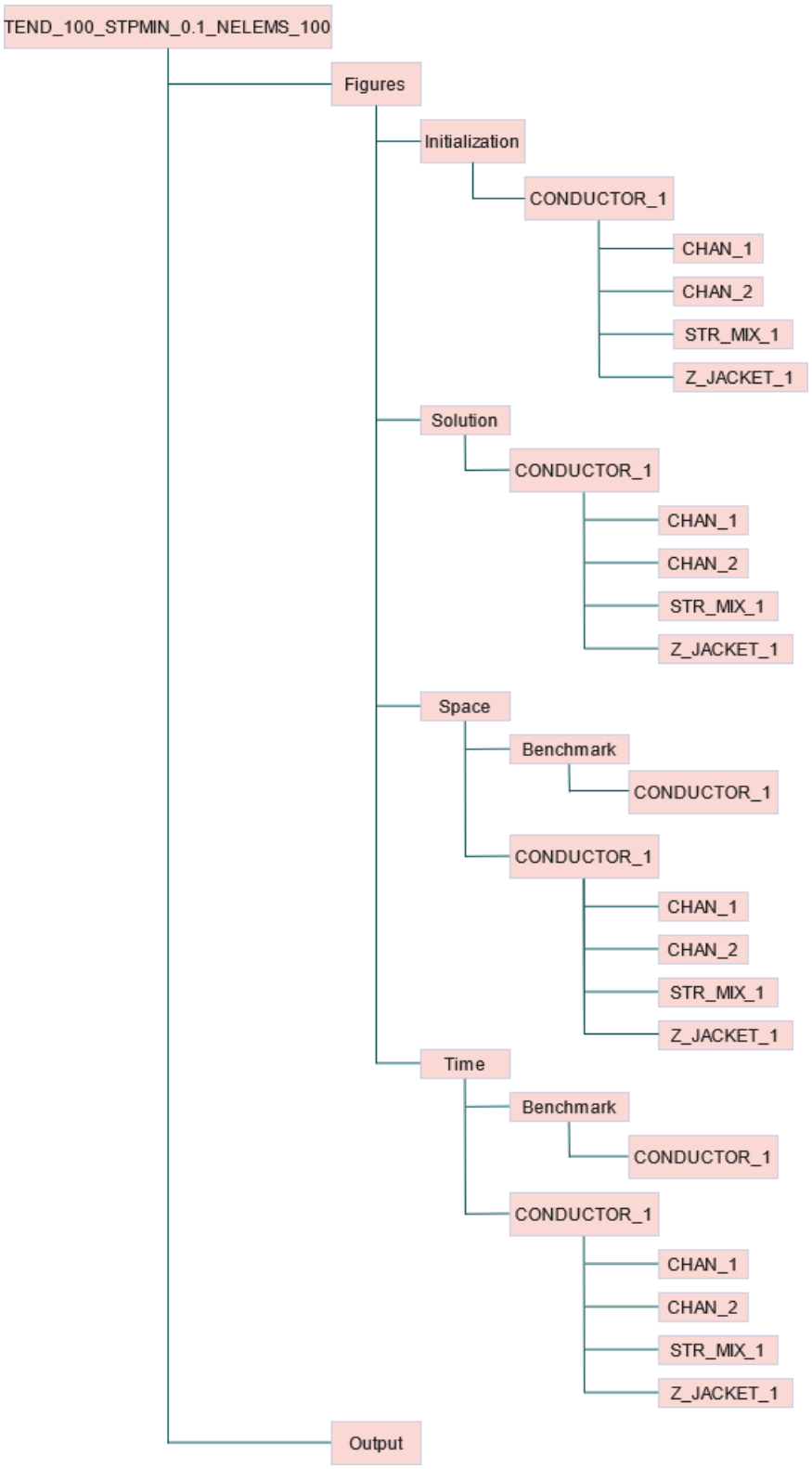


Figure 3.5-4 Tree of the subfolder Figures. Notice the recurrent inner structure of the tree.

The figures in Initialization and Solution folders are standard since in each figure a single curve is plotted, corresponding to the initial or final spatial distribution of the properties listed in. These figures are automatically realized by the code thanks to function Plot_properties, regardless of user input and can be useful to check if the simulation input data are correct and if the corresponding solution is reasonable.

The management of the spatial distributions and time evolutions is less straightforward because, as mentioned, user in conductor_diagnostic.xlsx can prescribe an arbitrary number of times and spatial locations; therefore, making a single figure for each value not is not practical to understand and compare the outcomes. Basing on these considerations, the spatial distributions and time evolutions plots are automatically grouped together into multi-axes figures exploiting the tools of matplotlib python library. Each figure can show at most twenty curves subdivided into no more than four horizontal subplots, that means that a maximum of five curves can be shown for each subplot. Subplots are filled with curves in such a way that their distribution is as uniform as possible to improve the readability of the figure. The data are tidied up according to the times and spatial locations respectively, sorted in ascending way. Each subplot is equipped with its own legend while the title and the horizontal axis are shared to enhance the readability. If user asks for more than twenty values, the curves are subdivided into several figures with just the same structure; they can be distinguished thanks to a numerical suffix in the name, for instance velocity_1.eps and velocity_2.eps and the legend labels are always sorted in ascending way. The function that performs this task is called Make_plots that works for both space distributions and time evolutions plots.

3.5.3 ADVANCED POST PROCESSING

Advanced post processing means analyzing data from convergence studies and benchmarking against other codes. These analyzes can be done with an ancillary tool of the program based on the GUI shown in Figure 3.5-5 that is a menu bar that allows to select between several options, briefly explained in Table 3.5-6.

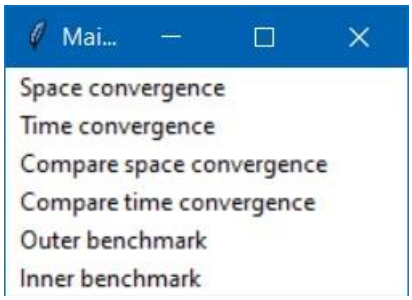


Figure 3.5-5 Graphical User Interface main window on the auxiliary tool for the advanced post processing of the data; menu bar composed by six cascades.

Table 3.5-6 List and description of all the available cascades in the GUI auxiliary tool for the advanced post processing.

Cascade	Description
Space convergence	performs the space convergence analysis using the data collected in folder Space_convergence.
Time convergence	performs the time convergence analysis with the data found in folder Time_convergence.
Compare space convergence	allows user to compare the results of the space convergence analyzes according to the time steps.
Compare time convergence	allows user to compare the results of time convergence analyzes according to the number of elements.
Outer benchmark	is the cascade devoted to make the benchmark of the solution obtained with the SC2 code against the solution obtained with other validated codes for the same simulation, in this work it is the 4C.
Innerbenhmark	can be used to check the inner consistency of the code once the benchmark against other validated codes is thoroughly verified.

The space convergence analysis is performed by changing the number of elements that are used to make the spatial discretization of the computational domain, keeping all the rest of the input frozen. On the other hand, the time convergence is performed changing the time step used to march in time. The former requires that a new steady state is reached while the latter must be done at a time for which the transient is not yet exhausted. In any case, the solution spatial distributions at the last time step (the end of the simulation) are used to make the analysis, together with the values of the mass and energy balances. These data are stored in folders Space_convergence and Time_convergence as described in section 3.5.1 and are used to evaluate the overall error of each simulation.

The outcome is stored in files in the Space_convergence or Time_convergence Output subfolder according to the same criteria described above (refers to For all of these inner benchmark kinds, user can chose three possible outcomes namely the full benchmark that make the comparison in both space and time, the space only and the time only that respectively performs the comparisons only on the spatial distributions and on the time evolutions.

Regardless of the selected benchmark (outer or inner), the figures obtained after the analysis are characterized by two horizontal subplots that share the abscissa axis. The top one compares no more than 2 pairs of curves to verify if they overlap; the bottom one is a semilogarithmic plot of the relative error that shows up to two curves. Each subplot has its legend and its title to better identify what they represent. If it is an outer benchmark, these figures are saved in the Benchmark subfolders of Space and Time folders within the Figures directory, while if it is an inner benchmark the figures are saved inside the directory Inner_Benchmark_results where the same nested structure discussed above is replied. In any case, their organization follows the schemes extensively discussed in the previous sections.

3.5 EASE OF POST PROCESSING

Examples of space and time convergence analyzes as well as benchmarks with respect to 4C and inner ones can be found in the next chapter.

Table 3.5-7); moreover, figures in logarithmic scale are automatically realized to easily control if the expected orders of convergence are achieved.

The outer benchmark against 4C code is executed comparing available data for spatial distributions and time evolutions not only of the solution, but also of the channel inlet and outlet mass flow rates, after that they are preprocessed in a format that is coherent to the one adopted by the SC2 code. The outer benchmark procedure consists of two steps: the check of the coherence of the data for both the codes and then the comparison of the spatial distributions and time evolutions, the strictly speaking benchmark.

The last available advanced post processing is the one that allows to perform the inner benchmark of the code. In this case, since the structure of the data files is always the same, the data preprocessing is not necessary. Exploiting the GUI, user can select the pair of simulations to be compared, how the results should be saved inside Inner_Benchmark_results, a directory at the same level of Simulation_results, the kind of benchmark to be done and the outcomes to get. At the time being three kind of inner benchmark are allowed:

1. Standard compares two simulations without acting on the data of the simulations;
2. Backflow compares the two simulations keeping into account that the flow directions are different in the two simulations;
3. Refined mesh compares the simulations knowing that they differ in the number of elements of the spatial discretizations, that impact on the error evaluation since the arrays of the spatial distributions do not have the same length.

For all of these inner benchmark kinds, user can chose three possible outcomes namely the full benchmark that make the comparison in both space and time, the space only and the time only that respectively performs the comparisons only on the spatial distributions and on the time evolutions.

Regardless of the selected benchmark (outer or inner), the figures obtained after the analysis are characterized by two horizontal subplots that share the abscissa axis. The top one compares no more than 2 pairs of curves to verify if they overlap; the bottom one is a semilogarithmic plot of the relative error that shows up to two curves. Each subplot has its legend and its title to better identify what they represent. If it is an outer benchmark, these figures are saved in the Benchmark subfolders of Space and Time folders within the Figures directory, while if it is an inner benchmark the figures are saved inside the directory Inner_Benchmark_results where the same nested structure discussed above is replied. In any case, their organization follows the schemes extensively discussed in the previous sections.

Examples of space and time convergence analyzes as well as benchmarks with respect to 4C and inner ones can be found in the next chapter.

Table 3.5-7 Naming scheme and content of the files saved to store the outcomes of the space and time convergence analyzes.

Object	Convergence analysis	Name scheme	Example	Content
CHAN	Space	CHAN.ID_sc_outcome.tsv	CHAN_1_sc_outcome.tsv	Number of elements, spatial discretization pitches and relative errors on velocity, pressure and temperature spatial distributions.
	Time	CHAN.ID_tc_outcome.tsv	CHAN_1_tc_outcome.tsv	Time steps and relative errors on velocity, pressure and temperature spatial distributions.
STR_MIX	Space	STR_MIX.ID_sc_outcome.tsv	STR_MIX_1_sc_outcome.tsv	Number of elements, spatial discretization pitches and relative errors on temperature spatial distribution.
	Time	STR_MIX.ID_tc_outcome.tsv	STR_MIX_1_tc_outcome.tsv	Time steps and relative errors on temperature spatial distribution.
STR_SC	Space	STR_SC.ID_sc_outcome.tsv	STR_SC_1_sc_outcome.tsv	Number of elements, spatial discretization pitches and relative errors on temperature spatial distribution.
	Time	STR_SC.ID_tc_outcome.tsv	STR_SC_1_tc_outcome.tsv	Time steps and relative errors on temperature spatial distribution.
STR_STAB	Space	STR_STAB.ID_sc_outcome.tsv	STR_STAB_1_sc_outcome.tsv	Number of elements, spatial discretization pitches and relative errors on temperature spatial distribution.
	Time	STR_STAB.ID_tc_outcome.tsv	STR_STAB_1_tc_outcome.tsv	Time steps and relative errors on temperature spatial distribution.
Z_JACKET	Space	Z_JACKET.ID_sc_outcome.tsv	Z_JACKET_1_sc_outcome.tsv	Number of elements, spatial discretization pitches and relative errors on temperature spatial distribution.

Object	Convergence analysis	Name scheme	Example	Content
CONDUCTOR	Time	Z_JACKET.ID_tc_outcome.tsv	Z_JACKET_1_tc_outcome.tsv	Time steps and relative errors on temperature spatial distribution.
	Space	CONDUCTOR.ID_sc_outcome_mass_energy.tsv	CONDUCTOR_1_sc_outcome_mass_energy.tsv	Number of elements, spatial discretization pitches and relative errors on mass and energy balance
	Time	CONDUCTOR.ID_tc_outcome_mass_energy.tsv	CONDUCTOR_1_tc_outcome_mass_energy.tsv	Time steps and relative errors on mass and energy balance

3.5.4 SUMMARY

The well-structured and nested organization of the output of the simulation and the post processing of the data are the main topics of this section. SC2 code automatically saves a large amount of data, from the initialization to the solution spatial distributions, as well as the inlet and outlet time evolutions of channels main properties as velocity, pressure, temperature, density and mass flow rate. User can ask to save other data at specific times and spatial coordinates compiling the main input file conductor_diagnostic.xlsx. Currently, the selected extension of the files is the Tab Separated Value format, but other possibilities will be considered. Both the default and user required data are then converted into suitable vectorial figures (.eps) that constitutes the outcome of the default post processing.

In order to perform the space and time convergence analyzes, the code saves for each simulation the final solution spatial distributions and the mass and energy balances computed at the last time step in “ad hoc” created folders. These files can be analyzed with a tool whose GUI allows several kinds of analyzes: alongside the convergence studies, the inner and outer benchmarks are also managed by this application, provided the required data are stored in the dedicated Benchmark folders if the latter is considered. These further elaborations are referred to as advanced postprocessing.

CHAPTER 4

4 SC2 VERIFICATION AND VALIDATION

The Verification and Validation phase of data is certainly required developing a new software. This can be carried out with respect to experiments conducted in the laboratory or by using other validated software. In this work, the second option was adopted, the reference code is 4C code [54], [55], [88].

A first fundamental step in code V&V is the check that the properties of materials are correctly evaluated from the corresponding functions. For the time being, these functions are an optimized python version of the 4C subroutine written in Fortran 90; whenever it is possible the *numpy array* smart notation is introduced to reduce the computational time and *numpy* available functions substitute the home-made ones. As mentioned in section 3.1.1, functions for material properties are grouped in modules by material; except the solid material density, that is assumed constant, all other properties are functions of variables which are passed as arguments to the function.

Generally, the benchmark is performed exploiting a parametric scan by fixing all the parameters that can vary except one, which is varied in order to evaluate the behavior of the function with respect to that parameter. For the sake of brevity, the results of this step are not discussed here, nonetheless the outcome of the comparisons is positive. An indirect proof of this statement are the results presented below.

The chapter is divided into three sections. The first describes the results of the space and time convergence analyses, followed by validation considerations with respect to 4C code in section 4.2, while internal verifications are left to the last section.

As a final remark, in order to limit the number of figures shown, only the most relevant ones are brought to the reader's attention, some of them being constructed to condense a considerable amount of information. Furthermore, the time evolutions and spatial distributions of the jacket are omitted because they do not bring further information; this helps to increase the clearness of the plots.

4.1 CONVERGENCE ANALYSES

The convergence analysis of the SC2 code is presented in this section. It is necessary to inspect if the theoretical order of convergence discussed in sections 2.2.1 and 2.2.2 are actually achieved.

The study is performed at low temperature considering the transient heating of an ITER-TF cable, whose general description, topology, and modeling are described in sections 2.1.4.1.1 and 0. The topology, material and geometrical data of the cable are reported in appendix D.2 while the other inputs of the simulation are summarized in the following Table 4.1-1.

4.1 CONVERGENCE ANALYSES

Table 4.1-1 Input data for the low temperature ITER-TF simulation launched to perform the space and the time convergence analysis.

Variable	Value	Unit
ITYMSH	0 (uniform)	–
TEND	15.0	s
INTIAL	1	–
TINL	4.5	K
PINL	6	bar
POUT	5.9	bar
Q0	250	W/m
XQBEG	4.0	m
XQEND	6.0	m
TQBEG	10.0	s
TQEND	20.0	s

The overall expected space convergence order of the code is between one and two since the finite element method applied here to discretize the space would lead to a second order which is preserved by the applied boundary conditions (homogeneous Neumann for the solid components and Dirichlet for the fluid components) but it is contaminated and downgraded by the extra upwind terms. The order of convergence in time of the θ-method is a function of θ. It is expected that the global order of convergence in time is exactly one if $\theta = 1$ (i.e., the Backward Euler numerical scheme), while the second order is foreseen for $\theta = \frac{1}{2}$, as this case corresponds to the Crank-Nicolson method.

The convergence analyses are carried out exploiting the post processing external tool mentioned in 3.5.3, the focus here are the global errors evaluated.

Firstly, the relative error for each variable is evaluated from the solution spatial distribution array:

$$\varepsilon = \frac{||\boldsymbol{\xi}_{nodal} - \boldsymbol{\xi}_{nodal,ref}||}{\left\|\boldsymbol{\xi}_{nodal,ref} - \xi_0\right\|} \tag{4.1-1}$$

according to the classical definition of the Euclidean norm:

$$\|\boldsymbol{y}\| = \sqrt{(\boldsymbol{y}, \boldsymbol{y})} = \sqrt{\sum_i y_i^2} \tag{4.1-2}$$

ξ_0 is an arbitrary reference value introduced to avoid that the denominator is always not null or too close to zero, especially with temperature arrays; it is chosen to be equal to the inlet value of the spatial distribution obtained with the most refined grid or time step, i.e.:

$$\xi_0 = \xi_{nodal,ref,1} \tag{4.1-3}$$

To obtain the global error, these errors should be reshaped as follows.

Arrays $\boldsymbol{\varepsilon}_i$ are constructed collecting the values of the error at a given spatial discretization pitch (or time step), then the average value of the elements for each of these arrays is computed that, in turn, constitutes the elements of the global error array $\boldsymbol{\varepsilon}_{\boldsymbol{global}}$. Practically, if N_{conv} is the total number of chosen spatial discretization pitches (or time steps) used to perform the space convergence (or the time convergence) analysis,

$$\boldsymbol{\varepsilon}_i = \begin{bmatrix} \boldsymbol{\varepsilon}_{\boldsymbol{v,ch,i}} \\ \boldsymbol{\varepsilon}_{\boldsymbol{p,ch,i}} \\ \boldsymbol{\varepsilon}_{\boldsymbol{T,ch,i}} \\ \boldsymbol{\varepsilon}_{\boldsymbol{T,st,i}} \\ \boldsymbol{\varepsilon}_{\boldsymbol{T,jk,i}} \end{bmatrix} for\ i = 1, \dots, N_{conv} \qquad (4.1\text{-}4)$$

being $\boldsymbol{\varepsilon}_{\boldsymbol{v,ch,i}}$ the array of the velocity errors for each channel evaluated with the i^{th} spatial discretization pitch (or time step) as described above, and so on. Finally, the elements of the global error array are defined as:

$$\varepsilon_{global,i} = \frac{\sum_{j=1}^{N_{eq}} \varepsilon_j}{N_{eq}}\ for\ i = 1, \dots, N_{conv} \qquad (4.1\text{-}5)$$

The space convergence is addressed first, followed by the time convergence of both BE and CN numerical schemes. The figures shown in this section are obtained with the "Compare space convergence" or "Compare time convergence" cascades of the post processing tool.

4.1.1 SPACE CONVERGENCE

The simulations for the space convergence are performed with inputs of Table 4.1-1 at fixed time step reducing the spatial discretization pitch of spatial discretization; the selected solution method is BE. Since it is not mandatory to perform a space convergence at the steady state, although recommended whenever is possible, in this case the end time of the simulation is in the middle of the heating phase where there are still transient conditions. The chosen time steps and spatial discretization pitches to perform the analysis are shown in Table 4.1-2 while the outcomes are plotted in *Figure 4.1-1*.

As can be seen from the logarithmic scale *Figure 4.1-1*, the slope of both the curves is such that to one order of magnitude on the abscissa axes the global error decreases of more than one order of magnitude but less than two resulting in a convergence order of about 1.2, that is in agreement with the previsions.

Table 4.1-2 Simulation performed with the ITER-TF configuration at low temperature. Number of elements for the spatial discretizations (discretization parameter in round brackets) used to perform the space convergence analysis and the two considered time steps.

$\boldsymbol{\Delta t\ s}$	$\mathbf{NELEMS_1}$ $(\boldsymbol{\Delta x_1\ m})$	$\mathbf{NELEMS_2}$ $(\boldsymbol{\Delta x_2\ m})$	$\mathbf{NELEMS_3}$ $(\boldsymbol{\Delta x_3\ m})$	$\mathbf{NELEMS_4}$ $(\boldsymbol{\Delta x_4\ m})$	$\mathbf{NELEMS_5}$ $(\boldsymbol{\Delta x_5\ m})$
0.5 0.1	2000 (0.005)	1000 (0.01)	200 (0.05)	100 (0.1)	50 (0.2)

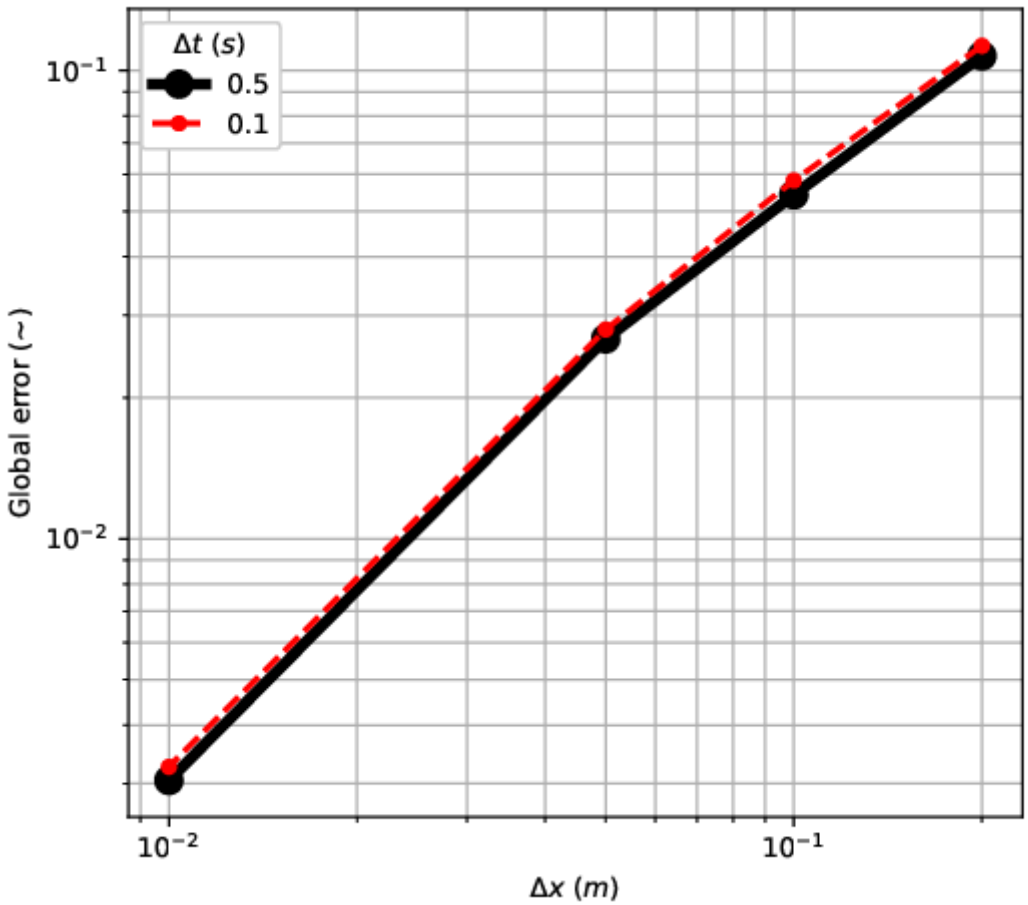


Figure 4.1-1 Simulation performed with the ITER-TF configuration at low temperature. Space convergence analysis in logarithmic scale at 15 s *for two time-steps. Solid circles represent actual values, solid or dashed lines give an idea of the trend.*

4.1.2 TIME CONVERGENCE

The time convergence analysis is the outcome of a set of simulations that share the same input data of Table 4.1-1, the same number of elements but characterized by different time steps to march in time. Both the time convergences are fulfilled with two spatial discretization parameters, the chosen values are collected in Table 4.1-3 and Table 4.1-4.

Table 4.1-3 Simulation performed with the ITER-TF configuration at low temperature. Time steps for the time convergence analysis with Backward Euler numerical schemes and the number of elements for the two set of simulations (discretization parameter in round brackets), about two orders of magnitude are explored.

$\boldsymbol{NELEMS}$ $(\boldsymbol{\Delta x}\ \boldsymbol{m})$	$\boldsymbol{\Delta t_1}\ \boldsymbol{s}$	$\boldsymbol{\Delta t_2}\ \boldsymbol{s}$	$\boldsymbol{\Delta t_3}\ \boldsymbol{s}$	$\boldsymbol{\Delta t_4}\ \boldsymbol{s}$	$\boldsymbol{\Delta t_5}\ \boldsymbol{s}$	$\boldsymbol{\Delta t_6}\ \boldsymbol{s}$
2000 (0.005) 200 (0.05)	0.02	0.05	0.1	0.2	0.5	1.0

Table 4.1-4 Simulation performed with the ITER-TF configuration at low temperature. Time steps for the time convergence analysis with Crank-Nicolson numerical schemes and the number of elements for the two set of simulations (discretization parameter in round brackets), more than two orders of magnitude are explored.

$\boldsymbol{NELEMS}$ $(\boldsymbol{\Delta x\ m})$	$\boldsymbol{\Delta t_1\ s}$	$\boldsymbol{\Delta t_2\ s}$	$\boldsymbol{\Delta t_3\ s}$	$\boldsymbol{\Delta t_4\ s}$	$\boldsymbol{\Delta t_5\ s}$	$\boldsymbol{\Delta t_6\ s}$	$\boldsymbol{\Delta t_7\ s}$	$\boldsymbol{\Delta t_8\ s}$	$\boldsymbol{\Delta t_9\ s}$	$\boldsymbol{\Delta t_{10}\ s}$
2000 (0.005) 200 (0.05)	0.005	0.01	0.02	0.03	0.05	0.1	0.2	0.3	0.5	1.0

Figure 4.1-2 collects the results of the time convergence analysis in two logarithmic subplots. The top one refers to Backward Euler numerical scheme while the bottom shows the Crank-Nicolson convergence. The errors shown are computed differently for the two methods. As far as BE is concerned, the points in the *Figure 4.1-2* (a) correspond to the global error of the simulation, equation (4.1-5); the subplot shows that the order of convergence meets the expectations being exactly one in the range $[0.1,1.0]\ s$ for both the considered number of elements.

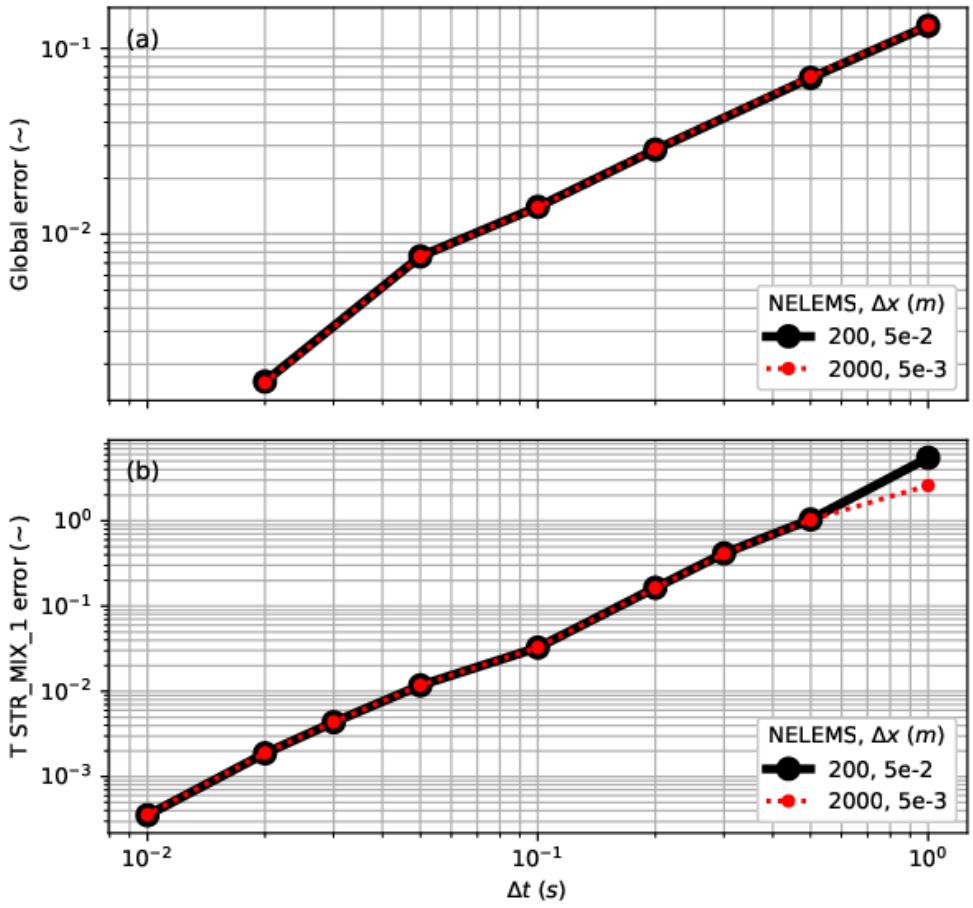


Figure 4.1-2 Simulation performed with the ITER-TF configuration at low temperature. Time convergence analysis results in logarithmic scale at 15 s for two different number of elements for the spatial discretization (listed in the legend). (a) Backward Euler the global error is shown; (b) Crank-Nicolson, only the error on the temperature of the strand is shown.

The algorithm implementing the CN method is more sensitive to errors introduced by the coupling of the conductor components. For this reason, the convergence performed by calculating the global error of a set of simulations in which the components are thermally coupled (i.e., non-zero heat transfer coefficients), is altered by the errors introduced by the coupling, returning a convergence order of one at most. Hence, the sets of simulations to study the convergence of this numerical scheme contemplate the fully decoupled configurations of the cable, obtained imposing that the heat transfer coefficients of appendix D.2 are all equal to 0. In this case the global error definition is no longer meaningful since only the temperature of STR_MIX object is changing; therefore, the time convergence considers only the errors on this variable computed according to equation (4.1-1), shown in Figure 4.1-2 (b). The second order convergence is verified for both the considered number of elements in the range [0.01,0.1], which can be extended up to $1\ s$ for 200 elements.

4.1.3 SUMMARY

The outcome of the convergence analyses both in space and time for the ITER-TF design is the topic of this section. The common input data to all the simulations can be found in appendix D, while the specific ones are collected in the tables of the current section. Having agreed on the definition of the error, the results are presented in Figure 4.1-1 and Figure 4.1-2 with an appropriate logarithmic scale on both axes. Globally, both the analyses reproduced the expected results: the space convergence order is between one and two due to the combination of the FEM for the spatial discretization and the upwind (boundary conditions do not affect the order of convergence); BE and CN returns the expected order of convergence as well, the difference being that for the former the overall error can be exploited, while for the latter the STR_MIX temperature error is considered since, in this case, the conductor components should be decoupled to obtain the foreseen results.

4.2 VALIDATION AGAINST THE 4C CODE

Once that the expected order of convergence of the algorithms implemented in the SC2 code are confirmed, the V&V proceeds with the comparison of the simulations with the results obtained from another widely validated and verified code, such as 4C code. The benchmark, involving both cable types described in sections 2.1.4.1.1 and 0, is the topic of the following two sections. Actually, in the two configurations considered only the geometry and the topology of the cable are different: both the same coolant and the same superconducting material are used for simulations. In order to test the different initializations allowed by the code, the INTIAL 1 and INTIAL 5 options are considered for both configurations. These options, as discussed in the section 4.3.1, imply two different sets of boundary conditions; besides, it shows that the INTIAL 2 option from the point of view of boundary conditions does not differ from INTIAL 5, so it is not considered here. Input data concerning the geometry, materials, topology, friction factors and heat transfer coefficients are reported in appendix D; here, the focus is on the main hypotheses made in order to limit the number of degrees of freedom in the simulation, effectively simplifying the comparison: namely, imposing both the values of the heat transfer coefficients (between fluid components, fluid and solid components and

between solid components) and the friction factors of the fluid components to a constant value, chosen to be representative of the typical order of magnitude of the parameters.

The simulation to achieve both benchmarks is a transient heating and cooling of the cable until the initial steady state is reached. The external heat source has the shape of a square wave in space and time and the heat is deposited directly on the strands.

This introductive section ends shortly describing what is behind the outer benchmark GUI commands of the post processing tool. It is a two steps procedure, the first being the check for the coherence of the data and the second the actual benchmark.

The coherence check is a preliminary condition to be verified aiming to guarantee that the saved data of the spatial distributions at given times and the time evolutions at given coordinates are coherent. In fact, the values corresponding to the space-time intersection must be the same irrespective of the array to which they belong, whether a spatial distribution or a time evolution. If this trivial but fundamental condition is verified for both the codes, the data can be trusted and the tool moves forward to the benchmark, that is the comparison of the saved spatial distributions and time evolutions of the solution variables. The generic relative error array is defined as:

$$\boldsymbol{\varepsilon_{outer\ bench}} = \frac{|\boldsymbol{\xi} - \boldsymbol{\xi_{4C}}|}{|\boldsymbol{\xi_{4C}} - \xi_0|} \tag{4.2-1}$$

where:

$$\xi_0 = 1.5 \max(|\boldsymbol{\xi_{4C}}|) \tag{4.2-2}$$

With this definition of ξ_0, the denominator never tends towards 0 and therefore the error never diverges, leading to a misunderstanding of the comparison.

The following sections share the same structure. They begin completing the picture of the input data needed to run the simulation, continue with a short physical description of the simulation outcomes and end discussing the comparison with the 4C code.

4.2.1 BENCHMARK WITH THE 4C CODE: AN HTS POWER CABLE

This section deals with the benchmark of the 3P-HTS configuration against the 4C code. Table 4.2-1 gathers the input data of the executed simulations.

When INTIAL 1 is considered, the inlet and outlet pressure are chosen such that the inlet mass flow rate is $0.1\ \mathrm{kg/s}$. The small pressure drop associated to this flow is related to the tiny value of the channel friction factor, see appendix D.1. Conversely when the flag is INTIAL 5, the outlet pressure is chosen such that the inlet pressure, given the inlet flow rate, is equal to the inlet pressure of the previous case. In this way the initialization is the same for the two simulations and the outcome is determined by the imposition of the boundary conditions.

The results of the benchmark with INTIAL 1 are summarized in Figure 4.2-1 and Figure 4.2-2, which show, respectively, the time evolution and the spatial distribution of the variables constituting the solution, which are the velocity, pressure and temperature of the fluid and

the temperature of the strand. To achieve better legibility, the temperature of the jacket is not shown as it does not add significant information to the graph.

To complete the picture, the global errors at three selected spatial locations and times are also shown in Figure 4.2-3.

Table 4.2-1 Input data for the simulation performed at high temperature with the 3P-HTS configuration; the second column refers to the simulation with INTIAL set equal to 1, the third column shows the data for the simulation with INTIAL set equal to 5.

Variable	INTIAL 1 Value	INTIAL 5 Value	Unit
METHOD	0 (BE)	0 (BE)	–
ITMESH	0 (uniform)	0 (uniform)	–
NELEMS	200	200	–
TEND	300.0	300.0	s
INTIAL	1	5	–
TINL	60	60	K
PINL	6	-	bar
POUT	5.99	5.99	bar
MDTIN	–	0.1	kg/s
Q0	3000	3000	W/m
XQBEG	4.0	4.0	m
XQEND	6.0	6.0	m
TQBEG	10.0	10.0	s
TQEND	25.0	25.0	s

As can be seen from the figures, the two codes calculate essentially the same solution (error of the order of 10^{-5}), except at initialization where there is the maximum global error ($3.2\ 10^{-5}$) for each of the coordinates considered. The reason for this is to be found in the different initialization algorithms of the two codes. As mentioned in section 0, the SC2 manipulates the equation of the hydraulic characteristic to carry out the initialization of the fluid variables; on the other hand, the 4C adopts an iterative process starting from an assumed velocity value and providing, at each iteration, the calculation of an equivalent hydraulic diameter weighted on the friction factor.

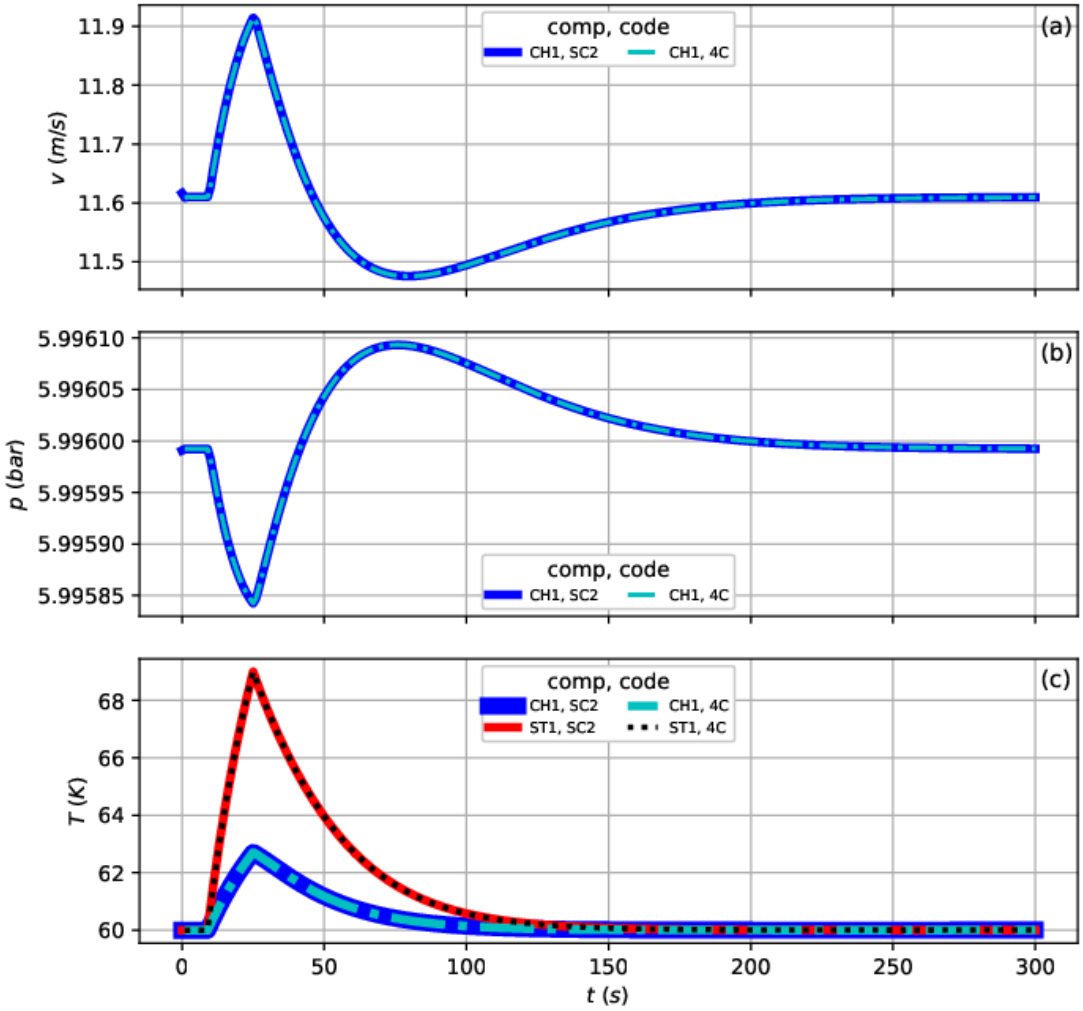


Figure 4.2-1 High temperature 3P-HTS simulation with INTIAL 1: time evolution variables at 5 m*. Solid lines refer to SC2 values, dot-dashed and dotted lines refer to 4C values: (a) fluid velocity, (b) fluid pressure, (c) fluid and strand temperatures*.

In any case, both codes give a good approximation of the real thermo-hydraulic behavior of the conductor. As a result of heating, the temperature of the strand increases, and the spatial shape of the source is clearly visible in Figure 4.2-2 (c). Another consequence of the heating is the pressurization of the channel, which, however, is inhibited by the low density of He at that high temperature. In fact, the channel pressure does not deviate significantly from the initial value during the entire duration of the transient (see Figure 4.2-1 (b)). However, the increase in fluid temperature due to convective heat exchange further reduces the density. Given the negligible change in pressure, the flow rate is practically constant and, consequently, a reduction in density corresponds to an increase in velocity as can be seen in Figure 4.2-1 (a) and Figure 4.2-2 (a). The marked difference between strand and coolant temperatures is due to the different volumetric heat capacity of the materials, being that of solid components two orders of magnitude larger than that of the fluid. This large temperature difference, together with the coolant flow rate, favors the removal of the heat

energy introduced by the source and justifies the need to use a linear power of $3000\ \mathrm{W/m}$ to obtain a peak temperature difference of about $10\ \mathrm{K}$.

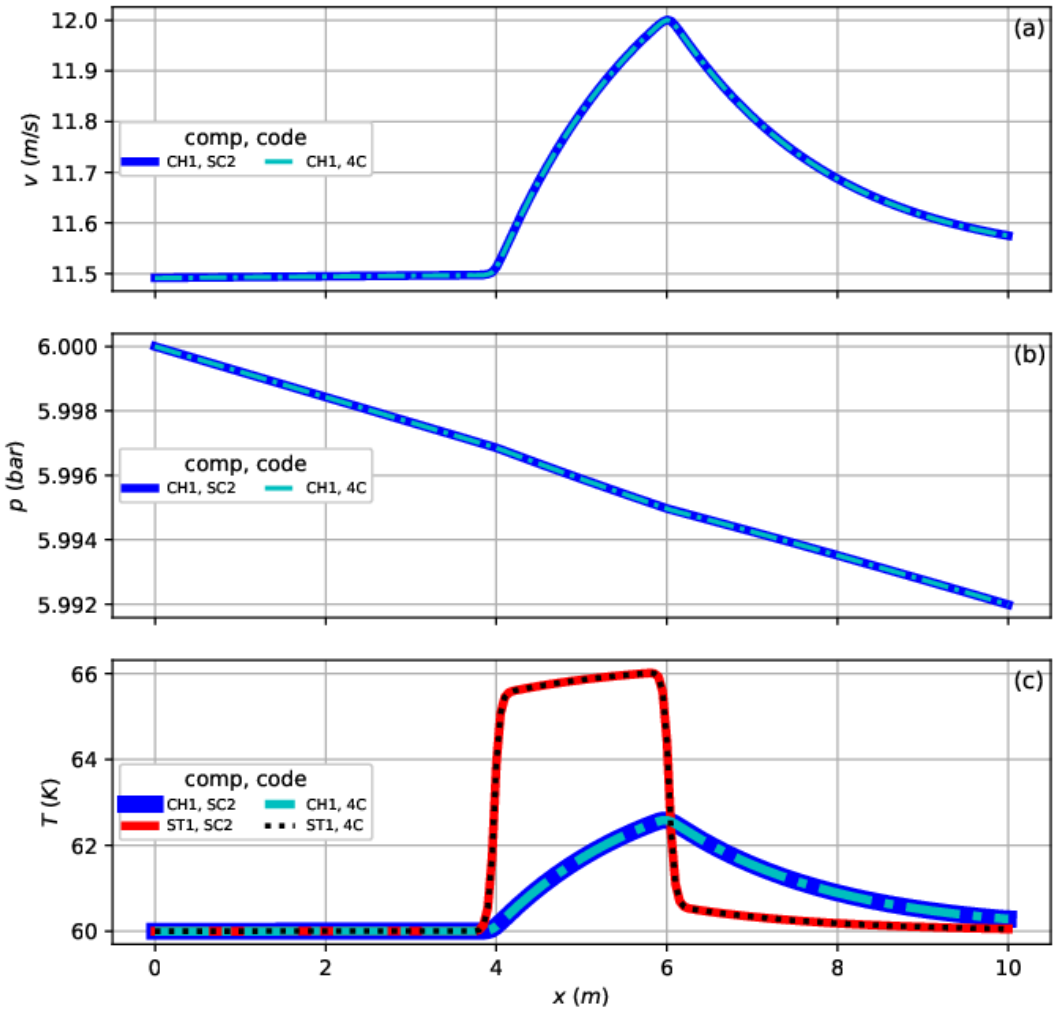


Figure 4.2-2 High temperature 3P-HTS simulation with INTIAL 1: spatial distribution variables at $18\ \mathrm{s}$. *Solid lines refer to SC2 values, dot-dashed and dotted lines refer to 4C values: (a) fluid velocity, (b) fluid pressure, (c) fluid and strand temperatures.*

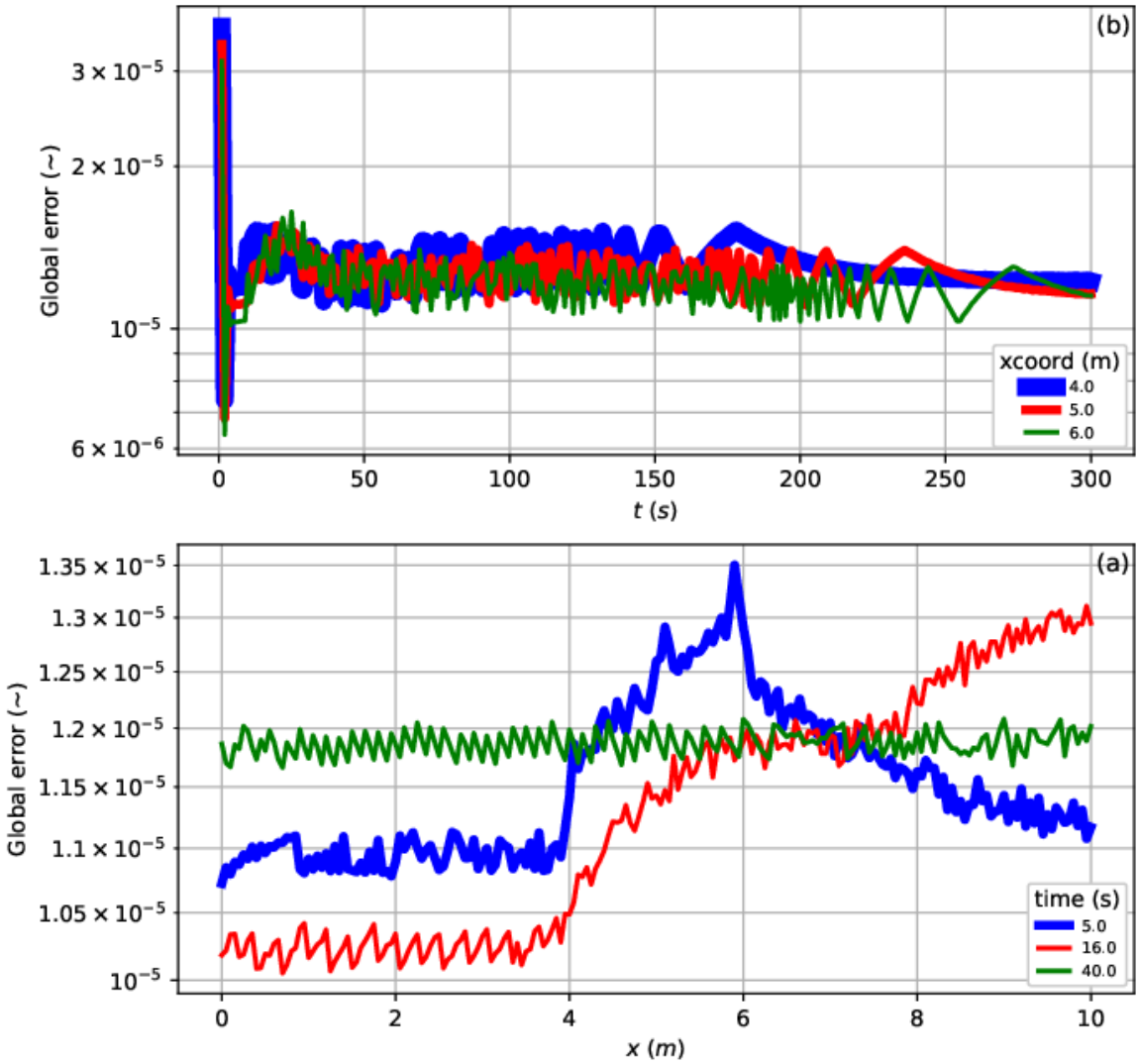


Figure 4.2-3 High temperature 3P-HTS simulation with INTIAL 1. Relative errors in semi logarithmic: (a) time evolution errors at $4.0\ m$, $5.0\ m$ *and* $6.0\ m$*; (b) spatial distribution errors at* $5.0\ s$, $16.0\ s$ *and* $40.0\ s$.

Attention is now shifted to the INTIAL 5 case, for which the benchmark results are shown in Figure 4.2-4 regarding the temporal evolution, in Figure 4.2-5 representing the spatial distribution, while the global error at three different spatial coordinates and times is shown in Figure 4.2-6. The physics of the simulation is now determined by the fact that the inlet velocity is fixed while the pressure can vary. However, since even in this case the pressure variation is not relevant, the time evolution and the spatial distribution are not very different from those shown for the previous case.

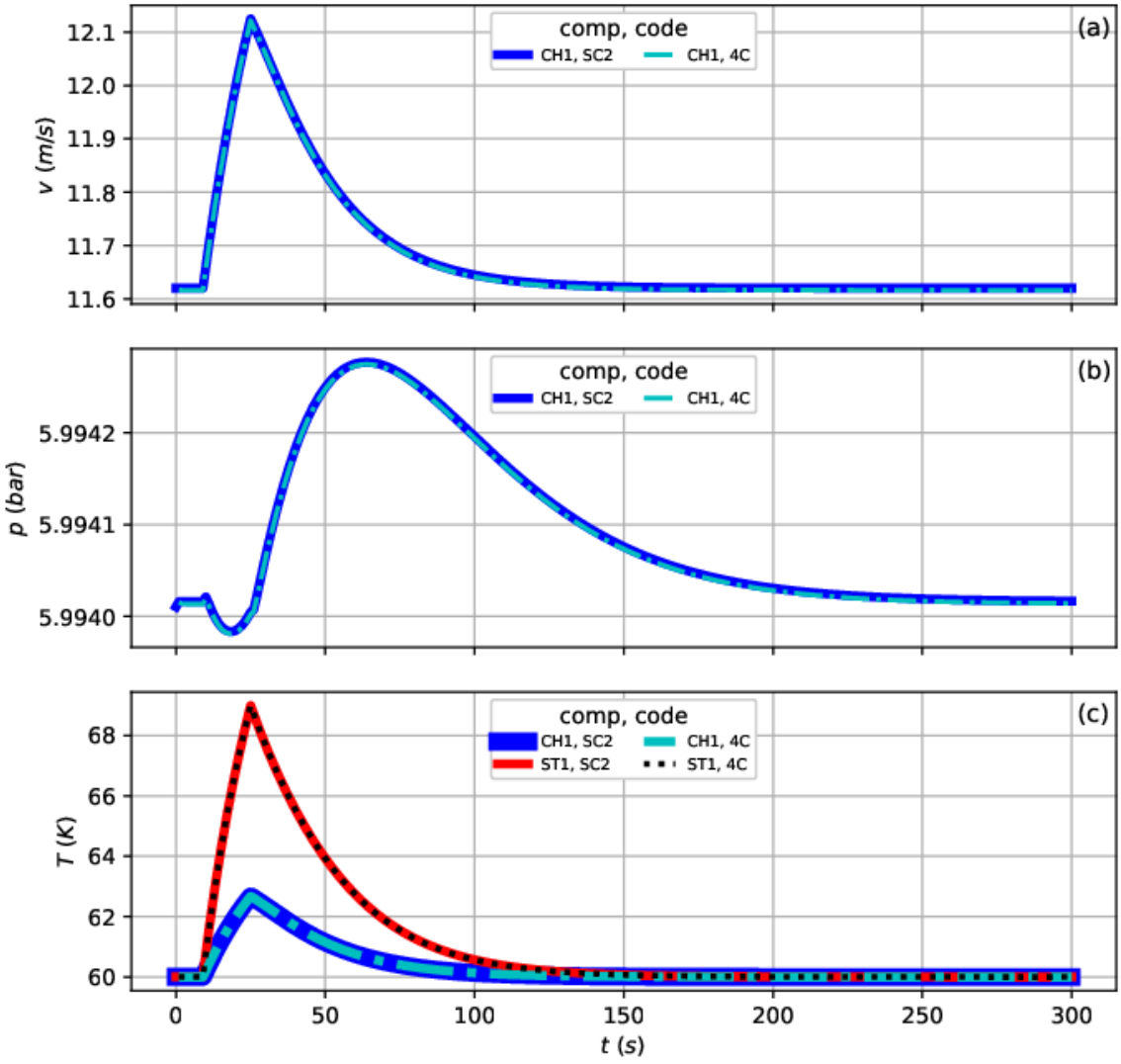


Figure 4.2-4 High temperature 3P-HTS simulation with INTIAL 5: time evolution variables at $5\ m$*. Solid lines refer to SC2 values, dot-dashed and dotted lines refer to 4C values: (a) fluid velocity, (b) fluid pressure, (c) fluid and strand temperatures.*

Observing the global error, this is about an order of magnitude greater than that calculated for the INTIAL 1 case, but its value is still such as to conclude that the two codes are equivalent.

As can be seen from Figure 4.2-3 and Figure 4.2-6, the average global error for the case where INTIAL is equal to 1 is of the order of magnitude 10^{-5}, while in the case where the flag is set equal to 5 it is of the order of magnitude 10^{-4}. The small value of these relative errors

ratifies the success of the benchmark; however, the result should not be taken for granted given the differences between the two codes, in particular the one concerning the calculation of properties in Gauss nodes. The comparison of the global errors of the helium, strand and jacket properties is proposed in Figure 4.2-7 for two spatial discretizations performed with 200 and 2000 elements, respectively. The properties are evaluated according to the strategies of the two codes using the spatial distributions of temperature and pressure at 18 s.

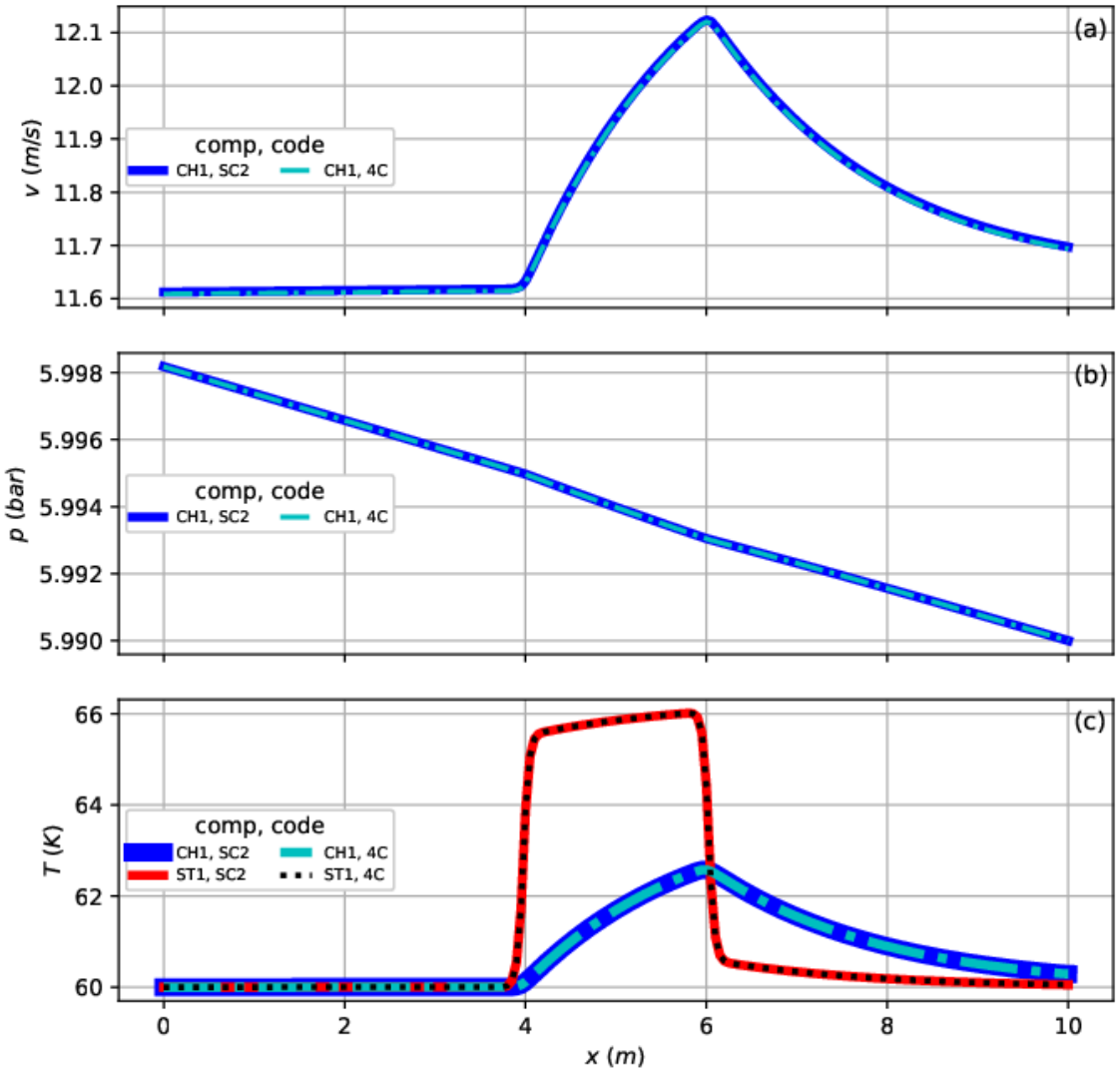


Figure 4.2-5 High temperature 3P-HTS simulation with INTIAL 5: spatial distribution variables at 18 *s. Solid lines refer to SC2 values, dot-dashed and dotted lines refer to 4C values: (a) fluid velocity, (b) fluid pressure, (c) fluid and strand temperatures.*

For consistency, the error on the generic property is defined as:

$$\varepsilon_{gauss} = \frac{\left|\chi_{gauss,SC2} - \chi_{gauss,4C}\right|}{\left|\chi_{gauss,4C}\right|} \qquad (4.2\text{-}3)$$

The global error is computed as the average error of the properties.

The maximum errors are located in correspondence of the temperature gradients close to the heated region and they are reported for clarity in Table 4.2-2. In fact, it is precisely at steep variations in temperature (and pressure when required) that the two strategies differ most: the overall error increases by many orders of magnitude, up to 10 for strand properties. The relevant contribution is due to the strand properties, however the errors are already small to give relevant effects.

The figure also shows that by increasing the number of elements by an order of magnitude, the difference between the values calculated by the two methods is reduced by almost two orders of magnitudes.

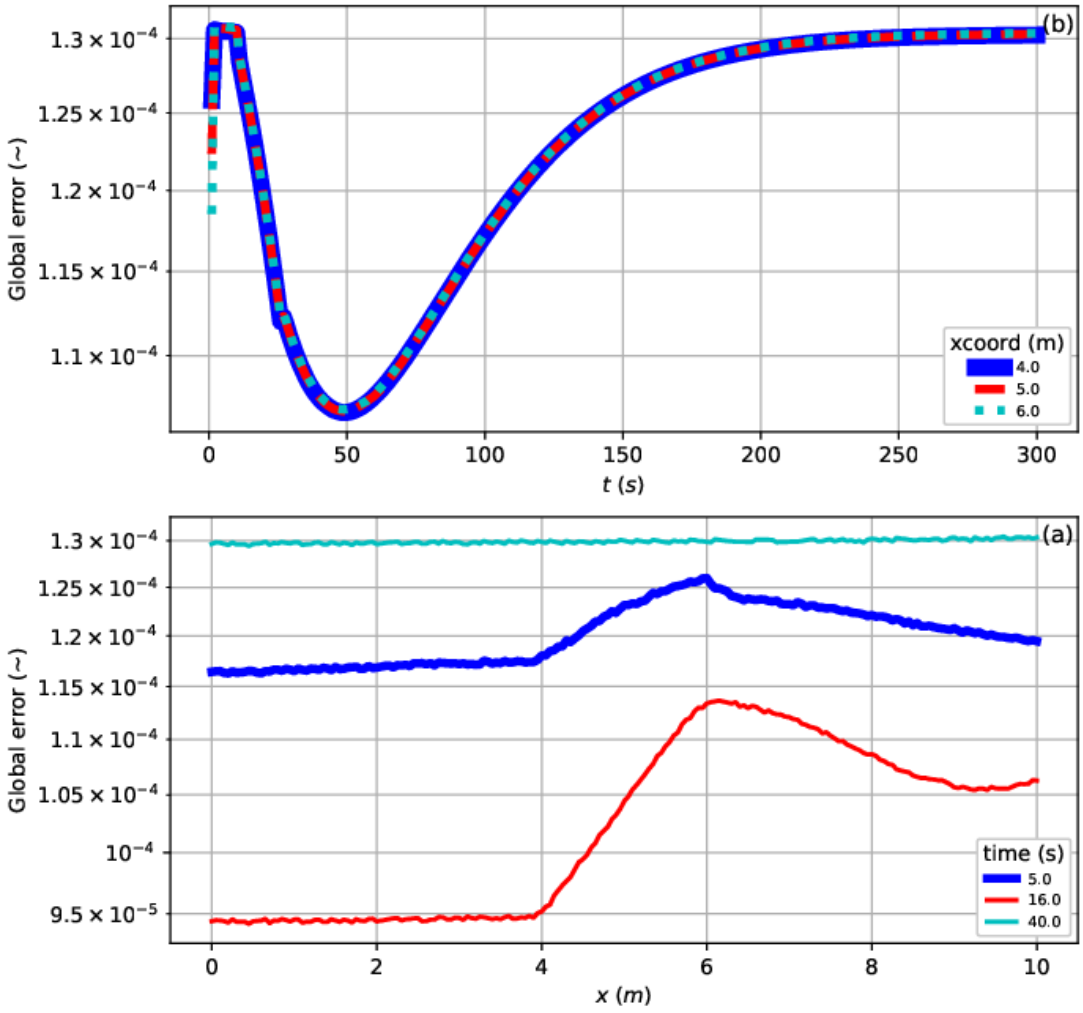


Figure 4.2-6 High temperature benchmark 3P-HTS with INTIAL 5. Relative errors in semi logarithmic scale: (a) time evolution errors at $4.0\ m$, $5.0\ m$ *and* $6.0\ m$*; (b) spatial distribution errors at* $5.0\ s$, $16.0\ s$ *and* $40.0\ s$.

Table 4.2-2 3P-HTS configuration with INTIAL 1: maximum global error from the comparison on the properties evaluated in Gauss points according to SC2 and 4C implementation at $18\ s$.

NELEMS	fluid	strand	Jacket
200	4.7e-10	4.4e-4	2.3e-8
2000	3.8e-12	7.1e-6	2.3e-10

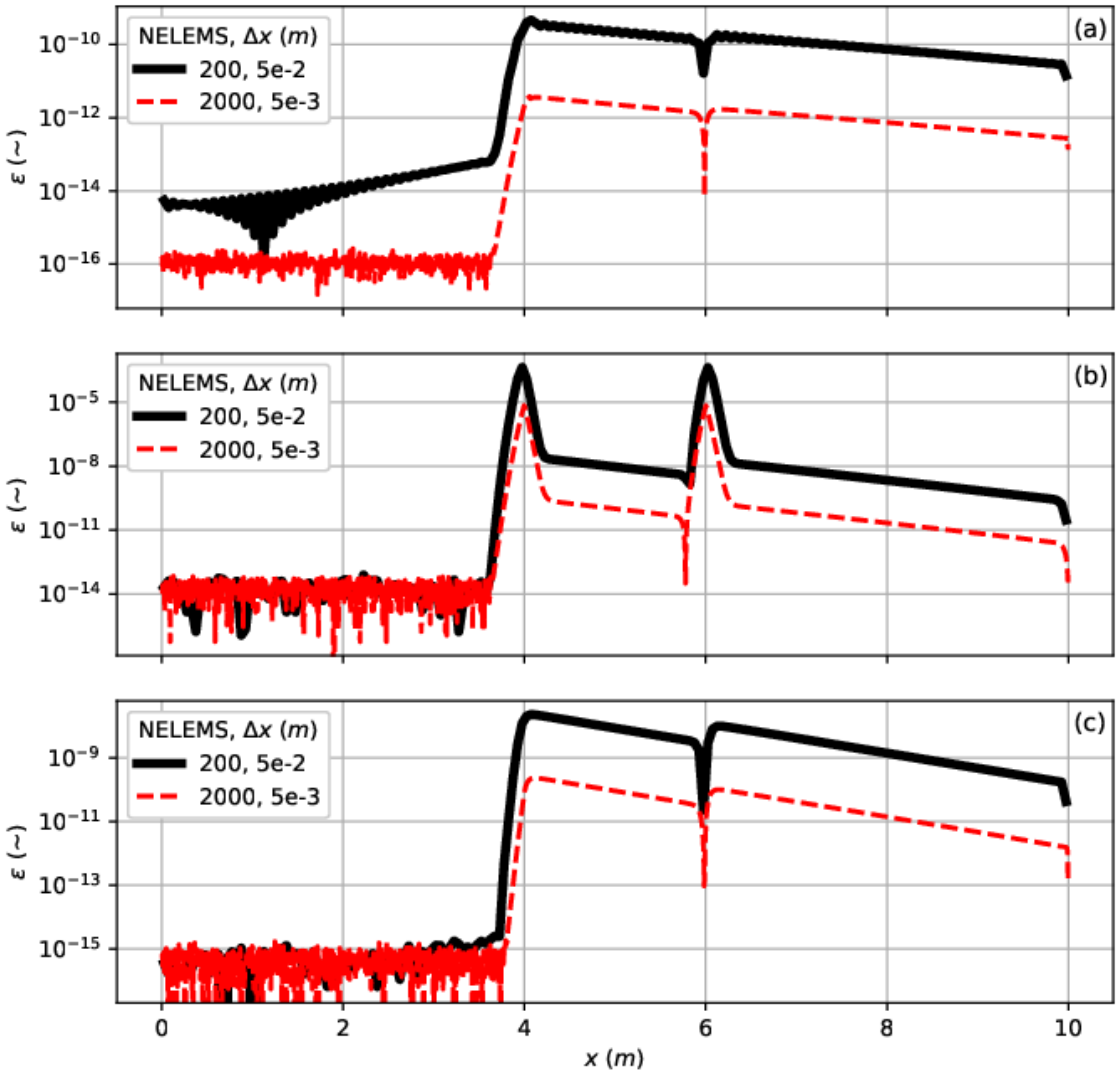


Figure 4.2-7 High temperature 3P-HTS INTIAL 1: spatial distribution of the global errors at $18\ s$ *from the comparison of the properties evaluated in the Gauss points according to the SC2 and 4C implementations, in semi logarithmic scale at high temperature. Two number of elements for the spatial discretizations are considered: (a) fluid properties, (b) strand properties, (c) jacket properties.*

In conclusion, for the considered temperature range, the solution at each time step is not strongly affected by the recipe used to evaluate the properties in the Gauss node, so the benchmark is successfully completed.

4.2.2 BENCHMARK WITH THE 4C CODE: AN ITER LTS TOROIDAL FIELD COIL CABLE

The benchmark results for the ITER-TF configuration are analyzed in this section. The input data that identify the simulations are collected in Table 4.2-3.

As mentioned in the introduction to the section, also for the ITER-TF geometry the comparison of the results obtained is carried out for two characteristic values of the INTIAL

flag. As there are two fluid components in hydraulic parallel, the value of the flag must be the same for both.

Table 4.2-3 Input data for the simulation performed at low temperature with the ITER-TF configuration; the second column refers to the simulation with INTIAL set equal to 1, the third column shows the data for the simulation with INTIAL set equal to 5.

Variable	INTIAL 1 Value	INTIAL 5 Value	Unit
METHOD	0 (BE)	0 (BE)	–
ITMESH	0 (uniform)	0 (uniform)	–
NELEMS	200	200	–
TEND	100.0	100.0	s
INTIAL	1	5	–
TINL	4.5	4.5	K
PINL	6	-	bar
POUT	5.9	5.9	bar
MDTIN_1	–	$8.4\ 10^{-3}$	kg/s
MDTIN_2	–	1.24510^{-2}	kg/s
Q0	250	250	W/m
XQBEG	4.0	4.0	m
XQEND	6.0	6.0	m
TQBEG	10.0	10.0	s
TQEND	20.0	20.0	s

For the sake of simplicity, the first set of input data applies the same initial values to both the conductor channels, but this is no longer true for the second set since for the two channels a different value of the inlet mass flow rate is assigned. These guess values approximate the initial inlet mass flow rates computed with the first simulation, used as input parameters in the second run. As discussed in section 4.3.1, these two simulations give different results and constitute a test case to check the capability of the SC2 code to manage different fluid dynamic operating conditions. For the time being they are considered separately, and the focus is on the benchmark with the 4C. In the following, first the results obtained are commented by comparing the simulations with INTIAL 1, and then those corresponding to INTIAL 5.

The curves shown in *Figure 4.2-8* and Figure 4.2-9 represent, respectively, the time evolutions and spatial distributions in an appropriate spatial coordinate and at a defined time step. The absence of the pressure and temperature curves for the first channel, the hole, in *Figure 4.2-8* is due to the fact that these time evolutions are not available in 4C for the case hand.

Looking at the above *Figure 4.2-8* and Figure 4.2-9, it could be concluded that, similarly to the previous section, the benchmark is positive. To quantify the accuracy of that agreement, Figure 4.2-10 shows the global error of the solution in three spatial coordinates and three time steps. As with the 3P-HTS configuration, before heating starts and once its effects ends, the two codes are in perfect agreement, except for the already discussed difference on initialization.

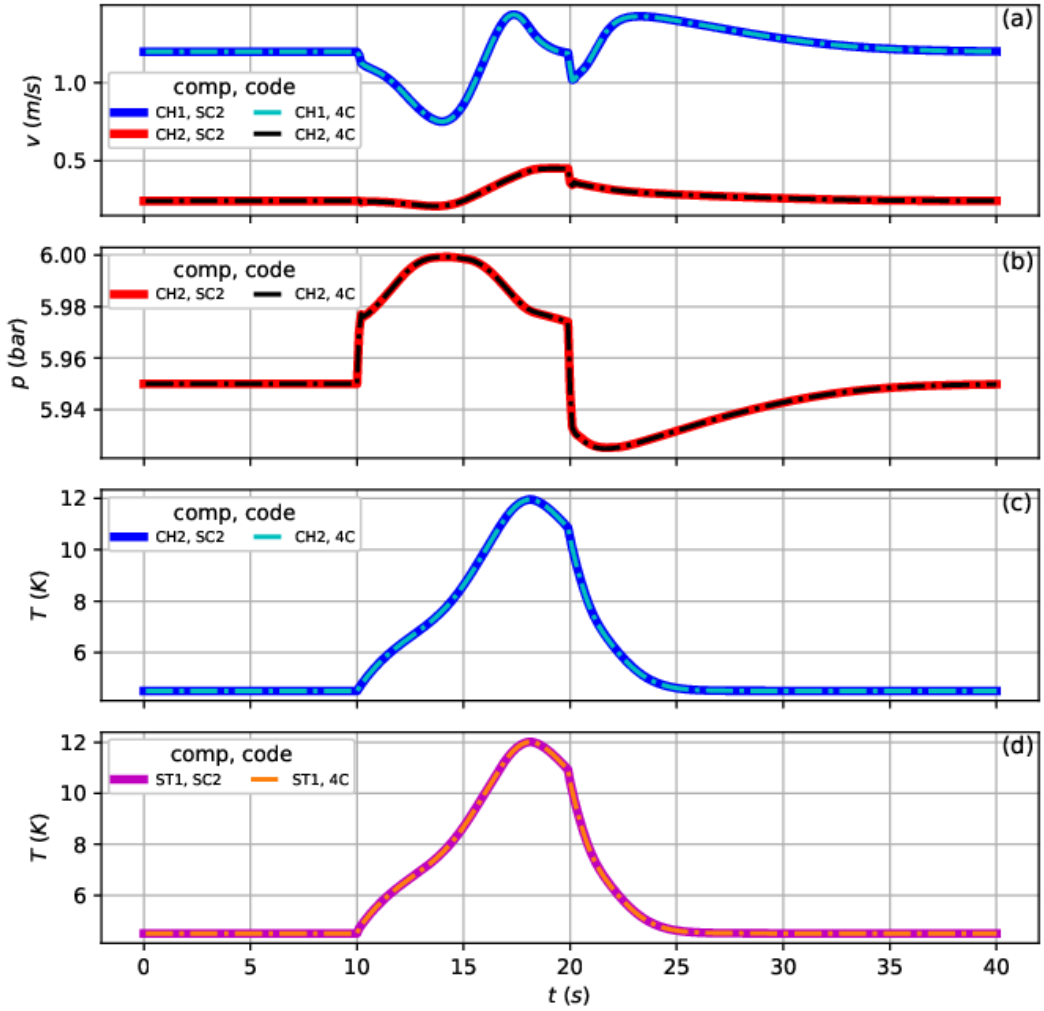


Figure 4.2-8 ITER-TF with INTIAL 1: time evolution variables at $5\ \mathrm{m}$ *. Solid lines refer to SC2 values, dot-dashed lines refer to 4C values: (a) fluid velocities, (b) fluid pressure, (c) fluid temperature (d) strand temperature.*

Note, however, that in this specific case, arises during the heating and cooling transient where the errors increase by up to three orders of magnitude. The main reason for this difference in results is the non-linearity of material properties at low temperatures, which is captured differently by the two strategies implemented in the codes for calculating properties in the Gauss node. This is supported by the *Figure 4.2-11* which compares, for two different spatial discretizations, the global error of the fluid, strand and jacket properties, calculated with the spatial distribution of temperature and pressure at $15\ \mathrm{s}$, i.e., in the middle of the heating phase of the transient. The errors are defined by equation (4.2-3), the maximum values are given in *Table 4.2-4* for ease of reading.

The global fluid error shown in Figure 4.2-11 (a) refers to the second channel, i.e., the bundle, as it is affected by a greater temperature variation than the first one, as can be seen from Figure 4.2-11 (c).

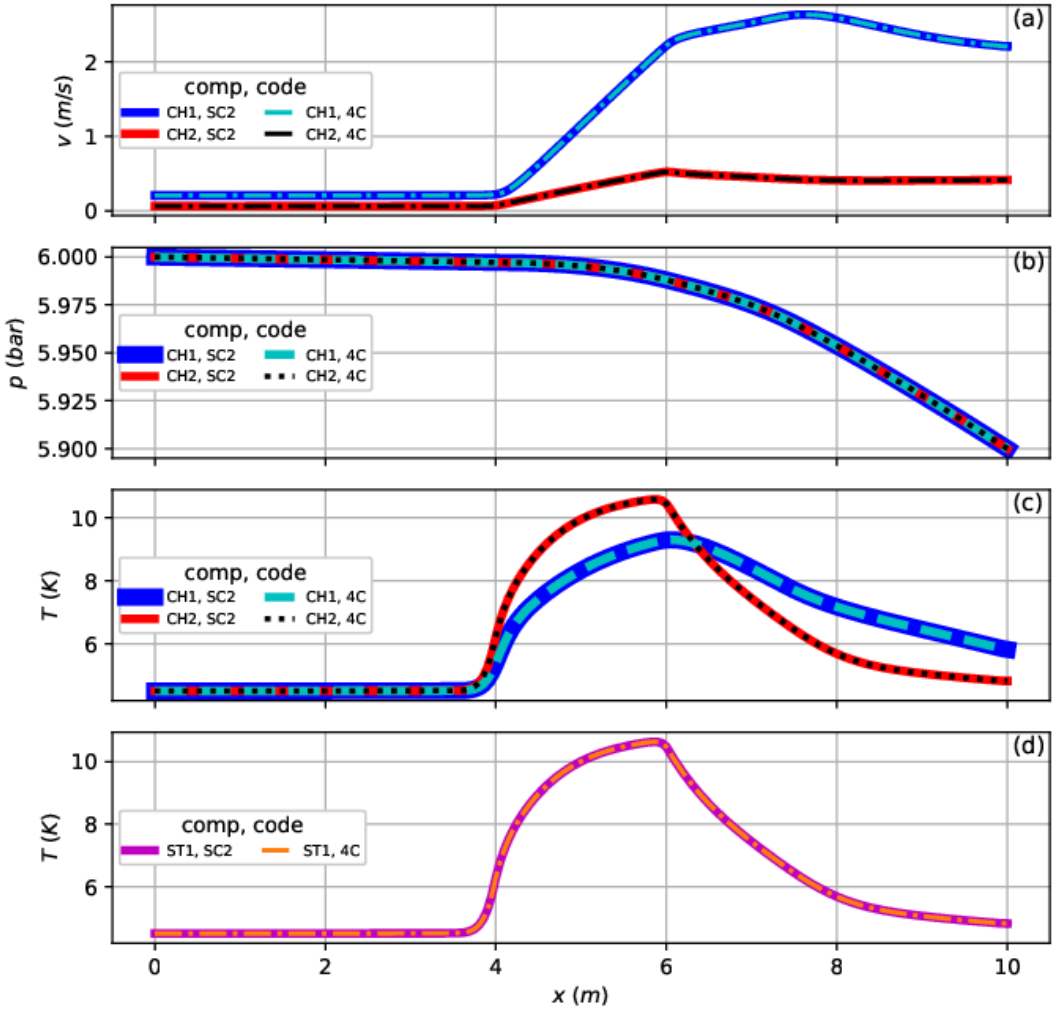


Figure 4.2-9 ITER-TF with INTIAL 1: spatial distribution variables at 16 s. Solid lines refer to SC2 values, dot-dashed and dotted lines refer to 4C values: (a) fluid velocities, (b) fluid pressures, (c) fluid temperatures, (d) strand temperature.

Table 4.2-4 ITER-TF configuration with INTIAL 1 and INTIAL 5: maximum global error from the comparison on the properties evaluated in Gauss points according to SC2 and 4C implementation at 15 s.

	INTIAL 1			INTIAL 5		
NELEMS	fluid	strand	Jacket	fluid	strand	Jacket
200	7.4e-3	3.0e-3	7e-4	6.5e-4	7.5e-4	1.2e-4
2000	2.5e-4	4.3e-5	9.7e-6	4.1e-5	1.1e-5	1.5e-6

It is important to note that the errors are in general much greater than those shown in the analogous Figure 4.2-7, reflecting the fact that in the temperature range considered, the properties are strongly non-linear and the two recipes give different values. Errors increase significantly (up to six orders of magnitude) at temperature gradients. By refining the mesh, the error tends to reduce as expected, but the reduction in maximum error is only one order of magnitude compared to two at high temperature.

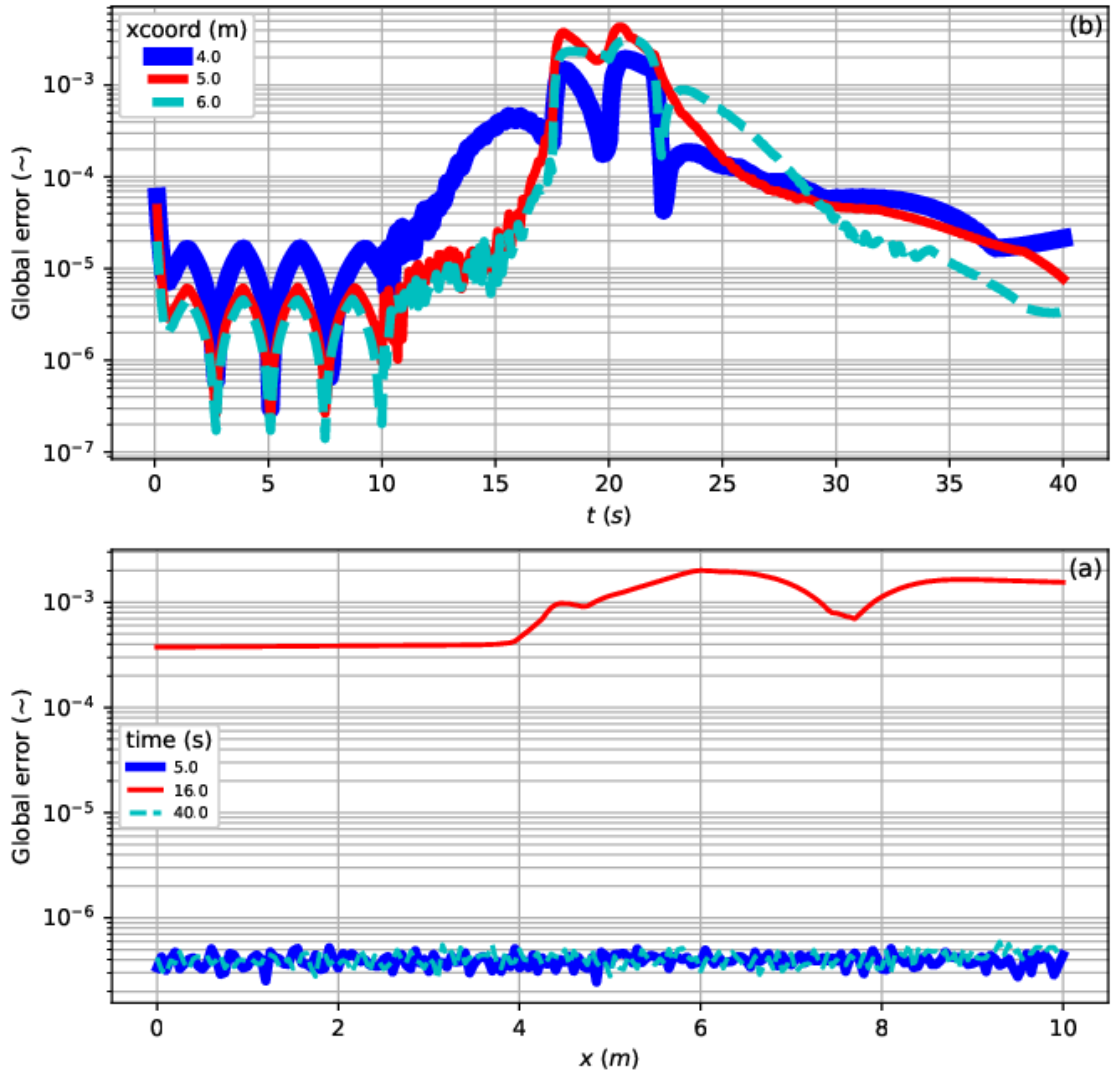


Figure 4.2-10 Benchmark ITER-TF with INTIAL 1. Relative global errors in semi logarithmic scale of the: (a) time evolution errors at $4.0\ m$, $5.0\ m$ *and* $6.0\ m$; *(b) spatial distribution errors at* $5.0\ s$, $16.0\ s$ *and* $40.0\ s$.

The consequence of the superimposition of these non-negligible errors on the properties of each component of the conductor is the different prediction of the solution of the problem during the heating and cooling phases of the cable, which leads to the errors shown in the Figure 4.2-10.

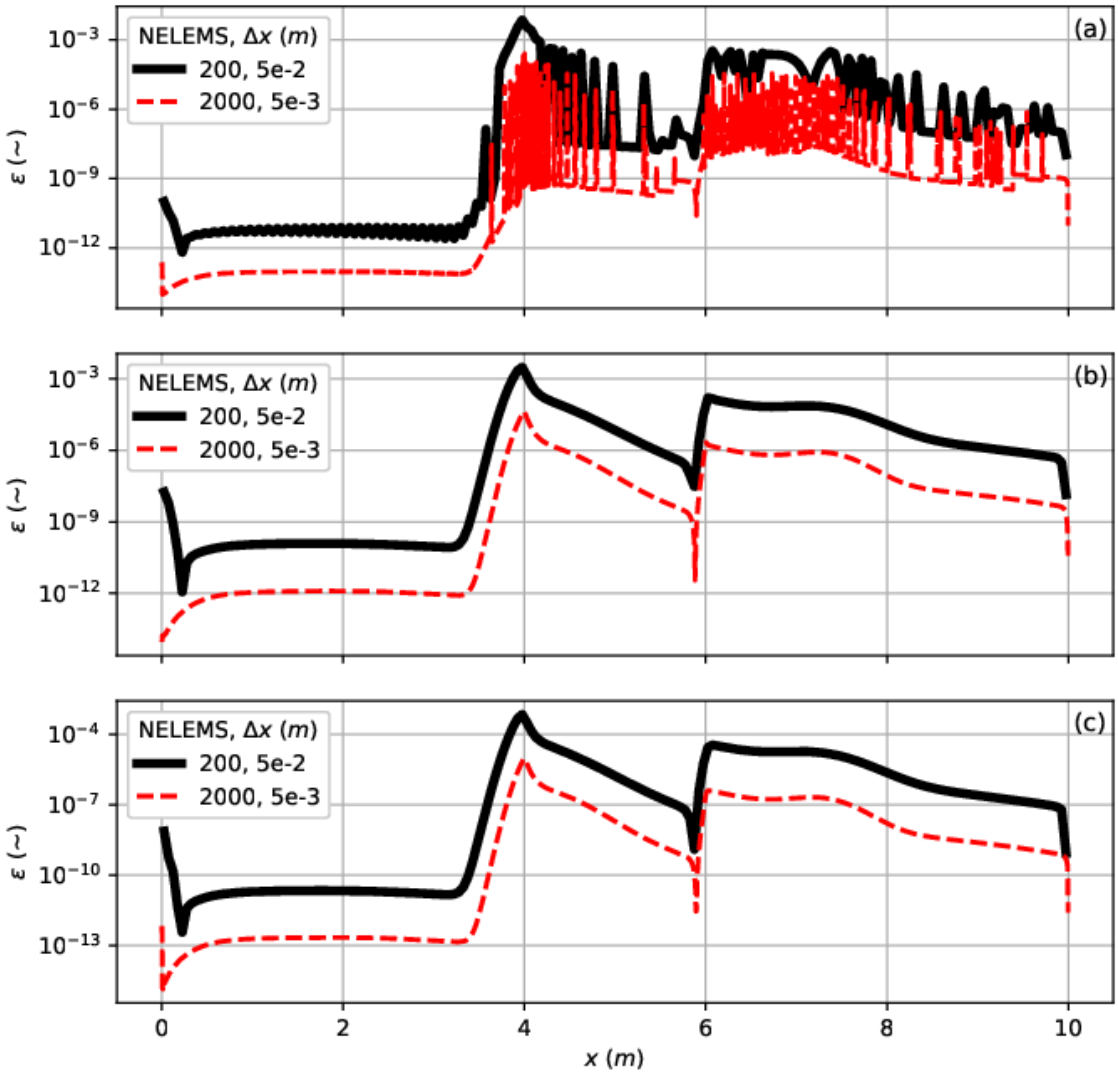


Figure 4.2-11 ITER-TF INTIAL 1: spatial distribution of the global errors at 15 *s from the comparison of the properties evaluated in the Gauss points according to the SC2 and 4C implementations, in semi logarithmic scale at low temperature. Two number of elements for the spatial discretizations are considered: (a) fluid properties, (b) strand properties, (c) jacket properties.*

Moving on to the comparison analysis of the INTIAL 5 case, for which refer to Figure 4.2-12, Figure 4.2-13 and *Figure 4.2-14* which show the time evolutions, spatial distributions and global errors respectively.

The comparison of these simulations has some peculiarities compared to the previous case. Specifically, observing *Figure 4.2-14* it can be seen that the error remains low during the significant phases of the transient: although increasing by an order of magnitude within the heated region, they are a couple of orders of magnitude smaller than in the INTIAL 1 case and comparable with those obtained by the benchmarks with the 3P-HTS geometry. On the other hand, the error during the initialization phase is increased at the inlet throughout the duration of the transient. Both phenomena are directly attributable to the different initialization and application of boundary conditions compared to the case where the INTIAL flag is initialized

at 1, but for different reasons. More details on the different strategies used by SC2 to initialize fluid components and boundary conditions are discussed in section 0.

The justification for the tendentially lower errors during the transient is twofold. Taking into account the figures concerning the time evolutions (Figure 4.2-12) and the spatial distributions (Figure 4.2-13), it is found that the temperature gradients are less steep and the temperature values lower than in the INTIAL 1 case and, consequently, the error resulting from the use of the two methods for the calculation of the properties in the Gauss node is reduced, with a consequent beneficial effect on the comparison of the results obtained with

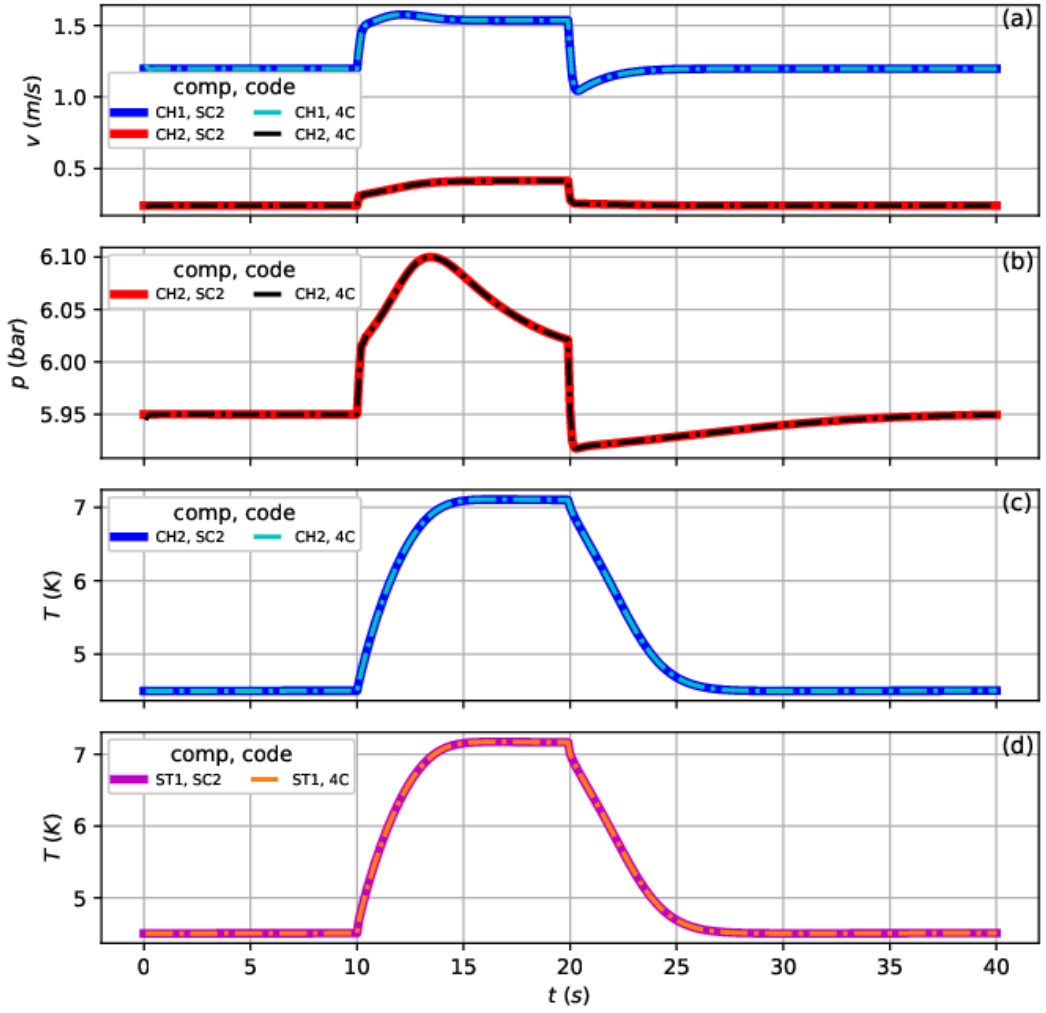


Figure 4.2-12 ITER-TF with INTIAL 5: time evolution variables at 5 m. Solid lines refer to SC2 values, dot-dashed lines refer to 4C values: (a) fluid velocities, (b) fluid pressure, (c) fluid temperature (d) strand temperature.

the two codes, (see Figure 4.2-15). The maximum errors are listed in Figure 4.2-9 *ITER-TF with INTIAL 1: spatial distribution variables at 16 s. Solid lines refer to SC2 values, dot-dashed and dotted lines refer to 4C values: (a) fluid velocities, (b) fluid pressures, (c) fluid temperatures, (d) strand temperature.*

Table 4.2-4 and from a direct comparison with the one obtained with INTIAL 1 results that their value is almost one order of magnitude lower for both the considered spatial discretizations.

4.2 VALIDATION AGAINST THE 4C CODE

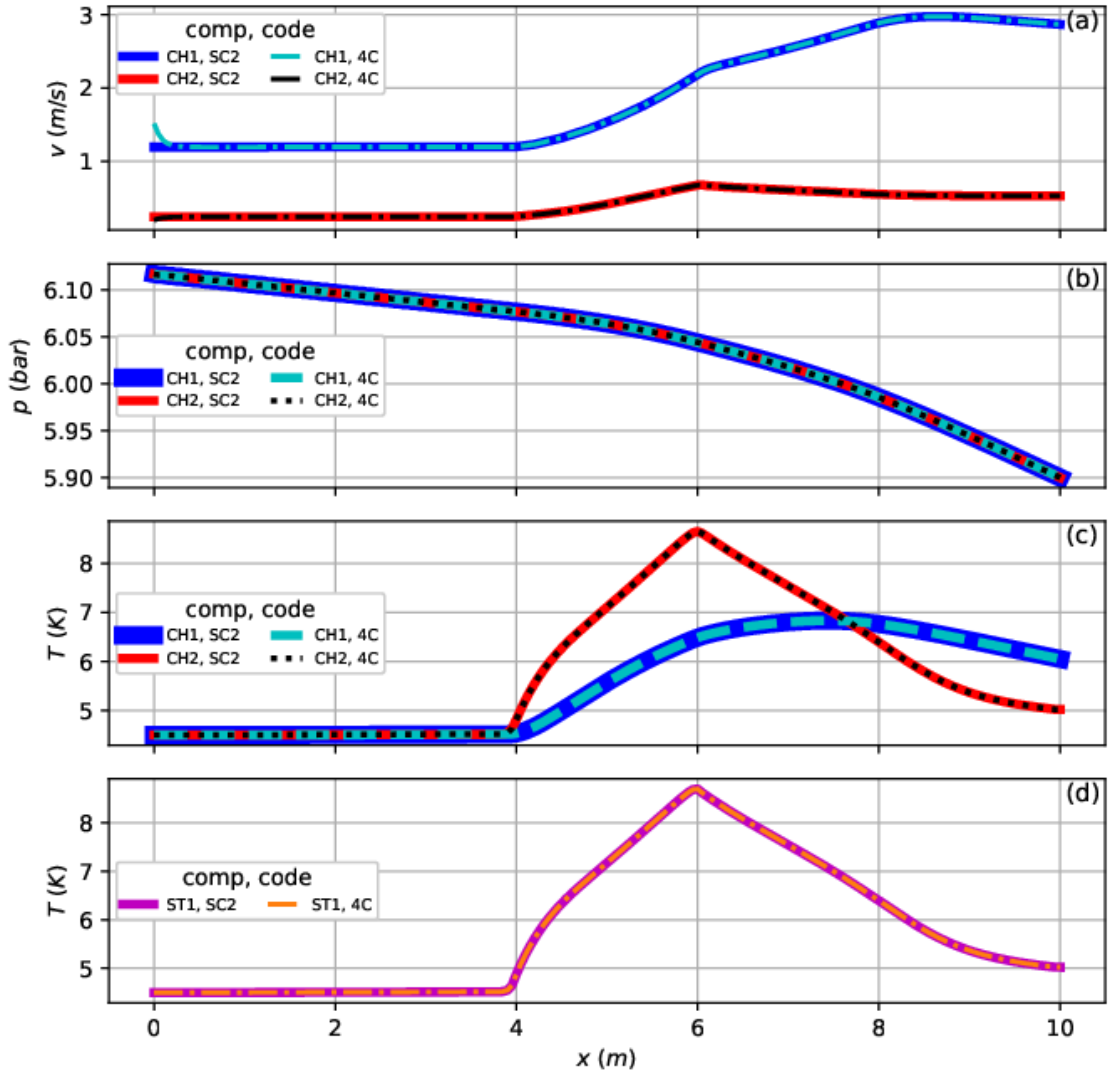


Figure 4.2-13 ITER-TF with INTIAL 5: spatial distribution variables at 16 *s. Solid lines refer to SC2 values, dot-dashed and dotted lines refer to 4C values: (a) fluid velocities, (b) fluid pressures, (c) fluid temperatures, (d) strand temperature.*

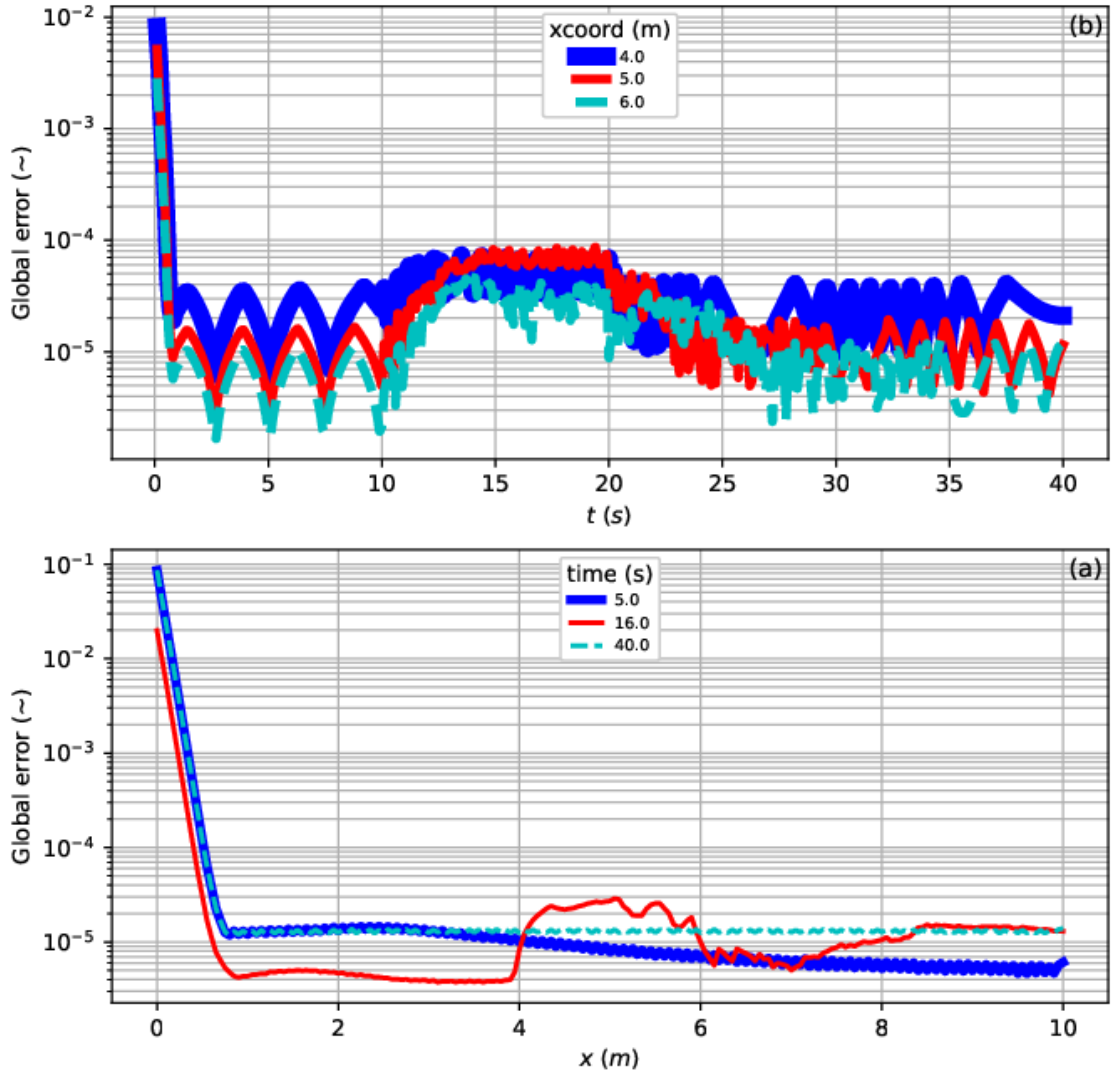


Figure 4.2-14 Benchmark ITER-TF with INTIAL 5. Relative global errors in semi logarithmic scale: (a) time evolution errors at 4.0 m, 5.0 m *and* 6.0 m*; (b) spatial distribution errors at* 5.0 s, 16.0 s *and* 40.0 s.

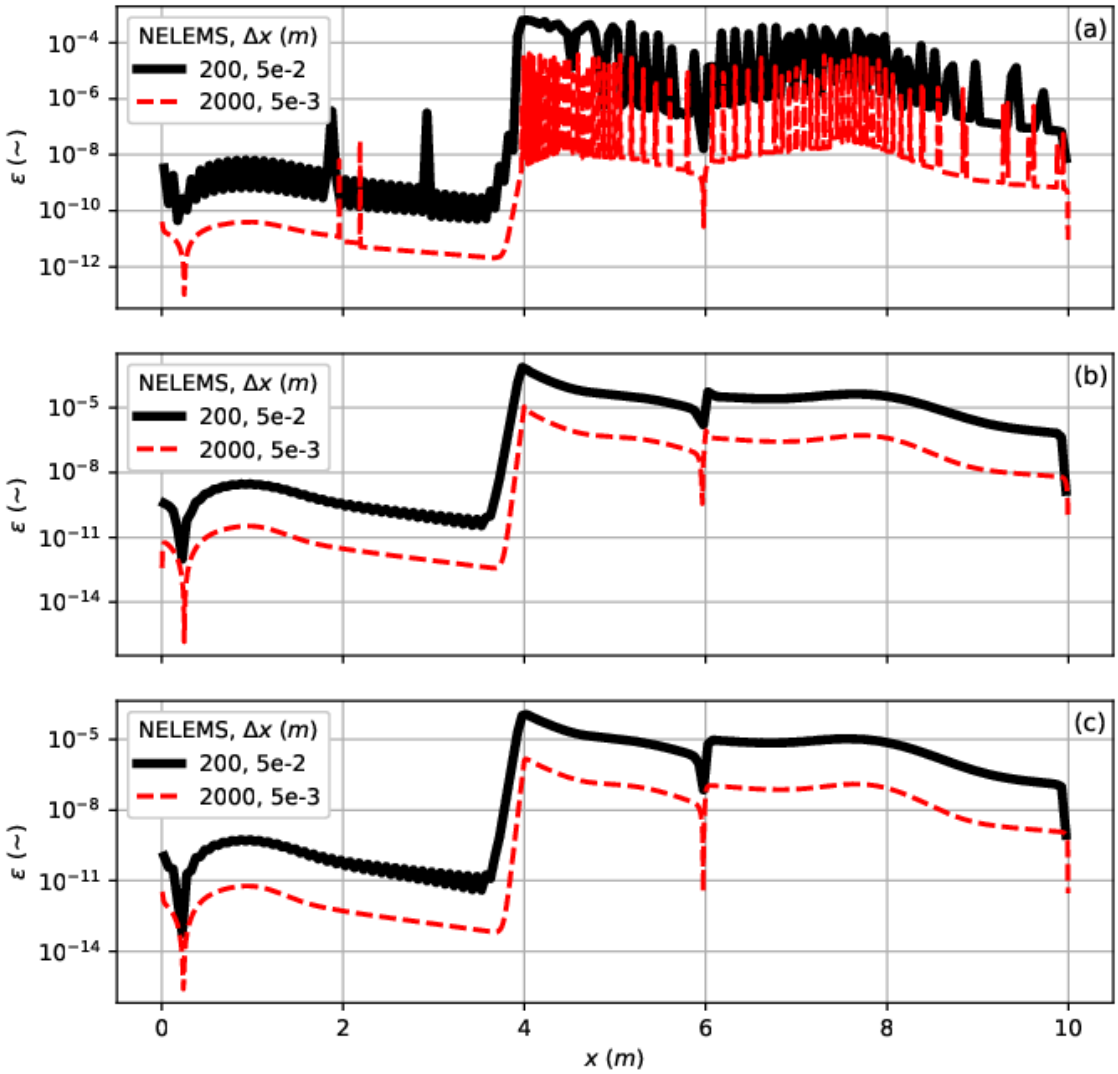


Figure 4.2-15 ITER-TF INTIAL 5: spatial distribution of the global errors at 15 s *from the comparison of the properties evaluated in the Gauss points according to the SC2 and 4C implementations, in semi logarithmic scale at low temperature. Two number of elements for the spatial discretizations are considered: (a) fluid properties, (b) strand properties, (c) jacket properties.*

Regarding the problem at initialization, reference is made to the Figure 4.2-16 which represents an enlargement of the spatial distribution of the He velocity in the two channels. The velocity imposed as boundary conditions on the inlet in the two codes is different, with relative error close to 10%. Both these values are calculated during initialization and then imposed as a boundary condition in the time steps following the initial one. The value calculated by SC2 code is immediately consistent with the one calculated as the solution of the problem in the other nodes of the spatial discretization; on the contrary, 4C predicts a value that proves to be unreliable and the solution better approximates the correct value after a settling length. At 0.6 m distance from the inlet the relative error for the velocity of both fluid components is less than 10^{-4}. It can therefore be concluded that the use of the hydraulic characteristic equation of the channel to compute the initialization of the fluid properties,

provides more accurate results over the whole spatial discretization than the iterative strategy proposed by 4C code.

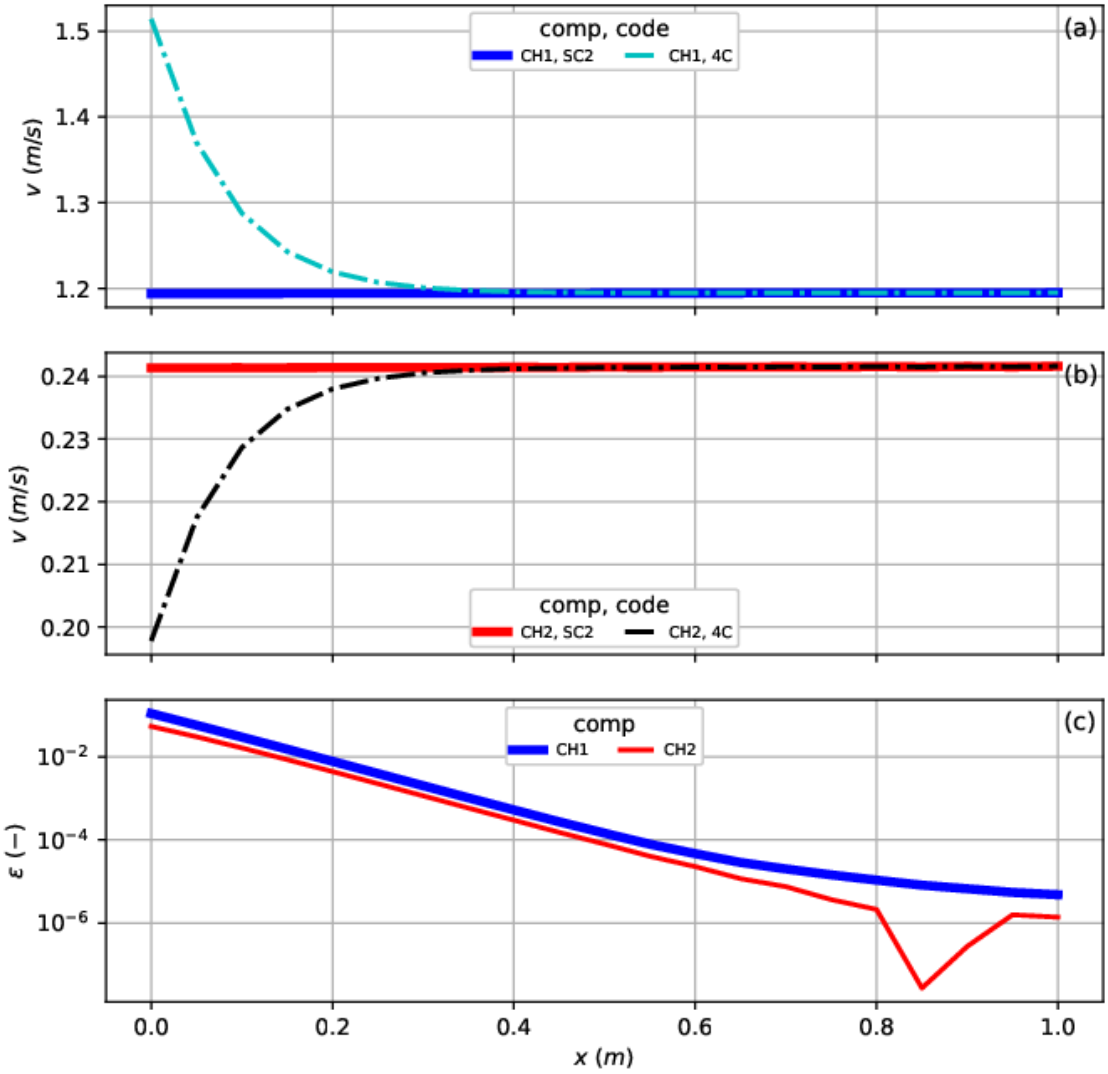


Figure 4.2-16 TER-TF configurations with INTIAL 5: enlargement of the fluid velocities spatial distributions at 16 s. (a) CHAN_1 velocities, (b) CHAN_2 velocities solid lines refer to SC2, dot-dashed lines refer to 4C; (c) relative error in semi logarithmic scale.

In conclusion, excepted the differences highlighted and justified above, the benchmark with INTIAL 5 has a positive result, see *Figure 4.2-14* .

4.2.3 SUMMARY

A series of mandatory benchmarks, with a widely validated and recognized code (such as 4C code), have been carry out to validate the results obtained with the SC2 code. In this running-in phase of the new code, an attempt has been made to minimize the sources of possible differences between the results obtained with the two codes, so the values of the heat transfer coefficients and friction factors have been set at constant and representative values. Two geometries are considered, respectively that of a high-temperature

superconducting cable for power transport (3P-HTS), and the ITER-TF configuration used to build the set of toroidal magnets in the tokamaks, characterized by low-temperature superconducting. For both, two different modes of initialization and application of boundary conditions were compared.

There are two main differences between the codes that lead to small differences in the results: the first is related to the different way in which the flow is initialized, the second is due to the calculation of the properties in the Gauss nodes, which are necessary to construct the elements of the matrix solving the set of equations relevant here.

The effects are in general quite unappreciable for the 3P-HTS cable since, as it consists of only one cooling channel, the two initializations give similar results; moreover, the properties at high temperature are globally more linear, also reducing the differences in the calculation of the properties in Gauss points.

On the other hand, the ITER-TF geometry is more complex due to the presence of two channels in hydraulic parallel for the coolant and consequently the number of degrees of freedom for initialization increases. In fact, the two strategies produce different results, with the one determined with SC2 proving to be more accurate: the benchmark shows a marked difference between the two alternatives, with errors of almost 10% at the inlet. In addition, the material properties are highly non-linear at low temperatures, so the calculation of the properties in Gauss points also has a negative effect on the comparison of the simulations. However, the method adopted in SC2 returns more accurate values as it evaluates the properties at these virtual nodes by re-invoking the functions, rather than assuming that they are the mean value of the properties at the nodes of the spatial discretization, as in 4C.

4.3 SC2 VERSATILITY CHECKS

A series of simulations have been accomplished to check the reliability and self-consistency of the code, in a word its versatility. These checks regard the possibility of imposing different initialization and boundary conditions (section 4.3.1) or to select different numerical schemes for the solution 4.3.2, the ability of the code to handle backward flow (when the outlet pressure is set to a value greater than the inlet one) discussed in section 4.3.3, together with the prospect of using a non-uniform mesh with a refined zone to optimize computational costs, which is the topic of section 4.3.4.

The so-called inner benchmark bases its veracity on the consideration that the validation against 4C was positive in the cases considered.

For the sake of conciseness, the simulations are carried out only for the ITER-TF design. As usual, the common input data can be found in appendix D.2 while in each subsection the specific data are reported to complete the picture.

The data analysis is done exploiting the “Inner Benchmark” cascade of the external post processing tool. Coherently to what already done for the outer benchmark, the inner benchmark error definition is given by:

$$\varepsilon_{inner\ bench} = \frac{|\xi_{Sim1} - \xi_{Sim2}|}{|\xi_{Sim1} - \xi_0|} \quad (4.3\text{-}1)$$

where:

$$\xi_0 = 1.5 \max(|\xi_{Sim1}|) \quad (4.3\text{-}2)$$

A suitable definition of ξ_{Sim1} and ξ_{Sim2} will be given the following sections 4.3.2, 4.3.3 and 4.3.4.

The global error is then defined as the average of the generic errors at a given spatial coordinate or time.

4.3.1 INNER BENCHMARK: INTIAL

The main ways of initializing the flow were presented in section 0 and previously used to benchmark with 4C (sections 4.2.1 and 4.2.2). A recap of the effects of the flag INTIAL on both initialization and boundary conditions according to its value is proposed in Table 4.3-1. Here the goal is twofold: to demonstrate that different outcomes of the simulation correspond to those options, and query them to check that they are in accordance with the physics of the problem being studied.

Expressly, it is expected that from the comparison of two simulations, one with INTIAL 1 the other with INTIAL 2 (or 5) the results will be quite different, while comparing INTIAL 2 with INTIAL 5 the code is expected to return almost the same solution since these options have more affinities than divergences.

Table 4.3-1 Effects on the initialization and on the application of the boundary conditions according to three possible values of the flag INTIAL, as far as fluid components are considered.

INTIAL	Initialization	Boundary condtions	Notes
± 1	p_{inl}	p_{inl}	
	T_{inl}	T_{inl}	Used as BC if $v_{inl} > 0$
	p_{out}	p_{out}	
	T_{out}	T_{out}	Used as BC if $v_{out} < 0$
± 2	$\dot{m}_{inl}$	v_{inl}	v_{inl} from $\dot{m}_{inl}$
	p_{inl}	T_{inl}	Used as BC if $v_{inl} > 0$
	T_{inl}	p_{out}	p_{out} evaluated in flow initialization
	T_{out}	T_{out}	Used as BC if $v_{out} < 0$
± 5	$\dot{m}_{inl}$	v_{inl}	v_{inl} from $\dot{m}_{inl}$
	T_{inl}	T_{inl}	Used as BC if $v_{inl} > 0$
	p_{out}	p_{out}	
	T_{out}	T_{out}	Used as BC if $v_{out} < 0$

4.3 SC2 VERSATILITY CHECKS

The specific input data used to execute the simulations are summarized in Table 4.3-2. The inlet mass flow rate used for the simulation with INTIAL 2 and 5 approximates the ones evaluated with the case INTIAL 1, furthermore note that the imposed pressure value is the same. Having the same input data in common, although calculated from different initialization conditions, the behavior of the simulations is only influenced by the application of boundary conditions.

Table 4.3-2 Input data for the simulation performed at low temperature with the ITER-TF configuration; the second column refers to the simulation with INTIAL 1, the third column shows the data for the simulation with INTIAL 2 while the fourth collects the values for the simulation performed with INTIAL 5

Variable	Value			Unit
METHOD	0 (BE)	0 (BE)	0 (BE)	–
ITMESH	0 (uniform)	0 (uniform)	0 (uniform)	–
NELEMS	200	200	200	–
TEND	15.0	15.0	15.0	s
STPMIN	0.1	0.1	0.1	s
INTIAL	1	2	5	–
TINL	4.5	4.5	4.5	K
PINL	6.	6	–	bar
POUT	5.9	–	5.9	bar
MDTIN_CH1	–	$8.4\ 10^{-3}$	$8.4\ 10^{-3}$	kg/s
MDTIN_CH2	–	$1.248\ 10^{-2}$	$1.248\ 10^{-2}$	kg/s
Q0	250	250	250	W/m
XQBEG	4.0	4.0	4.0	m
XQEND	6.0	6.0	6.0	m
TQBEG	10.0	10.0	10.0	s
TQEND	20.0	20.0	20.0	s

For this specific kind of inner benchmark, it is not meaningful to define an error since the simulations are different. Firstly, the inner benchmark INTIAL 1 against INTIAL 5 is considered and then the one INTIAL 2 against INTIAL 5.

On the basis of the above, the initial spatial distribution of the variables of both simulations must be the same as confirmed by *Figure 4.3-1*. The initialization of this simulation is characterized by a constant temperature spatial distribution at 4.5 K for all components (two fluids and solid ones) since the INTIAL 0 is considered for the lattes (see Table D.2-5 and

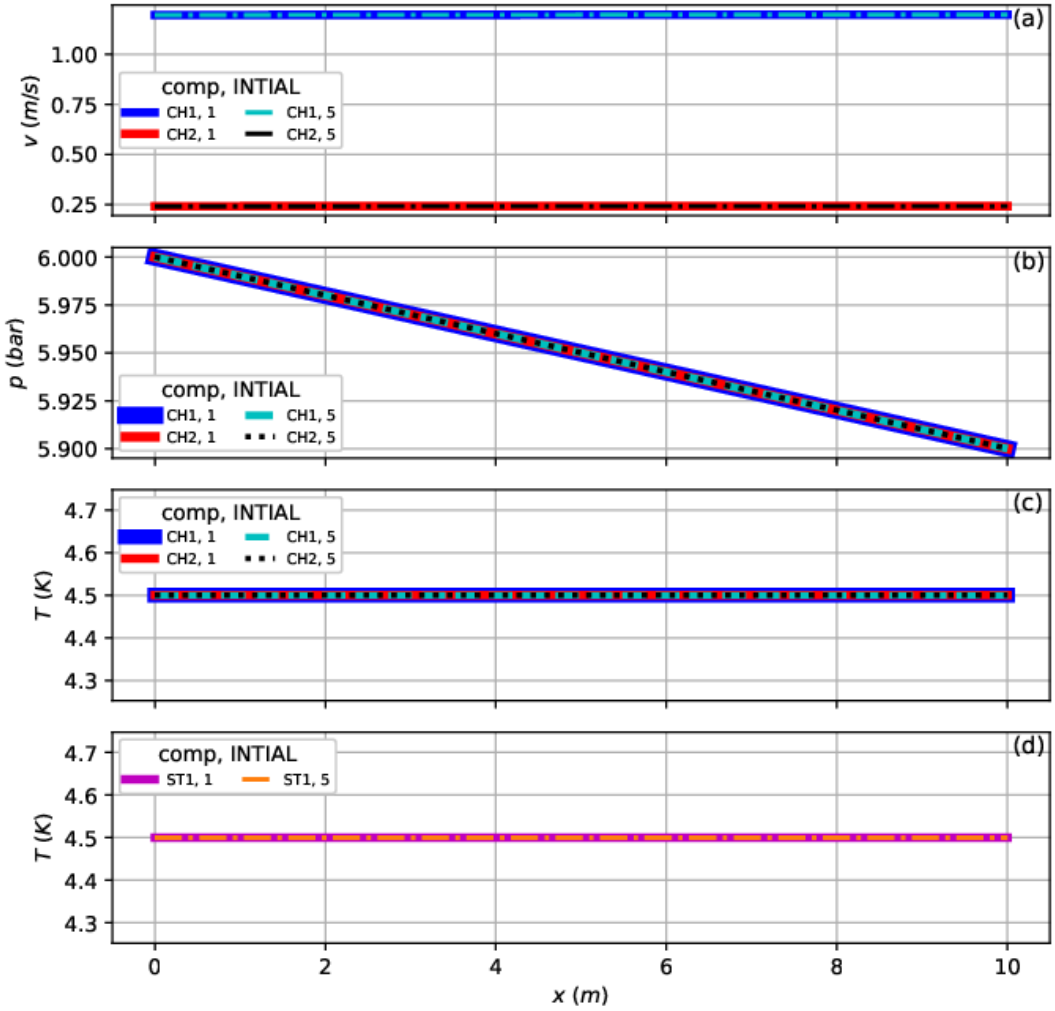


Table D.2-6), and a linear pressure spatial distribution from the inlet value to the outlet one. It is worth mention that the pressure behavior is the same in both the cooling channels, as expected since they are in hydraulic parallel, and thus the pressure is equalized. Also, the initial velocity spatial distribution is linear, it is not evident from the *Figure 4.3-1* due to the very small slope. Notice that curves perfectly overlap, meaning that the initializations are equivalent.

Figure 4.3-1 Comparison of the initial spatial distribution of the ITER-TF cable simulations with INTIAL 1 and INTIAL 5. Solid lines refer to INTIAL 1, dot-dashed and dotted lines refer to INTIAL 5. (a) fluid components velocities, (b) fluid components pressures, (c) fluid components temperatures, (d) strand temperature.

The effects of the different boundary conditions application can be seen in Figure 4.3-2 that compares the time evolutions of the solution variables at the inlet for INTIAL 1 and INTIAL 5, as well as in *Figure 4.3-3* and *Figure 4.3-4* that respectively show the variable time evolutions in the center of the conductor and the spatial distribution at 15 s, in the middle of the heating phase.

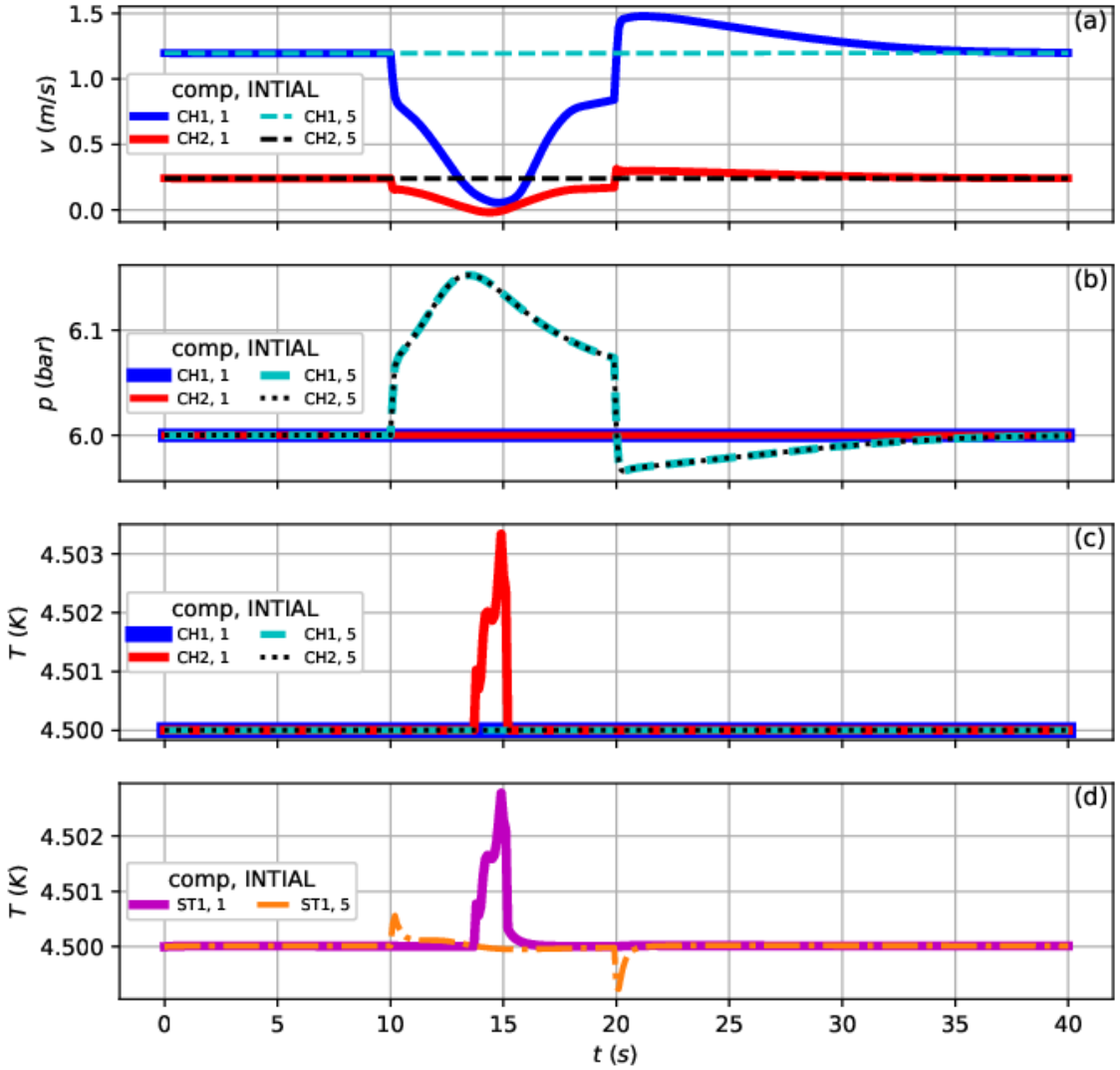


Figure 4.3-2 Comparison of the inlet time evolution of the ITER-TF cable simulations with INTIAL 1 and INTIAL 5. Solid lines refer to INTIAL 1, dot-dashed and dotted lines refer to INTIAL 5. (a) fluid components velocities, (b) fluid components pressures, (c) fluid components temperatures, (d) strand temperature. The simulations end at $100.0\ s$ *but only the first* $40.0\ s$ *are shown to improve the readability.*

It is noteworthy that the two initializations and set of boundary conditions are almost equivalent before the heating period starts and after that a new steady state is restored, confirming that the initializations are equivalent from the point of view of the calculated values. The differences raise during the heating and cooling phases of the transient because the physic is different for the two sets of boundary conditions.

During the first $10\ s$ of the transient the solution does not change that much as can be seen from both Figure 4.3-2 and *Figure 4.3-3*. The temperatures are flat and almost equal to the initial value (a small increase is due to the friction in the cable), both pressure and velocity distributions are still linear and very close to the initial ones.

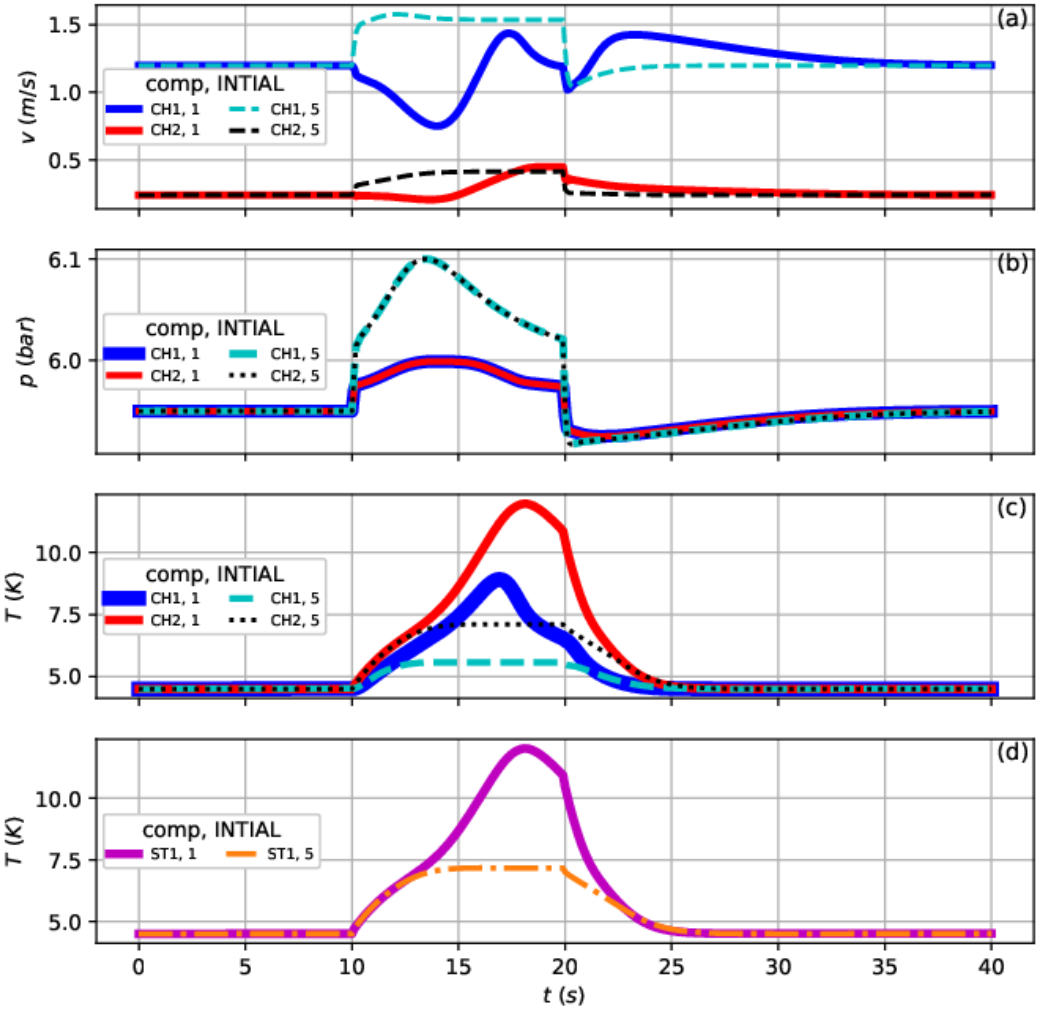


Figure 4.3-3 Comparison of the time evolution variables for the ITER-TF cable simulations performed with INTIAL 1 and INTIAL 5 at 5 m. *Solid lines refer to INTIAL 1, dot-dashed and dotted lines refer to INTIAL 5. (a) fluid components velocities, (b) fluid components pressures, (c) fluid components temperatures, (d) strand temperature. The simulations end at* 100.0 s *but only the first* 40.0 s *are shown to improve the readability.*

At 10 s the heating is switched on and the heating phase of the transient begins, the effects being the raise in temperature of the strand (and of the other cable components due to the heat transfer), and the coolant pressurization which manifests differently depending on the imposed boundary conditions. On the one hand, when the inlet and outlet pressure of the fluids are imposed, the inlet pressure is constant in time (as well as the outlet one) while the velocity changes according to the pressure drop, as visible in Figure 4.3-2 (a). On the other hand, when only the outlet pressure is imposed as boundary condition, its value at the inlet can vary such that the inlet velocity is constant, as can be inferred from Figure 4.3-2 (b). In both cases within the conductor, the pressure will change consistently with the physics described by the two cases, while the velocity is ruled by the local pressure drop, as shown in *Figure 4.3-3* (a, b) and *Figure 4.3-4*.

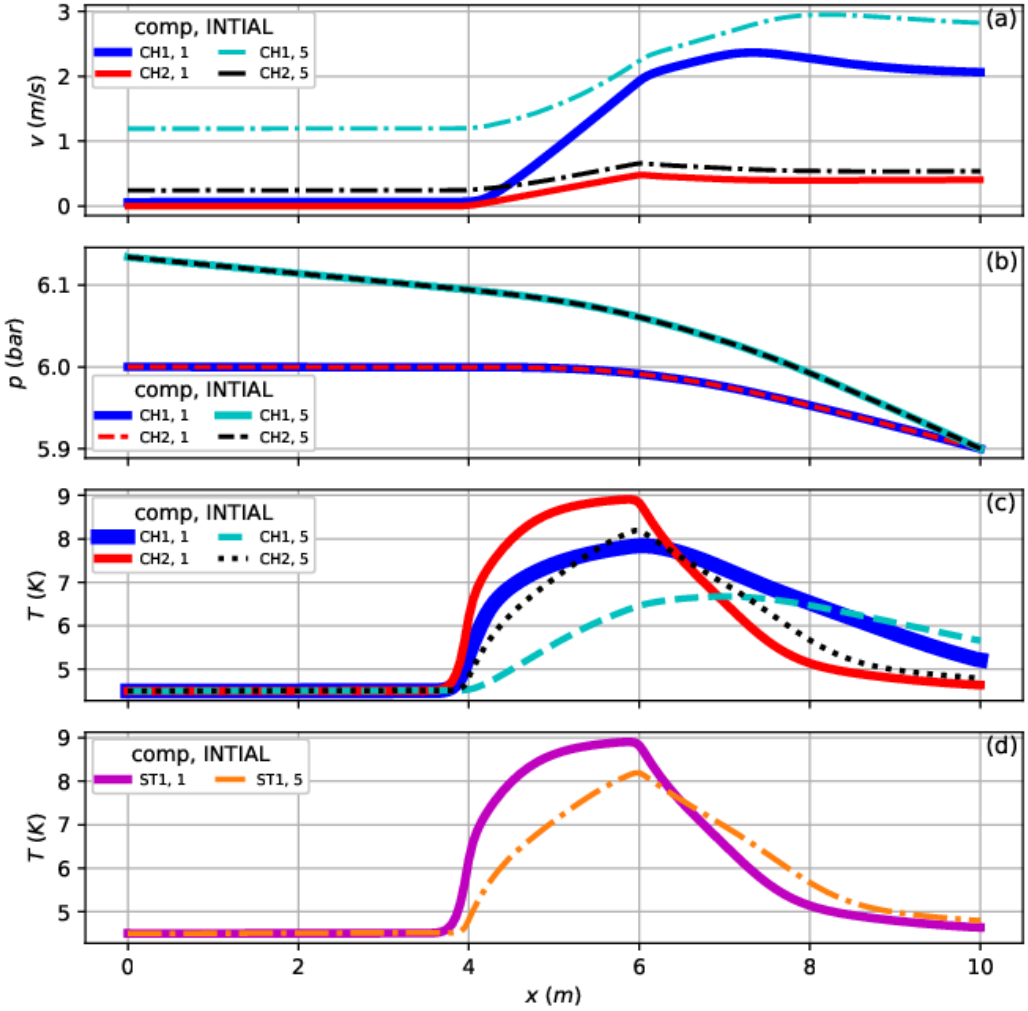


Figure 4.3-4 Comparison of the spatial distribution variables for the ITER-TF cable simulations performed with INTIAL 1 and INTIAL 5 at 15 s*. Solid lines refer to INTIAL 1, dot-dashed and dotted lines refer to INTIAL 5. (a) fluid components velocities, (b) fluid components pressures, (c) fluid components temperatures, (d) strand temperature.*

Focusing on the spatial distributions shown in *Figure 4.3-4*, at the considered time step both the heating effects are appreciable: the first is the pressurization on the channels, the second the global temperature raise of the conductor components. With INTIAL 1, the pressure is flat and almost equal to the inlet value up to the lower edge of the heated region, consequently the velocity is almost zero; beyond this spatial coordinate, the pressure starts to decrease, so the local pressure drop is not zero and the velocity raises. The different velocity distribution along the channels which, for part of its length, is characterized by a practically zero flow rate, justifies the spatial temperature distribution of the strand (and consequently those of the other components). As a matter of fact, along the first 4 m of the conductor, the main heat transfer mechanism is the conduction that has a limited effect in a region close to the beginning of the heated zone. Inside this region there is a sharp temperature gradient which decreases progressively as the velocity increases, since more heat is removed by convection.

Outside the heated region the velocity still increases leading to a sharp negative temperature spatial gradient.

Generally, when both the inlet and the outlet pressure are imposed as boundary conditions it could happens that, at some spatial coordinates, the local pressure value is larger than the inlet value. If such condition verifies, a backflow occurs: upstream the velocity is negative since the coolant flow is reversed, downstream the velocity is positive and large since the pressure drop is larger than the initial one, with dramatic consequences on the temperature of the strand.

When INTIAL 5 is taken into account, the physics is different from that described so far; the key is the different pressure distribution shown in *Figure 4.3-4* (b). In this case the cable pressurization involves the inlet as well, so the velocity never goes below the inlet value, increasing where the effects of pressurization are most pronounced. This means that in this configuration a back flow cannot begin since the maximum pressure is always at the inlet; moreover, the imposed mass flow rate allows a more efficient heat removal with respect to the previous case, as can be seen from the less steep gradients in *Figure 4.3-4* (c, d).

The same physics can be seen from a different point of view in *Figure 4.3-3*. The different shape of the temperature evolution is remarkable, as shown in *Figure 4.3-3*: very pronounced gradients for INTIAL 1, compared to the smoother shape of INTIAL 5, again related to the different pressurization which lead to a different velocity behavior, which has a smaller variation in the case of INTIAL 5 than INTIAL 1. It should also be noted that, with the same INTIAL, there is a different value for the He temperatures in the two channels. The lower temperature in the hole compared to the bundle can be understood for two reasons. Firstly, this channel is not in direct contact with the strand, but the heat flows by conduction and convection through the spiral separating it from the bundle, therefore it is not directly affected by the heating effects. In addition, as the velocities in the two channels are different, the information about the temperature change propagates with different timing. Table 4.3-3 collects the maximum values of the channel pressure and of the strand temperature at the considered time evolution and spatial distribution.

Table 4.3-3 Maximum values of the fluid components pressure and of the strand temperature for the low temperature ITER-TF simulation with INTIAL 1 and INTIAL 5. The relative difference is also shown in the last row of the table. In the first and third columns are reported the maximum values of the time evolution at $5\ m$*, while in the second and in the fourth columns are reported the maximum values of the space distribution at* $15\ s$*.*

	Pressure $\boldsymbol{MPa}$		Temperature $\boldsymbol{K}$	
	$\mathbf{5}\ \boldsymbol{m}$	$\mathbf{15}\ \boldsymbol{s}$	$\mathbf{5}\ \boldsymbol{m}$	$\mathbf{15}\ \boldsymbol{s}$
INTIAL 1	0.5999	0.6	12.02	7.84
INTIAL 5	0.6099	0.6134	7.17	6.67
Relative difference %	1.68	2.24	40.32	14.92

Figure 4.3-2 (c) shows that between 13.7 s and 15 s in the bundle (CH2) the inlet temperature is not equal to the imposed value. This may appear an odd event, but it can be explained remembering how the boundary conditions are applied. In Table 4.3-1 is remarked that the inlet temperature is imposed only if the inlet velocity is larger than 0; since between

13.7 s and 15 s the velocity is negative, this boundary condition is not imposed. Comparing the peak temperature of the strand and of the bundle, it turns out that the channel temperature is 0.1% larger than the strand one, but this is only related to numeric.

Twenty seconds after the start of the transient, the heat source is switched off and from the time evolution, see *Figure 4.3-3* (d), it can be seen that the temperature in the strand, and therefore in all the other conductor components decreases: the second phase of the transient, cooling down, begins. Fluid pressure drastically reduces, bringing velocities down to near pre-heating values. At 30 s the temperature at 5 m has almost reached the initial value but the heat is not yet exhausted since it propagates along the conductor.

At the end of the simulation (100 s) the steady state is restored.

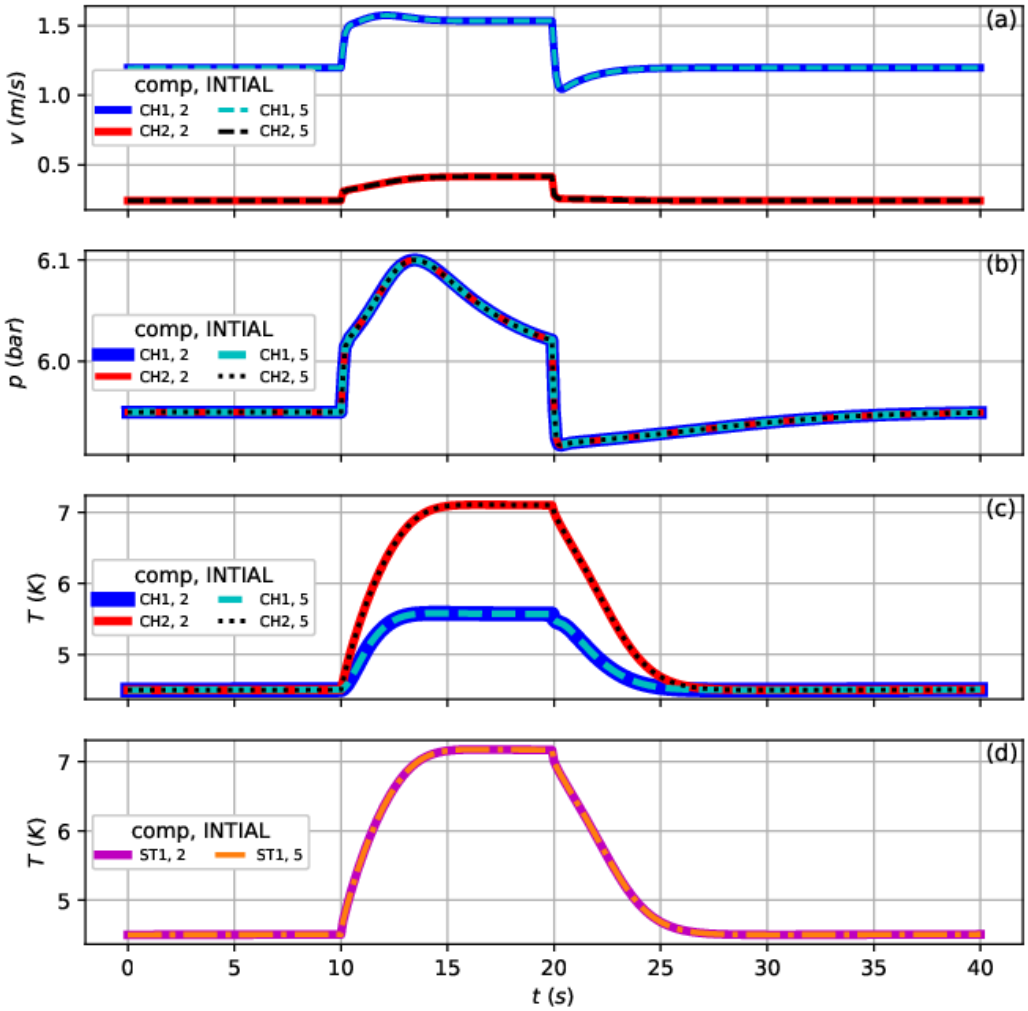


Figure 4.3-5 Comparison of the time evolution variables for the ITER-TF cable simulations performed with INTIAL 2 and INTIAL 5 at 5 m. Solid lines refer to INTIAL 2, dot-dashed and dotted lines refer to INTIAL 5. (a) fluid components velocities, (b) fluid components pressures, (c) fluid components temperatures, (d) strand temperature. The simulations end at 100.0 s but only the first 40.0 s are shown to improve the readability.

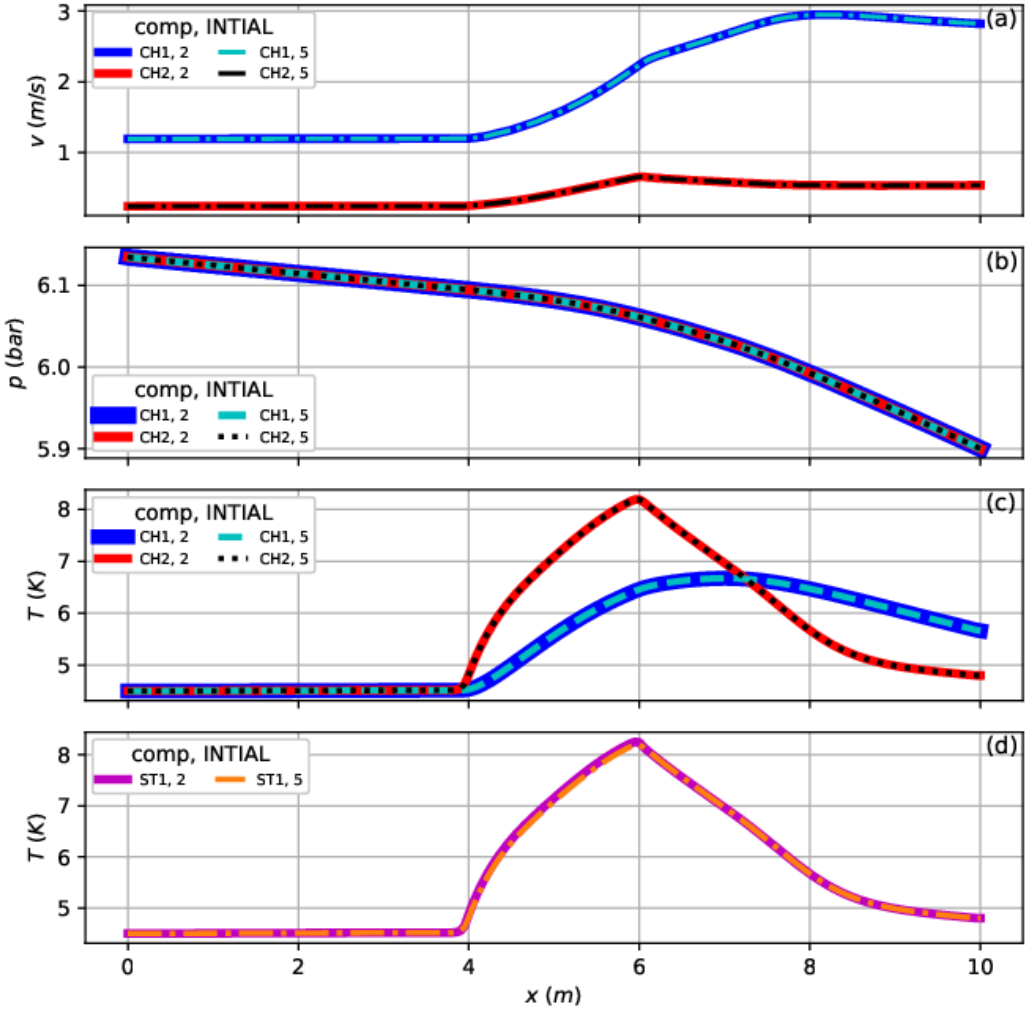


Figure 4.3-6 Comparison of the spatial distribution variables for the low temperature ITER-TF simulations performed with INTIAL 2 and INTIAL 5 at 15 s. *Solid lines refer to INTIAL 2, dot-dashed and dotted lines refer to INTIAL 5. (a) fluid components velocities, (b) fluid components pressures, (c) fluid components temperatures, (d) strand temperature.*

Ultimately, imposing both pressures at the inlet and outlet limits the pressurization of the cable but can lead to a partial absence of coolant flow, or worse a backflow, with negative effects on strand temperature. On the other hand, imposing the inlet velocity, while having the advantage of limiting the peak temperature, subjects the cable to non-negligible mechanical stress due to the inlet pressure increase.

The section ends commenting the inner benchmark INTIAL 2 against INTIAL 5. The main difference between the two options is that in the former the outlet pressure is evaluated starting from the information at the inlet in the initialization phase (as discussed in section 0), while in the latter the outlet pressure is given as input data. Also in this case, the input data presented in Table 4.3-2 are such that the simulations outcome is expected to be the same. The time evolutions and the spatial evolution of the solution variables are shown in Figure 4.3-5 and *Figure 4.3-6*. All the curves perfectly overlap largely confirming what was foreseen.

4.3.2 INNER BENCHMARK: BE VS CN

The time convergence analysis was performed considering both the methods available in the SC2 code to integrate the ODE system of equations. This section shows that both methods tend to the same solution when the same input data are given. The specific data are the same of the second column of Table 4.3-2, i.e. INTIAL 1 is considered, but four simulations are compared that differs for the time steps and the solution method according to the following Table 4.3-4, grouped into two cases by the former input data.

Table 4.3-4 Minimum time steps and methods to integrate the final ODE system use to compare the outcome obtained with Backward Euler and Crank-Nicolson numerical schemes for the ITER-TF configuration with INTIAL 1.

	S1	S2	S3	S4	Unit SI
STPMIN	0.1		0.01		s
METHOD	0 (BE)	1 (CN)	0 (BE)	1 (CN)	–

The global errors of the spatial distributions and the time evolutions are evaluated taking as reference simulation the one that implements the Backward Euler numerical scheme; therefore, for the first case:

$$\xi_{Sim1} = \xi_{BE,0.1s}$$

$$\xi_{Sim2} = \xi_{CN,0.1s}$$

The trend of the overall error is plotted in Figure 4.3-7. The order of magnitude of both the time evolutions errors (see Figure 4.3-7 (a)) and the spatial distributions errors (see Figure 4.3-7 (c)), when STPMIN is set to 0.1 s, is a sign that the two methods return a rather different solution regardless of the time considered for the spatial distribution and the spatial coordinates considered for the time evolutions. The only exception is the initialization ($time = 0$ s), when the properties have the same value by definition. Decreasing by one order of magnitude the time step the distance between the solutions is shortened, as evidenced by the fact that the overall error is reduced by several orders of magnitude in Figure 4.3-7 (b, d).

The larger reduction of the error is for times lower than TQBEG and much larger than TQEND when the cooling is almost completed. To give an idea, comparing the spatial distribution error at 5 s of Figure 4.3-7 (c, d) there are almost seven orders of magnitude of difference. Moreover, looking at the time evolutions Figure 4.3-7 (a, b), for the first case the error increases rapidly, irrespective of the spatial coordinate considered, and even after the end of heating, it does not decrease. On the contrary for the latter, at each spatial coordinate, the error stabilizes at a small value before the beginning of the heating and reduces as the heat introduced is dissipated. The error is still substantial during the heating up and cooling down of the strand, being related to the different numeric of the methods, and more visible when the heat source is switched on and off. To get reasonable results, a small time step should be considered when Crank-Nicolson scheme is selected.

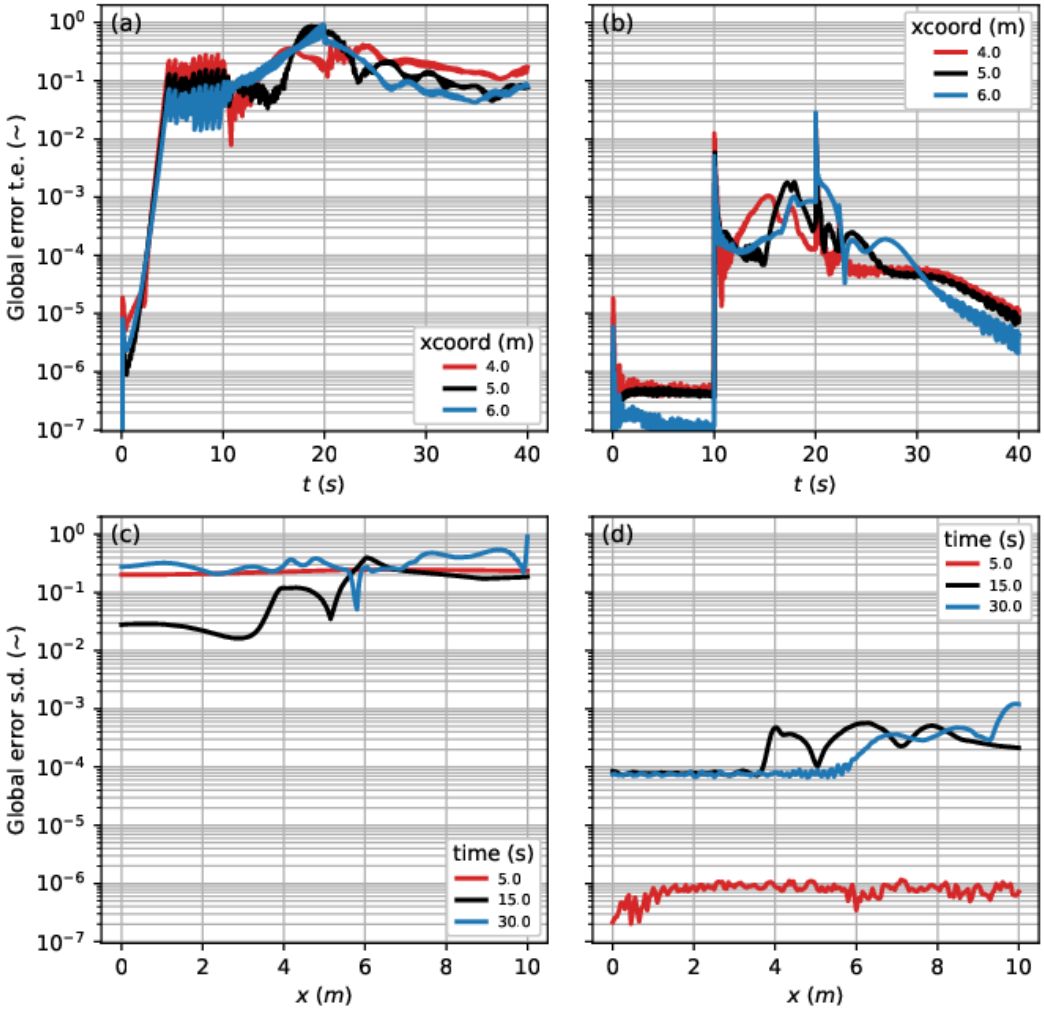


Figure 4.3-7 Global errors in semi logarithmic scale for the comparison between Backward Euler and Crank-Nicolson with INTIAL 1 for the ITER-TF cable configuration. (a) time evolutions with $STPMIN = 0.1\ s$ *at* $4.0\ m$, $5.0\ m$ *and* $6.0\ m$ *; (b) time evolutions with* $STPMIN = 0.01\ s$ *at* $4.0\ m$, $5.0\ m$ *and* $6.0\ m$ *; (c) spatial distributions with* $STPMIN = 0.1\ s$ *at* $5.0\ s$, $15.0\ s$ *and* $30.0\ s$ *;(d) spatial distributions with* $STPMIN = 0.01\ s$ *at* $5.0\ s$, $15.0\ s$ *and* $30.0\ s$.

4.3.3 INNER BENCHMARK: BACKFLOW

So far, a considerable fraction of the simulations has been carried out with the initialization option set to 1, in comments to the results the inlet corresponds to the left end of the cable and the outlet to the right one. This is only a convention, from a strictly physical point of view, as the inlet corresponds to the coordinate at which the pressure is maximum and the outlet to that at minimum pressure. Therefore, if practically the coordinates where the pressures are maximum and minimum are exchanged, i.e., the maximum pressure is imposed at the right end of the conductor and the minimum at the left one as shown in Figure 4.3-8, the inlet and outlet are also exchanged and the result, all else being equal, physically does not change.

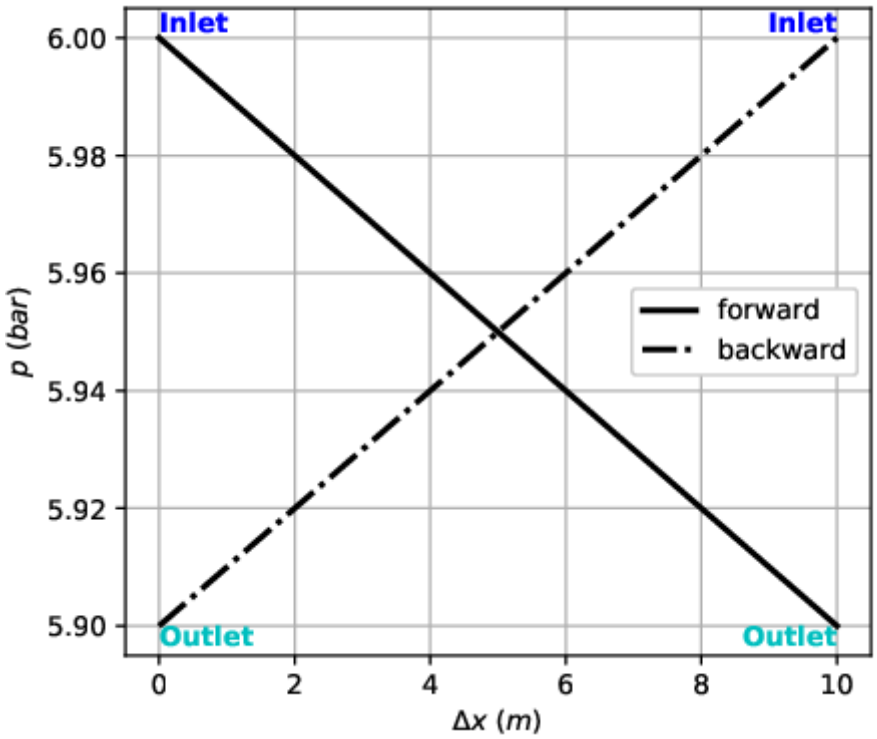


Figure 4.3-8 Relationship between the maximum and minimum pressure and inlet and outlet flow distribution for the forward and the backward configurations.

The aim of this section is to proof that the SC2 code can abide by this basic law of fluid dynamics. In the light of the above, the use of INTIAL 1 is again justified: it is the only option among those considered that allows the pressures at the ends of the cable to be imposed.

The analysis is carried out considering three sets of simulations each characterized by a given imposed pressure drop on the cable and composed by two simulations that differs from the values of the pressure at the ends of the conductor. The reference pressure drop, 0.1 bar, is the one adopted also in the previous sections; the other two are respectively one fifth and five times the reference value, namely 0.02 bar and 0.5 bar. Table 4.3-5 collects the input data shared by the six simulations, while Table 4.3-6 focuses on the initialization of the simulations; geometry, topology, heat transfer coefficients and the like are defined in appendix D.2. Please, note that to swap the inlet and the outlet in the simulation, the inlet and outlet pressure in the backward simulations are shifted:

$$p_{inl,back} = p_{out,forw} \tag{4.3-3}$$

$$p_{out,back} = p_{inl,forw} \tag{4.3-4}$$

Moreover, all the simulations have the same maximum (inlet) pressure, while the minimum (outlet) changes to get the desired pressure drop.

As far as the error evaluation is of concern, conventionally the reference simulation is the one with the flow from left to right, called forward flow (shortly forw), while the test simulation has the flow from right to left and it is called backward flow (or back), thus:

$$\xi_{Sim1} = \xi_{forw} \tag{4.3-5}$$

$$\xi_{Sim2} = \xi_{back} \tag{4.3-6}$$

Table 4.3-5 Input data for all six simulations used to check the consistency of a forward and a backward flow for the ITER-TF configuration at low temperature with INTIAL 1.

Variable	Value	Unit SI
METHOD	0 (BE)	–
ITMESH	0 (uniform)	–
NELEMS	200	–
INTIAL	1	–
TEND	15.0	s
STPMIN	0.1	s
Q0	250	W/m
XQBEG	4.0	m
XQEND	6.0	m
TQBEG	10.0	s
TQEND	20.0	s

Table 4.3-6 Initialization values for INTIAL1 according to the required pressure drop and the kind of simulation, namely forward or backward flow. Notice that the inlet and outlet pressure values exchanges moving from a forward to a backward simulation. Selected configuration is ITER-TF at low temperature.

	$\Delta p = 0.1$ bar		$\Delta p = 0.02$ bar		$\Delta p = 0.5$ bar		Unit
Variable	forward	backward	forward	backward	forward	backward	
TINL	4.5	4.5	4.5	4.5	4.5	4.5	K
PINL	6	5.9	6	5.98	6	5.5	bar
POUT	5.9	6	5.98	6	5.5	6	bar

To practically compute the error the array ξ_{back} should be tipped over; moreover, if it is a velocity, its sign should also be changed as its direction is opposite to the reference one.

Spatial distributions of the solution exemplify the outcomes of the comparison and are summarized in the following *Figure 4.3-9*, *Figure 4.3-10* and *Figure 4.3-11*.

4.3 SC2 VERSATILITY CHECKS

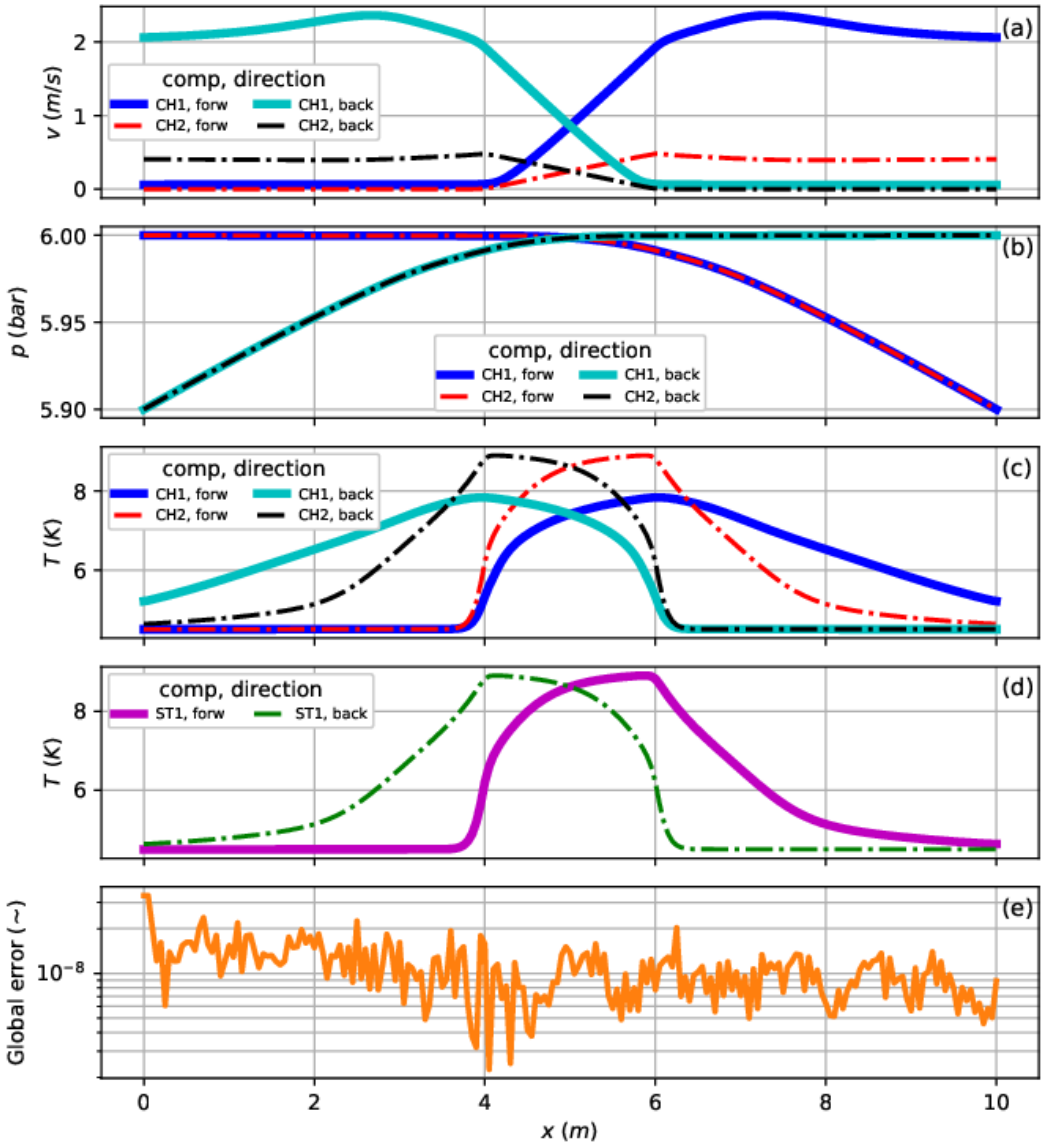


Figure 4.3-9 Comparison of the spatial distributions obtained with the forward and backward flow simulations for the low temperature ITER-TF configuration with INTIAL 1 and $\Delta p = 0.1$ bar *at* 15 s*. Solid lines refer to the forward flow, dot-dashed lines refer to the backward flow. (a) fluid components velocities; (b) fluid components pressure; (c) fluid components temperatures; (d) strand temperature; (e) global error in semi logarithmic scale.*

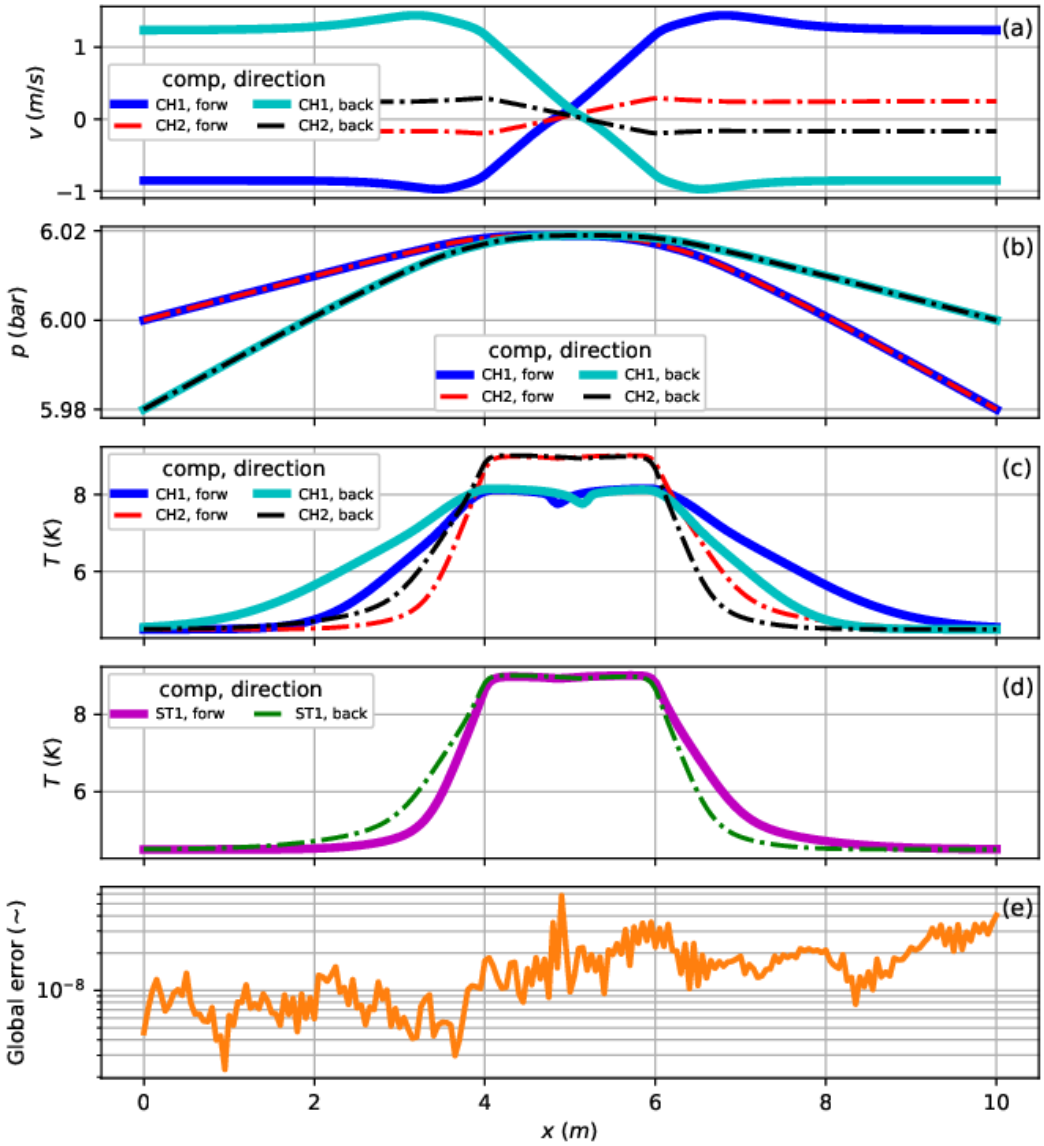


Figure 4.3-10 Comparison of the spatial distributions obtained with the forward and backward flow simulations for the low temperature ITER-TF configuration with INTIAL 1 and $\Delta p = 0.02$ bar *at* 15 s. *Solid lines refer to the forward flow, dot-dashed lines refer to the backward flow. (a) fluid components velocities; (b) fluid components pressure; (c) fluid components temperatures; (d) strand temperature; (e) global error in semi logarithmic scale.*

The pairs of curves represented in the first four subplots of *Figure 4.3-9*,*Figure 4.3-10* and *Figure 4.3-11* are symmetrical with respect to the midpoint of the spatial coordinate, that is the center of gravity of the cable, which shows that the code distinguishes the position of the inlet and outlet in the two cases. The final proof can be found in subplots (e) of *Figure 4.3-9*, *Figure 4.3-10* and *Figure 4.3-11*, that show, for all the three sets of simulation considered, an almost negligible value of the global relative error. That means that the SC2 recognizes where the inlet and the outlet are located according to the imposed pressure values and that the fluid dynamics implemented in the code is the same, regardless of their location.

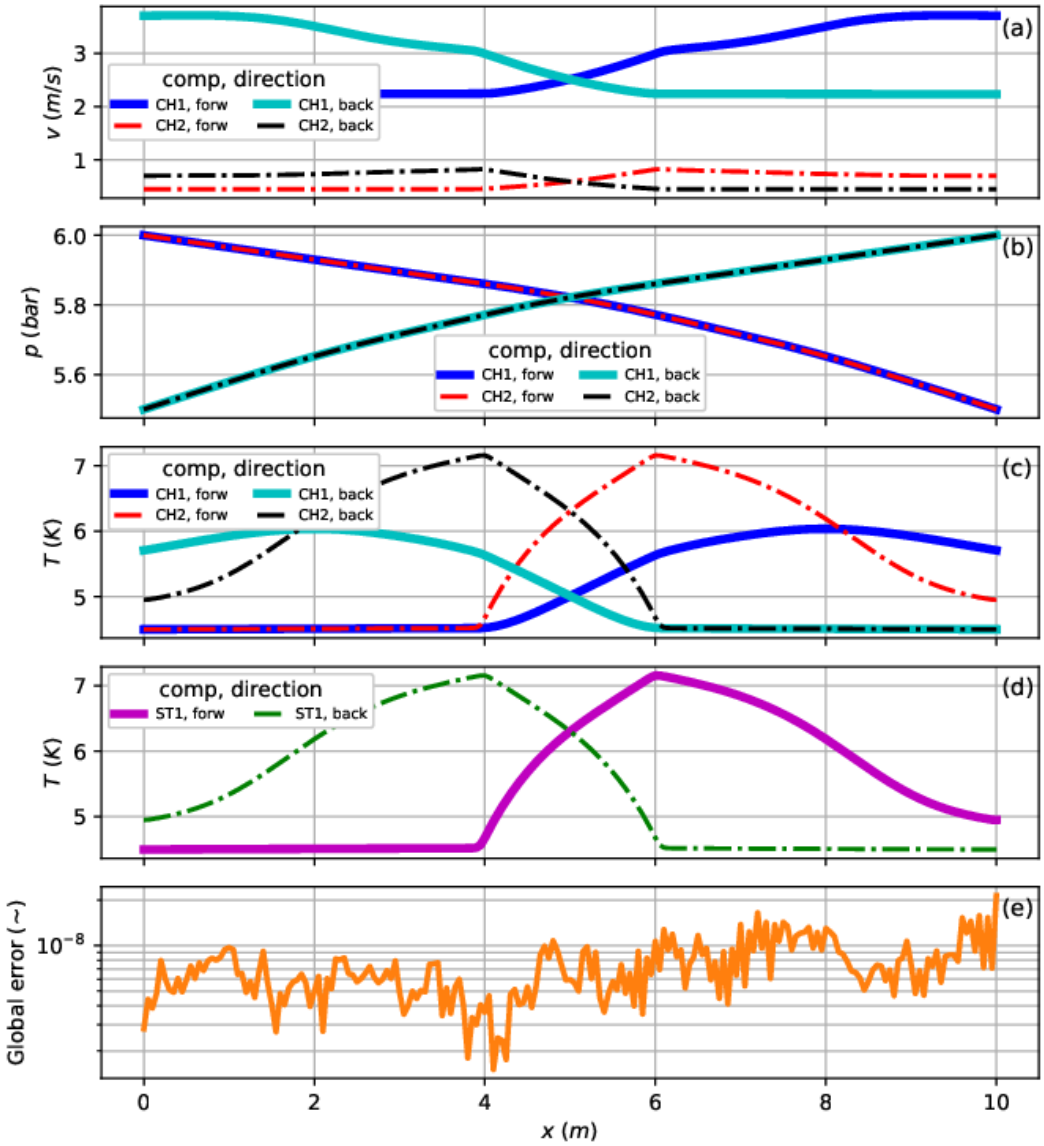


Figure 4.3-11 Compare spatial distributions results obtained with the forward and backward flow simulations for the low temperature ITER-TF configuration with INTIAL 1 and $\Delta p = 0.5$ bar *at* 15 s. *Solid lines refer to the forward flow, dot-dashed lines refer to the backward flow. (a) fluid components velocities; (b) fluid components pressure; (c) fluid components temperatures; (d) strand temperature; (e) global error in semi logarithmic scale.*

Figure 4.3-9, *Figure 4.3-10* and *Figure 4.3-11* may look very similar to each other, but on closer inspection they have differences worth mentioning, related to the different pressure drop imposed on the cable. Indeed, its behavior at the considered time is governed by the pressure, or better by the pressurization induced by the heat source. For the first set of simulations ($\Delta p = 0.1$ bar) the maximum pressure is still the one imposed at the inlet that is constant up to the lower edge of the heated zone (4.0 m). This occurs in both the channels since they are in hydraulic parallel. As a consequence, the coolant velocity is zero up to this coordinate, because the local pressure drop is zero, and then starts to increase as the local pressure becomes lower than the inlet. This has an impact on the shape of the temperature spatial

distributions: as the velocity increases in the heated region the spatial gradient decreases, upstream the temperature is practically equal to the inlet value and downstream the heat is transported both by conduction and, mainly, by convection.

Considering *Figure 4.3-10* characterized by ($\Delta p = 0.02$ bar), the pressurization is such that inside the heated region the pressure becomes larger than the inlet value so a backflow occurs almost in the center of the cable, as can be seen from *Figure 4.3-10* (a). The shape of the temperature profiles is flat in the heated region and almost spatially symmetric thanks to the backflow that transport the heat in both the directions.

Finally, looking at the last *Figure 4.3-11* another different scenario appears. In this case the pressure drop is such that the flow pressurization is almost negligible and localized in the heated region as can be seen from the velocity increase in *Figure 4.3-11* (a). This can be explained considering that the gain in pressure, due to the heat, is almost compensated by the local pressure drop related to a large velocity. The temperature distribution in *Figure 4.3-11* (c) is similar to the one shown in *Figure 4.3-9*, but the gradient in the heated region is less steep due to the larger coolant mass flow rate that allows to exhaust a larger amount of heat. The ultimate consequence is a lower hot spot value in the cable.

4.3.4 INNER BENCHMARK: REFINED MESH

The last section deals with the availability of a refined mesh that allows to reduce the computational time of the algorithm, reducing the density of spatial discretization nodes where they are least needed. A comparison of two simulations that differ in the type of mesh used is proposed aiming to demonstrate that the error on the solution does not diverge. The reference simulation employs a uniform mesh while the other make use of a refined mesh, hence:

$$\xi_{Sim1} = \xi_{uni} \tag{4.3-7}$$

$$\xi_{Sim2} = \xi_{ref} \tag{4.3-8}$$

As usual, the not modified data of the simulations can be found in appendix D.2, the specific input data are instead collected in Table 4.3-7; notice the different number of elements used in the two simulations and the needed of extra input data to set the simulation with the non-uniform mesh that do not appear in the set of input data for the uniform one.

As can be seen form Table 4.3-7, the heating interval is only 0.5 s, quite short compared to the other used in the previous simulations while the linear heat power is one order of magnitude larger, to simulate an impulse of heat that then propagates along the conductor, the so-called *heat slug*. The linear power source is chosen to achieve a temperature rise of at least two degrees in the bundle. Also, the time at which the heating starts is changed and it is such that the average coolant flow is already in the refined zone but not yet in the heated region. The He average inlet velocity value is 0.356 m/s, obtained from:

$$v_{ave,inl} = \frac{v_{ch1,inl}\,\Sigma_{ch1} + v_{ch2,inl}\,\Sigma_{ch2}}{\Sigma_{ch1} + \Sigma_{ch2}} \qquad (4.3\text{-}9)$$

Table 4.3-7 Input data for the simulations with the uniform and not uniform (refined) meshes for the ITER-TF configuration with INTIAL 1.

Variable	Value		Unit
METHOD	0 (BE)	0 (BE)	–
ITMESH	0 (uniform)	1 (refined)	–
NELEMS	2000	500	–
NELREF	–	400	–
XBREFI	–	4.0	m
XEREFI	–	6.0	m
DXINCRE	–	1.2	–
TEND	14.0	14.0	s
STPMIN	0.1	0.1	s
INTIAL	1	1	–
TINL	4.5	4.5	K
PINL	6	6	bar
POUT	5.9	5.9	bar
Q0	3000	3000	W/m
XQBEG	4.2	4.2	m
XQEND	5.8	5.8	m
TQBEG	11.5	11.5	s
TQEND	12.0	12.0	s

Some characteristic times are collected in Table 4.3-8, evaluated at the average inlet velocity; it also justifies the end time of the simulation.

Table 4.3-8 Times at which the average initial flow reaches some relevant coordinates.

Coordinates m	XBREFI	XQBEG	XQMID	XQEND	XEREFI	XLENGTH
Time s	11.2	11.8	14.0	16.3	16.9	20.1

Note that the heated region is included in the refined zoned, because here the largest temperature spatial gradients are expected during the heating times, therefore the need of a larger number of elements to catch these gradients. The coordinates of the refined zone that do not belong to the heated region allow to resolve the steepest part of the gradient.

The simulation with the refined mesh is carried out with one fourth of the elements of uniform one. The refined region is discretized using 400 elements, the same number of elements dedicated to the same interval in the uniform mesh; the remaining 100 elements are distributed outside the refined region. Thanks to the symmetry of the refined region with respect to the center of gravity of the cable, 50 elements are used to discretize the interval [0,4) and the other 50 are used to discretize the interval (6,10]. The algorithm to build the refined mesh is described in section 2.2.1.1.

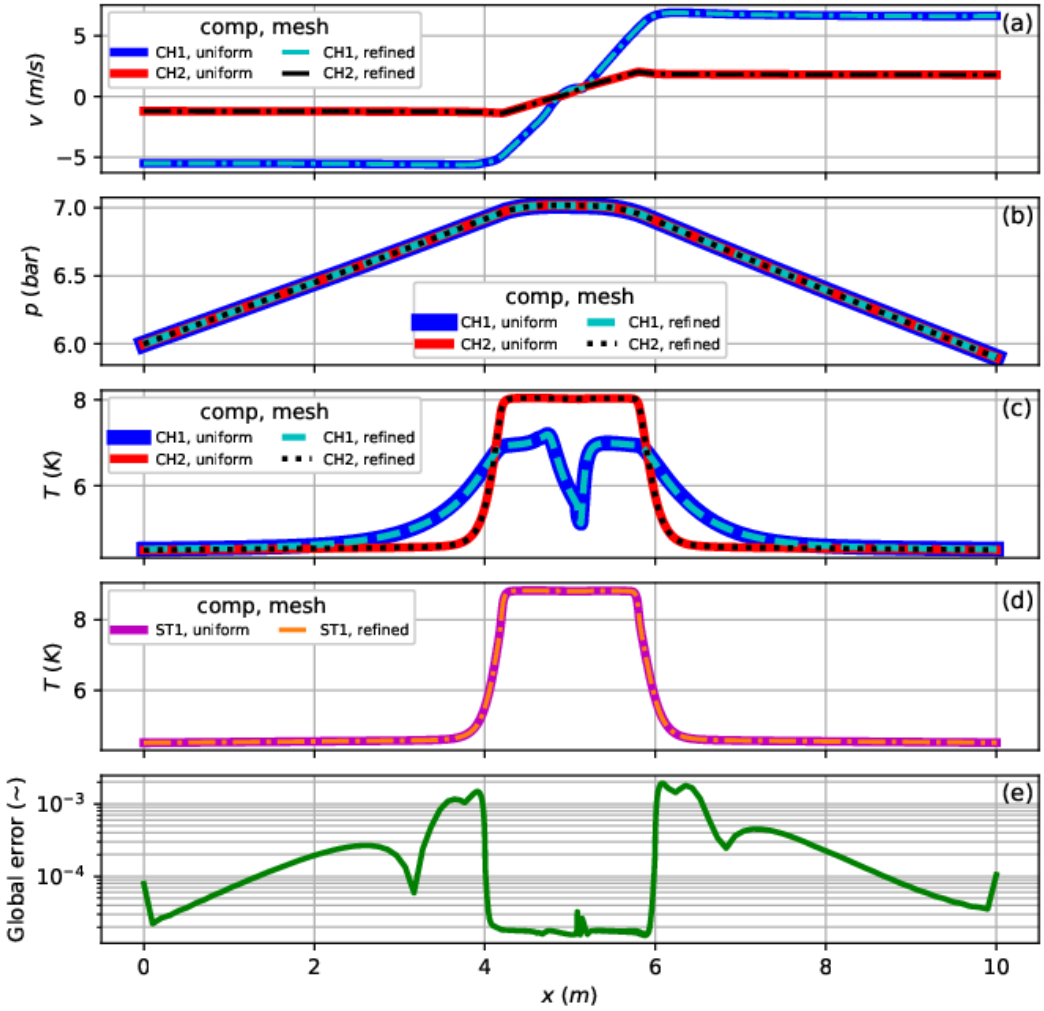


Figure 4.3-12 Comparison of the spatial distributions obtained with the uniform and refined meshes for the low temperature ITER-TF configuration with INTIAL 1 and at 11.8 s*. Solid lines refer to the uniform mesh, dot-dashed lines refer to the not uniform mesh. (a) fluid components velocities; (b) fluid components pressure; (c) fluid components temperatures; (d) strand temperature; (e) global error in semi logarithmic scale.*

When calculating the error, the code has to deal with vectors of different lengths, and the values must be compared at the same coordinates. The problem is outside the refined region since inside, by construction, the coordinates belonging to the range of the refined region coincide in both discretizations. Outside the refined region values in correspondence of the coordinates of the non-uniform mesh are evaluated interpolating the values obtained with the uniform one. In this way two arrays of the same length, equal to the number of nodes used in the non-uniform mesh, are built and they can be compared since their values are evaluated at the same spatial coordinates.

The results of the data processing are shown in *Figure 4.3-12*. The first four subplots (a-d) of *Figure 4.3-12* show the behavior of the solution spatial distribution for both considered meshes, demonstrating that the phenomena discussed several times are correctly captured even if fewer elements are used outside the heated region. The final proof is contained in the last subplot (*Figure 4.3-12* (e)) which shows the spatial distribution of the global error of the solution

at $11.8\,s$. The error is ruled mainly by the temperature gradients. As the temperature increases, the error increases outside the refined region, as the spatial gradient gets steeper the error tends to increase, until it enters the refined region where there is a reduction of a couple of orders of magnitude. A final remark concerns the unexpected behavior of the error at the boundary of the domain, which is related to the imposition of the fluid components temperature as boundary conditions. At the considered time step, the inlet velocity is negative and the outlet velocity is positive in both the channels, see *Figure 4.3-12* (a); so, recalling what summarized in Table 4.3-1, neither the inlet nor the outlet temperatures are imposed in both simulations. The evaluated temperatures are influenced by the different number of elements used for the spatial discretizations and, therefore, their value will be different giving a larger error.

4.3.5 SUMMARY

This section summarizes the results of the analysis of internal benchmarks, i.e., comparisons between different simulations launched with the same code. The possibility of performing this comparison is based on the assumption that, since the benchmarks with 4C are positive, the results of the simulations performed with SC2 are reliable.

The internal comparisons involve testing different initialization possibilities, such as using different methods for integrating the ODE system over time, making the results independent of the position of the inlet and outlet and, finally, defining a non-uniform mesh to reduce computational costs. The salient features of the results obtained are highlighted.

As regards the comparison of the different initializations, the three main possibilities were compared, namely INTIAL 1 INTIAL 2 and INTIAL 5. The results confirmed expectations, namely that setting the different flag leads to different results, depending on the initialization values, since the physics of the problem changes. In practice, the imposed boundary conditions change. Comparing instead INTIAL 2 with INTIAL 5 leads to the same results because the only difference is in how the initial values are obtained, while the boundary conditions are the same. It was also pointed out that the physics implemented by the code is plausible.

The comparison of the results obtained with BE and CN confirms, once again, that to obtain accurate results with the latter, small values of the time step must be adopted. In fact, the agreement between the two methods is better with STPMIN equal to $0.01\ s$ than with $0.1\ s$, but the errors are still high during the heating and cooling phases.

The three sets of simulations defined to test the fluid dynamics implemented in SC2, imposing three different pressure drops using INTIAL 1, gave consistent results that conformed to expectations: the code recognizes the location of the inlet and outlet according to the order relationship between the pressures assigned and consequently calculates the solution by attributing the correct sign to the velocity.

Finally, it has been shown that the use of the non-uniform mesh produces globally acceptable results; within the refined zone the error compared to the solution obtained with a uniform mesh decreases by approximately two orders of magnitude.

CHAPTER 5

5 CONCLUSIONS AND FUTURE PERSPECTIVES

The purpose of this thesis work can be declined in two key points of equal importance and interconnected: the first is the conceptual design and first implementation of a new tool for modeling LTS and HTS superconducting cables, the second consists in the rigorous procedure of verification and validation of the developed tool with respect to the existing and validated 4C code.

The novelty of the tool with respect to the ones already available in the scientific community relies in its object-oriented nature. Exploiting the potentialities of the classes and of the objects they instantiate, it has been shown the simplicity with which it is possible to study different topologies of conductors, supplying exclusively inputs on the number and the typology of the components that build the conductor, the geometry and the materials of which they are composed, as well as the operating conditions and the strategies to be applied for the solution of the problem.

In order to achieve the second objective, the verification of the algorithms is based on the control of the expected convergence orders in both space and time, together with the correct representation of the physics of the considered problem. On the other hand, validation is based on benchmarking against 4C code for two configurations that differ in topology, operating temperature, and application:

- the three-phase coaxial high critical temperature superconducting cables (3P-HTS);
- the ITER toroidal field coil low critical temperature superconductive cables (ITER-TF).

Both configurations have been studied with two different initializations and related boundary conditions: INTIAL 1 corresponds to an initialization that imposes the pressures at the ends of the cable and the initial temperature and applies the same values as boundary conditions; with INTIAL 5 the flow is initialized starting from the inlet mass flow rate and temperature together with the outlet pressure, so the boundary conditions imposed are the inlet velocity and temperature and the outlet pressure.

By means of a series of inner benchmarks and stability checks, the verification of the other characteristics of the code was then completed.

The correct representation of the physics of the problem, as well as being implicitly verified through the benchmark with 4C code, has been further investigated and confirmed with the benchmarks performed by adopting three values of the INTIAL flag, specifically 1, 2 and 5. It was found that adopting different flags (INTIAL 1 and INTIAL 5) leads to different results, consistent with the operating conditions of the cable, while comparing INTIAL 2 with INTIAL 5 confirms that the two initializations are equivalent, the only difference being the way in which the outlet pressure is obtained.

Comparison of the solution obtained with BE and with CN confirms that a smaller time step is required to obtain accurate solutions with the latter.

Further verification of the correct representation of physics was achieved by verifying that the position of the inlet and outlet of the conductor was related to the pressure values at the inlet and outlet, i.e., that the inlet is located where the pressure is maximum and the outlet where it is minimum. This study was carried out with INTIAL 1 for different values of the pressure drop (0.1 bar, 0.02 bar and 0.5 bar) with the inlet pressure set at 6 bar. Not only it was found that by exchanging the inlet pressure with the outlet one the direction of flow is correctly reversed, but it was also seen that different pressure drops correspond to different thermal-fluid dynamics behaviors of the cable, again confirming the ability of the tool to correctly simulate physics.

Finally, the consistency of the solution obtained with a non-uniform, refined mesh around the heated region was verified, compared to that obtained with a uniform mesh. Again, the benchmark was successful.

The validation with 4C code is influenced by the differences between the algorithms implemented in the two codes, in particular the different initialization procedure and the calculation of the properties in the Gauss nodes, essential for the construction of the matrix of the coefficients of the system. The effects of the former are particularly evident in the determination of the initial velocity distribution for the ITER-TF configuration with INTIAL 5, being the errors of the order of 10^{-1}. The recipe adopted for the calculation of the properties in the Gauss node has a non-negligible impact on the accuracy of the solution, a fortiori when considering the LTS due to the marked nonlinearity of the material properties in this temperature window 4.5 K - 20 K.

Based on the above analysis, it can be concluded that the SC2 algorithms are successfully verified and validated, the results are consistent and accurate. The two configurations considered sanction the versatility of the code and its ability to handle different topologies.

On this front, further simulation campaigns aimed at deepening the verification and validation of the SC2 code must necessarily be carried out. It is appropriate to perform tests with configurations that involve the use of more materials, a greater number of components and different topologies from those examined in this work, also allow to test more sophisticated computational strategies, and not yet implemented in SC2.

In fact, from this point of view, it still offers a limited number of options. Computational improvements can be obtained by implementing time step and grid adaptivity, as well as introducing new possibilities for numerical integration of the ODE system such as the fourth-order Adams-Moulton method (AM4). As far as the spatial discretization, higher order FEM should be considered to enhance the convergence of the solution. An interesting alternative to the FEM to consider is the VEM, already widely used in the literature for this type of analysis as highlighted in section 1.3. A further increase in performance would be achieved by replacing the methods currently used for solving the linear system of equations with python's built-in solvers.

5 CONCLUSIONS AND FUTURE PERSPECTIVES

SC2 code also needs to be extended as it is not yet able to handle the radiative heat transfer or to model cables electromagnetically. The modeling of different conductors, already possible in the developed tool, should be complemented by a suitable coupling between the conductors.

The graphical interface can and must be further developed. In particular, the “Simulation control panel” cascade must be completed to allow the user to interact with the simulation and “Simulation drivers” must be developed to definitively introduce the Run-AND-check philosophy (see Table 3.4-1). Moreover, new functionalities can be added, such as input data compilation and thus the construction of the conductor topology, the choice of the extension of the default figures and their structure (number of subplots and number of curves depicted in each of them).

As a final remark, the code also needs to be improved in terms of style and data organization. A good support for the first aspect can be found by following the suggestions and indications in PEP8 [106]. On the other hand, saving data in an orderly fashion [107] simplifies the routines dedicated to post processing and plotting. Finally, the use of binary and columnar format such as *hdf5* or *parquet* would allow an easier management of the data, making the most of the potential of pandas for advanced post processing.

APPENDIX A

A EXTENDED FORM OF EQUATIONS

This appendix shows the extended form of the general equations that constitute the system of PDEs to be solved. Specifically, the source terms of the fluid components set of equations are written explicitly, then the general expression of velocity, pressure and temperature equations are proposed.

The Euler like set of equations in non-conservative variables is:

$$\begin{cases} \dfrac{\partial v}{\partial t} + v\dfrac{\partial v}{\partial x} + \dfrac{1}{\rho}\dfrac{\partial p}{\partial x} = \dfrac{1}{\rho}\left(\Lambda_v - v\Lambda_\rho\right) \\ \dfrac{\partial p}{\partial t} + \rho c^2\dfrac{\partial v}{\partial x} + v\dfrac{\partial p}{\partial x} = \Phi\left[\Lambda_e - v\Lambda_v - \left(w - \dfrac{v^2}{2} - \dfrac{c^2}{\Phi}\right)\Lambda_\rho\right] \\ \dfrac{\partial T}{\partial t} + \Phi T\dfrac{\partial v}{\partial x} + v\dfrac{\partial T}{\partial x} = \dfrac{1}{\rho c_v}\left[\Lambda_e - v\Lambda_v - \left(w - \dfrac{v^2}{2} - \Phi c_v T\right)\Lambda_\rho\right] \end{cases} \quad (A\text{-}1)$$

There are three source terms, the mass Λ_ρ, the momentum Λ_v and the energy one Λ_e.

The derivation of the above equations can be found in [94]. The generic source terms for fluid component ca that interacts with $N_{ch} - 1$ fluid components are shown below.

$$\Lambda_\rho^{ca} = \frac{\sum_{ch\neq ca}^{N_{ch}} K'_{ca,ch}\left(p_{ch} - p_{ca}\right)}{L\Sigma_{ca}} \quad (A\text{-}2)$$

$$\Lambda_v^{ca} = \frac{\sum_{ch\neq ca}^{N_{ch}} K''_{ca,ch}\left(p_{ch} - p_{ca}\right)}{L\Sigma_{ca}} - \rho_{ca}F_{ca} \quad (A\text{-}3)$$

$$\begin{aligned} \Lambda_e^{ca} = {} & \frac{\sum_{ch\neq ca}^{N_{ch}} K'''_{ca,ch}\left(p_{ch} - p_{ca}\right)}{L\Sigma_{ca}} \\ & + \frac{\sum_{ch\neq ca}^{N_{ch}}\left(P^o_{ca,ch}h^o_{ca,ch} + P^c_{ca,ch}h^c_{ca,ch}\right)\left(T_{ch} - T_{ca}\right)}{\Sigma_{ca}} \\ & + \frac{\sum_{st=1}^{N_{st}} P_{ca,st}h_{ca,st}\left(T_{st} - T_{ca}\right)}{\Sigma_{ca}} + \frac{\sum_{jk=1}^{N_{jk}} P_{ca,jk}h_{ca,jk}\left(T_{jk} - T_{ca}\right)}{\Sigma_{ca}} \end{aligned} \quad (A\text{-}4)$$

$$F_{ca} = \frac{2f_{ca}v_{ca}^2}{D_{h.ca}} = v_{ca}\tilde{F}_{ca} \quad (A\text{-}5)$$

$$\tilde{F}_{ca} = \frac{2f_{ca}|v_{ca}|}{D_{h.ca}} \quad (A\text{-}6)$$

To see what is hidden behind the transport coefficients $K'_{ca,ch}$, $K''_{ca,ch}$ and $K'''_{ca,ch}$, the transversal velocity between two channels in hydraulic parallel should be introduced. It can be modeled as:

$$v_{\perp,ca,ch} = \begin{cases} \sqrt{\dfrac{2|p_{ch} - p_{ca}|}{k_{f,loc}\, p_{ch}}} \dfrac{p_{ch} - p_{ca}}{|p_{ch} - p_{ca}|}, p_{ch} - p_{ca} \geq 0 \\ \sqrt{\dfrac{2|p_{ch} - p_{ca}|}{k_{f,loc}\, p_{ca}}} \dfrac{p_{ch} - p_{ca}}{|p_{ch} - p_{ca}|}, p_{ch} - p_{ca} < 0 \end{cases} \quad \text{(A-7)}$$

The transport coefficients are defined as follows:

$$K'_{ca,ch} = \begin{cases} \Sigma_{\perp,ca,ch} \sqrt{\dfrac{2\rho_{ch}}{k_{f,loc}|p_{ch} - p_{ca}|}}, v_{\perp,ca,ch} \geq 0 \\ \Sigma_{\perp,ca,ch} \sqrt{\dfrac{2\rho_{ca}}{k_{f,loc}|p_{ch} - p_{ca}|}}, v_{\perp,ca,ch} < 0 \end{cases} \quad \text{(A-8)}$$

$$K''_{ca,ch} = \begin{cases} K'_{ca,ch}\lambda_v v_{ch}, v_{\perp,ca,ch} \geq 0 \\ K'_{ca,ch}\lambda_v v_{ca}, v_{\perp,ca,ch} < 0 \end{cases} \quad \text{(A-9)}$$

$$K'''_{ca,ch} = \begin{cases} K'_{ca,ch}\left[w_{ch} + \dfrac{(\lambda_v v_{ch})^2}{2}\right], v_{\perp,ca,ch} \geq 0 \\ K'_{ca,ch}\left[w_{ca} + \dfrac{(\lambda_v v_{ca})^2}{2}\right], v_{\perp,ca,ch} < 0 \end{cases} \quad \text{(A-10)}$$

The parameter λ_v allows to keep into account if the interface is able or not to absorb momentum, according to its structure. For an helicoidal interface $\lambda_v \cong 1$, while if the channels coupling is realized with small holes it can be assumed that $\lambda_v \cong 0$.

The unit of measure of the transport coefficients and of the source terms are reported in Table B.1-1.

Table B.1-1 Transport coefficient and source terms units of measure.

Quantity	$\boldsymbol{K'}$	$\boldsymbol{K''}$	$\boldsymbol{K'''}$	$\boldsymbol{\Lambda_\rho}$		$\boldsymbol{\Lambda_v}$		$\boldsymbol{\Lambda_e}$	
Unit SI	ms	m^2	m^3/s	$\mathrm{kg}/(\mathrm{m}^3\,\mathrm{s})$	J/m^4	$\equiv \mathrm{kg}/(\mathrm{m}^2\,\mathrm{s}^2)$	W/m^3	$\equiv \mathrm{kg}/(\mathrm{ms}^3)$

From the above definitions results that:

$$K'_{ch,ca} = K'_{ca,ch}\ ca, ch = 1, \dots, N_{ch} \quad \text{(A-11)}$$

$$K''_{ch,ca} = K''_{ca,ch}\ ca, ch = 1, \dots, N_{ch} \quad \text{(A-12)}$$

$$K'''_{ch,ca} = K'''_{ca,ch}\ ca, ch = 1, \dots, N_{ch} \quad \text{(A-13)}$$

Substituting the full expression of the source terms in equation (A-1) the extended version of the equations can be obtained. Their final forms are proposed here.

A EXTENDED FORM OF EQUATIONS

Velocity equation for the ca-th channel that interacts with $N_{ch}-1$ fluid components, N_{st} strands components and N_{jk} jacket components:

$$\frac{\partial v_{ca}}{\partial t}+v_{ca}\frac{\partial v_{ca}}{\partial x}+\frac{1}{\rho_{ca}}\frac{\partial p_{ca}}{\partial x}+v_{ca}\tilde{F}_{ca} - \frac{p_{ca}}{L\rho_{ca}\Sigma_{ca}}\sum_{ch\neq ca}^{N_{ch}}\left(v_{ca}K'_{ca,ch}-K''_{ca,ch}\right) + \frac{1}{L\rho_{ca}\Sigma_{ca}}\sum_{ch\neq ca}^{N_{ch}}\left(v_{ca}K'_{ca,ch}-K''_{ca,ch}\right)p_{ch}=0 \quad \text{(A-14)}$$

Pressure equation for the ca-th channel that interacts with $N_{ch}-1$ fluid components, N_{st} strands components and N_{jk} jacket components:

$$\begin{aligned}&\frac{\partial p_{ca}}{\partial t}+\rho_{ca}c_{ca}^2\frac{\partial v_{ca}}{\partial x}+v_{ca}\frac{\partial p_{ca}}{\partial x}-\Phi_{ca}\rho_{ca}\tilde{F}_{ca}v_{ca}^2\\&+\frac{\Phi_{ca}p_{ca}}{L\Sigma_{ca}}\sum_{ch\neq ca}^{N_{ch}}\left[K'''_{ca,ch}-v_{ca}K''_{ca,ch}-\left(w_{ca}-\frac{v_{ca}^2}{2}-\frac{c_{ca}^2}{\Phi_{ca}}\right)K'_{ca,ch}\right]\\&-\frac{\Phi_{ca}}{L\Sigma_{ca}}\sum_{ch\neq ca}^{N_{ch}}\left[K'''_{ca,ch}-v_{ca}K''_{ca,ch}-\left(w_{ca}-\frac{v_{ca}^2}{2}-\frac{c_{ca}^2}{\Phi_{ca}}\right)K'_{ca,ch}\right]p_{ch}\\&+\frac{\Phi_{ca}T_{ca}}{\Sigma_{ca}}\left[\sum_{ch\neq ca}^{N_{ch}}\left(P_{ca,ch}^{O}\,h_{ca,ch}^{O}+P_{ca,ch}^{C}\,h_{ca,ch}^{C}\right)+\sum_{st=1}^{N_{st}}P_{ca,st}\,h_{ca,st}\right.\\&\left.+\sum_{jk=1}^{N_{jk}}P_{ca,jk}\,h_{ca,jk}\right]\\&-\frac{\Phi_{ca}}{\Sigma_{ca}}\left[\sum_{ch\neq ca}^{N_{ch}}\left(P_{ca,ch}^{O}\,h_{ca,ch}^{O}+P_{ca,ch}^{C}\,h_{ca,ch}^{C}\right)T_{ch}\right.\\&\left.+\sum_{st=1}^{N_{st}}P_{ca,st}\,h_{ca,st}\,T_{st}+\sum_{jk=1}^{N_{jk}}P_{ca,jk}\,h_{ca,jk}\,T_{jk}\right]=0\end{aligned} \quad \text{(A-15)}$$

Temperature equation for the ca -th channel that interacts with $N_{ch}-1$ fluid components, N_{st} strands components and N_{jk} jackets components:

$$
\begin{aligned}
&\frac{\partial T_{ca}}{\partial t} + \Phi_{ca} T_{ca} \frac{\partial v_{ca}}{\partial x} + v_{ca} \frac{\partial T_{ca}}{\partial x} - \frac{\tilde{F}_{ca} v_{ca}^2}{c_{v,ca}} \\
&\quad + \frac{p_{ca}}{L \rho_{ca} \Sigma_{ca} c_{v,ca}} \sum_{ch \neq ca}^{N_{ch}} \left[K'''_{ca,ch} - v_{ca} K''_{ca,ch} \right. \\
&\quad \left. - \left(w_{ca} - \frac{v_{ca}^2}{2} - \Phi_{ca} c_{v,ca} T_{ca} \right) K'_{ca,ch} \right] \\
&\quad - \frac{1}{L \rho_{ca} \Sigma_{ca} c_{v,ca}} \sum_{ch \neq ca}^{N_{ch}} \left[K'''_{ca,ch} - v_{ca} K''_{ca,ch} \right. \\
&\quad \left. - \left(w_{ca} - \frac{v_{ca}^2}{2} - \Phi_{ca} c_{v,ca} T_{ca} \right) K'_{ca,ch} \right] p_{ch} \\
&\quad + \frac{T_{ca}}{\rho_{ca} \Sigma_{ca} c_{v,ca}} \left[\sum_{ch \neq ca}^{N_{ch}} \left(P^O_{ca,ch} h^O_{ca,ch} + P^C_{ca,ch} h^C_{ca,ch} \right) \right. \\
&\quad \left. + \sum_{st=1}^{N_{st}} P_{ca,st} h_{ca,st} + \sum_{jk=1}^{N_{jk}} P_{ca,jk} h_{ca,jk} \right] \\
&\quad - \frac{1}{\rho_{ca} \Sigma_{ca} c_{v,ca}} \left[\sum_{ch \neq ca}^{N_{ch}} \left(P^O_{ca,ch} h^O_{ca,ch} + P^C_{ca,ch} h^C_{ca,ch} \right) T_{ch} \right. \\
&\quad \left. + \sum_{st=1}^{N_{st}} P_{ca,st} h_{ca,st} T_{st} + \sum_{jk=1}^{N_{jk}} P_{ca,jk} h_{ca,jk} T_{jk} \right] = 0
\end{aligned}
\qquad \textit{(A-16)}
$$

For the sake of completeness, the extended version of the 1D transient heat transfer equations for the genetic strand and jacket objects are also shown in this appendix.

Full extension of the transient one-dimensional heat transfer equation for the fi-th strand that interacts with N_{ch} fluid components, $N_{st} - 1$ strands component and N_{jk} jackets components:

$$
\begin{aligned}
&\Sigma_{fi} \rho_{fi} c_{fi} \frac{\partial T_{fi}}{\partial t} - \Sigma_{fi} \frac{\partial}{\partial x} \left(k_{fi} \frac{\partial T_{fi}}{\partial x} \right) \\
&\quad + T_{fi} \left(\sum_{ch=1}^{N_{ch}} P_{fi,ch} h_{fi,ch} + \sum_{st \neq fi}^{N_{st}} P_{fi,st} h_{fi,st} + \sum_{jk=1}^{N_{jk}} P_{fi,jk} h_{fi,jk} \right) \\
&\quad - \sum_{ch=1}^{N_{ch}} P_{fi,ch} h_{fi,ch} T_{ch} - \sum_{st=fi}^{N_{st}} P_{fi,st} h_{fi,st} T_{st} - \sum_{jk=1}^{N_{jk}} P_{fi,jk} h_{fi,jk} T_{jk} \\
&\quad = Q_{fi,Joule} + Q_{fi,ext}
\end{aligned}
\qquad \textit{(A-17)}
$$

Full extension of the transient one-dimensional heat transfer equation for the in-th jacket that interacts with N_{ch} fluid components, N_{st} strands component and $N_{jk} - 1$ jackets components:

A EXTENDED FORM OF EQUATIONS

$$\begin{aligned}
&\Sigma_{in}\rho_{in}c_{in}\frac{\partial T_{in}}{\partial t} - \Sigma_{in}\frac{\partial}{\partial x}\left(k_{in}\frac{\partial T_{in}}{\partial x}\right) \\
&\quad + T_{in}\left(\sum_{ch=1}^{N_{ch}} P_{in,ch}h_{in,ch} + \sum_{st=1}^{N_{st}} P_{in,st}h_{in,st} + \sum_{jk\neq in}^{N_{jk}} P_{in,jk}h_{in,jk}\right) \\
&\quad - \sum_{ch=1}^{N_{ch}} P_{in,ch}h_{in,ch}T_{ch} - \sum_{st=1}^{N_{st}} P_{in,st}h_{in,st}T_{st} - \sum_{jk\neq in}^{N_{jk}} P_{in,jk}h_{in,jk}\,T_{jk} \\
&\quad = Q_{in,Joule} + Q_{in,ext}
\end{aligned} \qquad \textit{(A-18)}$$

From this form of the equations it is possible to get the elements of the matrices and the known term vector that appears in equation (2.1-10). These are proposed in appendix B.1.

APPENDIX B

B MATRIX ELEMENTS

This appendix is divided into three sections. First the general form of the elements of the matrices and vectors that appear in the matrix equation of the system are shown, then the real shape of these matrices are proposed for the two study cases.

B.1 GENERAL FORM OF MATRICES AND VECTORS

The matrix form of the PDEs system is:

$$M\frac{\partial \boldsymbol{u}}{\partial t} + A\frac{\partial \boldsymbol{u}}{\partial x} + \frac{\partial}{\partial x}\left(K\frac{\partial \boldsymbol{u}}{\partial x}\right) + S\boldsymbol{u} = \boldsymbol{s} \tag{B-1}$$

This section describes the general shape of both vectors and matrices that appears in the above equation.

Recall that the number of degrees of freedom, the number of unknowns, is given by:

$$N_{eq} = 3N_{ch} + N_{st} + N_{jk} \tag{B-2}$$

This implies that $\boldsymbol{u}$ and $\boldsymbol{s}$ are columns vectors ∈ $\mathbb{R}^{N_{eq},1}$; their general structure is:

$$\boldsymbol{u} = \begin{bmatrix} \boldsymbol{v_{ch}} \\ \boldsymbol{p_{ch}} \\ \boldsymbol{T_{ch}} \\ \boldsymbol{T_{st}} \\ \boldsymbol{T_{jk}} \end{bmatrix} \tag{B-3}$$

$$\boldsymbol{s} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{Q_{st}} \\ \boldsymbol{Q_{jk}} \end{bmatrix} \tag{B-4}$$

with $\boldsymbol{v_{ch}}$, $\boldsymbol{p_{ch}}$ and $\boldsymbol{T_{ch}}$ ∈ $\mathbb{R}^{N_{ch},1}$, $\boldsymbol{T_{st}}$ ∈ $\mathbb{R}^{N_{st},1}$ and $\boldsymbol{T_{jk}}$ ∈ $\mathbb{R}^{N_{jk},1}$; $\boldsymbol{0}$ is the null vector in $\mathbb{R}^{3N_{ch},1}$, finally $\boldsymbol{Q_{strand}}$ and $\boldsymbol{Q_{jacket}}$ are respectively vectors of $\mathbb{R}^{N_{st},1}$ and $\mathbb{R}^{N_{jk},1}$.

$$\boldsymbol{v_{ch}} = [v_{ch}]\ ch = 1, \dots, N_{ch} \tag{B-5}$$

$$\boldsymbol{p_{ch}} = [p_{ch}]\ ch = 1, \dots, N_{ch} \tag{B-6}$$

$$\boldsymbol{T_{ch}} = [T_{ch}]\ ch = 1, \dots, N_{ch} \tag{B-7}$$

$$\boldsymbol{T_{st}} = [T_{st}]\ st = 1, \dots, N_{st} \tag{B-8}$$

$$\boldsymbol{T_{jk}} = \left[T_{jk}\right]\ jk = 1, \dots, N_{jk} \tag{B-9}$$

$$\boldsymbol{Q_{st}} = \left[Q_{st,Joule} + Q_{st,ext}\right]\ st = 1, \dots, N_{st} \tag{B-10}$$

B.1 GENERAL FORM OF MATRICES AND VECTORS

$$\boldsymbol{Q_{jk}} = \left[Q_{jk,Joule} + Q_{jk,ext}\right] jk = 1, \dots, N_{jk} \quad \textit{(B-11)}$$

The matrices of equation (2.1-10) are sparse square matrices of $\mathbb{R}^{N_{eq},N_{eq}}$. Looking at the extended general form of the equations listed in appendix A, their elements can be deduced.

The M matrix is diagonal:

$$M = diag\left(\mathbf{1}, \boldsymbol{M_{st}}, \boldsymbol{M_{jk}}\right) \quad \textit{(B-12)}$$

where $\mathbf{1}$ is the vector of ones in $\mathbb{R}^{3N_{ch},1}$, $\boldsymbol{M_{st}} \in \mathbb{R}^{N_{st},1}$ and $\boldsymbol{M_{jk}} \in \mathbb{R}^{N_{jk},1}$ with:

$$\boldsymbol{M_{st}} = \left[\Sigma_{st}\rho_{st}c_{p,st}\right] \; st = 1, \dots, N_{st} \quad \textit{(B-13)}$$

$$\boldsymbol{M_{jk}} = \left[\Sigma_{jk}\rho_{jk}c_{p,jk}\right] \; jk = 1, \dots, N_{jk} \quad \textit{(B-14)}$$

The A matrix is banded with four not null diagonals including the main one: specifically, the upper N_{ch}-th super diagonal ($\boldsymbol{A_{sup,v}}$nd the lowers N_{ch}-th ($\boldsymbol{A_{sub,p}}$) and $2N_{ch}$-th ($\boldsymbol{A_{sub,T}}$) diagonals are not null together with the main diagonal ($\boldsymbol{A_{main}}$). The elements of these diagonals are shown in their vectorial form.

$$\boldsymbol{A_{main}} = \begin{bmatrix} \boldsymbol{v_{ch}} \\ \boldsymbol{v_{ch}} \\ \boldsymbol{v_{ch}} \\ \mathbf{0_{main}} \end{bmatrix} \in \mathbb{R}^{N_{eq},1} \quad \textit{(B-15)}$$

$$\boldsymbol{A_{sup,v}} = \begin{bmatrix} \boldsymbol{A_v} \\ \mathbf{0_{sup,v}} \end{bmatrix} \in \mathbb{R}^{2N_{ch}+N_{st}+N_{jk},1} \quad \textit{(B-16)}$$

$$\boldsymbol{A_{sub,p}} = \begin{bmatrix} \boldsymbol{A_p} \\ \mathbf{0_{sub,p}} \end{bmatrix} \in \mathbb{R}^{2N_{ch}+N_{st}+N_{jk},1} \quad \textit{(B-17)}$$

$$\boldsymbol{A_{sub,T}} = \begin{bmatrix} \boldsymbol{A_T} \\ \mathbf{0_{sub,T}} \end{bmatrix} \in \mathbb{R}^{N_{ch}+N_{st}+N_{jk},1} \quad \textit{(B-18)}$$

$\boldsymbol{A_{main}}$ is the vector corresponding to the main diagonal of A: its elements are the channels velocities and $N_{st} + N_{jk}$ zeros collected into the vector $\mathbf{0_{main}} \in \mathbb{R}^{N_{st}+N_{jk},1}$.

The values of the not null super diagonal are collected in vector $\boldsymbol{A_{sup,v}}$at is composed in turn by a not null vector $\boldsymbol{A_v} \in \mathbb{R}^{N_{ch},1}$ and a null vector $\mathbf{0_{sup,v}} \in \mathbb{R}^{N_{ch}+N_{st}+N_{jk},1}$he subscript v refers to the fact that these terms came from the coefficient of partial derivative $\frac{\partial p}{\partial x}$ in the velocity equations:

$$\boldsymbol{A_v} = \left[\frac{1}{\rho_{ch}}\right] \; ch = 1, \dots, N_{ch} \quad \textit{(B-19)}$$

The two not null sub-diagonals are also composed by a combination of a not null and a null vectors. The former comes from partial derivative $\frac{\partial v}{\partial x}$ in the pressure equations, so it is called $\boldsymbol{A_{sub,p}}$; its not null elements are the coefficients that multiply this derivative, that is:

B MATRIX ELEMENTS

$$\boldsymbol{A_p} = [\rho_{ch} c_{ch}^2]\ ch = 1, \dots, N_{ch} \tag{B-20}$$

so it has the same dimension of $\boldsymbol{A_v}$, $\boldsymbol{A_p} \in \mathbb{R}^{N_{ch},1}$, and this implies that $\boldsymbol{0_{sub,p}}$ is identical to $\boldsymbol{0_{sup,v}}$inally, $\boldsymbol{A_T}$ groups the not null elements of the second sub-diagonal that correspond to the coefficients that multiply the derivative $\frac{\partial v}{\partial x}$ in the temperature equation, therefore also $\boldsymbol{A_T} \in \mathbb{R}^{N_{ch},1}$; since the global dimension of the diagonal is $N_{ch} + N_{st} + N_{jk}$, the null vector $\boldsymbol{0_{sub,T}} \in \mathbb{R}^{N_{st}+N_{jk},1}$ as $\boldsymbol{0_{main}}$.

$$\boldsymbol{A_T} = [\Phi_{ch} T_{ch}]\ ch = 1, \dots, N_{ch} \tag{B-21}$$

The K matrix is diagonal and collects the coefficients of the second derivative in space:

$$K = diag(\boldsymbol{0}, \boldsymbol{K_{st}}, \boldsymbol{K_{jk}}) \tag{B-22}$$

where $\boldsymbol{0}$ is the already mentioned vector of zeros in $\mathbb{R}^{3N_{ch},1}$, $\boldsymbol{K_{st}} \in \mathbb{R}^{N_{st},1}$ and $\boldsymbol{K_{jk}} \in \mathbb{R}^{N_{jk},1}$ with:

$$\boldsymbol{K_{st}} = [-\Sigma_{st} k_{st}]\ \ st = 1, \dots, N_{st} \tag{B-23}$$

$$\boldsymbol{K_{jk}} = [-\Sigma_{jk} k_{jk}]\ \ jk = 1, \dots, N_{jk} \tag{B-24}$$

The last matrix is S that collects all the remaining addendums on the left-hand side of the equations. Its construction is not trivial as the previous matrices and its sparsity pattern is function of the cable topology. It is composed by twenty fife sub-matrices whose elements will be discussed in the last part of this section.

$$S = \begin{bmatrix} S_{v_{ch},v_{ch}} & S_{v_{ch},p_{ch}} & S_{v_{ch},T_{ch}} & S_{v_{ch},T_{st}} & S_{v_{ch},T_{jk}} \\ S_{p_{ch},v_{ch}} & S_{p_{ch},p_{ch}} & S_{p_{ch},T_{ch}} & S_{p_{ch},T_{st}} & S_{p_{ch},T_{jk}} \\ S_{T_{ch},v_{ch}} & S_{T_{ch},p_{ch}} & S_{T_{ch},T_{ch}} & S_{T_{ch},T_{st}} & S_{T_{ch},T_{jk}} \\ S_{T_{st},v_{ch}} & S_{T_{st},p_{ch}} & S_{T_{st},T_{ch}} & S_{T_{st},T_{st}} & S_{T_{st},T_{jk}} \\ S_{T_{jk},v_{ch}} & S_{T_{jk},p_{ch}} & S_{T_{jk},T_{ch}} & S_{T_{jk},T_{st}} & S_{T_{jk},T_{jk}} \end{bmatrix} \tag{B-25}$$

The first important point to clarify is the dimensions of the sub-matrices: nine groups can be identified as shown in equation (B-25) The upper left three by three block of matrices (upper side) is characterized by square matrices $\in \mathbb{R}^{N_{ch},N_{ch}}$. The first three matrices of the fourth and fifth columns are rectangular matrices $\in \mathbb{R}^{N_{ch},N_{st}}$ and $\in \mathbb{R}^{N_{ch},N_{jk}}$ respectively; while the first three matrices on the fourth and fifth rows belongs respectively to $\mathbb{R}^{N_{st},N_{ch}}$ and $\mathbb{R}^{N_{jk},N_{ch}}$. The last four matrices belong each to a specific group. The two on the main diagonal are squared $\in \mathbb{R}^{N_{st},N_{st}}$ and $\in \mathbb{R}^{N_{jk},N_{jk}}$ respectively, finally the last two are rectangular $S_{T_{jk},T_{st}} \in \mathbb{R}^{N_{jk},N_{st}}$ and $S_{T_{st},T_{jk}} \in \mathbb{R}^{N_{st},N_{jk}}$.

In the following, the matrices are described proceeding by column.

B.1 GENERAL FORM OF MATRICES AND VECTORS

$S_{v_{ch},v_{ch}}$, $S_{p_{ch},v_{ch}}$ and $S_{T_{ch},v_{ch}}$ are diagonal matrices. Their elements correspond to the coefficients that multiply the channel velocity of the fourth addendum in velocity, pressure and temperature equations.

$$S_{v_{ch},v_{ch}} = diag(\widetilde{\boldsymbol{F}}) \tag{B-26}$$

$$S_{p_{ch},v_{ch}} = diag(\boldsymbol{S_{p_{ch},v_{ch}}}) \tag{B-27}$$

$$S_{T_{ch},v_{ch}} = diag(\boldsymbol{S_{T_{ch},v_{ch}}}) \tag{B-28}$$

Being

$$\widetilde{\boldsymbol{F}} = [\tilde{F}_{ch}]\ ch = 1, \dots, N_{ch} \tag{B-29}$$

$$\boldsymbol{S_{p_{ch},v_{ch}}} = [-\tilde{F}_{ch}\rho_{ch}\Phi_{ch}v_{ch}]\ ch = 1, \dots, N_{ch} \tag{B-30}$$

$$\boldsymbol{S_{T_{ch},v_{ch}}} = \left[-\frac{\tilde{F}_{ch}v_{ch}}{c_{v,ch}}\right]\ ch = 1, \dots, N_{ch} \tag{B-31}$$

Matrices $S_{T_{st},v_{ch}}$ and $S_{T_{jk},v_{ch}}$ are identically null since there are no terms multiplied by channel velocity in the solid equations.

Next the terms multiplied by pressure are considered that contribute to build-up the five matrices of the second column. Analogously to the previous case, the heat equation does not have such addendums so matrices $S_{T_{st},p_{ch}}$ and $S_{T_{jk},p_{ch}}$ are full of zeros.

$S_{v_{ch},p_{ch}}$, $S_{p_{ch},p_{ch}}$ and $S_{T_{ch},p_{ch}}$ are square matrices $\in \mathbb{R}^{N_{ch},N_{ch}}$.

$$S_{v_{ch},p_{ch}} = \begin{bmatrix} S_{1,1}^{v,p} & \cdots & S_{1,N_{ch}}^{v,p} \\ \vdots & \ddots & \vdots \\ S_{N_{ch},1}^{v,p} & \cdots & S_{N_{ch},N_{ch}}^{v,p} \end{bmatrix} \tag{B-32}$$

$$S_{ca,ca}^{v,p} = -\frac{\sum_{ch\neq ca}^{N_{ch}}\left(v_{ca}K'_{ca,ch} - K''_{ca,ch}\right)}{L\rho_{ca}\Sigma_{ca}}\ ca = 1, \dots, N_{ch} \tag{B-33}$$

$$S_{ca,ch}^{v,p} = \frac{v_{ca}K'_{ca,ch} - K''_{ca,ch}}{L\rho_{ca}\Sigma_{ca}}\ ca, st = 1, \dots, N_{ch} \tag{B-34}$$

$$S_{p_{ch},p_{ch}} = \begin{bmatrix} S_{1,1}^{p} & \cdots & S_{1,N_{ch}}^{p} \\ \vdots & \ddots & \vdots \\ S_{N_{ch},1}^{p} & \cdots & S_{N_{ch},N_{ch}}^{p} \end{bmatrix} \tag{B-35}$$

$$S_{ca,ca}^{p} = \frac{\Phi_{ca}}{L\Sigma_{ca}}\sum_{ch\neq ca}^{N_{ch}}\left[K'''_{ca,ch} - v_{ca}K''_{ca,ch} - \left(w_{ca} - \frac{v_{ca}^2}{2} - \frac{c_{ca}^2}{\Phi_{ca}}\right)K'_{ca,ch}\right]\ ca = 1, \dots, N_{ch} \tag{B-36}$$

B MATRIX ELEMENTS

$$S^{p}_{ca,ch} = -\frac{\Phi_{ca}}{L\Sigma_{ca}}\left[K'''_{ca,ch} - v_{ca}K''_{ca,ch} - \left(w_{ca} - \frac{v_{ca}^2}{2} - \frac{c_{ca}^2}{\Phi_{ca}}\right)K'_{ca,ch}\right] ca, st = 1, \dots, N_{ch} \quad (B\text{-}37)$$

$$S_{T_{ch},p_{ch}} = \begin{bmatrix} S^{T_{ch},p}_{1,1} & \cdots & S^{T_{ch},p}_{1,N_{ch}} \\ \vdots & \ddots & \vdots \\ S^{T_{ch},p}_{N_{ch},1} & \cdots & S^{T_{ch},p}_{N_{ch},N_{ch}} \end{bmatrix} \quad (B\text{-}38)$$

$$S^{T_{ch},p}_{ca,ca} = \frac{\sum_{ch\neq ca}^{N_{ch}}\left[K'''_{ca,ch} - v_{ca}K''_{ca,ch} - \left(w_{ca} - \frac{v_{ca}^2}{2} - \Phi_{ca}c_{v,ca}\,T_{ca}\right)K'_{ca,ch}\right]}{L\rho_{ca}\Sigma_{ca}c_{v,ca}} ca = 1, \dots, N_{ch} \quad (B\text{-}39)$$

$$S^{T_{ch},p}_{ca,ch} = -\frac{K'''_{ca,ch} - v_{ca}K''_{ca,ch} - \left(w_{ca} - \frac{v_{ca}^2}{2} - \Phi_{ca}c_{v,ca}\,T_{ca}\right)K'_{ca,ch}}{L\rho_{ca}\Sigma_{ca}c_{v,ca}} ca, st = 1, \dots, N_{ch} \quad (B\text{-}40)$$

The attention is now moved towards the set of matrices grouped in the third column of equation (B-25) whose elements are the coefficients that multiply the channel temperature. It is possible to notice from equation (A-14) that there is no reference to T_{ch}, so the elements of $S_{v_{ch},T_{ch}}$ are 0 every where. On the contrary, both pressure and temperature show these addendums thus $S_{p_{ch},T_{ch}}$ and $S_{T_{ch},T_{ch}}$ are not identically null.

$$S_{p_{ch},T_{ch}} = \begin{bmatrix} S^{p,T_{ch}}_{1,1} & \cdots & S^{p,T_{ch}}_{1,N_{ch}} \\ \vdots & \ddots & \vdots \\ S^{p,T_{ch}}_{N_{ch},1} & \cdots & S^{p,T_{ch}}_{N_{ch},N_{ch}} \end{bmatrix} \quad (B\text{-}41)$$

$$S^{p,T_{ch}}_{ca,ca} = \frac{\Phi_{ca}}{\Sigma_{ca}}\left[\sum_{ch\neq ca}^{N_{ch}}\left(P^{O}_{ca,ch}\,h^{O}_{ca,ch} + P^{C}_{ca,ch}\,h^{C}_{ca,ch}\right) + \sum_{st=1}^{N_{st}} P_{ca,st}\,h_{ca,st} + \sum_{jk=1}^{N_{jk}} P_{ca,jk}\,h_{ca,jk}\right] ca = 1, \dots, N_{ch} \quad (B\text{-}42)$$

$$S^{p,T_{ch}}_{ca,ch} = -\frac{\Phi_{ca}}{\Sigma_{ca}}\left(P^{O}_{ca,ch}\,h^{O}_{ca,ch} + P^{C}_{ca,ch}\,h^{C}_{ca,ch}\right) ca, st = 1, \dots, N_{ch} \quad (B\text{-}43)$$

$$S_{T_{ch},T_{ch}} = \begin{bmatrix} S^{T_{ch}}_{1,1} & \cdots & S^{T_{ch}}_{1,N_{ch}} \\ \vdots & \ddots & \vdots \\ S^{T_{ch}}_{N_{ch},1} & \cdots & S^{T_{ch}}_{N_{ch},N_{ch}} \end{bmatrix} \quad (B\text{-}44)$$

$$S_{ca,ca}^{T_{ch}} = \frac{\sum_{ch\neq ca}^{N_{ch}}\left(P_{ca,ch}^{O}\, h_{ca,ch}^{O} + P_{ca,ch}^{C}\, h_{ca,ch}^{C}\right) + \sum_{st=1}^{N_{st}} P_{ca,st}\, h_{ca,st} + \sum_{jk=1}^{N_{jk}} P_{ca,jk}\, h_{ca,jk}}{\rho_{ca}\Sigma_{ca}c_{v,ca}} ca \quad (B\text{-}45)$$

$$= 1, \dots, N_{ch}$$

$$S_{ca,ch}^{T_{ch}} = -\frac{\left(P_{ca,ch}^{O}\, h_{ca,ch}^{O} + P_{ca,ch}^{C}\, h_{ca,ch}^{C}\right)}{\rho_{ca}\Sigma_{ca}c_{v,ca}} ca, st = 1, \dots, N_{ch} \quad (B\text{-}46)$$

Due to the coupling between fluid and solid components, also the last two matrices show elements that differ from 0.

$$S_{T_{st},T_{ch}} = \left[S_{st,ch}^{T_{st},T_{ch}}\right] = \left[-P_{st,ch}h_{st,ch}\right] st = 1, \dots, N_{st}; ch = 1, \dots, N_{ch} \quad (B\text{-}47)$$

$$S_{T_{jk},T_{ch}} = \left[S_{jk,ch}^{T_{jk},T_{ch}}\right] = \left[-P_{jk,ch}h_{jk,ch}\right] jk = 1, \dots, N_{jk}; ch = 1, \dots, N_{ch} \quad (B\text{-}48)$$

The last two columns refer to the temperature of the solid components, respectively strands and jackets temperatures and they are analyzed in parallel since they are quite similar. One more time, in the velocity equation there are no terms that refers to the temperature so both matrices $S_{v_{ch},T_{st}}$ and $S_{v_{ch},T_{jk}}$ are filled with 0.

From the pressure equations (A-15) the contributions are of the form:

$$S_{p_{ch},T_{st}} = \left[S_{ch,st}^{p,T_{st}}\right] = \left[-\frac{\Phi_{ch}P_{ch,st}\, h_{ch,st}}{\Sigma_{ch}}\right] ch = 1, \dots, N_{ch}; st = 1, \dots, N_{st} \quad (B\text{-}49)$$

$$S_{p_{ch},T_{jk}} = \left[S_{ch,jk}^{p,T_{jk}}\right] = \left[-\frac{\Phi_{ch}P_{ch,jk}\, h_{ch,jk}}{\Sigma_{ch}}\right] ch = 1, \dots, N_{ch}; jk = 1, \dots, N_{jk} \quad (B\text{-}50)$$

respectively for coefficients that multiply strands and jacket temperatures.

The analogous matrices that came from the terms of the temperature equation (A-16) are:

$$S_{T_{ch},T_{st}} = \left[S_{ch,st}^{T_{ch},T_{st}}\right] = \left[-\frac{P_{ch,st}\, h_{ch,st}}{\rho_{ch}\Sigma_{ch}c_{v,ch}}\right] ch = 1, \dots, N_{ch}; st = 1, \dots, N_{st} \quad (B\text{-}51)$$

$$S_{T_{ch},T_{jk}} = \left[S_{ch,jk}^{T_{ch},T_{jk}}\right] = \left[-\frac{P_{ch,jk}\, h_{ch,jk}}{\rho_{ch}\Sigma_{ch}c_{v,ch}}\right] ch = 1, \dots, N_{ch}; jk = 1, \dots, N_{jk} \quad (B\text{-}52)$$

These are the last matrices that have elements coming from the fluid components equations. The remaining four matrices are built considering only the 1-D transient heat transfer equations. Let focus on the matrices on the main diagonal $S_{T_{st},T_{st}}$ and $S_{T_{jk},T_{jk}}$; their elements are the coefficients that multiply the strand (jacket) temperature.

$$S_{T_{st},T_{st}} = \begin{bmatrix} S_{1,1}^{T_{st}} & \cdots & S_{1,N_{st}}^{T_{st}} \\ \vdots & \ddots & \vdots \\ S_{N_{st},1}^{T_{st}} & \cdots & S_{N_{st},N_{st}}^{T_{st}} \end{bmatrix} \quad (B\text{-}53)$$

B MATRIX ELEMENTS

$$S_{fi,fi}^{T_{st}} = \sum_{ch=1}^{N_{ch}} P_{fi,ch} h_{fi,ch} + \sum_{st \neq fi}^{N_{st}} P_{fi,st} h_{fi,st} + \sum_{jk=1}^{N_{jk}} P_{fi,jk} h_{fi,jk} \;\; fi = 1, \dots, N_{st} \tag{B-54}$$

$$S_{fi,st}^{T_{st}} = -P_{fi,st} h_{fi,st} \; fi, st = 1, \dots, N_{st} \tag{B-55}$$

$$S_{T_{jk},T_{jk}} = \begin{bmatrix} S_{1,1}^{T_{jk}} & \cdots & S_{1,N_{jk}}^{T_{jk}} \\ \vdots & \ddots & \vdots \\ S_{N_{jk},1}^{T_{jk}} & \cdots & S_{N_{jk},N_{jk}}^{T_{jk}} \end{bmatrix} \tag{B-56}$$

$$S_{in,in}^{T_{jk}} = \sum_{ch=1}^{N_{ch}} P_{in,ch} h_{in,ch} + \sum_{st=1}^{N_{st}} P_{in,st} h_{in,st} + \sum_{jk \neq in}^{N_{jk}} P_{in,jk} h_{in,jk} \;\; in = 1, \dots, N_{jk} \tag{B-57}$$

$$S_{in,jk}^{T_{jk}} = -P_{in,jk} h_{in,jk} \; in, jk = 1, \dots, N_{jk} \tag{B-58}$$

Finally, matrices $S_{T_{jk},T_{st}}$ and $S_{T_{st},T_{jk}}$ keep into account the coupling between strands and jackets, their elements are the coefficients that multiply the strand (jacket) temperature in the jacket (strand) equations, respectively.

$$S_{T_{jk},T_{st}} = \left[S_{jk,st}^{T_{jk},T_{st}} \right] = \left[-P_{jk,st} h_{jk,st} \right] jk = 1, \dots, N_{jk}; st = 1, \dots, N_{st} \tag{B-59}$$

$$S_{T_{st},T_{jk}} = \left[S_{st,jk}^{T_{st},T_{jk}} \right] = \left[-P_{st,jk} h_{st,jk} \right] st = 1, \dots, N_{st}; jk = 1, \dots, N_{jk} \tag{B-60}$$

It is worthy to notice that matrix S is not symmetric.

B.2 3P-HTS CASE STUDY: MATRICES AND VECTORS

In this section the explicit form of the matrices and of the vectors for the case study 3P-HTS are proposed. Here, the cable is discretized using one channel, one strand and one jacket; therefore, there are three fluid equations and two solid components equations to be solved for a total of five unknowns, i.e. $N_{eq} = 5$. The vectors belong to the vectorial space $\mathbb{R}^{5,1}$ while the matrices $\mathbb{R}^{5,5}$. The channel is in contact with both the strand and the jacket, but the solid components are not in mutual contact. The only driver is the external heating in the strand.

$$\boldsymbol{u} = \begin{bmatrix} v_{CH1} \\ p_{CH1} \\ T_{CH1} \\ T_{ST1} \\ T_{JK1} \end{bmatrix} \tag{B-61}$$

$$\boldsymbol{s} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ Q_{ST1,ext} \\ 0 \end{bmatrix} \tag{B-62}$$

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \Sigma_{ST1}\rho_{ST1}c_{p,ST1} & 0 \\ 0 & 0 & 0 & 0 & \Sigma_{JK1}\rho_{JK1}c_{p,JK1} \end{bmatrix} \tag{B-63}$$

$$A = \begin{bmatrix} v_{CH1} & \dfrac{1}{\rho_{CH1}} & 0 & 0 & 0 \\ \rho_{CH1}c_{CH1}^2 & v_{CH1} & 0 & 0 & 0 \\ \Phi_{CH1}T_{CH1} & 0 & v_{CH1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{B-64}$$

$$K = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\Sigma_{ST1}k_{ST1} & 0 \\ 0 & 0 & 0 & 0 & -\Sigma_{JK1}k_{JK1} \end{bmatrix} \tag{B-65}$$

$$S = \begin{bmatrix} \tilde{F}_{CH1} & 0 & 0 & 0 & 0 \\ S_{CH1,CH1}^{p,v} & 0 & S_{CH1,CH1}^{p,T_{ch}} & S_{CH1,ST1}^{p,T_{st}} & S_{CH1,JK1}^{p,T_{jk}} \\ S_{CH1,CH1}^{T_{ch},v} & 0 & S_{CH1,CH1}^{T_{ch}} & S_{CH1,ST1}^{T_{ch},T_{st}} & S_{CH1,JK1}^{T_{ch},T_{jk}} \\ 0 & 0 & S_{CH1,ST1}^{T_{st},T_{ch}} & -S_{CH1,ST1}^{T_{st},T_{ch}} & 0 \\ 0 & 0 & S_{CH1,JK1}^{T_{jk},T_{ch}} & 0 & -S_{CH1,JK1}^{T_{jk},T_{ch}} \end{bmatrix} \tag{B-66}$$

B MATRIX ELEMENTS

$$S^{p,v}_{CH1,CH1} = -\tilde{F}_{CH1}\rho_{CH1}\Phi_{CH1}v_{CH1} \quad \textit{(B-67)}$$

$$S^{p,T_{ch}}_{CH1,CH1} = \frac{\Phi_{CH1}\left(P_{CH1,ST1}\,h_{CH1,ST1} + P_{CH1,JK1}\,h_{CH1,JK1}\right)}{\Sigma_{CH1}} \quad \textit{(B-68)}$$

$$S^{p,T_{st}}_{CH1,ST1} = -\frac{\Phi_{CH1}P_{CH1,ST1}\,h_{CH1,ST1}}{\Sigma_{CH1}} \quad \textit{(B-69)}$$

$$S^{p,T_{jk}}_{CH1,JK1} = -\frac{\Phi_{CH1}P_{CH1,JK1}\,h_{CH1,JK1}}{\Sigma_{CH1}} \quad \textit{(B-70)}$$

$$S^{T_{ch},v}_{CH1,CH1} = -\frac{\tilde{F}_{CH1}v_{CH1}}{c_{v,CH1}} \quad \textit{(B-71)}$$

$$S^{T_{ch}}_{CH1,CH1} = \frac{P_{CH1,ST1}\,h_{CH1,ST1} + P_{CH1,JK1}\,h_{CH1,JK1}}{\rho_{CH1}\Sigma_{CH1}c_{v,CH1}} \quad \textit{(B-72)}$$

$$S^{T_{ch}}_{CH1,CH1} = \frac{P_{CH1,ST1}\,h_{CH1,ST1} + P_{CH1,JK1}\,h_{CH1,JK1}}{\rho_{CH1}\Sigma_{CH1}c_{v,CH1}} \quad \textit{(B-73)}$$

$$S^{T_{ch},T_{st}}_{CH1,ST1} = -\frac{P_{CH1,ST1}\,h_{CH1,ST1}}{\rho_{CH1}\Sigma_{CH1}c_{v,CH1}} \quad \textit{(B-74)}$$

$$S^{T_{ch},T_{jk}}_{CH1,JK1} = -\frac{P_{CH1,JK1}\,h_{CH1,JK1}}{\rho_{CH1}\Sigma_{CH1}c_{v,CH1}} \quad \textit{(B-75)}$$

$$S^{T_{st},T_{ch}}_{CH1,ST1} = -P_{CH1,ST1}h_{CH1,ST1} \quad \textit{(B-76)}$$

$$S^{T_{st}}_{ST1,ST1} = P_{CH2,ST1}h_{CH2,ST1} = -S^{T_{st},T_{ch}}_{CH1,ST1} \quad \textit{(B-77)}$$

$$S^{T_{jk},T_{ch}}_{CH1,JK1} = -P_{CH1,JK1}h_{CH1,JK1} \quad \textit{(B-78)}$$

$$S^{T_{jk}}_{JK1,JK1} = P_{CH2,JK1}h_{CH2,JK1} = -S^{T_{jk},T_{ch}}_{CH1,JK1} \quad \textit{(B-79)}$$

B.3 ITER-TF CASE STUDY: MATRICES AND VECTORS

In this section the explicit form of the matrices and of the vector for the case study ITER-TF are proposed. In the present case, the cable is discretized with two channels, one strand and one jacket; therefore, there are six fluid equations and two solid components equations to be solved for a total of eight unknowns, i.e. $N_{eq} = 8$. The vectors belong to the vectorial space $\mathbb{R}^{8,1}$ while the matrices $\mathbb{R}^{8,8}$. The two channels are in hydraulic parallel, the two solid components are in thermal contact while only the second channel is in contact with both the strand and the jacket. The only driver is the external heating in the strand.

$$\boldsymbol{u} = [v_{CH1} \quad p_{CH1} \quad T_{CH1} \quad v_{CH2} \quad p_{CH2} \quad T_{CH2} \quad T_{ST1} \quad T_{JK1}]^T \tag{B-80}$$

$$\boldsymbol{s} = [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad Q_{ST1,ext} \quad 0]^T \tag{B-81}$$

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \Sigma_{ST1}\rho_{ST1}c_{p,ST1} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \Sigma_{JK1}\rho_{JK1}c_{p,JK1} \end{bmatrix} \tag{B-82}$$

$$A = \begin{bmatrix} v_{CH1} & 0 & \frac{1}{\rho_{CH1}} & 0 & 0 & 0 & 0 & 0 \\ 0 & v_{CH2} & 0 & \frac{1}{\rho_{CH2}} & 0 & 0 & 0 & 0 \\ \rho_{CH1}c_{CH1}^2 & 0 & v_{CH1} & 0 & 0 & 0 & 0 & 0 \\ 0 & \rho_{CH2}c_{CH2}^2 & 0 & v_{CH2} & 0 & 0 & 0 & 0 \\ \Phi_{CH1}T_{CH1} & 0 & 0 & 0 & v_{CH1} & 0 & 0 & 0 \\ 0 & \Phi_{CH2}T_{CH2} & 0 & 0 & 0 & v_{CH2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{B-83}$$

$$K = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\Sigma_{ST1}k_{ST1} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\Sigma_{JK1}k_{JK1} \end{bmatrix} \tag{B-84}$$

B MATRIX ELEMENTS

$$S = \begin{bmatrix} \tilde{F}_{CH1} & 0 & S^{v,p}_{CH1,CH1} & -S^{v,p}_{CH1,CH1} & 0 & 0 & 0 & 0 \\ 0 & \tilde{F}_{CH2} & -S^{v,p}_{CH2,CH2} & S^{v,p}_{CH2,CH2} & 0 & 0 & 0 & 0 \\ S^{p,v}_{CH1,CH1} & 0 & S^{p}_{CH1,CH1} & -S^{p}_{CH1,CH1} & S^{p,T_{ch}}_{CH1,CH1} & -S^{p,T_{ch}}_{CH1,CH1} & 0 & 0 \\ 0 & S^{p,v}_{CH2,CH2} & -S^{p}_{CH2,CH2} & S^{p}_{CH2,CH2} & S^{p,T_{ch}}_{CH2,CH1} & S^{p,T_{ch}}_{CH2,CH2} & S^{p,T_{st}}_{CH2,ST1} & S^{p,T_{jk}}_{CH2,JK1} \\ S^{T_{ch},v}_{CH1,CH1} & 0 & S^{T_{ch},p}_{CH1,CH1} & -S^{T_{ch},p}_{CH1,CH1} & S^{T_{ch}}_{CH1,CH1} & -S^{T_{ch}}_{CH1,CH1} & 0 & 0 \\ 0 & S^{T_{ch},v}_{CH2,CH2} & -S^{T_{ch},p}_{CH2,CH2} & S^{T_{ch},p}_{CH2,CH2} & S^{T_{ch}}_{CH2,CH1} & S^{T_{ch}}_{CH2,CH2} & S^{T_{ch},T_{st}}_{CH2,ST1} & S^{T_{ch},T_{jk}}_{CH2,JK1} \\ 0 & 0 & 0 & 0 & 0 & S^{T_{st},T_{ch}}_{CH2,ST1} & S^{T_{st}}_{ST1,ST1} & S^{T_{st},T_{jk}}_{ST1,JK1} \\ 0 & 0 & 0 & 0 & 0 & S^{T_{jk},T_{ch}}_{CH2,JK1} & S^{T_{jk},T_{st}}_{JK1,ST1} & S^{T_{jk}}_{JK1,JK1} \end{bmatrix} \quad (B\text{-}85)$$

$$S^{v,p}_{CH1,CH1} = -\frac{v_{CH1}K'_{CH1,CH2} - K''_{CH1,CH2}}{L\rho_{CH1}\Sigma_{CH1}} \quad (B\text{-}86)$$

$$S^{v,p}_{CH1,CH2} = \frac{v_{CH1}K'_{CH1,CH2} - K''_{CH1,CH2}}{L\rho_{CH1}\Sigma_{CH1}} = -S^{v,p}_{CH1,CH1} \quad (B\text{-}87)$$

$$S^{v,p}_{CH2,CH2} = -\frac{v_{CH2}K'_{CH1,CH2} - K''_{CH1,CH2}}{L\rho_{CH2}\Sigma_{CH2}} \quad (B\text{-}88)$$

$$S^{v,p}_{CH2,CH1} = \frac{v_{CH2}K'_{CH1,CH2} - K''_{CH1,CH2}}{L\rho_{CH2}\Sigma_{CH2}} = -S^{v,p}_{CH2,CH2} \quad (B\text{-}89)$$

$$S^{p,v}_{CH1,CH1} = -\tilde{F}_{CH1}\rho_{CH1}\Phi_{CH1}v_{CH1} \quad (B\text{-}90)$$

$$S^{p}_{CH1,CH1} = \frac{\Phi_{CH1}}{L\Sigma_{CH1}}\left[K'''_{CH1,CH2} - v_{CH1}K''_{CH1,CH2} - \left(w_{CH1} - \frac{v^2_{CH1}}{2} - \frac{c^2_{CH1}}{\Phi_{CH1}}\right)K'_{CH1,CH2}\right] \quad (B\text{-}91)$$

$$S^{p}_{CH1,CH2} = -\frac{\Phi_{CH1}}{L\Sigma_{CH1}}\left[K'''_{CH1,CH2} - v_{CH1}K''_{CH1,CH2} - \left(w_{CH1} - \frac{v^2_{CH1}}{2} - \frac{c^2_{CH1}}{\Phi_{CH1}}\right)K'_{CH1,CH2}\right] = -S^{p}_{CH1,CH1} \quad (B\text{-}92)$$

$$S^{p,T_{ch}}_{CH1,CH1} = \frac{\Phi_{CH1}\left(P^{O}_{CH1,CH2}\,h^{O}_{CH1,CH2} + P^{C}_{CH1,CH2}\,h^{C}_{CH1,CH2}\right)}{\Sigma_{CH1}} \quad (B\text{-}93)$$

$$S^{p,T_{ch}}_{CH1,CH2} = -\frac{\Phi_{CH1}\left(P^{O}_{CH1,CH2}\,h^{O}_{CH1,CH2} + P^{C}_{CH1,CH2}\,h^{C}_{CH1,CH2}\right)}{\Sigma_{CH1}} = -S^{p,T_{ch}}_{CH1,CH1} \quad (B\text{-}94)$$

$$S^{p,v}_{CH2,CH2} = -\tilde{F}_{CH2}\rho_{CH2}\Phi_{CH2}v_{CH2} \quad (B\text{-}95)$$

$$S^{p}_{CH2,CH2} = \frac{\Phi_{CH2}}{L\Sigma_{CH2}}\left[K'''_{CH1,CH2} - v_{CH2}K''_{CH1,CH2} - \left(w_{CH2} - \frac{v^2_{CH2}}{2} - \frac{c^2_{CH2}}{\Phi_{CH2}}\right)K'_{CH1,CH2}\right] \quad (B\text{-}96)$$

$$S^{p}_{CH2,CH1} = -\frac{\Phi_{CH2}}{L\Sigma_{CH2}}\left[K'''_{CH1,CH2} - v_{CH2}K''_{CH1,CH2} - \left(w_{CH2} - \frac{v^2_{CH2}}{2} - \frac{c^2_{CH2}}{\Phi_{CH2}}\right)K'_{CH1,CH2}\right] = -S^{p}_{CH2,CH2} \quad (B\text{-}97)$$

B.3 ITER-TF CASE STUDY: MATRICES AND VECTORS

$$S_{CH2,CH1}^{p,T_{ch}} = -\frac{\Phi_{CH2}\left(P_{CH1,CH2}^{O}\, h_{CH1,CH2}^{O} + P_{CH1,CH2}^{C}\, h_{CH1,CH2}^{C}\right)}{\Sigma_{CH2}} \quad \text{(B-98)}$$

$$S_{CH2,CH2}^{p,T_{ch}} = \frac{\Phi_{CH2}}{\Sigma_{CH2}}\left(P_{CH1,CH2}^{O}\, h_{CH1,CH2}^{O} + P_{CH1,CH2}^{C}\, h_{CH1,CH2}^{C} + P_{CH2,ST1}\, h_{CH2,ST1} + P_{CH2,JK1}\, h_{CH2,JK1}\right) \quad \text{(B-99)}$$

$$S_{CH1,CH1}^{T_{ch},v} = -\frac{\tilde{F}_{CH1} v_{CH1}}{c_{v,CH1}} \quad \text{(B-100)}$$

$$S_{CH1,CH1}^{T_{ch},p} = \frac{K_{CH1,CH2}''' - v_{CH1} K_{CH1,CH2}'' - \left(w_{CH1} - \frac{v_{CH1}^2}{2} - \Phi_{CH1} c_{v,CH1}\, T_{CH1}\right) K_{CH1,CH2}'}{L \rho_{CH1} \Sigma_{CH1} c_{v,CH1}} \quad \text{(B-101)}$$

$$S_{CH1,CH2}^{T_{ch},p} = -\frac{K_{CH1,CH2}''' - v_{CH1} K_{CH1,CH2}'' - \left(w_{CH1} - \frac{v_{CH1}^2}{2} - \Phi_{CH1} c_{v,CH1}\, T_{CH1}\right) K_{CH1,CH2}'}{L \rho_{CH1} \Sigma_{CH1} c_{v,CH1}} = -S_{CH1,CH1}^{T_{ch},p} \quad \text{(B-102)}$$

$$S_{CH1,CH1}^{T_{ch}} = \frac{P_{CH1,CH2}^{O}\, h_{CH1,CH2}^{O} + P_{CH1,CH2}^{C}\, h_{CH1,CH2}^{C}}{\rho_{CH1} \Sigma_{CH1} c_{v,CH1}} \quad \text{(B-103)}$$

$$S_{CH1,CH2}^{T_{ch}} = -\frac{P_{CH1,CH2}^{O}\, h_{CH1,CH2}^{O} + P_{CH1,CH2}^{C}\, h_{CH1,CH2}^{C}}{\rho_{CH1} \Sigma_{CH1} c_{v,CH1}} = -S_{CH1,CH1}^{T_{ch}} \quad \text{(B-104)}$$

$$S_{CH2,CH2}^{T_{ch},v} = -(F\tilde{}\_CH2\ v\_CH2)/c\_(v, CH2) \quad \text{(B-105)}$$

$$S_{CH2,CH2}^{T_{ch},p} = \frac{K_{CH1,CH2}''' - v_{CH2} K_{CH1,CH2}'' - \left(w_{CH2} - \frac{v_{CH2}^2}{2} - \Phi_{CH2} c_{v,CH2}\, T_{CH2}\right) K_{CH1,CH2}'}{L \rho_{CH2} \Sigma_{CH2} c_{v,CH2}} \quad \text{(B-106)}$$

$$S_{CH2,CH1}^{T_{ch},p} = -\frac{K_{CH1,CH2}''' - v_{CH2} K_{CH1,CH2}'' - \left(w_{CH2} - \frac{v_{CH2}^2}{2} - \Phi_{CH2} c_{v,CH2}\, T_{CH2}\right) K_{CH1,CH2}'}{L \rho_{CH2} \Sigma_{CH2} c_{v,CH2}} = -S_{CH2,CH2}^{T_{ch},p} \quad \text{(B-107)}$$

$$S_{CH2,CH1}^{T_{ch}} = -\frac{P_{CH1,CH2}^{O}\, h_{CH1,CH2}^{O} + P_{CH1,CH2}^{C}\, h_{CH1,CH2}^{C}}{\rho_{CH2} \Sigma_{CH2} c_{v,CH2}} \quad \text{(B-108)}$$

$$S_{CH2,CH2}^{T_{ch}} = \frac{P_{CH1,CH2}^{O}\, h_{CH1,CH2}^{O} + P_{CH1,CH2}^{C}\, h_{CH1,CH2}^{C} + P_{CH2,ST1}\, h_{CH2,ST1} + P_{CH2,JK1}\, h_{CH2,JK1}}{\rho_{CH2} \Sigma_{CH2} c_{v,CH2}} \quad \text{(B-109)}$$

$$S_{CH2,ST1}^{T_{ch},T_{st}} = -\frac{P_{CH2,ST1}\, h_{CH2,ST1}}{\rho_{CH2} \Sigma_{CH2} c_{v,CH2}} \quad \text{(B-110)}$$

$$S_{CH2,JK1}^{T_{ch},T_{jk}} = -\frac{P_{CH2,JK1}\, h_{CH2,JK1}}{\rho_{CH2} \Sigma_{CH2} c_{v,CH2}} \quad \text{(B-111)}$$

$$S_{CH2,ST1}^{T_{st},T_{ch}} = -P_{CH2,ST1} h_{CH2,ST1} \quad \text{(B-112)}$$

B MATRIX ELEMENTS

$$S^{T_{st}}_{ST1,ST1} = P_{CH2,ST1}h_{CH2,ST1} + P_{ST1,JK1}h_{ST1,JK1} \quad \textit{(B-113)}$$

$$S^{T_{st},T_{jk}}_{ST1,JK1} = -P_{ST1,JK1}h_{ST1,JK1} \quad \textit{(B-114)}$$

$$S^{T_{jk},T_{ch}}_{CH2,JK1} = -P_{CH2,JK1}h_{CH2,JK1} \quad \textit{(B-115)}$$

$$S^{T_{jk},T_{st}}_{JK1,ST1} = -P_{ST1,JK1}h_{ST1,JK1} \quad \textit{(B-116)}$$

$$S^{T_{jk}}_{JK1,JK1} = P_{CH2,JK1}h_{CH2,JK1} + P_{ST1,JK1}h_{ST1,JK1} \quad \textit{(B-117)}$$

APPENDIX C

C EVALUATION OF THE PROPERTIES OF THE SOLID COMPONENTS

The correct formulation of the average properties for solid components are discussed in this appendix. The proposed formulation is validated since it allows the respect of the first principle of the Thermodynamic, the energy conservation. The appendix ends showing that a different formulation of the average specific heat leads to the violation of the energy conservation.

First, let show that the currently used average formulas are correct. If the solid components object is constituted by N_{mat} materials the average density, specific heat, thermal conductivity and heat capacity are evaluated according to the following definitions:

$$\rho = \frac{\Sigma_1\rho_1 + \cdots + \Sigma_i\rho_i + \cdots + \Sigma_{N_{mat}}\rho_{N_{mat}}}{\Sigma_1 + \cdots + \Sigma_i + \cdots + \Sigma_{N_{mat}}} = \frac{\sum_{i=1}^{N_{mat}}\Sigma_i\rho_i}{\sum_{i=1}^{N_{mat}}\Sigma_i} \tag{C-1}$$

$$c_p = \frac{\Sigma_1\rho_1 c_{p_1} + \cdots + \Sigma_i\rho_i c_{p_i} + \cdots + \Sigma_{N_{mat}}\rho_{N_{mat}} c_{p_{N_{mat}}}}{\Sigma_1\rho_1 + \cdots + \Sigma_i\rho_i + \cdots + \Sigma_{N_{mat}}\rho_{N_{mat}}} = \frac{\sum_{i=1}^{N_{mat}}\Sigma_i\rho_i c_{p_i}}{\sum_{i=1}^{N_{mat}}\Sigma_i\,\rho_i} \tag{C-2}$$

$$k = \frac{\Sigma_1 k_1 + \cdots + \Sigma_i k_i + \cdots + \Sigma_{N_{mat}} k_{N_{mat}}}{\Sigma_1 + \cdots + \Sigma_i + \cdots + \Sigma_{N_{mat}}} = \frac{\sum_{i=1}^{N_{mat}}\Sigma_i k_i}{\sum_{i=1}^{N_{mat}}\Sigma_i} \tag{C-3}$$

$$C = \frac{\Sigma_1\rho_1 c_{p_1} + \cdots + \Sigma_i\rho_i c_{p_i} + \cdots + \Sigma_{N_{mat}}\rho_{N_{mat}} c_{p_{N_{mat}}}}{\Sigma_1 + \cdots + \Sigma_i + \cdots + \Sigma_{N_{mat}}} = \frac{\sum_{i=1}^{N_{mat}}\Sigma_i\rho_i c_{p_i}}{\sum_{i=1}^{N_{mat}}\Sigma_i\,\rho_i} \tag{C-4}$$

The transient one-dimensional heat transfer equation for a multi material solid component, like mixed strands or jackets, with no energy source reads:

$$\sum_{i=1}^{N_{mat}}\Sigma_i\rho_i c_{p_i}\frac{\partial T_i}{\partial t} - \sum_{i=1}^{N_{mat}}\Sigma_i\frac{\partial}{\partial x}\left(k_i\frac{\partial T_i}{\partial x}\right) = 0 \tag{C-5}$$

Assuming that cross sections are constant and that the temperature dependence in space and time is always the same for each component, this equation can be written in a different form. Since:

$$T_1(x,t) = T_2(x,t) = \cdots = T_i(x,t) = \cdots = T_{N_{mat}}(x,t) = T(x,t) \tag{C-6}$$

it is possible to collect both time and space temperature derivatives, moreover the cross section is constant, therefore:

$$\begin{aligned}\left(\Sigma_1\rho_1 c_{p_1} + \cdots + \Sigma_i\rho_i c_{p_i} + \cdots + \Sigma_{N_{mat}}\rho_{N_{mat}} c_{p_{N_{mat}}}\right)\frac{\partial T}{\partial t} \\ -\frac{\partial}{\partial x}\left[\left(\Sigma_1 k_1 + \cdots + \Sigma_i k_i + \cdots + \Sigma_{N_{mat}} k_{N_{mat}}\right)\frac{\partial T}{\partial x}\right] = 0\end{aligned} \tag{C-7}$$

Let divide both left- and right-hand side by the total cross section A:

$$\frac{\Sigma_1\rho_1 c_{p_1} + \cdots + \Sigma_i\rho_i c_{p_i} + \cdots + \Sigma_{N_{mat}}\rho_{N_{mat}} c_{p_{N_{mat}}}}{\Sigma}\frac{\partial T}{\partial t} - \frac{\partial}{\partial x}\left(\frac{\Sigma_1 k_1 + \cdots + \Sigma_i k_i + \cdots + \Sigma_{N_{mat}} k_{N_{mat}}}{\Sigma}\frac{\partial T}{\partial x}\right) = 0 \qquad \textit{(C-8)}$$

Exploiting the definitions (C-2)and (C-3), the above equation can be written as:

$$C\frac{\partial T}{\partial t} - \frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right) = 0 \qquad \textit{(C-9)}$$

that is the transient 1D heat equation for a homogeneous material, with no external source.

In SC2 code, instead of the average heat capacity, the average density and the average specific heat at constant pressure are used. The two formulations are equivalent if the averages are defined as in (C-1)-(C-4). To proof this, the starting equation is:

$$\Sigma\rho c_p\frac{\partial T}{\partial t} - A\frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right) = 0 \qquad \textit{(C-10)}$$

Introducing equation (C-1) and (C-2) inside equation (C-5) yields:

$$\Sigma\frac{\sum_{i=1}^{N_{mat}}\Sigma_i\rho_i}{\sum_{i=1}^{N_{mat}}\Sigma_i}\frac{\sum_{i=1}^{N_{mat}}\Sigma_i\rho_i c_{p_i}}{\sum_{i=1}^{N_{mat}}\Sigma_i\,\rho_i}\frac{\partial T}{\partial t} - \Sigma\frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right) = 0 \qquad \textit{(C-11)}$$

That simplifies to:

$$\Sigma\frac{\sum_{i=1}^{N_{mat}}\Sigma_i\rho_i c_{p_i}}{\sum_{i=1}^{N_{mat}}\Sigma_i}\frac{\partial T}{\partial t} - \Sigma\frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right) = 0 \qquad \textit{(C-12)}$$

Dividing by Σ and remembering the definition (C-4) the above equation becomes equal to equation (C-9).

To complete the proof, let show that from equation (C-9) it is possible to get (C-10).

Substituting in (C-9) the definition (C-4) gives:

$$\frac{\sum_{i=1}^{N_{mat}}\Sigma_i\rho_i c_{p_i}}{\sum_{i=1}^{N_{mat}}\Sigma_i}\frac{\partial T}{\partial t} - \frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right) = 0 \qquad \textit{(C-13)}$$

Multiplying and dividing the first addendum on the left-hand side by $\sum_{i=1}^{N_{mat}}\Sigma_i\rho_i$, adjusting the terms results in:

$$\frac{\sum_{i=1}^{N_{mat}}\Sigma_i\rho_i}{\sum_{i=1}^{N_{mat}}\Sigma_i}\frac{\sum_{i=1}^{N_{mat}}\Sigma_i\rho_i c_{p_i}}{\sum_{i=1}^{N_{mat}}\Sigma_i\rho_i}\frac{\partial T}{\partial t} - \frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right) = 0 \qquad \textit{(C-14)}$$

Recognizing the definitions of the average density and average specific heat given in equations (C-1) and (C-2), this equation is the same of equation (C-10). Therefore, with the

definitions given in (C-1)-(C-4)equations (C-9) and (C-10)are equivalent, so the current choice does not violate the first principle of the thermodynamics.

In the end, if the definition of the specific heat at constant pressure (C-2) is replaced by the following one (C-15), the energy conservation is no longer guaranteed since the resulting equation is not equivalent to (C-9).

$$c_p = \frac{\sum_{i=1}^{N_{mat}} \Sigma_i c_{p_i}}{\sum_{i=1}^{N_{mat}} \Sigma_i} \qquad \textit{(C-15)}$$

Substituting (C-1) and the above in equation (C-5) yields:

$$\Sigma \frac{\sum_{i=1}^{N_{mat}} \Sigma_i \rho_i}{\sum_{i=1}^{N_{mat}} \Sigma_i} \frac{\sum_{i=1}^{N_{mat}} \Sigma_i c_{p_i}}{\sum_{i=1}^{N_{mat}} \Sigma_i} \frac{\partial T}{\partial t} - \Sigma \frac{\partial}{\partial x}\left(k \frac{\partial T}{\partial x}\right) = 0 \qquad \textit{(C-16)}$$

which simplifies to:

$$\frac{\left(\sum_{i=1}^{N_{mat}} \Sigma_i \rho_i\right)\left(\sum_{i=1}^{N_{mat}} \Sigma_i c_{p_i}\right)}{\sum_{i=1}^{N_{mat}} \Sigma_i} \frac{\partial T}{\partial t} - \Sigma \frac{\partial}{\partial x}\left(k \frac{\partial T}{\partial x}\right) = 0 \qquad \textit{(C-17)}$$

that, being not equivalent to equation (C-9), does not respect the conservation of energy.

APPENDIX D

D INPUT DATA OF THE SIMULATIONS

This appendix contains the data tables needed to perform the simulations used to carry out the convergence studies and the external and internal benchmarks which are common to all the simulations. The first section contains the data for the 3P-HTS geometry, the second one summarizes the data for the ITER-TF geometry.

D.1 3P-HTS INPUT DATA

Figure D.1-1 shows a sketch of the conductor geometry and topology.

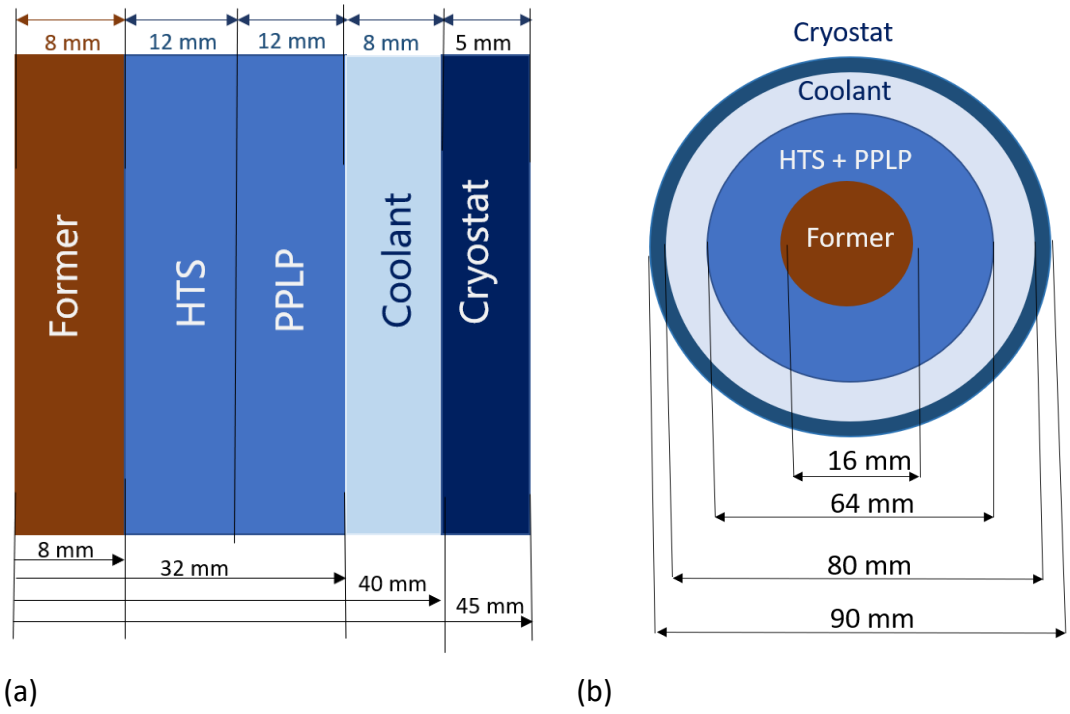


Figure D.1-1 Three phase coaxial HTS cable configuration: (a) radial view; (c) cross section. Note that in (b) the region HTS and PPLP are collapsed in a single one. Non-scale figure.

Table D.1-1 Input data to be set in the workbooks Transitory_Input.xlsx and conductor_definition.xlsx.

Variable	Value	Unit SI	Meaning
IADAPTIME	0	-	Flag for the time adaptivity: no adaptivity (constant time step)
XLENGHT	10.0	m	Conductor length
IOPFUN	0	-	Flag for the current function: constant
IOP0_TOT	0	A	Total current transported at time = 0 s
UPWIND	1	-	Flag to switch on/off the upwind discretization in all the fluid equations: on

D.1 3P-HTS INPUT DATA

Table D.1-2 Input data to be set in the CHAN sheet of the workbook conductor_input.xlsx. Cross section and hydraulic diameter computed from data in [77].

Variable	Value	Unit SI	Meaning
CROSSECTION	$1.81\ 10^{-3}$	m^2	Channel cross section
FLUID_TYPE	He	-	Type of coolant: helium
HYDIAMETER	$1.601\ 10^{-2}$	m	Hydraulic diameter
COSTETA	1	-	Cosine of the angle between the cable and the x axis
VOID_FRACTION	1	-	Void fraction of the channel
IFRICTION	-99	-	Flag to select the correlation for the friction factor: user defined
FRICTION_MULTIPLIER	$1.00\ 10^{-3}$	-	Multiplier of the friction factor: the actual value in this case
ISRECTANGULAR	0	-	Flag to set the shape of the channel: circular
SIDE1	0.00	m	Side of the rectangle if ISRECTANGULAR equal 1
SIDE2	0.00	m	Side of the rectangle if ISRECTANGULAR equal 1

Table D.1-3 Input data to be set in the STR_MIX sheet of the workbook conductor_input.xlsx. Cross section and stabilizer non-stabilizer ratio computed from data in [77], copper residual resistance ratio] and superconductor scaling properties from [87].

Variable	Value	Unit SI	Meaning
CROSSECTION	$3.22\ 10^{-3}$	m^2	Strand total cross section
COSTETA	1.00	-	Cosine of the angle between the cable and the x axis
STAB_NON_STAB	2.047	-	Stabilizer non stabilizer ratio
NUM_MATERIAL_TYPES	2	-	Number of materials making up the strand
ISTABILIZER	1	-	Flag for the stabilizer material: copper (Cu)
RRR	100	-	Copper residual resistance ratio
ISUPERCONDUCTOR	2	-	Flag for the superconductor material: Nb_3Sn
c0	$1.21508\ 10^{11}$	$\mathrm{AT/m}^2$	Property of superconductor scaling
Bc20m	32.35	T	Property of superconductor scaling
Tc0m	16.22	K	Property of superconductor scaling

Table D.1-4 Input data to be set in the Z_JACKET sheet of the workbook conductor_input.xlsx. Cross section computed from data in [77].

Variable	Value	Unit SI	Meaning
CROSSECTION_JK	$1.33\ 10^{-3}$	m^2	Pure jacket cross section
CROSSECTION_IN	0.00	m^2	Insulation cross section
NUM_MATERIAL_TYPES	1	-	Number of materials making up the jacket
IMATERIAL_JK	1	-	Flag for pure jacket material: stainless steel
IMATERIAL_IN	0	-	Flag for insulation material: no insulation
COSTETA	1.00	-	Cosine of the angle between the cable and the x axis

Table D.1-5 Input data to be set in the STR_MIX sheet of the workbook conductor_operation.xlsx.

Variable	Value	Unit SI	Meaning
IEPS	0	-	Flag to define the strain along the strand: no strain
EPS	0	-	Value of the strain along the strand
IQFUN	1	-	Flag for the heat source shape: square wave in space and time
INTIAL	0	-	Flag to define how to initialize the STR_MIX temperature spatial distribution: from channel initial temperature

Table D.1-6 Input data to be set in the Z_JACKET sheet of the workbook conductor_operation.xlsx.

Variable	Value	Unit SI	Meaning
IQFUN	0	-	Flag for the heat source shape: no heat source
INTIAL	0	-	Flag to define how to initialize the Z_JACKET temperature spatial distribution: from channel initial temperature

Table D.1-7 Contact perimeter flags, dimensionless. If 1 cable components are in contact, if 0 there is no contact between the components. Values to be inserted in sheet contact_perimeter_flag of workbook conductor_coupling.xlsx.

	CHAN_1	STR_MIX_1	Z_JACKET_1
CHAN_1	0	1	1
STR_MIX_1	0	0	0
Z_JACKET_1	0	0	0

Table D.1-8 Contact perimeter in m; values to be inserted in sheet contact_perimeter of workbook conductor_coupling.xlsx. Values computed from data in [77].

	CHAN_1	STR_MIX_1	Z_JACKET_1
CHAN_1	0	0.20096	0.25133
STR_MIX_1	0	0	0
Z_JACKET_1	0	0	0

Table D.1-9 Flags to select the correlation to compute the heat transfer coefficients, dimensionless. Flag on the channel main diagonal: correlation to evaluate the steady heat transfer coefficients for channel objects (not used in this case); other values: flag to select the correlation to evaluate interface heat transfer coefficients, if -1 they are user defined. Values to be inserted in sheet HTC_choice of workbook conductor_coupling.xlsx.

	CHAN_1	STR_MIX_1	Z_JACKET_1
CHAN_1	0	−1	−1
STR_MIX_1	0	0	0
Z_JACKET_1	0	0	0

Table D.1-10 Values of the heat transfer coefficient if user defined, in $W/m^2/K$. Values to be inserted in sheet contact_HTC of workbook conductor_coupling.xlsx.

	CHAN_1	STR_MIX_1	Z_JACKET_1
CHAN_1	0	1000	1000
STR_MIX_1	0	0	0
Z_JACKET_1	0	0	0

Table D.1-11 Values for the heat transfer coefficient dimensionless multipliers, when needed; to be inserted in sheet HTC_multiplier of workbook conductor_coupling.xlsx.

	CHAN_1	STR_MIX_1	Z_JACKET_1
CHAN_1	0	1	1
STR_MIX_1	0	0	0
Z_JACKET_1	0	0	0

Table D.1-12 Fraction of the contact perimeter between fluid components which is actually open, dimensionless. Values to be inserted in sheet open_perimeter_fract of workbook conductor_coupling.xlsx.

	CHAN_1	STR_MIX_1	Z_JACKET_1
CHAN_1	0	0	0
STR_MIX_1	0	0	0
Z_JACKET_1	0	0	0

Table D.1-13 Thickness of the interface between fluid elements, in m. Values to be inserted in sheet interf_thickness of workbook conductor_coupling.xlsx.

	CHAN_1	STR_MIX_1	Z_JACKET_1
CHAN_1	0	0	0
STR_MIX_1	0	0	0
Z_JACKET_1	0	0	0

Table D.1-14 Multiplier for the transport properties in the open fraction of the perimeter between fluid components, dimensionless. Values to be inserted in sheet trans_transp_multiplier of workbook conductor_coupling.xlsx.

	CHAN_1	STR_MIX_1	Z_JACKET_1
CHAN_1	0	0	0
STR_MIX_1	0	0	0
Z_JACKET_1	0	0	0

D.2 ITER-TF INPUT DATA

Figure D.2-1 shows a sketch of the conductor geometry.

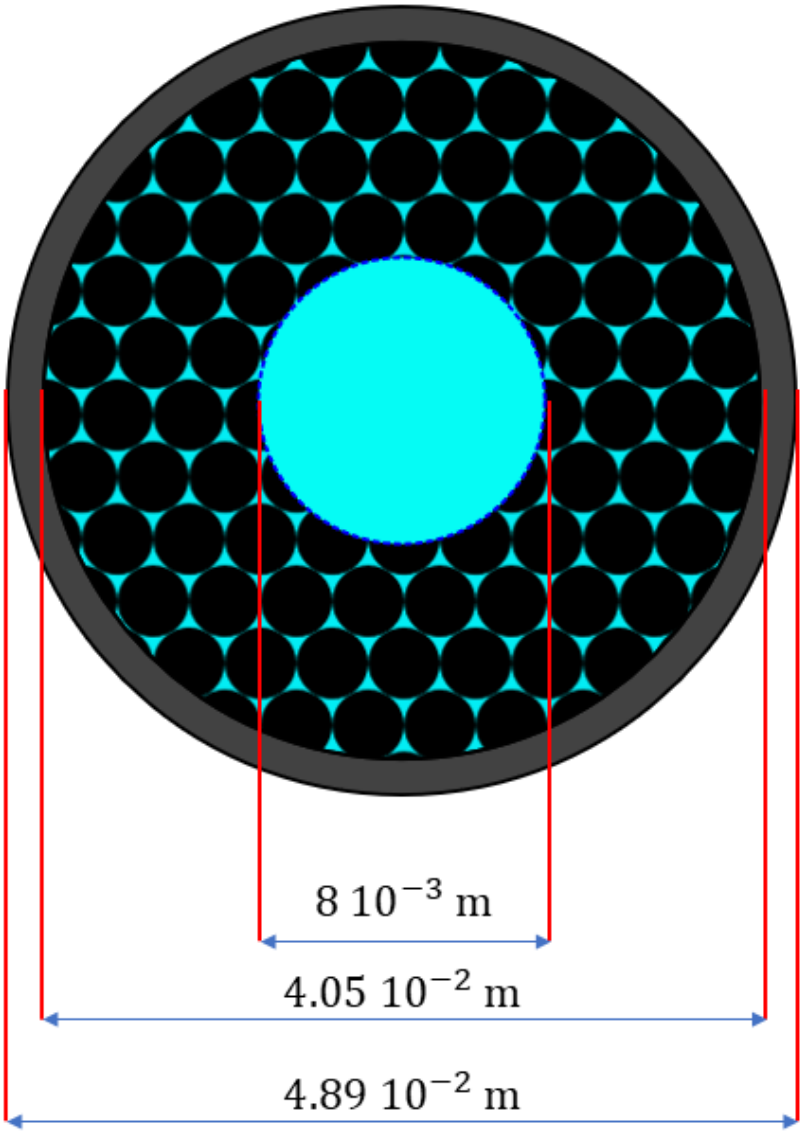


Figure D.2-1 Sketch of the cross section of the ITER-TF configuration. Coolant He in cyan, mixed strand of Cu and NB_3Sn in black and jacket/insulation in grey. Not scaled figure.

D.2 ITER-TF INPUT DATA

Table D.2-1 Input data to be set in the workbooks Transitory_Input.xlsx and conductor_definition.xlsx.

Variable	Value	Unit SI	Meaning
IADAPTIME	0	-	Flag for the time adaptivity: no adaptivity (constant time step)
XLENGHT	10.0	m	Conductor length
IOPFUN	0	-	Flag for the current function: constant
IOP0_TOT	0	A	Total current transported at time = 0 s
UPWIND	1	-	Flag to switch on/off the upwind discretization in all the fluid equations: on

Table D.2-2 Input data to be set in the CHAN sheet of the workbook conductor_input.xlsx. Cross sections and hydraulic diameters computed from data in [87].

Variables	Value		Unit SI	Meaning
	CHAN_1	CHAN_2	m^2	Channel cross section
CROSSECTION	$5.0265\ 10^{-5}$	$3.6965\ 10^{-4}$	-	Type of coolant: helium
FLUID_TYPE	He	He	m	Hydraulic diameter
HYDIAMETER	$8.0\ 10^{-3}$	$3.2676\ 10^{-4}$	-	Cosine of the angle between the cable and the x axis
COSTETA	1	1	-	Void fraction of the channel
VOID_FRACTION	1	0.313	-	Flag to select the correlation for the friction factor: user defined
IFRICTION	−99	−99	-	Multiplier of the friction factor: the actual value in this case
FRICTION_MULTIPLIER	0.02	0.02	-	Flag to set the shape of the channel: circular
ISRECTANGULAR	0	0	m	Side of the rectangle if ISRECTANGULAR equal 1
SIDE1	0.00	$0.00E$	m	Side of the rectangle if ISRECTANGULAR equal 1
SIDE2	0.00	0.00	m^2	Channel cross section

D INPUT DATA OF THE SIMULATIONS

Table D.2-3 Input data to be set in the STR_MIX sheet of the workbook conductor_input.xlsx. Cross section and stabilizer non-stabilizer ratio computed from data in [87], copper residual resistance ratio and Nb_3Sn scaling properties came from the same paper.

Variable	Value	Unit SI	Meaning
CROSSECTION	$7.54\ 10^{-4}$	m^2	Strand total cross section
COSTETA	0.9699	-	Cosine of the angle between the cable and the x axis
STAB_NON_STAB	2.0846	-	Stabilizer non stabilizer ratio
NUM_MATERIAL_TYPES	2	-	Number of materials making up the strand
ISTABILIZER	1	-	Flag for the stabilizer material: copper (Cu)
RRR	197.71	-	Copper residual resistance ratio
ISUPERCONDUCTOR	2	-	Flag for the superconductor material: Nb_3Sn
c0	$1.21508\ 10^{11}$	$\frac{\mathrm{AT}}{\mathrm{m}^2}$	Property of superconductor scaling
Bc20m	32.35	T	Property of superconductor scaling
Tc0m	16.22	K	Property of superconductor scaling

Table D.2-4 Input data to be set in the Z_JACKET sheet of the workbook conductor_input.xlsx. Cross sections computed from data in [87].

Variable	Value	Unit SI	Meaning
CROSSECTION_JK	$3.0699\ 10^{-4}$	m^2	Pure jacket cross section
CROSSECTION_IN	$2.7966\ 10^{-4}$	m^2	Insulation cross section
NUM_MATERIAL_TYPES	2	-	Number of materials making up the jacket
IMATERIAL_JK	1	-	Flag for pure jacket material: stainless steel
IMATERIAL_IN	1	-	Flag for insulation material: glass-epoxy
COSTETA	$1.00E+00$	-	Cosine of the angle between the cable and the x axis

Table D.2-5 Input data to be set in the STR_MIX sheet of the workbook conductor_operation.xlsx.

Variable	Value	Unit	Meaning
IEPS	0	-	Flag to define the strain along the strand: no strain
EPS	0	-	Value of the strain along the strand
IQFUN	1	-	Flag for the heat source shape: square wave in space and time
INTIAL	0	-	Flag to define how to initialize the STR_MIX temperature spatial distribution: from channel initial temperature

Table D.2-6 Input data to be set in the Z_JACKET sheet of the workbook conductor_operation.xlsx.

Variable	Value	Unit	Meaning
IQFUN	0	-	Flag for the heat source shape: no heat source
INTIAL	0	-	Flag to define how to initialize the Z_JACKET temperature spatial distribution: from channel initial temperature

Table D.2-7 Contact perimeter flags, dimensionless. If 1 cable components are in contact, if 0 there is no contact between the components. Values to be inserted in sheet contact_perimeter_flag of workbook conductor_coupling.xlsx.

	CHAN_1	CHAN_2	STR_MIX_1	Z_JACKET_1
CHAN_1	0	1	0	0
CHAN_2	0	0	1	1
STR_MIX_1	0	0	0	1
Z_JACKET_1	0	0	0	0

Table D.2-8 Contact perimeter in m*; values to be inserted in sheet contact_perimeter of workbook conductor_coupling.xlsx. Values computed from data in [87].*

	CHAN_1	CHAN_2	STR_MIX_1	Z_JACKET_1
CHAN_1	0	$0.009\,\pi$	0	0
CHAN_2	0	0	3.7275	0.094356
STR_MIX_1	0	0	0	0.031452
Z_JACKET_1	0	0	0	0

Table D.2-9 Flags to select the correlation to compute the heat transfer coefficients, dimensionless. Flag on the channel main diagonal: correlation to evaluate the steady heat transfer coefficients for channel objects (not used in this case); other values: flag to select the correlation to evaluate interface heat transfer coefficients, if -1 they are user defined. Values to be inserted in sheet HTC_choice of workbook conductor_coupling.xlsx.

	CHAN_1	CHAN_2	STR_MIX_1	Z_JACKET_1
CHAN_1	0	−1	0	0
CHAN_2	0	0	−1	−1
STR_MIX_1	0	0	0	−1
Z_JACKET_1	0	0	0	0

Table D.2-10 Values of the heat transfer coefficient if user defined, in $W/m^2/K$. Values to be inserted in sheet contact_HTC of workbook conductor_coupling.xlsx.

	CHAN_1	CHAN_2	STR_MIX_1	Z_JACKET_1
CHAN_1	0	1000	0	0
CHAN_2	0	0	1000	1000
STR_MIX_1	0	0	0	500
Z_JACKET_1	0	0	0	0

Table D.2-11 Values for the heat transfer coefficient dimensionless multipliers, when needed; to be inserted in sheet HTC_multiplier of workbook conductor_coupling.xlsx.

	CHAN_1	CHAN_2	STR_MIX_1	Z_JACKET_1
CHAN_1	0	1	0	0
CHAN_2	0	0	1	1
STR_MIX_1	0	0	0	1
Z_JACKET_1	0	0	0	0

Table D.2-12 Fraction of the contact perimeter between fluid components which is actually open, dimensionless. Values to be inserted in sheet open_perimeter_fract of workbook conductor_coupling.xlsx. Values from data in [87].

	CHAN_1	CHAN_2	STR_MIX_1	Z_JACKET_1
CHAN_1	0	0.293	0	0
CHAN_2	0	0	0	0
STR_MIX_1	0	0	0	0
Z_JACKET_1	0	0	0	0

Table D.2-13 Thickness of the interface between fluid elements, in m. Values to be inserted in sheet interf_thickness of workbook conductor_coupling.xlsx from [87].

	CHAN_1	CHAN_2	STR_MIX_1	Z_JACKET_1
CHAN_1	0	0.001	0	0
CHAN_2	0	0	0	0
STR_MIX_1	0	0	0	0
Z_JACKET_1	0	0	0	0

Table D.2-14 Multiplier for the transport properties in the open fraction of the perimeter between fluid components, dimensionless. Values to be inserted in sheet trans_transp_multiplier of workbook conductor_coupling.xlsx.

	CHAN_1	CHAN_2	STR_MIX_1	Z_JACKET_1
CHAN_1	0	1	0	0
CHAN_2	0	0	0	0
STR_MIX_1	0	0	0	0
Z_JACKET_1	0	0	0	0

APPENDIX E

E FURTHER DETAILS

This appendix is devoted to describing aspects that are too detailed to be mentioned within the main body of the thesis and are not essential to the overall understanding of the work, although it does provide additional information about them. Section E.1 is a nod to the compilation of the input files, while section E.2 describes the contents of the input file.

E.1 HOW TO COMPILE INPUT FILES

All the main input files apart from conductor_coupling.xlsx, are organized by columns; since they share the same structure, the following illustrates how to compile file conductor_definition.xlsx.

The workbook is composed of a single sheet called CONDUCTOR, that is also the root of the object identifier (ID) reported in cell A1. The first four columns of this sheet define the variable name, its unit including the specification “flag” for flags variables, the variable type (float, integer or string) and the variable description respectively; starting from the fifth column user can define cables. To do this, it is sufficient to write an integer number in the first row of the fifth column that must be larger or equal than one; automatically the sheet will compile the second and third rows of the same columns which hold respectively a counter and the cable identifier. Object ID is constructed starting from the root or object name (CONDUCTOR in this example) and the integer number provided by the user, joined by an underscore. The total value of the cable counter is reported in cell B1 and it is updated when a new column is initialized.

Bear in mind that basic components objects with the same IDs are not allowed within the same cable as well as different cables with the same IDs, so for each new column a not already used integer number must be provided.

The conductor_coupling.xlsx workbook is composed by a set of eight sheets, each of them characterized by a square matrix. The rows and columns of this matrices corresponds to the IDs of the cable basic components objects previously defined; since they are symmetric, only the upper triangular region must be filled by the user. In these sheets the conditional formatting is introduced to help user with the compilation of matrices.

E.2 INTERFACES AND COUPLING BETWEEN COMPONENTS

In the eight sheets of conductor_coupling.xlsx primary input file, user sets the interfaces between components, the heat transfer coefficients and the multiplier to determine the transported mass, momentum and energy coefficients through fluid components open contact perimeter.

In sheet contact_perimeter_flag, matrix element meaning is the wetted or contact perimeter flags: if 1 there is at least thermal contact between fluid components, fluid and solid components and between solid components, if 0 there is no interfaces. The contact perimeter

length must be provided in the second sheet (contact_perimeter) and not 0 values must be inserted in the cells with a value of the contact perimeter flag equal to 1; this rule applies to all the sheets. Conditional formatting is thought and introduced to guide user in matrix filling phase of the input file compilation. To fully describe the interfaces between fluid components, user should compile the open_perimeter_fraction matrix whose elements represent the open fraction of the contact perimeters. It is a number in the range $[0,1)$ and if larger than zero it means that the channels are in hydraulic parallel.

Heat transfer coefficient (HTC) are evaluated starting from the flag values of the matrix in sheet HTC_choice. On the main diagonal the elements corresponding to channels are used to choose the correlation with which the channels steady heat transfer coefficient is to be computed; flag values on the upper triangular matrix allows to select the value of the interface heat transfer coefficient: if positive it is evaluated with a correlation, the final value is obtained correcting the computed heat transfer coefficient by the multiplier provided by the user in sheet HTC_multiplier; if negative it is directly imposed by user writing its value in sheet contact_HTC. The thickness of the interface between channels is exploited to evaluate the wall thermal resistance that appears in the formula to evaluate the heat transfer coefficient between channels. Heat transfer coefficients and channels friction factors are evaluated by method Get_transp_coeff of class Conductor.

The last sheet trans_transp_multiplier holds the multiplier for the transport properties through the open fraction of the channels contact perimeter, but it is not used at the time being in SC2.

BIBLIOGRAPHY

[1] L. Rossi, “Superconductivity: Its role, its success and its setbacks in the LARGE HADRON COLlider of CERN,” *Supercond. Sci. Technol.*, 2010, doi: 10.1088/0953-2048/23/3/034001.

[2] P. Bruzzone, “Superconductivity and fusion energy - The inseparable companions,” *Supercond. Sci. Technol.*, 2015, doi: 10.1088/0953-2048/28/2/024001.

[3] H. Jones, “Superconductors in the transmission of electricity and networks,” *Energy Policy*, 2008, doi: 10.1016/j.enpol.2008.09.063.

[4] A. Horvath and E. Rachlew, “Nuclear power in the 21st century: Challenges and possibilities,” *Ambio*, 2016, doi: 10.1007/s13280-015-0732-y.

[5] J. Adelerhof, M. B. Thoopal, D. Lee, and C. Hardy, “Clean and Sustainable Fusion Energy for the Future,” *PAM Rev. Energy Sci. Technol.*, 2015, doi: 10.5130/pamr.v1i0.1384.

[6] J. L. Johnson, “Stellarator and Heliotron Devices,” *Nucl. Fusion*, 1999, doi: 10.1088/0029-5515/39/2/701.

[7] M. Wanner *et al.*, “Design and construction of WENDELSTEIN 7-X,” *Fusion Eng. Des.*, 2001, doi: 10.1016/S0920-3796(01)00239-3.

[8] R. C. Wolf *et al.*, “Wendelstein 7-X Program - Demonstration of a Stellarator Option for Fusion Energy,” *IEEE Trans. Plasma Sci.*, 2016, doi: 10.1109/TPS.2016.2564919.

[9] V. Erckmann *et al.*, “W7-X project: scientific basis and technical realization,” 1998, doi: 10.1109/fusion.1997.685662.

[10] M. Fujiwara *et al.*, “Overview of LHD experiments,” *Nuclear Fusion*. 2001, doi: 10.1088/0029-5515/41/10/305.

[11] A. Iiyoshi *et al.*, “Overview of the large helical device project,” *Nucl. Fusion*, 1999, doi: 10.1088/0029-5515/39/9y/313.

[12] T. Rummel *et al.*, “The superconducting magnet system of the stellarator wendelstein 7-X,” 2012, doi: 10.1109/TPS.2012.2184774.

[13] F. Warmer, C. D. Beidler, A. Dinklage, and R. Wolf, “From W7-X to a HELIAS fusion power plant: Motivation and options for an intermediate-step burning-plasma stellarator,” *Plasma Phys. Control. Fusion*, 2016, doi: 10.1088/0741-3335/58/7/074006.

[14] V. Queral, F. A. Volpe, D. Spong, S. Cabrera, and F. Tabarés, “Initial Exploration of High-Field Pulsed Stellarator Approach to Ignition Experiments,” *J. Fusion Energy*, 2018, doi: 10.1007/s10894-018-0199-5.

[15] N. Holtkamp, “An overview of the ITER project,” *Fusion Eng. Des.*, 2007, doi: 10.1016/j.fusengdes.2007.03.029.

[16] K. Fiore, “Nuclear energy and sustainability: Understanding ITER,” *Energy Policy*, 2006, doi: 10.1016/j.enpol.2005.07.008.

[17] C. Llewellyn Smith and D. Ward, “Fusion,” *Energy Policy*, 2008, doi: 10.1016/j.enpol.2008.09.071.

[18] Y. Wan *et al.*, “Overview of the present progress and activities on the CFETR,” *Nuclear Fusion*. 2017, doi: 10.1088/1741-4326/aa686a.

[19] L. Muzzi, G. De Marzi, A. Di Zenobio, and A. Della Corte, “Cable-in-conduit conductors: Lessons from the recent past for future developments with low and high temperature superconductors,” *Superconductor Science and Technology*. 2015, doi: 10.1088/0953-2048/28/5/053001.

[20] T. Donné, “European Research Roadmap to the Realisation of Fusion Energy,” *EUROfusion*, 2018.

[21] K. Sedlak *et al.*, “Advance in the conceptual design of the European DEMO magnet system,” *Supercond. Sci. Technol.*, 2020, doi: 10.1088/1361-6668/ab75a9.

[22] K. Tobita *et al.*, “Overview of the DEMO conceptual design activity in Japan,” *Fusion Eng. Des.*, 2018, doi: 10.1016/j.fusengdes.2018.04.059.

[23] K. Kim *et al.*, “Design concept of K-DEMO for near-term implementation,” *Nucl. Fusion*, 2015, doi: 10.1088/0029-5515/55/5/053027.

[24] H. W. Kim *et al.*, “Design updates of magnet system for Korean fusion demonstration reactor, K-DEMO,” *Fusion Eng. Des.*, 2019, doi: 10.1016/j.fusengdes.2019.02.012.

[25] B. N. Sorbom *et al.*, “ARC: A compact, high-field, fusion nuclear science facility and demonstration power plant with demountable magnets,” *Fusion Eng. Des.*, 2015, doi: 10.1016/j.fusengdes.2015.07.008.

[26] D. G. Whyte, J. Minervini, B. LaBombard, E. Marmar, L. Bromberg, and M. Greenwald, “Smaller & Sooner: Exploiting High Magnetic Fields from New Superconductors for a More Attractive Fusion Energy Development Path,” *J. Fusion Energy*, 2016, doi: 10.1007/s10894-015-0050-1.

[27] D. Whyte, “Small, modular and economically attractive fusion enabled by high temperature superconductors,” 2019, doi: 10.1098/rsta.2018.0354.

[28] T. Tsunematsu, “Broader Approach to fusion energy,” *Fusion Eng. Des.*, 2009, doi: 10.1016/j.fusengdes.2009.02.029.

[29] P. Barabaschi, Y. Kamada, and H. Shirai, “Progress of the JT-60SA project,” *Nucl. Fusion*, vol. 59, no. 11, 2019, doi: 10.1088/1741-4326/ab03f6.

[30] A. Pizzuto *et al.*, “JT-60SA toroidal field magnet system,” 2008, doi: 10.1109/TASC.2008.920827.

[31] A. Zappatore *et al.*, “Modeling Quench Propagation in the ENEA HTS Cable-In-Conduit Conductor,” *IEEE Trans. Appl. Supercond.*, 2020, doi: 10.1109/TASC.2020.3001035.

[32] H. Jones, “Superconductors in the transmission of electricity and networks,” *Energy Policy*, vol. 36, no. 12, pp. 4342–4345, Dec. 2008, doi: 10.1016/j.enpol.2008.09.063.

[33] A. P. Malozemoff, J. Yuan, and C. M. Rey, "5 - High-temperature superconducting (HTS) AC cables for power grid applications," in *Superconductors in the Power Grid*, C. Rey, Ed. Woodhead Publishing, 2015, pp. 133–188.

[34] D. Uglietti, "A review of commercial high temperature superconducting materials for large magnets: from wires and tapes to cables and conductors," *Supercond. Sci. Technol.*, vol. 32, no. 5, p. 053001, Apr. 2019, doi: 10.1088/1361-6668/ab06a2.

[35] G. Venkataramanan and B. K. Johnson, "A superconducting DC transmission system based on VSC transmission technologies," *IEEE Trans. Appl. Supercond.*, vol. 13, no. 2, pp. 1922–1925, Jun. 2003, doi: 10.1109/TASC.2003.812947.

[36] D. I. Doukas, "Superconducting Transmission Systems: Review, Classification, and Technology Readiness Assessment," *IEEE Trans. Appl. Supercond.*, vol. 29, no. 5, pp. 1–5, Aug. 2019, doi: 10.1109/TASC.2019.2895395.

[37] K. Sato, "Present Status and Future Perspective of High-Temperature Superconductors," p. 13.

[38] Y. Xie *et al.*, "Second-Generation HTS Conductor Design and Engineering for Electrical Power Applications," *IEEE Trans. Appl. Supercond.*, vol. 19, no. 3, pp. 3009–3013, Jun. 2009, doi: 10.1109/TASC.2009.2018499.

[39] V. E. Sytnikov, V. S. Vysotsky, I. P. Radchenko, and N. V. Polyakova, "1G versus 2G-comparison from the practical standpoint for HTS power cables use," *J. Phys. Conf. Ser.*, vol. 97, p. 012058, Feb. 2008, doi: 10.1088/1742-6596/97/1/012058.

[40] J. P. Stovall *et al.*, "Installation and operation of the Southwire 30-meter high-temperature superconducting power cable," *IEEE Trans. Appl. Supercond.*, vol. 11, no. 1, pp. 2467–2472, Mar. 2001, doi: 10.1109/77.920363.

[41] J. F. Maguire *et al.*, "Development and Demonstration of a HTS Power Cable to Operate in the Long Island Power Authority Transmission Grid," *IEEE Trans. Appl. Supercond.*, vol. 17, no. 2, pp. 2034–2037, Jun. 2007, doi: 10.1109/TASC.2007.898359.

[42] D. Zhang *et al.*, "Testing Results for the Cable Core of a 360 m/10 kA HTS DC Power Cable Used in the Electrolytic Aluminum Industry," *IEEE Trans. Appl. Supercond.*, vol. 23, no. 3, pp. 5400504–5400504, Jun. 2013, doi: 10.1109/TASC.2012.2236812.

[43] D.-H. Yoon, "A Feasibility Study on HTS Cable for the Grid Integration of Renewable Energy," *Phys. Procedia*, vol. 45, pp. 281–284, Jan. 2013, doi: 10.1016/j.phpro.2013.05.022.

[44] L. Xiao *et al.*, "Development of a 10 kA HTS DC Power Cable," *IEEE Trans. Appl. Supercond.*, vol. 22, no. 3, pp. 5800404–5800404, Jun. 2012, doi: 10.1109/TASC.2011.2176090.

[45] H. J. Kim and K. Hur, "Expanded Adoption of HTS Cables in a Metropolitan Area and its Potential Impact on the Neighboring Electric Power Grid," *IEEE Trans. Appl. Supercond.*, vol. 22, no. 3, pp. 5800704–5800704, Jun. 2012, doi: 10.1109/TASC.2011.2176700.

[46] S. K. Shrivastava, "Application of Superconductivity in Electric Power and Transportation System," vol. 11, no. 4, p. 9.

[47] B. Gamble, G. Snitchler, and T. MacDonald, “Full Power Test of a 36.5 MW HTS Propulsion Motor,” *IEEE Trans. Appl. Supercond.*, vol. 21, no. 3, pp. 1083–1088, Jun. 2011, doi: 10.1109/TASC.2010.2093854.

[48] L. Savoldi Richard, F. Casella, B. Fiori, and R. Zanino, “The 4C code for the cryogenic circuit conductor and coil modeling in ITER,” *Cryogenics*, 2010, doi: 10.1016/j.cryogenics.2009.07.008.

[49] M. Bagnasco, D. Bessette, L. Bottura, C. Marinucci, and C. Rosso, “Progress in the integrated simulation of thermal-hydraulic operation of the ITER magnet system,” 2010, doi: 10.1109/TASC.2010.2043836.

[50] D. Bessette, N. Shatil, and E. Zapretilina, “Simulations of the ITER toroidal field coil operation with the VINCENTA code,” 2006, doi: 10.1109/TASC.2006.873258.

[51] R. Zanino and L. S. Richard, “Multiscale approach and role of validation in the thermal-hydraulic modeling of the ITER superconducting magnets,” *IEEE Trans. Appl. Supercond.*, 2013, doi: 10.1109/TASC.2012.2236134.

[52] L. Savoldi and R. Zanino, “M & M: Multi-conductor Mithrandir code for the simulation of thermal-hydraulic transients in superconducting magnets,” *Cryogenics*, 2000, doi: 10.1016/S0011-2275(00)00027-8.

[53] R. Zanino, P. Santagati, L. Savoldi, and C. Marinucci, “Joint+conductor thermal-hydraulic experiment and analysis on the Full Size Joint Sample using MITHRANDIR 2.1,” *IEEE Trans. Appl. Supercond.*, 2000, doi: 10.1109/77.828427.

[54] R. Zanino, R. Bonifetto, C. Hoa, and L. S. Richard, “Verification of the predictive capabilities of the 4C code cryogenic circuit model,” 2014, doi: 10.1063/1.4860896.

[55] R. Zanino, R. Bonifetto, F. Casella, and L. Savoldi Richard, “Validation of the 4C code against data from the HELIOS loop at CEA Grenoble,” 2013, doi: 10.1016/j.cryogenics.2012.04.010.

[56] A. Zappatore, R. Heller, L. Savoldi, M. J. Wolf, and R. Zanino, “A new model for the analysis of quench in HTS cable-in-conduit conductors based on the twisted-stacked-tape cable concept for fusion applications,” *Supercond. Sci. Technol.*, 2020, doi: 10.1088/1361-6668/ab895b.

[57] M. Lewandowska, A. Dembkowska, R. Heller, and M. Wolf, “Thermal-hydraulic analysis of an HTS DEMO TF coil,” *Cryogenics*, 2018, doi: 10.1016/j.cryogenics.2018.10.014.

[58] A. Zappatore, W. H. Fietz, R. Heller, L. Savoldi, M. J. Wolf, and R. Zanino, “A critical assessment of thermal–hydraulic modeling of HTS twisted-stacked-tape cable conductors for fusion applications,” *Supercond. Sci. Technol.*, Aug. 2019, doi: 10.1088/1361-6668/ab20a9.

[59] V. Amoskov *et al.*, “Validation of VINCENTA modelling based on an experiment with the central solenoid model coil of the international thermonuclear experimental reactor,” *Plasma Devices Oper.*, 2006, doi: 10.1080/10519990500518001.

[60] S. Nicollet, D. Bessette, D. Ciazynski, J. L. Duchateau, and B. Lacroix, "Cross checking of Gandalf and Vincenta on the CS behaviour during ITER reference scenario," 2010, doi: 10.1063/1.3422315.

[61] V. D. Arp, "Stability and thermal quenches in force-cooled superconducting cables. Final report," National Bureau of Standards, Washington, DC (USA), PB-85-141018, May 1979. Accessed: Mar. 12, 2021. [Online]. Available: https://www.osti.gov/biblio/6032647.

[62] L. Bottura, "Thermal, Hydraulic, and Electromagnetic Modeling of Superconducting Magnet Systems," *IEEE Trans. Appl. Supercond.*, vol. 26, no. 3, pp. 1–7, Apr. 2016, doi: 10.1109/TASC.2016.2544253.

[63] J. A. Souza, J. C. Ordonez, R. Hovsapian, and J. V. C. Vargas, "Thermal Modeling of Helium Cooled High-Temperature Superconducting DC Transmission Cable," *IEEE Trans. Appl. Supercond.*, vol. 21, no. 3, pp. 947–952, Jun. 2011, doi: 10.1109/TASC.2010.2099196.

[64] C. L. Buiar and J. V. C. Vargas, "Thermodynamic Analisis Applied to Superconducting DC Cable Model," *14th Braz. Congr. Therm. Sci. Eng.*, p. 8, 22-09 2012.

[65] C. L. Buiar, J. V. C. Vargas, and J. C. Ordonez, "Dimensionless High Temperature Superconducting (HTS) DC Cable Model," p. 7, 2013.

[66] D. I. Doukas, A. I. Chrysochos, T. A. Papadopoulos, D. P. Labridis, L. Harnefors, and G. Velotto, "Volume Element Method for Thermal Analysis of Superconducting DC Transmission Cable," *IEEE Trans. Appl. Supercond.*, vol. 27, no. 4, pp. 1–8, Jun. 2017, doi: 10.1109/TASC.2017.2656785.

[67] C. E. Bruzek *et al.*, "Cable Conductor Design for the High-Power MgB2 DC Superconducting Cable Project of BEST PATHS," *IEEE Trans. Appl. Supercond.*, vol. 27, no. 4, pp. 1–5, Jun. 2017, doi: 10.1109/TASC.2016.2641338.

[68] "MATLAB - Il linguaggio del calcolo tecnico." https://it.mathworks.com/products/matlab.html (accessed Mar. 14, 2021).

[69] D. I. Doukas, A. I. Chrysochos, T. A. Papadopoulos, D. P. Labridis, L. Harnefors, and G. Velotto, "Coupled Electro-Thermal Transient Analysis of Superconducting DC Transmission Systems Using FDTD and VEM Modeling," *IEEE Trans. Appl. Supercond.*, vol. 27, no. 8, pp. 1–8, Dec. 2017, doi: 10.1109/TASC.2017.2749500.

[70] T. Masuda *et al.*, "Test Results of a 30 m HTS Cable for Yokohama Project," *IEEE Trans. Appl. Supercond.*, vol. 21, no. 3, pp. 1030–1033, Jun. 2011, doi: 10.1109/TASC.2010.2093109.

[71] Y. Sato, K. Agatsuma, X. Wang, and A. Ishiyama, "Temperature and Pressure Simulation of a High-Temperature Superconducting Cable Cooled by Subcooled $\mboxLN_2$ With Fault Current," *IEEE Trans. Appl. Supercond.*, vol. 25, no. 3, pp. 1–5, Jun. 2015, doi: 10.1109/TASC.2014.2387119.

[72] "Horizon Technologies...GASPAK Software." http://www.htess.com/gaspak.htm (accessed Mar. 14, 2021).

[73] E. Shabagin, C. Heidt, S. Strauß, and S. Grohmann, “Modelling of 3D temperature profiles and pressure drop in concentric three-phase HTS power cables,” *Cryogenics*, vol. 81, pp. 24–32, Jan. 2017, doi: 10.1016/j.cryogenics.2016.11.004.

[74] M. Stemmle, F. Merschel, M. Noe, and A. Hobl, “Ampacity project - worldwide first superconducting cable and fault current limiter installation in a German city center,” *22nd Int. Conf. Electr. Distrib.*, pp. 0742–0742, Jan. 2013, doi: 10.1049/cp.2013.0905.

[75] W. T. B. de Sousa, D. Kottonau, J. Bock, and M. Noe, “Investigation of a Concentric Three-Phase HTS Cable Connected to an SFCL Device,” *IEEE Trans. Appl. Supercond.*, vol. 28, no. 4, pp. 1–5, Jun. 2018, doi: 10.1109/TASC.2018.2794586.

[76] D. Kottonau, W. T. B. de Sousa, J. Bock, and M. Noe, “Design Comparisons of Concentric Three-Phase HTS Cables,” *IEEE Trans. Appl. Supercond.*, vol. 29, no. 6, pp. 1–8, Sep. 2019, doi: 10.1109/TASC.2019.2897893.

[77] S.-J. Lee, H.-J. Sung, M. Park, D. Won, J. Yoo, and H. S. Yang, “Analysis of the Temperature Characteristics of Three-Phase Coaxial Superconducting Power Cable according to a Liquid Nitrogen Circulation Method for Real-Grid Application in Korea,” *Energies*, vol. 12, no. 9, p. 1740, May 2019, doi: 10.3390/en12091740.

[78] “SINDA/FLUINT.” https://www.crtech.com/products/sindafluint (accessed Mar. 14, 2021).

[79] C. Lee, D. Kim, S. Kim, D. Y. Won, and H. S. Yang, “Thermo-Hydraulic Analysis on Long Three-Phase Coaxial HTS Power Cable of Several Kilometers,” *IEEE Trans. Appl. Supercond.*, vol. 29, no. 5, pp. 1–5, Aug. 2019, doi: 10.1109/TASC.2019.2900710.

[80] “Best Paths - Project.” http://www.bestpaths-project.eu/en/project (accessed Mar. 14, 2021).

[81] G. Angeli, M. Bocchi, M. Ascade, V. Rossi, A. Valzasina, and L. Martini, “Development of Superconducting Devices for Power Grids in Italy: Update About the SFCL Project and Launching of the Research Activity on HTS Cables,” *IEEE Trans. Appl. Supercond.*, vol. 27, no. 4, pp. 1–6, Jun. 2017, doi: 10.1109/TASC.2016.2639022.

[82] F. Grilli, “Numerical Modeling of HTS Applications,” *IEEE Trans. Appl. Supercond.*, vol. 26, no. 3, pp. 1–8, Apr. 2016, doi: 10.1109/TASC.2016.2520083.

[83] T.-T. Nguyen *et al.*, “A Simplified Model of Coaxial, Multilayer High-Temperature Superconducting Power Cables with Cu Formers for Transient Studies,” *Energies*, vol. 12, no. 8, p. 1514, Apr. 2019, doi: 10.3390/en12081514.

[84] “Home | PSCAD.” https://www.pscad.com/ (accessed Mar. 14, 2021).

[85] S. S. Fetisov, V. V. Zubko, A. A. Nosov, S. Y. Zanegin, and V. S. Vysotsky, “Review of the design, production and tests of compact AC HTS power cables,” *Prog. Supercond. Cryog.*, vol. 22, no. 4, pp. 31–39, 2020, doi: 10.9714/psac.2020.22.4.031.

[86] K. Glinos, “EU Open Science.” Dec. 13, 2019, Accessed: Mar. 15, 2021. [Online]. Available: https://ec.europa.eu/info/sites/info/files/research_and_innovation/knowledge_publications_tools_and_data/documents/ec_rtd_factsheet-open-science_2019.pdf.

[87] R. Zanino, D. Bessette, and L. S. Richard, "Quench analysis of an ITER TF coil," *Fusion Eng. Des.*, vol. 85, no. 5, pp. 752–760, Aug. 2010, doi: 10.1016/j.fusengdes.2010.04.056.

[88] R. Bonifetto, F. Buonora, L. S. Richard, and R. Zanino, "4C Code Simulation and Benchmark of ITER TF Magnet Cool-Down From 300 K to 80 K," *IEEE Trans. Appl. Supercond.*, vol. 22, no. 3, pp. 4902604–4902604, Jun. 2012, doi: 10.1109/TASC.2011.2177233.

[89] L. S. Richard, R. Bonifetto, A. Foussat, N. Mitchell, K. Seo, and R. Zanino, "Mitigation of the Temperature Margin Reduction due to the Nuclear Radiation on the ITER TF Coils," *IEEE Trans. Appl. Supercond.*, vol. 23, no. 3, pp. 4201305–4201305, Jun. 2013, doi: 10.1109/TASC.2013.2238977.

[90] L. S. Richard *et al.*, "Analysis of the Effects of the Nuclear Heat Load on the ITER TF Magnets Temperature Margin," *IEEE Trans. Appl. Supercond.*, vol. 24, no. 3, pp. 1–4, Jun. 2014, doi: 10.1109/TASC.2013.2280720.

[91] R. Zanino, S. Giors, and R. Mondino, "CFD Modeling of ITER Cable-in-Conduit Superconductors. Part I: Friction in the Central Channel," *AIP Conf. Proc.*, vol. 823, no. 1, pp. 1009–1016, Apr. 2006, doi: 10.1063/1.2202514.

[92] R. Zanino, S. Giors, and R. Mondino, "CFD modeling of ITER cable-in-conduit superconductors: Part II. Effects of spiral geometry on the central channel pressure drop," *Fusion Eng. Des.*, vol. 81, no. 23, pp. 2605–2610, Nov. 2006, doi: 10.1016/j.fusengdes.2006.07.006.

[93] R. Zanino and S. Giors, "Cfd modeling of iter cable-in-conduit superconductors. part v: combined momentum and heat transfer in rib roughened pipes," *AIP Conf. Proc.*, vol. 985, no. 1, pp. 1261–1268, Mar. 2008, doi: 10.1063/1.2908481.

[94] R. Zanino, S. De Palo, and L. Bottura, "A two-fluid code for the thermohydraulic transient analysis of CICC superconducting magnets," *J. Fusion Energy*, vol. 14, no. 1, pp. 25–40, Mar. 1995, doi: 10.1007/BF02214031.

[95] R. Zanino and C. Marinucci, "Heat slug propagation in QUELL. Part I: Experimental setup and 1-fluid GANDALF analysis," *Cryogenics*, vol. 39, no. 7, pp. 585–593, Jul. 1999, doi: 10.1016/S0011-2275(99)00074-0.

[96] R. Zanino and C. Marinucci, "Heat slug propagation in QUELL. Part II: 2-fluid MITHRANDIR analysis," *Cryogenics*, vol. 39, no. 7, pp. 595–608, Jul. 1999, doi: 10.1016/S0011-2275(99)00075-2.

[97] C. Johnson, *Numerical solution of partial differential equations by the finite element method*, Dover ed. Mineola, N.Y: Dover Publications, 2009.

[98] J. Cooper, *Introduction to Partial Differential Equations with MATLAB*. Boston, MA: Birkhäuser Boston, 1998.

[99] A. Quarteroni, F. Saleri, and P. Gervasio, *Scientific Computing with MATLAB and Octave*, vol. 2. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014.

[100] "The Python Standard Library — Python 3.9.2 documentation." https://docs.python.org/3/library/ (accessed Mar. 18, 2021).

[101] “NumPy.” https://numpy.org/ (accessed Mar. 18, 2021).

[102] “SciPy — SciPy v1.6.1 Reference Guide.” https://docs.scipy.org/doc/scipy/reference/ (accessed Mar. 18, 2021).

[103] “pandas - Python Data Analysis Library.” https://pandas.pydata.org/ (accessed Mar. 18, 2021).

[104] “openpyxl - A Python library to read/write Excel 2010 xlsx/xlsm files — openpyxl 3.0.7 documentation.” https://openpyxl.readthedocs.io/en/stable/ (accessed Mar. 18, 2021).

[105] “Matplotlib: Python plotting — Matplotlib 3.3.4 documentation.” https://matplotlib.org/ (accessed Mar. 18, 2021).

[106] “PEP 8 -- Style Guide for Python Code,” *Python.org*. https://www.python.org/dev/peps/pep-0008/ (accessed Mar. 17, 2021).

[107] H. Wickham, “Tidy Data,” *J. Stat. Softw.*, vol. 059, no. i10, 2014, Accessed: Mar. 17, 2021. [Online]. Available: https://ideas.repec.org/a/jss/jstsof/v059i10.html.