

Horizon 2020



Reduced Order Modelling, Simulation and Optimization of Coupled systems

Software-based representation of selected benchmark hierarchies equipped with publically available data

Deliverable number: D5.2

Version 2.0



Funded by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie Grant Agreement No. 765374

Project Acronym: ROMSOC
Project Full Title: Reduced Order Modelling, Simulation and Optimization of Coupled systems
Call: H2020-MSCA-ITN-2017
Topic: Innovative Training Network
Type of Action: European Industrial Doctorates
Grant Number: 765374

Editors:	Andrés Prieto, Peregrina Quintela, ITMATI
Deliverable nature:	Report (R)
Dissemination level:	Public (PU)
Contractual Delivery Date:	30/08/2019
Actual Delivery Date	04/10/2019
Number of pages:	161
Keywords:	Benchmarks, Model hierarchies
Authors:	<p>Marcus W.F.M. Bannenberg, BUW Patricia Barral, ITMATI-USC Jean-David Benamou, INRIA Andreas Baermann, FAU Federico Bianco, Danieli Andres Binder, MathConsult Riccardo Conte, Danieli Guillaume Chazareix, INRIA Sören Dittmer, U-HB Daniel Fernández, Microflown Technologies Michele Girfoglio, SISSA M. Günther, BUW José Carlos Gutiérrez Pérez, U-HB Lena Hauberg-Lotte, U-HB Michael Hintermüller, WIAS Berlin Wilbert Ijzerman, Signify Onkar Jadhav, MathConsult Tobias Kluth, U-HB Karl Knall, MathTec Alejandro Lengomin, AMIII Peter Maass, U-HB Gianfranco Marconi, Danieli Alexander Martin, FAU Marco Martinolli, MOX, PoliMi Volker Mehrmann, TU Berlin Pier Paolo Monticone, CorWave SA Umberto Emil Morelli, ITMATI Ashwin Nayak, ITMATI Andreas Obereder, MathConsult Daniel Otero Bager, U-HB Luc Polverelli, CorWave Inc. Andrés Prieto, ITMATI-UDC Peregrina Quintela, ITMATI-USC Ronny Ramlau, Industrial Mathematics Institute JKU Conte Riccardo, Danieli Gianluigi Rozza, SISSA Giorgi Rukhaia, INRIA Nirav Shah, SISSA Jonasz Staszek, Giovanni Stabile, SISSA Bernadett Stadler, Industrial Mathematics Institute JKU Jonasz Staszek, FAU Christian Vergara, MOX, PoliMi</p>
Peer review:	<p>Andrés Prieto, ITMATI-UDC Lena Scholz, TU Berlin</p>

Abstract

Based on the multitude of industrial applications, benchmarks for model hierarchies will be created that will form a basis for the interdisciplinary research and for the training programme. These will be equipped with publically available data and will be used for training in modelling, model testing, reduced order modelling, error estimation, efficiency optimization in algorithmic approaches, and testing of the generated MSO/MOR software. The present document includes a detailed description of the computer implementation of these benchmarks involving not only the required publically available data but also the used software packages, libraries and any other relevant information, which guarantee a fully reproducibility of the reported numerical results. The present document has been structured in three main parts to distinguish those contributions which are focused on coupling methods, model order reduction methods, and optimization methods.

Disclaimer & acknowledgment

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie Grant Agreement No. 765374.

This document reflects the views of the author(s) and does not necessarily reflect the views or policy of the European Commission. The REA cannot be held responsible for any use that may be made of the information this document contains.

Reproduction and translation for non-commercial purposes are authorised, provided the source is acknowledged and the publisher is given prior notice and sent a copy.

Contents

I. Coupling methods	1
1. Implementing acoustic scattering simulations for external geometries within a porous enclosure	2
<i>Ashwin Nayak, Andrés Prieto, Daniel Fernández</i>	
1.1. Introduction	2
1.2. Mathematical Formulation	2
1.2.1. Model Hierarchy	3
1.2.1.1. Fluid region	3
1.2.1.2. Porous region	3
1.2.1.3. Perfectly matched layer	4
1.2.2. Coupling	5
1.3. Implementation	6
1.3.1. Geometry and Meshing	7
1.3.2. Solver	8
1.3.3. Post-processing and Visualization	9
1.4. Computer Requirements	9
1.5. Case study : Acoustic transmission of a vibrating sphere in a porous enclosure	9
1.5.1. Description of test case sphere	9
1.5.2. Exact solution	10
1.5.3. Geometry and Mesh	11
1.5.4. Solver	13
1.5.5. Visualization	16
1.6. Conclusion	17
Bibliography	18
2. Multirate time integration and model order reduction for coupled thermal electrical systems	19
<i>Marcus W.F.M. Bannenberg, Michael Günther</i>	
2.1. Introduction	19
2.1.1. Thermal-Electric Benchmark Circuit	19
2.2. Mathematical Formulation	20
2.2.1. Multirate time integration	20
2.2.2. Model order reduction	21
2.3. Implementation	21
2.3.1. Parameters and functions	22
2.3.2. Setting up the IRK iteration	22
2.3.3. The IRK iteration	22
2.3.4. The POD algorithm	23
2.4. Requirements	23
2.5. Numerical examples	23
2.5.1. Results	24
2.6. Conclusion	25
Bibliography	26

3. Validation of fluid-structure interaction simulations in membrane-based blood pumps 27

Marco Martinolli, Christian Vergara, Luc Polverelli

3.1. Introduction	27
3.2. Mathematical Formulation	28
3.2.1. The Fluid-Structure Interaction Model	28
3.2.2. Numerical Method	30
3.2.3. Hierarchical Modeling	30
3.2.3.1. Geometry of the domain	31
3.2.3.2. Meshing	31
3.2.3.3. Geometric coupling	32
3.2.3.4. Modeling the Contact	33
3.3. Implementation	34
3.4. Computer Requirements	36
3.5. Numerical Example	37
3.5.1. Experimental Data	37
3.5.2. Benchmark Plan	38
3.A. Appendix	39
3.A.1. Licences of Use	39
3.A.1.1. LIFEV (release version)	39
3.A.1.2. TetGen	39
3.A.1.3. Triangle	39
3.A.2. Configuration files	40
Bibliography	42

II. Model order reduction methods 44

4. Model order reduction for parametric high dimensional interest rate models in the analysis of financial risk 45

Andreas Binder, Onkar Jadhav, Volker Mehrmann

4.1. Introduction	45
4.2. Mathematical Formulation: Model Hierarchy	47
4.3. Numerical Methods	48
4.3.1. Parametric Model Order Reduction	50
4.3.2. Greedy Sampling Method	52
4.3.3. Adaptive Greedy Sampling Method	53
4.3.4. Adaptive Greedy Sampling Algorithm	55
4.4. Numerical Example	57
4.4.1. Computational cost	62
4.4.1.1. Floater Scenario Values	63
4.5. Conclusion	64
4.A. Relation between a singular value decomposition and a principal component analysis	64
Bibliography	65

5. Software-based representation of an inverse heat conduction problem 68

Patricia Barral, Federico Bianco, Riccardo Conte, Umberto Emil Morelli, Peregrina Quintela, Gianluigi Rozza, Giovanni Stabile

5.1. Introduction	68
5.2. Mathematical Formulation	70
5.2.1. Computational Domain and Notation	70

5.2.2.	Direct Heat Transfer Problem	71
5.2.3.	Inverse Problem	72
5.2.3.1.	Alifanov's Regularization	74
5.3.	Implementation	75
5.3.1.	Geometry and Meshing	75
5.3.2.	Direct Problem Solver	76
5.3.3.	Inverse Problem Solver	76
5.3.4.	Post-processing and Visualization	77
5.4.	Computer Requirements	77
5.5.	Benchmark Case	77
5.5.1.	Direct Problem	78
5.5.2.	Inverse Problem	81
	Bibliography	82

6. Coupled parameterized reduced order modelling of thermo-mechanical phenomena arising in blast furnaces 85

Nirav Shah, Patricia Barral, Michele Girfoglio, Alejandro Lengomin, Peregrina Quintela, Gianluigi Rozza

6.1.	Introduction	85
6.1.1.	Conceptual model	85
6.1.2.	Mathematical problem and Benchmark cases	86
6.1.2.1.	Benchmark for the thermal model	88
6.1.2.2.	Benchmark for the mechanical model	88
6.1.2.3.	Benchmark for the coupling	88
6.2.	Mathematical formulation	88
6.2.1.	Axisymmetric model	88
6.2.2.	Weak formulation for thermal model	90
6.2.3.	Weak formulation of the mechanical model	91
6.2.4.	Finite element method	93
6.2.5.	Parameter space	94
6.2.6.	Model order reduction	95
6.3.	Implementation	95
6.4.	Computer requirements	96
6.5.	Numerical examples	96
6.5.1.	Benchmark tests	97
6.5.1.1.	Thermal model	98
6.5.1.2.	Mechanical model	99
6.5.1.3.	Coupled model	102
6.5.1.4.	Simulation for actual problem	104
6.5.2.	Test problem for reduced basis method	105
6.5.2.1.	Thermal system	106
6.5.2.2.	Mechanical system	107
6.5.2.3.	Coupling system	107
6.3.	Hierarchy of thermal model	107
6.4.	Hierarchy of mechanical model	107
	Bibliography	107

III. Optimization methods	111
7. A benchmark for atmospheric tomography	112
<i>Bernadett Stadler, Ronny Ramlau, Andreas Obereder</i>	
7.1. Introduction	112
7.2. Mathematical formulation	113
7.2.1. Adaptive optics	113
7.2.1.1. Guide stars	113
7.2.1.2. Operating modes	115
7.2.1.3. Turbulence statistic in the atmosphere	116
7.2.1.4. Deformable mirror	116
7.2.1.5. Wavefront sensor	117
7.2.2. Problem formulation - atmospheric tomography	118
7.2.3. Solution method - Matrix Vector Multiplication	120
7.3. Implementation	120
7.4. Computer requirements	121
7.5. Numerical example	121
7.5.1. Input parameters	121
7.5.2. Output - DM commands	122
Bibliography	122
8. Acceleration of Sinkhorn Algorithm using ϵ scaling with applications to the Reflector Problem	124
<i>Jean-David Benamou, Guillaume Chazareix, Wilbert Ijzerman, Giorgi Rukhaia</i>	
8.1. Introduction	124
8.1.1. Optimal Transport model	124
8.1.2. Entropic Regularization of Optimal Transport	126
8.1.3. Sinkhorn Algorithm for Regularized Optimal Transport	127
8.1.4. Benchmark Cases	127
8.2. Hierarchical approach to Sinkhorn Algorithm	128
8.2.1. ϵ scaling	128
8.2.2. Discretization scaling	128
8.3. Entropic Bias	130
8.3.1. Entropic Bias and Sinkhorn Divergences	130
8.4. Implementation	130
8.5. Computer Requirements	131
8.6. Numerical Demonstration	132
Bibliography	133
9. Data driven model adaptations of coil sensitivities in magnetic particle imaging	135
<i>Sören Dittmer, José Carlos Gutiérrez Pérez, Lena Hauberg-Lotte, Tobias Kluth, Peter Maass, Daniel Otero Bager</i>	
9.1. Introduction and literature	135
9.1.1. Magnetic Particle Imaging	136
9.1.1.1. Scenarios in MPI	137
9.1.2. Deep Learning and Inverse Problems	140
9.1.2.1. A: Learned Penalty Terms	140
9.1.2.2. B: Plug-and-Play Prior Methods	141
9.1.2.3. C: Gradient-Descent-by-Gradient-Descent type Methods	141
9.1.2.4. D: Regularization by Architecture	142

9.1.2.5. E: Image Post-Processing via Deep Learning	143
9.1.3. Applying Deep Learning to Magnetic Particle Imaging	143
9.2. Implementation	143
9.3. Computer requirements	145
9.4. Numerical examples	145
Bibliography	147
10.A mixed-inter programming (MIP) model for a joint assignment of drivers and locomotives to trains at a rail freight company	151
<i>Jonasz Staszek, Andreas Bärmann, Alexander Martin</i>	
10.1. Introduction	151
10.2. Methods	151
10.3. Implementation	155
10.4. Results	156
10.5. Conclusions and Contributions	158
Bibliography	160

List of Acronyms

ITMATI	Technological Institute of Industrial Mathematics
USC	University of Santiago de Compostela
UDC	University of A Coruña
JKU	Johannes Kepler University Linz
ELT	Extremely Large Telescope
MAP	Maximum a-posteriori Estimate
AO	Adaptive Optics
MVM	Matrix-Vector-Multiplication
DM	Deformable Mirror
IDE	Integrated Development Environment
WFS	Wavefront sensor
GS	Guide Star
NGS	Natural Guide Star
LGS	Laser Guide Star
SCAO	Single Conjugate AO
LTAO	Laser Tomography AO
MOAO	Multi Object AO
MCAO	Multi Conjugate AO
FWHM	Full Width at Half Maximum
CCD	Charge-Coupled Device
BLAS	Basic Linear Algebra Subprograms
LAPACK	Linear Algebra Package
FFTW	Fastest Fourier Transform in the West
ADMM	alternating direction method of multipliers
CNN	convolutional neural network
CT	computer tomography
DL	deep learning
FFP	field free point
FFL	field free line
FOV	field-of-view
LISTA	Learning Fast Approximations of Sparse Coding
LSTM	long-short-term-memory network
MPI	magnetic particle imaging
MRI	magnetic resonance imaging
NN	neural networks
PET	positron emission tomography
RNN	recurrent neural network
SPIO	superparamagnetic iron oxide nanoparticles
SISSA	Scuola Internazionale Superiore di Studi Avanzati
CIP	Continuous Interior Penalty

DG	Discontinuous Galerkin
FSI	Fluid-Structure Interaction
LIFEV	Library of Finite Elements V
LVADs	Left Ventricular Assist Devices
NS	Navier-Stokes Equations
PDEs	Partial Differential Equations
HQ	Head pressure-Flow
NS	Navier-Stokes
XFEM	Extended Finite Element Method
3D	three dimensions
SISSA	Scuola Internazionale Superiore di Studi Avanzati

Part I.

Coupling methods



1. Implementing acoustic scattering simulations for external geometries within a porous enclosure

Ashwin Nayak¹, Andrés Prieto¹, Daniel Fernández²

¹*Instituto Tecnológico de Matemática Industrial, Universidade da Coruña*

²*Microflown Technologies*

Abstract. The details in implementing an acoustic scattering simulation on a rigid exterior domain enclosed in a porous layer are outlined. Acoustic transmission in fluid and porous media require different physical models and require conditions at interfaces to enable coupling. Time-harmonic models for static acoustic fluid and rigid-frame porous layers are explained and a coupling strategy is discussed. The coupled system is implemented using the finite-element method to obtain an approximate solution. An elaborate end-to-end strategy using open-source software tools is highlighted. The tools used to create meshes, to specify and solve the variational system of equations and to postprocess the obtained results are thoroughly explained. The workflow is illustrated for a test case with simple geometry such that the reader may easily be able to reproduce the results.

Keywords: Scattering, aeroacoustics, porous materials.

1.1. Introduction

Understanding the transmission of sound through different media is key in designing and improving accuracy of acoustic sensors to filter *noise* - a critical component in acoustic measurements. Of relevance in the project is the Microflown sensor, distinguished by their ability to measure both the magnitude and direction of sound. The sensor is commercially available in a variety of housings and enclosures, suiting different acoustic environments. Particularly, porous enclosures are used to improve measurement quality in wind although their effectiveness is sensitive to flow conditions. A coupled computational model can help accurately predict the influence on an incident sound signal by effectively integrating the influence of sound transmission in porous and turbulent media. An earlier report [1] proposed a progressive development of such a model in stages, given as benchmark cases. This article highlights the software implementation of the model-in-development, currently highlighting the implementation of coupling between a still fluid and a rigid-frame porous media. A similar procedure is utilized in further developments and will be reported in future.

This article illustrates the implementation of one of the benchmark stages of the project i.e. to solve for the acoustic scattering effect of a rigid object represented by the external domain Ω_S , enclosed entirely by a porous layer Ω_P . The setup is placed in an acoustic field represented by the unbounded domain Ω_F , as shown in the schematic Fig.1.1. To be more generic with possible configurations - a fluid-filled gap is considered between the structure and the porous enclosure. An acoustic wave of a certain kind (plane wave, spherical etc.) is assumed to be incident on the setup and a model is sought to compute the scattering of the incident wave due to the object.

1.2. Mathematical Formulation

The problem can be mathematically formulated in various physically-relevant variables e.g., scalar fields like pressure, displacement potential or velocity potential; or vector fields like displacement or velocity, the choice often being the vector fields for coupled systems [2]. In this particular implementation, the acoustic oscillations are chosen to be represented by the displacement vector field.



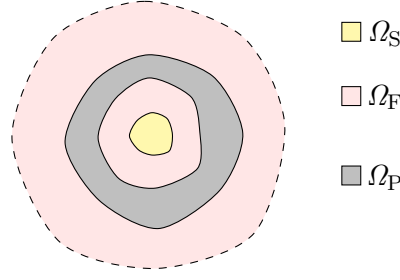


Figure 1.1: Schematic of the original problem configuration on unbounded domain

1.2.1. Model Hierarchy

A series of hierarchical models and associated assumptions are considered to arrive at a feasible overall numerical solution for the problem. The acoustic fluid is assumed to be homogeneous, non-viscous, compressible, isotropic and isentropic. Also, the porous layer is considered to be made of homogeneous, isotropic and isothermal material. The acoustic fields on the domain are assumed to be time-harmonic. The problem configuration is also posed in an unbounded domain which ensures a complete dissipation of all outgoing waves. This effect is ensured by with a finite truncation of the domain and an artificial fluid layer enclosing it with absorbing properties, known in literature as the perfectly matched layers (PML) technique [3, 4].

1.2.1.1. Fluid region

The mathematical model taking into account the transmission of sound oscillations must satisfy the Navier-Stokes equations. Utilizing the earlier assumptions, the equations reduce to being linear and for the time harmonic case, it simplifies into the Helmholtz equation. For a given angular frequency ω , it is expressed as,

$$-\nabla(\rho_F c_F^2 \operatorname{div} \mathbf{u}_F) - \rho_F \omega^2 \mathbf{u}_F = \mathbf{f}_F \quad \text{in } \Omega_F, \quad (1.1)$$

where, ρ_F and c_F are the mass density of the acoustic fluid and speed of sound. The forcing function \mathbf{f}_F represents any sound sources in the domain.

Boundary conditions may be given as,

$$\mathbf{u}_F \cdot \mathbf{n} = g, \quad (1.2)$$

where, g is normal displacement field at Γ_S . If the structure is assumed rigid, $g = 0$ else it may be expressed as a function. The fluid region has interfaces with porous layer and contact conditions with it will be elaborated in the following sections.

1.2.1.2. Porous region

Porous materials come in various structural configuration like fibrous, foam or granular kinds, requiring different models to characterize their acoustic behaviour. Considering the simplifying assumptions of isotropy accounts for the macroscopic behaviour of the material. The structural frame of the layer is also considered rigid for further simplification allowing for an *fluid-equivalent* formulation.

$$-\nabla(K_P(\omega) \operatorname{div} \mathbf{u}_P) - \rho_P(\omega) \omega^2 \mathbf{u}_P = \mathbf{f}_P \quad \text{in } \Omega_P, \quad (1.3)$$

where, $\rho_P(\omega)$ and K_P are the equivalent dynamic density and the equivalent dynamic bulk modulus of the porous material. The material properties are frequency dependent and are determined either through experiments conducted *a priori* or through suitable models. A wide range of porous material models provide the material response along a range of frequencies e.g the Zwikker-Kosten model, Miki model, Johnson-Champoux-



Allard-Lafarge Model, the Johnson-Champoux-Allard-Pride-Lafarge model, among others [4, 5].

The fairly detailed six-parameter Johnson-Champoux-Allard-Lafarge (JCAL) model is chosen in the current article to obtain the dynamic porous mass density and bulk modulus, given by equations,

$$\rho_P(\omega) = \frac{\rho_F}{\phi} \alpha_\infty \left(1 - i \frac{\sigma \phi}{\omega \rho_F \alpha_\infty} \sqrt{1 + i \frac{4\alpha_\infty^2 \eta \rho_F \omega}{\sigma^2 \Lambda^2 \phi^2}} \right), \quad (1.4)$$

$$K_P(\omega) = \frac{\gamma P_F / \phi}{\gamma - (\gamma - 1) \left(1 - i \frac{\eta \phi}{\rho_F k'_0 \omega \text{Pr}} \sqrt{1 + i \frac{4k'_0{}^2 \rho_F \omega \text{Pr}}{\eta \Lambda'^2 \phi^2}} \right)^{-1}}. \quad (1.5)$$

The JCAL model is reliable for porous materials with arbitrarily shaped pores. The parameters in the model: porosity ϕ , flow resistivity σ , tortuosity α_∞ , viscous characteristic length Λ , thermal characteristic length Λ' and static thermal permeability k'_0 ; effectively capture the macroscopic thermal, viscous and inertial characteristics of the porous material. The fluid state properties like density ρ_F , specific heat ratio γ , Prandtl Number Pr , and equilibrium fluid pressure P_F encapsulate the properties of the saturating fluid.

Contact conditions between the fluid and porous layers ensure continuity in normal-direction displacement and pressure fields. Hence,

$$\mathbf{u}_F \cdot \mathbf{n} = \mathbf{u}_P \cdot \mathbf{n}, \quad (1.6)$$

$$\text{extend } \rho_F c^2 \text{div } \mathbf{u}_F = K_P \text{div } \mathbf{u}_P. \quad (1.7)$$

1.2.1.3. Perfectly matched layer

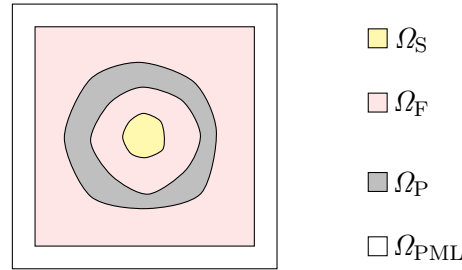


Figure 1.2: Schematic of the domain with the perfectly matched layer

The unbounded domain of the acoustic fluid implies a complete dissipation of all outgoing waves. Since an actual computation needs a finite domain, the outer boundary needs to absorb any outward incident waves. In theory, this is expressed as the Sommerfeld boundary condition given as the limiting condition at large distances,

$$\lim_{|\mathbf{x}| \rightarrow \infty} |\mathbf{x}| \left(\text{div } \mathbf{u}_F - i k_F \mathbf{u}_F \cdot \frac{\mathbf{x}}{|\mathbf{x}|} \right) = 0. \quad (1.8)$$

This condition is realistically hard to impose and is typically accomplished by a "sponge" layer with dissipating properties known as the perfectly matched layer (PML). The Helmholtz-like governing equation with auxiliary complex stretching factors within the PML are imposed to replicate this as (see [6]),

$$- \text{div}(\rho_F c_F^2 \tilde{\mathbf{C}}(\nabla \mathbf{u}_{\text{PML}})) - \rho_F \omega^2 \tilde{\mathbf{M}} \mathbf{u}_{\text{PML}} = \mathbf{0}. \quad (1.9)$$



Here, $\tilde{\mathbf{C}}$ and $\tilde{\mathbf{M}}$ are fourth-order and second-order tensors given by,

$$\tilde{\mathbf{C}}(\nabla \mathbf{w}) = \left(\sum_{j=1}^3 \frac{1}{\gamma_j} \frac{\partial w_j}{\partial x_j} \right) \mathbf{I}, \quad (1.10)$$

$$\text{extend} \quad \tilde{\mathbf{M}} = \sum_{j=1}^3 \gamma_j \mathbf{e}_j \otimes \mathbf{e}_j, \quad (1.11)$$

with \mathbf{I} being the fourth-order identity tensor and \mathbf{e}_j 's being the unit vectors along the spatial directions. The optimally-tuned functions provided by Bermudez et al. [7] are chosen among the various choices for the complex stretching functions γ_j 's, giving,

$$\gamma_j(x_j) = \begin{cases} 1, & \text{if } |x_j| \leq L_j, \\ 1 + i \frac{c_F}{\omega(L_j^\infty - |x_j|)}, & \text{if } L_j \leq |x_j| \leq L_j^\infty. \end{cases} \quad (1.12)$$

Here, L_j and L_j^∞ are respectively the lengths of the Cartesian box of the truncated fluid domain and the PML domain, along the direction x_j from the origin. The piece-wise definition of γ_j ensures the absorption of waves only along the outward direction of propagation. The interface conditions between the PML and fluid layers are just the continuity in displacement and pressure fields, namely,

$$\mathbf{u}_F \cdot \mathbf{n} = \mathbf{u}_{\text{PML}} \cdot \mathbf{n}, \quad (1.13)$$

$$\text{extend} \quad \rho_F c_F^2 \operatorname{div} \mathbf{u}_F = \rho_F c^2 \operatorname{div} \mathbf{u}_{\text{PML}}. \quad (1.14)$$

1.2.2. Coupling

The entire formulation for the coupled problem is given by the following system of equations: for a particular frequency, ω ,

$$-\nabla(\rho_F c_F^2 \operatorname{div} \mathbf{u}_F) - \rho_F \omega^2 \mathbf{u}_F = \mathbf{f}_F \quad \text{in } \Omega_F, \quad (1.15)$$

$$-\nabla(K_P(\omega) \operatorname{div} \mathbf{u}_P) - \rho_P(\omega) \omega^2 \mathbf{u}_P = \mathbf{0} \quad \text{in } \Omega_P, \quad (1.16)$$

$$-\operatorname{div}(\rho_F c_F^2 \tilde{\mathbf{C}}(\nabla \mathbf{u}_{\text{PML}})) - \rho_F \omega^2 \tilde{\mathbf{M}} \mathbf{u}_{\text{PML}} = \mathbf{0} \quad \text{in } \Omega_{\text{PML}}, \quad (1.17)$$

$$\mathbf{u}_F \cdot \mathbf{n} = g \quad \text{on } \Gamma_S, \quad (1.18)$$

$$\mathbf{u}_F \cdot \mathbf{n} - \mathbf{u}_P \cdot \mathbf{n} = 0 \quad \text{on } \Gamma_C, \quad (1.19)$$

$$\rho_F c_F^2 \operatorname{div} \mathbf{u}_F - K_P(\omega) \operatorname{div} \mathbf{u}_P = 0 \quad \text{on } \Gamma_C, \quad (1.20)$$

$$\mathbf{u}_F \cdot \mathbf{n} - \mathbf{u}_{\text{PML}} \cdot \mathbf{n} = 0 \quad \text{on } \Gamma_{\text{PML}}, \quad (1.21)$$

$$\operatorname{div} \mathbf{u}_F - \operatorname{div} \mathbf{u}_{\text{PML}} = 0 \quad \text{on } \Gamma_{\text{PML}}. \quad (1.22)$$

The coupling between different layers occurs only at interfaces, and the displacement vector fields, \mathbf{u}_F , \mathbf{u}_P and \mathbf{u}_{PML} , are defined in exclusive domains with different smoothing requirements. It can be unified to be a member of a functional space \mathbf{V} introduced as,

$$\mathbf{V} = \left\{ \mathbf{v} \in [\mathbf{L}^2(\Omega)]^3 : \mathbf{v}|_{\Omega_F} \in \mathbf{H}(\operatorname{div}, \Omega_F), \mathbf{v}|_{\Omega_P} \in \mathbf{H}(\operatorname{div}, \Omega_P), \tilde{\mathbf{M}} \mathbf{v}|_{\Omega_{\text{PML}}} \in [\mathbf{L}^2(\Omega_{\text{PML}})]^3, \right. \\ \left. \sum_{j=1}^3 \frac{1}{\gamma_j} \frac{\partial v_j}{\partial x_j} \Big|_{\mathbf{v} \in \Omega_{\text{PML}}} \in \mathbf{L}^2(\Omega_{\text{PML}}), \mathbf{v} \cdot \mathbf{n} = 0 \text{ on } \Gamma_\infty \right\}, \quad (1.23)$$

provided the necessary continuity and differentiable properties are satisfied at the interfaces. The variational



form can then be deduced from Equations (1.15)-(1.9) by multiplying a test function $\mathbf{v} \in \mathbf{V}$ and utilizing the Green's theorem : Find $\mathbf{u} \in \mathbf{V}$ such that,

$$\begin{aligned} \int_{\Omega_F} \rho_F c^2 (\operatorname{div} \mathbf{u})(\operatorname{div} \mathbf{v}) \, dV - \int_{\Omega_F} \rho_F \omega^2 \mathbf{u} \cdot \mathbf{v} \, dV \\ + \int_{\Omega_P} K_P(\omega) (\operatorname{div} \mathbf{u})(\operatorname{div} \mathbf{v}) \, dV - \int_{\Omega_P} \rho_P(\omega) \omega^2 \mathbf{u} \cdot \mathbf{v} \, dV \\ + \int_{\Omega_{PML}} \rho_F c^2 \tilde{\mathbf{C}}(\nabla \mathbf{u}) : \nabla \mathbf{v} \, dV - \int_{\Omega_{PML}} \rho_F \omega^2 \tilde{\mathbf{M}} \mathbf{u} \cdot \mathbf{v} \, dV = \int_{\Omega_F} \mathbf{f}_F \cdot \mathbf{v} \, dV \end{aligned} \quad (1.24)$$

holds for all $\mathbf{v} \in \mathbf{V}$ and also, $\mathbf{v} = 0$ at Γ_S . This equation maybe more conveniently expressed in the general form of a linear variational problem with \mathcal{A} and \mathcal{L} being the sesquilinear and linear functionals as,

$$\mathcal{A}(\mathbf{u}, \mathbf{v}) = \mathcal{L}(\mathbf{v}). \quad (1.25)$$

A practical implementation of this model would require the approximation of the infinite dimensional functional space \mathbf{V} , with a discrete n -dimensional space \mathbf{V}_h with a finite set of basis functions ψ_h , $h = 1, 2, \dots, n$. This reforms Equation (1.25) as,

$$\sum_{r=1}^n \mathcal{A}(\psi_r, \psi_s) \mu_r = \mathcal{L}(\psi_s) \quad \text{for } s = 1, 2, \dots, n; \quad (1.26)$$

with μ_r 's as coefficients of the basis functions. A solution may then be obtained by solving this linear system of equations. The following sections describe the implementation of this model along with a specific example of acoustic transmission across a porous layer around a vibrating sphere.

1.3. Implementation

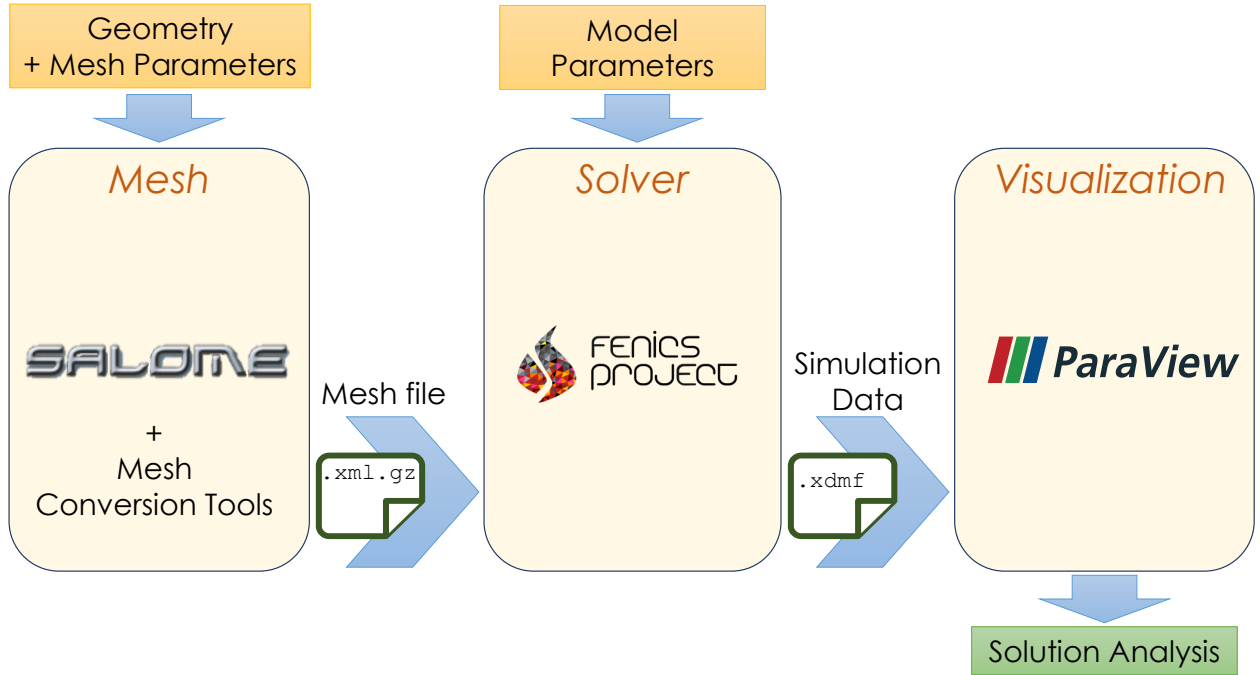


Figure 1.3: Workflow representing implementation stages and their interfaces.

The implementation follows the requirements of the model and may be divided into three main stages viz., mesh generation, solving equations and visualizing solutions. The different stages of the implementation and the overall workflow is illustrated in Fig.1.3. The mesh generation stage requires the user inputs on geometrical configuration of the setup. This includes the exact dimensions of the structure, porous layer, fluid domain and PML. Considering that the variational form includes integrals which differ in sub-domains, it is necessary to mark the mesh cells according to region requiring conformality of the mesh with the geometry of sub-domains. Furthermore, user inputs may be needed to suggest local refining of the mesh in a particular region or surface to capture the geometry accurately. The generated mesh also needs to be adapted to the file format compatible with the solver. The solver imports the mesh data and categorizes cells according the sub-domain regions. It is responsible for implementing the finite-element method - defining the discrete functional space with chosen basis functions and assembling the system of equations before solving them. The solution obtained may also include processing for analysis before being saved in a memory-efficient storage format. Finally, the visualization tool reads the simulated solution from the disk to provide graphical representations aiding the user in deriving information and performing analysis. The following sections explain the usage of each of the stages and the related tools in detail.

1.3.1. Geometry and Meshing

The digital representation of the setup is first done by modeling the geometry and then discretizing it to form a mesh. While several tools and techniques are available for this, the open-source modules offered by SALOME are used in this article, which provides capabilities for interfacing with various numerical simulation tools. It has a flexible cross-platform architecture made of reusable components allowing for customized integration and handling of complex geometrical objects. It allows for creation of geometry and meshes using either (or both) the graphical user interface (GUI) and a text user interface (TUI). The following sections explains the usage of creating geometries and meshes using the TUI, a powerful Python-based scripting interface. The same may also be achieved either in part or entirety by using the GUI. The GUI also allows for exporting the equivalent state in a TUI script.

The TUI provides `geomBuilder` class for the creating and editing geometry. The class instance contains a list of function attributes for operations useful in creating complex geometrical objects. It allows for creating basic objects and primitives in 1D, 2D, 3D; perform boolean operations like fuse, common, cut and section operations; execute extrusion, rotation and other linear operations; create higher order topological objects like solids and compounds grouped from primitives; and implement an advanced partition or gluing between geometrical structures, among others. Table 1.1 lists some useful TUI commands available as function attributes, whereas a detailed description along with other functions are available in the documentation [8].

The meshing section is again accessed through another Python module, `smeshBuilder`. It presents different algorithms to create meshes on the basis of geometrical models created by `geomBuilder`. The module also provides control on mesh generation like maintaining conformality between subgroups, splitting, editing, boolean operations and marking. These are handy in segregating subdomains accurately in a mesh. The NETGEN-1D2D3D algorithm is utilized for the purpose of the project which provides a range of control parameters like tetrahedral or hexahedral mesh elements, specifying the global maximum or minimum size of the edges and also control locally-permissible edge sizes on a lower-order geometrical construct - useful in refining the mesh near a point or a face.

The instance of the `smeshBuilder` class is provided with the geometrical object to mesh and the algorithm specifications. The snippet code below illustrates access to the parameters of NETGEN-1D2D3D algorithm and specify the relevant conditions for the mesh. The most useful functions are listed in table 1.1. The mesh is computed after all the parameters are set.

```
from salome.smesh import smeshBuilder
smesh = smeshBuilder.New() #Instantiate the smeshBuilder class
domain_mesh = smesh.Mesh(Domain) #Domain is a geometrical object
```



```
# Select type of elements and algorithm to compute them
NETGEN_3D = domain_mesh.Tetrahedron(algo=smeshBuilder.NETGEN_1D2D3D)
NETGEN_3D_Params = NETGEN_3D.Parameters() #Access parameters.
```

GeomBuilder	MakeVertex	Creates a vertex
	MakeVectorDXDYDZ	Creates a vector
	MakeBoxTwoPnt	Creates a Cuboid defined by ends of a body diagonal
	MakeSphereR	Creates a Sphere at origin given the radius
	MakeCylinder	Creates a Cylinder
	TranslateDXDYDZ	Linear translation of a geometrical object
	MakeCutList	Boolean operation between two geometrical object
	MakePartition	Group various sub-shapes into one geometrical object
Mesh	SubShapeSortedCentres	Obtain sub-structures of a complex object
	Tetrahedron	Set cells to be tetrahedral
	Compute	Compute the Mesh
NETGEN Parameters	Reorient2DBy3D	Order the cell normals to face either outward or inward
	SetMaxSize	Limit global maximum edge length in mesh
	SetMinSize	Set global minimum edge length in mesh
	SetLocalSizeOnShape	Set local size on a sub-shape

Table 1.1: Useful functions in SALOME's geom and smesh modules

The SALOME classes are provided only in its own environment and the scripts are executed through a wrapper offered as,

```
$ salome -t script.py # Runs script.py in SALOME's shell
```

1.3.2. Solver

The solver of choice is FEniCS, a popular open-source finite-element library for solving partial differential equations (PDEs). It offers a rich interface with data-structures and optimized algorithms for finite-element code which makes it easy to write PDEs. The library is hardware-optimized and parallel by design and is easily deployed to high-performance computing clusters. With its Python and C++ interfaces, FEniCS offers powerful capabilities to integrate into various scientific computing workflows.

The FEniCS library offers a number of component modules and the interfacing is done mainly through its DOLFIN and UFL modules. DOLFIN is a highly optimized computational back-end responsible for finite-element machinery and offers a rich Python interface. It provides abstract data-structures similar to mathematical terms such as mesh, finite element, function spaces and functions. It also includes compute-intensive algorithms such as finite-element assembly and mesh refinement, and, interfaces to various linear algebra solvers and libraries like PETSc and Eigen. UFL, on the other hand, provides an abstract mathematical language to express variational problems which are interpreted automatically and connected to DOLFIN classes.

The powerful feature of the solver is its ability to interpret the variational form in an easily readable framework. The Python module also allows for finer control through a detailed interface to the underlying C++ code enabling sub-classing and base class overloading. Among others, it provides an `Expression` class which can be used for user-defined expressions specified by C++ code and compiled during execution by a just-in-time (JIT) compiler for efficiency. A detailed documentation along with numerous examples are offered by Langtangen et al. [9] and at the official FEniCS documentation webpage [10]. It is to be noted that, currently the



solver is limited by its inability to handle complex numbers and needs additional care to ensure that the real and imaginary parts of the equations and function spaces are represented separately.

1.3.3. Post-processing and Visualization

Visualization of generated simulation data is critical in understanding the physical process. Implementing and representing this data in a simple and effective manner is extremely useful for deriving information, presenting results, and also in testing and debugging. The post-processing operations on the solution is dependent on the study undertaken by the user. In this specific use case, some routine cases of analysis include validation of the solver for a test case, measure of a field at a particular point in space and directivity patterns of fields around the object amongst many others. The implementation of these could either be included in the solver phase or during the visualization phase. Within the solver phase, these could just be operations on the solution data done using Python and plotted using some common user preferred graphing libraries like Matplotlib [11]. The approach quickly gets overwhelming while dealing with 3D datasets and it is useful to use a dedicated visualization tool like ParaView. It is a widely used open-source visualization tool for plotting and viewing solutions and graphs. It offers a powerful and an intuitive 3D visualization interface allowing for heavy in-situ customization and processing of simulation data. Furthermore, it also provides a Python scripting interface to automate visualization for batch processing.

ParaView uses a three-stage procedure for visualization of data: reading, filtering and rendering, all done using the user interface. The simulation data from the solver is read into memory through many supported file formats. The dataset being typically large, the XDMF (eXtensible Data Model and Format) file format is used for storage, which is able to manage extremely large datasets and is scalable for parallel systems. Filters provide the ability to extract or analyze this data into information. There are a wide range of filters available for analysis and visualization including plotting graphs, contours, surface plots, vector field plots etc. In addition, it is also possible to define user-defined filters to perform customized operations. The rendering stage deals with generating images or interactive plots from the filtered information. ParaView provides a user guide [12] and many tutorials [13] highlighting the usage and relevance of each of the stages along with the available functionalities to fully exploit its potential.

1.4. Computer Requirements

The software tools used in the implementation of the project require a minimal UNIX system with atleast 1GB of memory and about 500MB of disk space (swap) for execution. It is recommended to have some higher configuration that would ease the workflow and be capable of handling problems of larger order. The requirements however are decided by the finite-element solver, which handles assembling and solving Equation (1.26). The matrix in the equation is dependent on the mesh configuration and in a typical 3D use-case scenario with an unstructured tetrahedral mesh, the author suggests a system with 64GB memory for 1 million cells.

1.5. Case study : Acoustic transmission of a vibrating sphere in a porous enclosure

The implementation considered is a test case to validate our model, the acoustic transmission of a vibrating sphere placed in a spherical porous enclosure. The example illustrates the use of the tools described in the earlier sections. The solution for the case can also be computed analytically.

1.5.1. Description of test case sphere

A sphere of radius R_0 is placed at the origin within an acoustic fluid of density ρ_F . It is enclosed in a hollow spherical porous disk with an inner radius of R_1 and an outer radius of R_2 . Given that the surface of the sphere vibrates at a constant rate producing oscillations of frequency ω , the problem is then to compute the distortion of the acoustic field due to the porous layer.



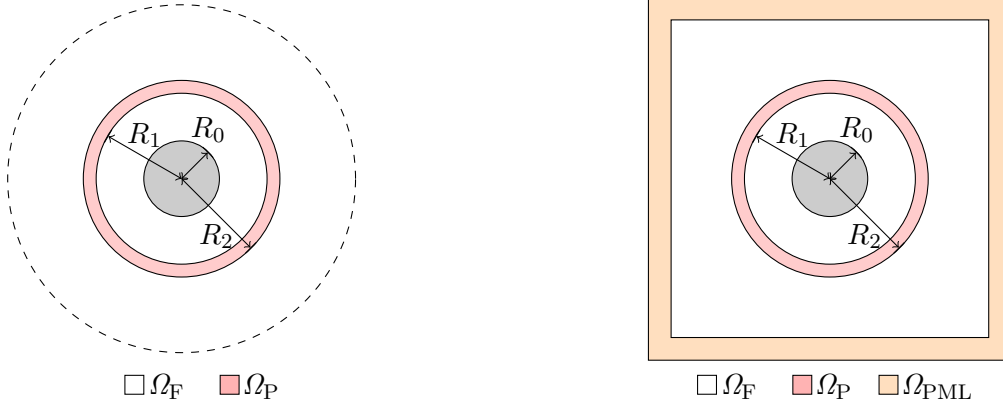


Figure 1.4: Schematic of the spherical test case posed in an unbounded domain (left) and modeled using perfectly matched layers (right).

1.5.2. Exact solution

The exact solution may be obtained by enforcing a constant Neumann boundary condition on the surface of the sphere. The radial symmetry of the configuration ensures that the Helmholtz equation has only radially dependent solutions which may be expressed as a linear combination of incoming and outgoing waves. Writing in term of pressure fields the exact solution is obtained as,

$$p(r) = \begin{cases} A_1 \frac{\exp(ik_F r)}{r} + B_1 \frac{\exp(-ik_F r)}{r}, & r \in [R_0, R_1], \\ A_2 \frac{\exp(ik_P r)}{r} + B_2 \frac{\exp(-ik_P r)}{r}, & r \in [R_1, R_2], \\ B_3 \frac{\exp(-ik_F r)}{r}, & r \in [R_2, \infty). \end{cases} \quad (1.27)$$

where the A_l 's correspond to the coefficients of incoming waves (for $l = 1, 2$), and B_m 's to the coefficients of outgoing waves (for $m = 1, 2, 3$). In an unbounded domain, since there are no incoming waves from infinity, the coefficient of the incoming component is zero. Considering that the pressure and displacement field needs to satisfy conditions on structure boundary (at R_0) and continuity conditions at fluid-porous media interfaces (R_1 and R_2), it follows that,

$$\frac{\partial p}{\partial r}(R_0) = \rho_F \omega^2 g_0, \quad g_0 = \text{constant}, \quad (1.28)$$

$$p(R_1^-) = p(R_1^+), \quad (1.29)$$

$$\rho_P \frac{\partial p}{\partial r}(R_1^-) = \rho_F \frac{\partial p}{\partial r}(R_1^+), \quad (1.30)$$

$$p(R_2^-) = p(R_2^+), \quad (1.31)$$

$$\rho_F \frac{\partial p}{\partial r}(R_2^-) = \rho_P \frac{\partial p}{\partial r}(R_2^+). \quad (1.32)$$

Substituting the general solution (1.27) in (1.28)-(1.32) and rewriting in terms of the coefficients, A_l 's and B_m 's, a 5×5 system of equations is obtained and is solved to compute the exact solution.



1.5.3. Geometry and Mesh

The scripting interface of SALOME is utilized to prepare the geometry and mesh for the problem. The parameters of the sphere is set initially in the script. Once the geometry building module is instantiated, it is conventional to create an origin and the axes. The volumetric spherical object may be created using the `MakeSphereR` function. The porous layer is then created by an intersection of two concentric spheres using the `MakeCutList` function to perform a boolean deletion of the volume of one by another. The two-part fluid region is also made using a similar boolean operation on a Cartesian box where the "unbounded" domain is truncated.

```

from salome.geom import geomBuilder

# Parameters
L_0 = L_1 = L_2 = 0.2
R_0 = 0.5
R_1 = 2 * R_0
R_2 = 2.25 * R_0

# Create an instance of the geometry builder class
geompy = geomBuilder.New()

# Origin and Axes
O = geompy.MakeVertex(0, 0, 0)
OX = geompy.MakeVectorDXDYDZ(1, 0, 0)
OY = geompy.MakeVectorDXDYDZ(0, 1, 0)
OZ = geompy.MakeVectorDXDYDZ(0, 0, 1)

# Sphere
Sphere = geompy.MakeSphereR(R_0)

# Porous Layer
porous_in = geompy.MakeSphereR(R_1)
porous_out = geompy.MakeSphereR(R_2)
porous_layer = geompy.MakeCutList(porous_out, [porous_in], True)

# Fluid Domain
b0 = geompy.MakeVertex(L_0, L_1, L_2)
b1 = geompy.MakeVertex(-L_0, -L_1, -L_2)
B = geompy.MakeBoxTwoPnt(b0, b1)
Fluid1 = geompy.MakeCutList(B, [porous_out], True)
Fluid2 = geompy.MakeCutList(porous_in, [Sphere], True)

```

While the basic domains of interest are created in an obvious manner outlined above, care is taken in constructing the PML layer. The integration of the variational form (1.24) within the PML domain contains the terms \tilde{C} and \tilde{M} which are dependent on the axial orientation. To ensure mesh conformality, the layer is formed by blocked subdomains along each axis. Highlighted below is creation of one such layer enveloping the fluid domain on the x_0 direction,

```

# x0-direction subdomain
be0 = b0

```



```
be1 = geompy.MakeVertex( L_0+L_pml, -L_1, -L_2)
Be = geompy.MakeBoxTwoPnt (be0, be1)
```

Similar such blocks are created enveloping the fluid domain along all the directions - two each along the x_0 , x_1 and x_2 directions, four each along x_0x_1 , x_1x_2 and x_2x_0 directions and eight along the corners i.e. the $x_0x_1x_2$ direction. All the layers formed are unified in a single geometrical object achieved by the function attribute `MakePartition` as,

```
Domain = geompy.MakePartition(Domainlist)
```

where, `Domainlist` is a Python list of all the separate subdomains. The function `SubShapeSortedCentres` is then used to obtain any sub-structure identifiers, useful in specifying local mesh refinements. Figure 1.5a shows a cross-section of the generated geometry.

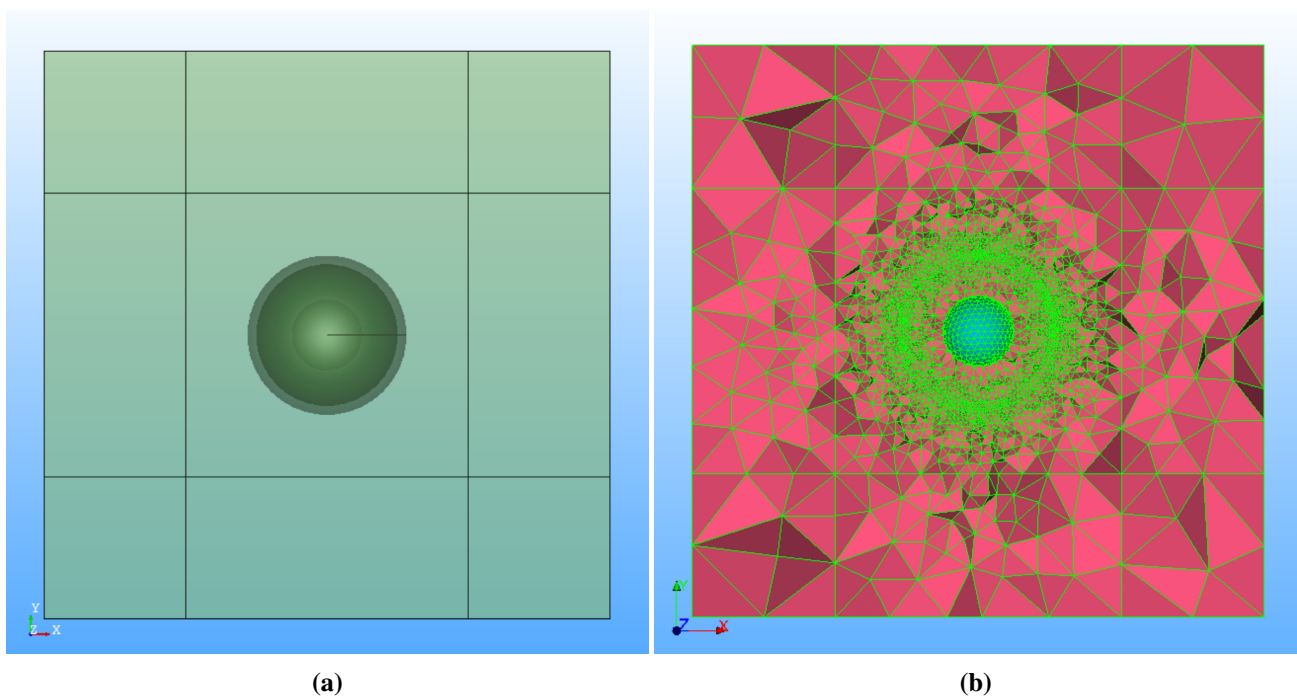


Figure 1.5: View of the geometry(left) and the cross-section of the Mesh(right).

The meshing operation is then carried out using the `smesh` module of SALOME as was similarly suggested in 1.3.1 . Figure 1.5b shows a cross-sectional view of the computed mesh volume along with local refinements within the porous layers and around the spherical boundary using the `SetLocalSizeOnShape` function. The `Reorient2DBy3D` function was also used to ensure that all boundary face normals pointed outwards of the enclosure. The step is crucial considering that the boundary condition is specified across these faces using expressions. The computed mesh is then be exported to file in `.unv` file format using function attributes of the mesh object. Conversion to `.xml.gz` for interfacing with the solver is achieved using `FEConv` [14] and `dolfin-convert` (provided by FEniCS) utilities as,

```
$ feconv -gm mesh.unv mesh.msh
$ dolfin-convert mesh.msh mesh.xml
$ gzip mesh.xml
```



1.5.4. Solver

The solver section handles details of the finite element implementation of the model and is made concise only due to the features of the FEniCS library. The geometrical parameters regarding the physical dimensions of the various subdomains and the model parameters related to frequency of choice, the user-defined boundary parameters etc are provided. Subsequently, the mesh is read into memory and each of the subdomains are marked by different identifiers, which is illustrated below.

```

### Fluid domain lengths : L[0], L[1], L[2] in x,y,z directions
### Porous domain radii : r_min (inner) and r_max (outer)
### Mesh object : mesh

# Obtain list of all cell identifiers
subdomains = MeshFunction("size_t", mesh, mesh.topology().dim())

# # Sub-domain specifications
tol = DOLFIN_EPS_LARGE

pml_cond = "fabs(x[0])>Lx-tol || fabs(x[1])>Ly-tol || fabs(x[2])>Lz-tol"
pml_layer = CompiledSubDomain(pml_cond,
                               Lx=L[0], Ly=L[1], Lz=L[2], tol=tol)

por_cond = "x.norm() > r_min-tol && x.norm() < r_max + tol"
por_layer = CompiledSubDomain(por_cond,
                               r_min=r_min, r_max=r_max, tol=tol)

# # Mark Sub-domains
subdomains.set_all(0)           # Fluid_Marker = 0
pml_layer.mark(subdomains, 1)  # PML_Marker = 1
por_layer.mark(subdomains, 2)  # Porous_Marker = 2

```

A similar procedure is also used to mark the boundary faces by different identifiers so as to specify boundary conditions. The function space is initialized with Raviart-Thomas finite elements, which define basis functions as unit vectors along the normals of the faces. To substitute a complex function space, it is necessary to initialize a hybrid function space made of two real parts,

```

# # Define function space (1st order Raviart-Thomas elements)
RT = d.FiniteElement("RT", mesh.ufl_cell(), 1)
Q = d.FunctionSpace(mesh, RT)

# # Define 2-part real function spaces to substitute for complex space
V = d.FunctionSpace(mesh, RT * RT)

(u_re, u_im) = d.TrialFunctions(V)
(v_re, v_im) = d.TestFunctions(V)

```

The variational form maybe expressed in three separate stages - one each for the fluid, porous and PML subdomains. The primary task is to represent each term on the Equation (1.24) in real and imaginary portions. This may be achieved easily by splitting Equations (1.15)-(1.22) into real and imaginary parts and following in the



same process to achieve a similar variational form like (1.24). The wave equations being linear help separate each of the term into two,

$$\begin{aligned}
 & \int_{\Omega_F} \rho_F c^2 \Re(\operatorname{div} \mathbf{u}) \Re(\operatorname{div} \mathbf{v}) \, dV - \int_{\Omega_F} \rho_F \omega^2 \Re(\mathbf{u}) \cdot \Re(\mathbf{v}) \, dV \\
 & \int_{\Omega_F} \rho_F c^2 \Im(\operatorname{div} \mathbf{u}) \Im(\operatorname{div} \mathbf{v}) \, dV - \int_{\Omega_F} \rho_F \omega^2 \Im(\mathbf{u}) \cdot \Im(\mathbf{v}) \, dV \\
 & + \int_{\Omega_P} \Re(K_P(\omega) \operatorname{div} \mathbf{u}) \Re(\operatorname{div} \mathbf{v}) \, dV - \int_{\Omega_P} \Re(\rho_P(\omega) \omega^2 \mathbf{u}) \cdot \Re(\mathbf{v}) \, dV \\
 & + \int_{\Omega_P} \Im(K_P(\omega) \operatorname{div} \mathbf{u}) \Im(\operatorname{div} \mathbf{v}) \, dV - \int_{\Omega_P} \Im(\rho_P(\omega) \omega^2 \mathbf{u}) \cdot \Im(\mathbf{v}) \, dV \\
 & + \int_{\Omega_{\text{PML}}} \rho_F c^2 \Re(\tilde{\mathbf{C}}(\nabla \mathbf{u})) : \Re(\nabla \mathbf{v}) \, dV - \int_{\Omega_{\text{PML}}} \rho_F \omega^2 \Re(\tilde{\mathbf{M}} \mathbf{u}) \cdot \Re(\mathbf{v}) \, dV \\
 & + \int_{\Omega_{\text{PML}}} \rho_F c^2 \Im(\tilde{\mathbf{C}}(\nabla \mathbf{u})) : \Im(\nabla \mathbf{v}) \, dV - \int_{\Omega_{\text{PML}}} \rho_F \omega^2 \Im(\tilde{\mathbf{M}} \mathbf{u}) \cdot \Im(\mathbf{v}) \, dV = 0. \quad (1.33)
 \end{aligned}$$

Note that the right-hand term is null since there are no source terms in the configuration. The Equation (1.33) can be easily specified in FEniCS using UFL. Since the subdomains are marked with identifiers already, the bilinear form within the fluid layer maybe specified directly as,

```

# FLUID LAYER : SubDomain Marker 0
# Define Bilinear form within the fluid layer
a = (rho * c**2 * div(u_re) * div(v_re) * dx(0)
     + rho * c**2 * div(u_im) * div(v_im) * dx(0)
     - rho * omega**2 * inner(u_re, v_re) * dx(0)
     - rho * omega**2 * inner(u_im, v_im) * dx(0))
    
```

The PML and the porous layer require operating between complex arithmetic and UFL operations. These are made simpler by declaring a complex container class with the necessary operations like multiplication, division and conjugation. This reduces the complex arithmetic into UFL operations and increases readability of the code. The Complex class declared below shows a simple structure of such a container class which contains member functions to return just the real or imaginary parts of the common operations. The return values of each of them translates the arithmetic into UFL operations.

```

class Complex(object):
    def __init__(self, real_part, imag_part):
        self.real=real_part
        self.imag=imag_part

    # Define inner-(conjugated) product
    def dot_re(f, g):
        return f.real*g.real + f.imag*g.imag

    def dot_im(f, g):
        return f.imag*g.real - f.real*g.imag

    # Define product
    def prod_re(f, g):
        return f.real*g.real - f.imag*g.imag
    
```



```

def prod_im(f, g):
    return f.imag*g.real + f.real*g.imag

# Define division
def div_re(f, g):
    return dot_re(f, g)/dot_re(g, g)

def div_im(f, g):
    return dot_im(f, g)/dot_re(g, g)

```

Further consideration is needed to specify piece-wise functions (γ_j 's) into the variational form. A direct approach would be segregating the PML domain into smaller domains defined by intervals where γ_j 's are resolved, and specify the operations separately. While this approach saves the assembling time, it can quickly get tedious and instead be expressed using logical operators offered by UFL [15], specifically the conditional operator. This leaves the evaluations to be performed during the assembling phase. The following code illustrates the bilinear form expression using the Complex class and the UFL logical operators. The real and imaginary parts of the tensor product $\tilde{C}(\nabla u) : \nabla v$ and the inner product $\tilde{M}u \cdot v$ are expressed by simplifying the operation.

```

# PML LAYER : SubDomain Marker 1
def s(j):
    return conditional(gt(abs(x[j]), L[j] + tol),
                      c / abs(abs(x[j]) - (L[j] + Lpml)) / omega,
                      Constant("0."))

def gamma(j):
    return Complex(Constant("1.0"), s(j))

def du_dx(u_re, u_im, i):
    return Complex(Dx(ure[i], i), Dx(uim[i], i))

# # Divergence operator in the PML domain
Div_re = sum(prod_re(Complex(1., 0.) / gamma(i), du_dx(u_re, u_im, i))
             for i in range(3))
Div_im = sum(prod_im(Complex(1., 0.) / gamma(i), du_dx(u_re, u_im, i))
             for i in range(3))

# # Scaled PML displacement vector to be used in the mass matrix
def coef(ure, uim, i)
    return Complex(ure[i], uim[i])

u_scaled_re = as_vector([prod_re(gamma(i), coef(u_re, u_im, i))
                        for i in range(3)])
u_scaled_im = as_vector([prod_im(gamma(i), coef(u_re, u_im, i))
                        for i in range(3)])

dx_pml = dx(1, scheme='default', degree=6)

# # Define bilinear form in the PML layer
a += (rho * c**2 * Div_re * d.div(v_re) * dx_pml

```



```

+ rho * c**2 * Div_im * d.div(v_im) * dx_pml
- rho * omega2 * inner(u_scaled_re, v_re) * dx_pml
- rho * omega2 * inner(u_scaled_im, v_im) * dx_pml

```

The expression for porous layer requires the porous density and bulk modulus obtained from the JCAL model. These are easily computed from expressions (1.4) and (1.5) using Python's inbuilt complex arithmetic support and the results are subsequently cast into `Complex` class type. The resulting bilinear expressions can then be specified as,

```

# POROUS LAYER : SubDomain Marker 2
div_u = Complex(div(u_re), div(u_im))
a += (prod_re(BulkMod_P, div_u) * div(v_re) * dx(2)
      + prod_im(BulkMod_P, div_u) * div(v_im) * dx(2)
      - omega**2 * inner(prod_re(Rho_P, u), v_re) * dx(2)
      - omega**2 * inner(prod_im(Rho_P, u), v_im) * dx(2))

```

The boundary conditions at the structure are also divided into real and imaginary parts and the expressions are specified using the `Expression` class. They can then be applied over the relevant faces using the `DirichletBC` class.

```

# BC at Structure Boundary (Marker 2)
bc = [DirichletBC(V.sub(0), g_re, bdry_markers, 2), # Real subspace
      DirichletBC(V.sub(1), g_im, bdry_markers, 2)] # Imag subspace

```

The system is now completely determined and can be assembled and solved using the MUMPS (MULTifrontal Massively Parallel Sparse direct Solver) linear algebra solver. The resulting solution is separated into real and imaginary parts using the `split` attribute of the `Function` class and are then written into XDMF files using the DOLFIN provided `XDMFFile` class.

1.5.5. Visualization

The saved XDMF file are directly compatible and can be easily visualized using ParaView. The user interface is very intuitive and once a file is opened it allows the user to select the solution fields to import into memory. Once the datasets are imported, it renders the volumetric data on the viewer. The toolbar offers some commonly used filters and are also accessible from the menu options. Initially, the dataset is bifurcated into truncated fluid domain and the PML. This can be achieved using the `ExtractCellsByRegion` filter used with its 'box' configuration scaled appropriately to omit the cells in the PML. To obtain cross-sections of the volumetric data, as shown in Fig. 1.6, the filter `Clip` or `Slice` is used and the specifications of the cutting plane is provided. It is also possible to compute from the saved fields on the interface directly by using the filter `Calculator`. This filter allows the user to define an expression using the fields in memory and compute a derived field. It is useful in computing the total displacement field putting together the real and imaginary portions. The same filter can also be used to compute the errors provided the exact solution is also in memory. Figure 1.7 shows the contours of obtained error fields on a plane passing through the origin. ParaView offers a lot more features like plotting, thresholding etc. which can help the user gain further understanding from the simulation data. Certain computations like pressure-at-a-point and directivity computations are better handled in the solver stage and can then be visualized using other Python graphing libraries.



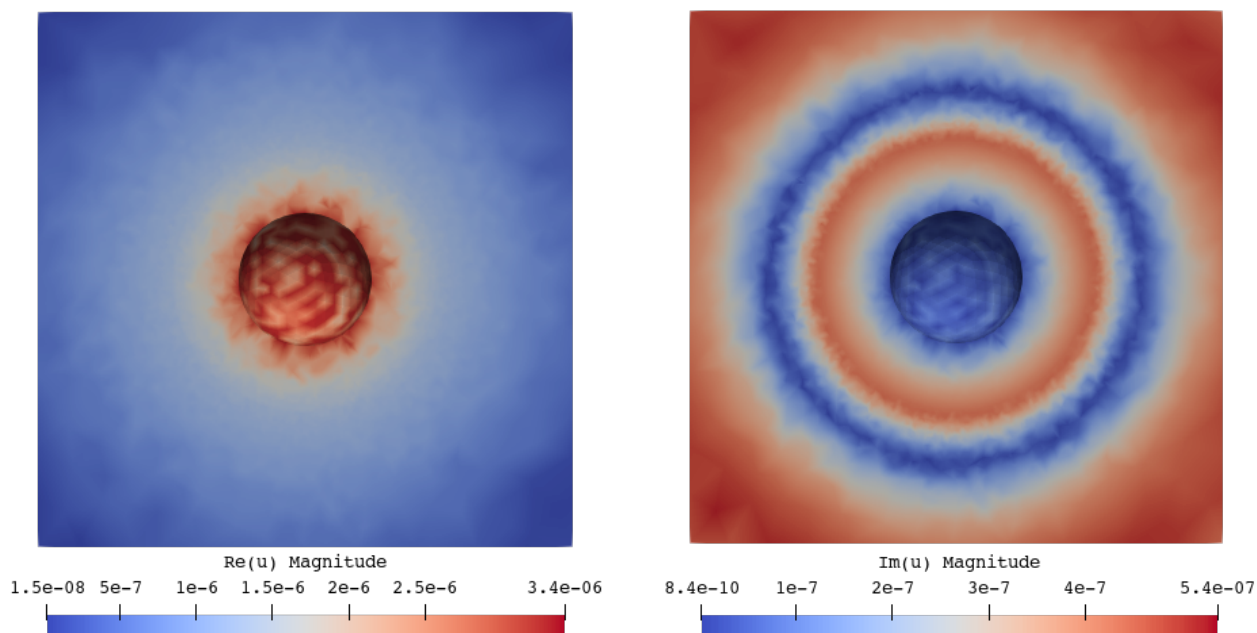


Figure 1.6: Cross-sectional contour views of the magnitude of the real part (left) and the magnitude of the imaginary part (right) of the computed displacement field.

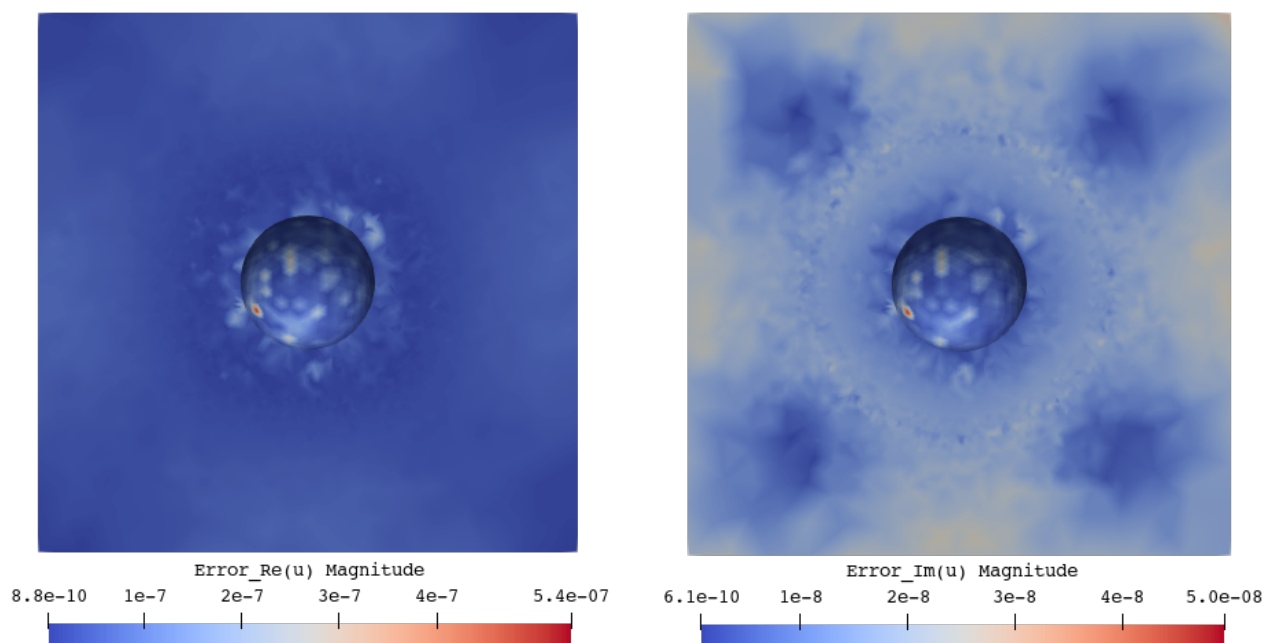


Figure 1.7: Cross-sectional contour views of the errors in the magnitude of the real part (left) and magnitude of the imaginary part (right) of the displacement field.

1.6. Conclusion

The objective of developing a coupled model for fluid-porous media interactions in designing an acoustic sensor is approached in stages. One such benchmark case with a porous coupling under no-flow conditions has been highlighted and a model has been developed. The porous layer is represented in the model by a fluid-equivalent

layer with absorption properties and certain interface conditions. The material properties of the porous medium is modeled by a fairly detailed 6-parameter Johnson-Champoux-Allard-Lafarge (JCAL) model. The perfectly-matched layers (PML) technique is also utilized to replicate the behaviour of an unbounded domain.

A comprehensive implementation of the model is provided in the article entirely using open-source software. The scripting interface interlace with the graphical interface for mesh generation provided by SALOME is useful in meshing complex geometry and automating the procedure either entirely or partly. The intricate finite element machinery of the solver stage is handled by the FEniCS library offering ease-of-use to the user while helping focus their attention towards model development and prototyping. The visualization tool, ParaView is feature-rich, providing access to many post-processing algorithms along with an intuitive interface. The entire toolchain can also be controlled on Python scripts allowing for easier development and future customization.

The workflow is meticulously described in solving for acoustic transmission of a vibrating sphere within a porous enclosure. The implementation of this benchmark case sheds light into every detail on running a simulation and obtaining and visualizing results, and hopes to enable the reader to reproduce the results in its entirety.

Bibliography

- [1] N. Auer, P. Barral, J.-D. Benamou, D. F. Comesaña, M. Girfoglio, L. Hauberg-Lotte, M. Hintermüller, W. Ijzerman, K. Knall, P. Maass, G. Marconi, M. Martinolli, P. P. Monticone, U. Morelli, A. Nayak, L. Polverelli, A. Prieto, P. Quintela, R. Ramlau, C. Riccardo, G. Rozza, G. Rukhaia, N. Shah, B. Stadler, and C. Vergara, “Reports about 8 selected benchmark cases of model hierarchies,” Sep. 2018, project Deliverable D5.1. [Online]. Available: <https://doi.org/10.14279/depositonce-7412>
- [2] S. Marburg and B. Nolte, *Computational Acoustics of Noise Propagation in Fluids - Finite and Boundary Element Methods*. Springer-Verlag Berlin Heidelberg, 2008.
- [3] J.-P. Berenger, “A perfectly matched layer for the absorption of electromagnetic waves,” *Journal of Computational Physics*, vol. 114, no. 2, pp. 185 – 200, 1994.
- [4] X. Sagartzazu, L. Hervella-Nieto, and J. M. Pagalday, “Review in sound absorbing materials,” *Archives of Computational Methods in Engineering*, vol. 15, no. 3, pp. 311–342, Sep 2008.
- [5] J. Allard and N. Atalla, *Propagation of Sound in Porous Media: Modelling Sound Absorbing Materials 2e*. Wiley, 2009.
- [6] A. Bermúdez, L. Hervella-Nieto, A. Prieto, and R. Rodríguez, “Perfectly matched layers for time-harmonic second order elliptic problems,” *Archives of Computational Methods in Engineering*, vol. 17, no. 1, pp. 77–107, Mar 2010.
- [7] A. Bermúdez, L. Hervella-Nieto, A. Prieto, and R. Rodríguez, “An optimal perfectly matched layer with unbounded absorbing function for time-harmonic acoustic scattering problems,” *Journal of Computational Physics*, vol. 223, no. 2, pp. 469 – 488, 2007. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0021999106004487>
- [8] “SALOME documentation.” [Online]. Available: <https://www.salome-platform.org/user-section/documentation/current-release>
- [9] H. P. Langtangen and A. Logg, *Solving PDEs in Python*. Springer, 2017.
- [10] “FEniCS documentation.” [Online]. Available: <https://fenicsproject.org/documentation/>
- [11] J. D. Hunter, “Matplotlib: A 2D graphics environment,” *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.
- [12] U. Ayachit, *The ParaView Guide: A Parallel Visualization Application*. USA: Kitware, Inc., 2015.
- [13] “ParaView tutorials.” [Online]. Available: https://www.paraview.org/Wiki/The_ParaView_Tutorial
- [14] “FEConv project page.” [Online]. Available: <http://victorsndvg.github.io/FEconv/>
- [15] “UFL documentation.” [Online]. Available: <https://fenics.readthedocs.io/projects/ufl/en/latest/>



2. Multirate time integration and model order reduction for coupled thermal electrical systems

Marcus W.F.M. Bannenberg¹, Michael Günther¹

¹*Bergische Universität Wuppertal*

Abstract. The coupled multiphysics systems arising in circuit simulation are both high dimensional and exhibit different internal time scales. These two properties can be exploited by numerical techniques. The high dimensional systems can be reduced by model order reduction. Then the different intrinsic time scales are efficiently handled by a multirate time integration scheme. The combination of these techniques is applied to coupled differential-algebraic circuit equations and a nonlinear thermal system. The result is a significant decrease in the computational effort.

Keywords: Model Order Reduction, Multirate, Differential Algebraic Equations, Coupled Systems, Circuit Simulation.

2.1. Introduction

In the simulation of nanoelectronics there are a great many things to consider. Trying to accurately model the natural phenomena happening inside a microchip leads to very large coupled systems. Which can become unfeasible to solve numerically. As specific type of coupled system this project focuses on systems with different intrinsic time scales i.e. due to thermal-electric coupling. The objective of this project is to combine multirate time integration and model order reduction to drastically improve the simulation speed. By using MR one can exploit the different intrinsic time scales of subsystems as to improve the overall computation efficiency, whilst preserving a level of global accuracy. MOR aims to reduce the size of large subsystems and decrease the numerical effort for each time step, again within a certain level of global accuracy. The outline of the report is as follows: First the benchmark problem is described. Next a mathematical definition of both techniques is given, followed by a detailed description of the implementation. Then to demonstrate the effect of this combination this new approach will be applied to generate a numerical example.

2.1.1. Thermal-Electric Benchmark Circuit

For this benchmark a test circuit is needed which contains both coupling and different intrinsic time scales. To this end the thermal-electric test circuit as described in [1] is used, Figure 2.1.

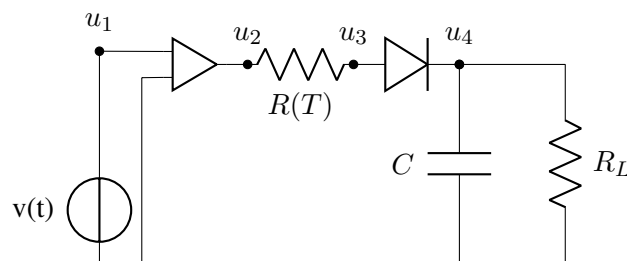


Figure 2.1: Electric description of the benchmark circuit.

This circuit consists of an operational amplifier, two resistors, a diode and a capacitor. The resistor $R(T)$ produces and transports heat and is temperature dependent. The amplifier is a heat source and the diode has a temperature dependent characteristic curve. The electric behaviour of the circuit is modelled by nodal analysis



yielding from Kirchhoff's laws. Following the discretisation as done in the original paper we arrive at the following thermal-electric system:

Electric network:

$$0 = (Av(t) - u_3)/R(T) + i_{di}(u_3 - u_4, T_{di}),$$

$$C\dot{u}_4 = i_{di}(u_3 - u_4, T_{di}) - u_4/R_L,$$

Coupling interfaces:

$$P_{op} = |(v_{op} - |v(t)|) \cdot (Av(t) - u_3)/R|, \quad P_w = (Av(t) - u_3)^2/R,$$

$$R(T) = \left(\frac{1}{2}(\rho(0, T_0) + \sum_{i=1}^{N-1} \rho(X_i, T_i) + \frac{1}{2}\rho(l, T_N)) \right) \cdot h,$$

Heat equation:

$$M'_{w,i} h \dot{T}_i = \Lambda \frac{T_{i+1} - 2T_i + T_{i-1}}{h} + P_w \frac{\tilde{\rho}(X_i, T_i)}{R} h,$$

$$- \gamma S'_{w,i} h (T_i - T_{env}), \quad (i = 1, \dots, N-1),$$

$$(M'_{w,0} \cdot \frac{h}{2} + M_{op}) \dot{T}_0 = \Lambda \frac{T_1 - T_0}{h} + P_w \frac{\tilde{\rho}(0, T_0)}{R} \frac{h}{2}$$

$$- \gamma (S'_{w,0} \frac{h}{2} + S_{op}) \cdot (T_0 - T_{env}) + P_{op},$$

$$(M'_{w,N} \cdot \frac{h}{2} + M_{di}) \dot{T}_N = \Lambda \frac{T_{N-1} - T_N}{h} + P_w \frac{\tilde{\rho}(X_N, T_N)}{R} \frac{h}{2}$$

$$- \gamma (S'_{w,N} \frac{h}{2} + S_{di}) \cdot (T_N - T_{env})$$

2.2. Mathematical Formulation

2.2.1. Multirate time integration

To efficiently integrate the thermal-electric system through time we use a multirate scheme, [2]. First consider a more general notation of the thermal-electric system:

$$M_e \dot{\mathbf{u}} = f_e(t, \mathbf{u}, \mathbf{T}),$$

$$M_t \dot{\mathbf{T}} = f_t(t, \mathbf{u}, \mathbf{T}).$$

We can distinguish a natural partition by splitting the system into thermal and electric equations. Note that the coupling between the two systems is handled by a coupling interface which is omitted from the general notation. Now the multirate scheme starts by integrating the whole system with a macro-step H . This yields solutions for time $t_0 + H$. From these solutions only one for the slow changing subsystem is accepted. Then the fast changing subsystem is integrated from t_0 to $t_0 + H$ with micro-steps h . For the coupling linearly interpolated values from the slow subsystem are used. As the system total system is an index 1 DAE it can be integrated by implicit Runge-Kutta schemes, following the approach as outlined in chapter VI of [3]. In this report the *stiffly accurate* implicit Runge-Kutta schemes are used as means to use the *State Space Form Method*.



This results in the system, where A and c are obtained from the Butcher Tableau,

$$\mathbf{k}^u = \begin{cases} \mathbf{u}_0 + H \cdot f(t_0 + c \cdot H, \mathbf{k}^u, \mathbf{k}^T) \cdot A & \text{Differential,} \\ f(t_0 + c \cdot H, \mathbf{k}^u, \mathbf{k}^T) & \text{Algebraic,} \end{cases}$$

$$\mathbf{k}^T = \mathbf{T}_0 + H \cdot f(t_0 + c \cdot H, \mathbf{k}^u, \mathbf{k}^T) \cdot A,$$

Which has to be solved with respect to \mathbf{k}^u and \mathbf{k}^T . For the macro-step the complete system is solved. Then for the intermediate micro-steps the system is solved with interpolated values $\bar{\mathbf{k}}^u$ for \mathbf{k}^u in the coupling interface. To solve this system different methods are available. In this implementation the multidimensional root finder algorithm by GSL is used for this system of nonlinear equations, [4].

2.2.2. Model order reduction

As the discretised number of thermal equations can be made arbitrarily large model order reduction is applied to this part of the system. To avoid linearisation of the thermal part the proper orthogonal decomposition (POD) method is chosen as it can handle nonlinear systems. Following [5], POD starts out with a snapshot matrix $\mathbf{X} = [\mathbf{x}(t_0), \dots, \mathbf{x}(t_{N_s})] \in \mathbb{R}^{n \times N_s}$ of collected state evolutions of the whole system. From this snapshot matrix an orthonormal basis \mathbf{V} is derived. This is done by computing the singular value decomposition of \mathbf{X} :

$$\mathbf{X} = \mathbf{U}\Sigma\mathbf{T}^T$$

From the scaled singular values matrix $\Sigma = \text{diag}(\sigma_1, \dots, \sigma_n) / \sum \sigma_i$ the degree of truncation r can be obtained by a threshold value ϵ . To construct basis \mathbf{V} , POD chooses the left singular vectors corresponding to the r largest singular values.

$$\mathbf{V} = [\mathbf{u}_1, \dots, \mathbf{u}_r].$$

With this orthonormal basis the reduced system is given by

$$\mathbf{W}^T \mathbf{M}_t \mathbf{V} \dot{\tilde{\mathbf{T}}} = \mathbf{W}^T f_t(t, \mathbf{u}, \mathbf{V} \tilde{\mathbf{T}}).$$

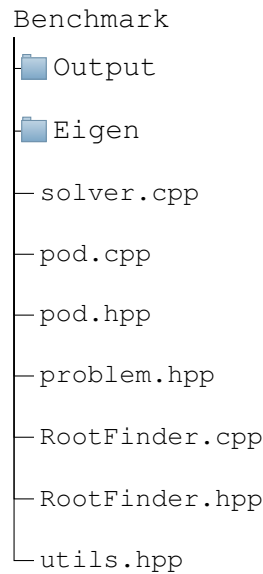
And thus the total system with the reduced thermal subsystem is

$$\begin{aligned} \mathbf{M}_e \dot{\mathbf{u}} &= f_e(t, \mathbf{u}, \mathbf{V} \tilde{\mathbf{T}}), \\ \mathbf{W}^T \mathbf{M}_t \mathbf{V} \dot{\tilde{\mathbf{T}}} &= \mathbf{W}^T f_t(t, \mathbf{u}, \mathbf{V} \tilde{\mathbf{T}}). \end{aligned}$$

2.3. Implementation

The code for the implementation of the problem sketched in the introduction is written in C++. In this section of the report an overview of the structure of the software is given.





2.3.1. Parameters and functions

In the Benchmark folder the main function is in the file `solver.cpp`, compiled this results in `main` that can to be run. In this file first all the numerical parameters of the benchmark problem are declared. These are declared as global variables for accessibility and for single allocation purposes. The problem specific parameters are declared in the file `problem.hpp` which is included in `solver.cpp`, see below. After the numerical parameter declaration the objective function declaration takes place. As these are linked to the numerical scheme they are declared in `solver.cpp` and use the problem specific functions from `problem.hpp`. Following the declaration style of `solver.cpp` the thermal and circuit parameter declarations in `problem.hpp` are defined as globals. Then the following functions are defined:

- `v, i_di, rho, a, S_prime`: Declared for their respective functions in the system.
- `coupling_update`: Function for updating the coupling parameters. Note that there are multiple versions depending on the integration rate.
- `f`: Function values of the DAEs, again there are several depending on activity and MOR.

2.3.2. Setting up the IRK iteration

In the main function, `solver.cpp`, first the fifth order Radau IIA Butcher Tableau is set up by filling matrix `A` and vector `c`. Then index arrays, `iA` and `iD`, for the algebraic and dynamic equations are filled. Next the fast and slow index arrays are constructed and their local counterparts, `indices_A/L` and `iA/D_local`. This is done for ease of index access in the `objective_f_A` function. The solution destination array, `y_tot`, is then setup and filled with the initial conditions, `y_0`, of this specific problem. Then the IRK iteration is started. For the IRK iteration a `RootFinder` object is used to solve the k_i . This solver is encapsulated in its own class.

2.3.3. The IRK iteration

In this `for`-loop the computation of the solution for each time-step takes place. As the goal is to use the multirate IRK loop this is the one that will be described in the section below. The loop starts with initialising the seed of the solver, for which the current state is used. Then a macro-step of m micro steps is performed. For this large step size the solution is calculated and stored in the the destination matrix. Then the interpolated m values and if needed extrapolated values are constructed by linear interpolation of the current state value and the received solution. Note that this is only done for the latent values. Then the internal loop for the m micro-steps is started. This begins again with the construction of a seed vector for the `RootFinder` object. However this time



the seed is only the size of the active dimension. The needed interpolated values is stored in the global array to be accessible for the objective function. The active subsystem is then solved for the seed value and the results are stored in the destination matrix. Then the time and global state values are updated.

2.3.4. The POD algorithm

To encapsulate this process this functionality is stored in the POD class as defined by `pod.cpp` and `pod.hpp`. The class is first initiated with the desired reduction number as to allocate global storage arrays. Then once the snapshot matrix is obtained from the solution matrix the POD object is updated. The update starts by constructing the SVD of the snapshot matrix. This is done by a EIGEN subroutine, for which two types can be chosen. Note that one of them is commented in the file. After the SVD the necessary matrices are constructed and stored in the POD object for ease of access.

2.4. Requirements

The program is compiled by the running the `make` command inside the Benchmark folder. For this the following libraries are used: `-lgsl -lgslcblas`. Furthermore the folder of the 3.3.4 version of third party library Eigen, [6], must be included in the Benchmark folder. The program is currently compiled with `g++` version 4.2.1. Note that the `-O3` optimiser flag is used for a faster runtime. The specs of the computer which is used for the numerical examples are:

- 2,6 GHz Intel Core i5.
- 8 GB 1600 Mhz DDR3.

2.5. Numerical examples

To get an impression of the impact of applying model order reduction and multirate time integrating a numerical experiment is done. For this test case the following simulation parameter values are chosen:

$$\begin{array}{ll}
 t_0 = 0 & t_{end} = 0.025, \\
 H_{sr} = 1.25e - 4 & H_{mr} = 2.5e - 4, \\
 h_{mr} = 2.5e - 5 & m = 10, \\
 N = 100 & r = 8,
 \end{array}$$

Butcher Tableau

$\frac{2}{5} - \frac{\sqrt{6}}{10}$	$\frac{11}{45} - \frac{7\sqrt{6}}{360}$	$-\frac{37}{225} - \frac{169\sqrt{6}}{1800}$	$-\frac{2}{225} + \frac{\sqrt{6}}{75}$
$\frac{2}{5} + \frac{\sqrt{6}}{10}$	$-\frac{37}{225} + \frac{169\sqrt{6}}{1800}$	$\frac{11}{45} + \frac{7\sqrt{6}}{360}$	$-\frac{2}{225} - \frac{\sqrt{6}}{75}$
1	$\frac{4}{9} - \frac{\sqrt{6}}{36}$	$\frac{4}{9} + \frac{\sqrt{6}}{36}$	$\frac{1}{9}$
	$\frac{4}{9} - \frac{\sqrt{6}}{36}$	$\frac{4}{9} + \frac{\sqrt{6}}{36}$	$\frac{1}{9}$

The experiment is setup in following way. First the full system is simulated with single rate time integration. Then the full system is simulated with multirate time integration. The singlerate solution is then used to construct the snapshot matrix for the POD procedure. Then lastly the reduced system is simulated using multirate time integration.



2.5.1. Results

To give an impression of the solution of the circuit a reference solution is shown in Figure 2.2. This solution is obtained by a very coarse time and spatial discretisation and using the singlerate time integration. Shown is the desired output, the voltage at u_3 and the development of the heat in the middle voxel of the thermal resistor.

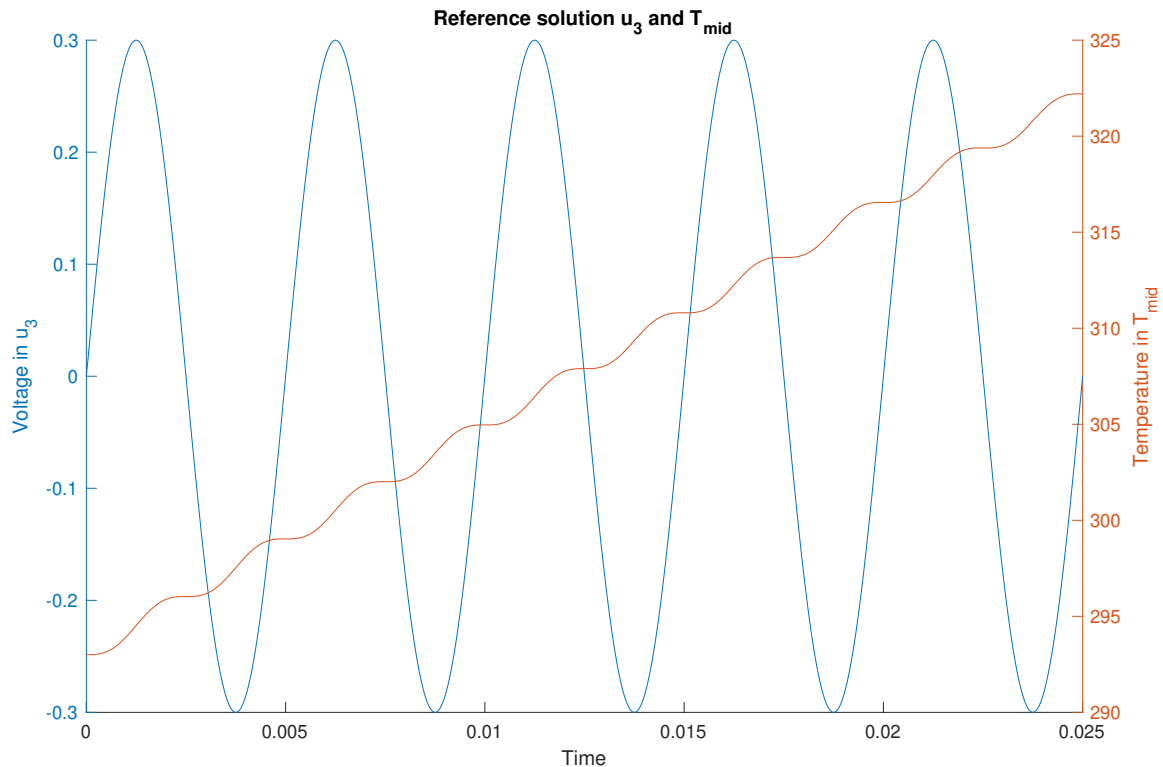


Figure 2.2: Reference solution

To measure the impact of the multirate time integration and model order reduction the error between the reference solution and the experiment solution is defined as the difference between the voltage at node u_3 of both solutions, as this would be the desired output of the circuit simulation. In Figure 2.3 the results of the experiments are shown. First we see that the solutions for u_3 overlap nicely and thus that the simulations are correct. Then below the first plot the error introduced by the singlerate and multirate time integration techniques are shown for the full system, the second and third plot. We see that although the error in the multirate plot is slightly higher, the errors are within the same magnitude. Then the fourth and final plot illustrates the error induced by the multirate time integration and POD methods combined. Here we see that due to the POD reduction the error is increased and more irregular. However, the error does not stray from the previous errors' magnitudes.

Lastly, in Table 2.1 the computation time of the three different approaches is shown. These times have been measured for 20 runs and then the average is shown. It now shows that the multirate approach roughly dou-

	SR	MR	MR+MOR
Time	54.8243	28.3296	4.0325

Table 2.1: Computation time of the solver.

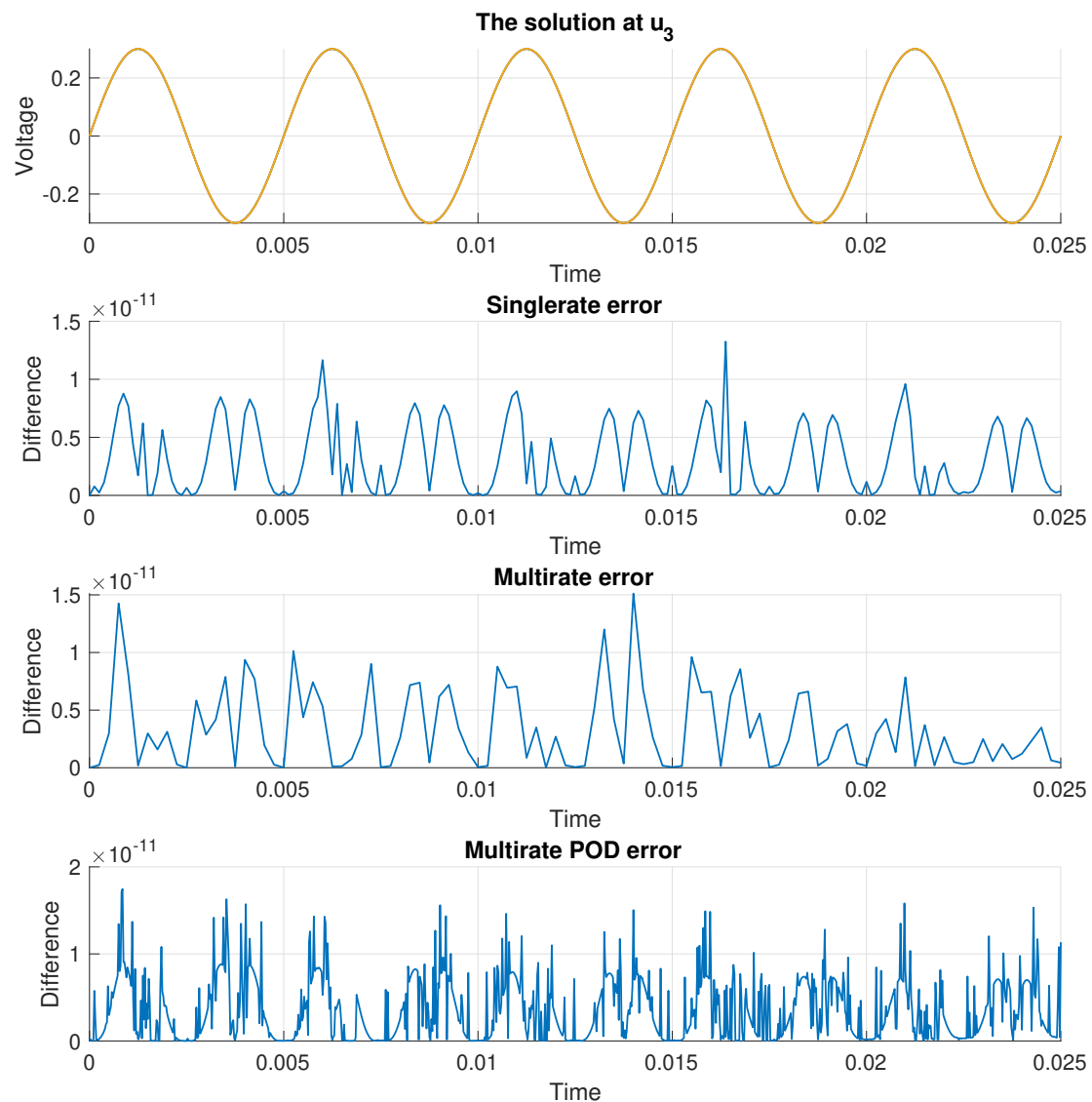


Figure 2.3: The results of the simulations

bles the integration speed whilst maintaining accurate. If this is combined with the POD the solver is almost fourteen times as fast for the same error magnitude.

2.6. Conclusion

From the numerical example it shows that the combination of multirate time integration and model order reduction looks quite promising. A decrease of computation time can be seen by applying the two methods whilst the accuracy is preserved. These preliminary results are a positive indicator for further research. As only one type of nonlinear model order reduction has been introduced in this benchmark there is much to gain in this territory. Furthermore different types of coupling need to be elaborated on.



Bibliography

- [1] A. Bartel, M. Günther, and M. Schulz, “Modeling and discretization of a thermal-electric test circuit,” in *Modeling, Simulation, and Optimization of Integrated Circuits*. Springer, 2003, pp. 187–201.
- [2] M. Günther, A. Kvaernø, and P. Rentrop, “Multirate partitioned runge-kutta methods,” *BIT Numerical Mathematics*, vol. 41, no. 3, pp. 504–514, 2001.
- [3] G. Wanner and E. Hairer, *Solving ordinary differential equations II*. Springer Berlin Heidelberg, 1996.
- [4] B. Gough, *GNU scientific library reference manual*. Network Theory Ltd., 2009.
- [5] A. Verhoeven, J. Ter Maten, M. Striebel, and R. Mattheij, “Model order reduction for nonlinear ic models,” in *IFIP Conference on System Modeling and Optimization*. Springer, 2007, pp. 476–491.
- [6] G. Guennebaud, B. Jacob *et al.*, “Eigen v3,” <http://eigen.tuxfamily.org>, 2010.



3. Validation of fluid-structure interaction simulations in membrane-based blood pumps

Marco Martinolli¹, Christian Vergara¹, Luc Polverelli²

¹*MOX, Dipartimento di Matematica, PoliMi*

²*CorWave Inc.*

Abstract. The benchmark consists in the validation of a numerical model for the fluid-structure interaction arising in membrane-based blood pumps against experimental data obtained by in vitro testings at CorWave Inc. The goal is to numerically reproduce the pump system under the same working conditions of the documented experimental sessions, in order to predict the pump outflow rates and the hydraulic power for different pressure conditions in the pump and finally compare the results with the experimental HQ curves. The software for the solution of the benchmark will be implemented in the C++ parallel library of finite elements LIFEV using the Extended Finite Element Method. The report includes the plan for the Docker installation of the LIFEV environment and the third-part packages, information on the future online availability of the software, the licences of use of the main libraries and the computer requirements to run the benchmark.

Keywords: Fluid-structure interaction, blood pumps, model validation, HQ curves.

3.1. Introduction

Blood pumps are medical devices used to support cardiac function in patients affected by end-stage heart failure. These devices are implanted at the apex of the heart, where they expel the oxygenated blood collected in the left ventricle into the ascending aorta via a flexible outlet cannula. Hence, these devices are called Left Ventricular Assist Devices (LVADs). In this case, we will focus on a novel prototype of blood pumps that is under development at CorWave Inc.(Paris), said progressive wave blood pumps [1]. Progressive wave blood pumps are based on the interaction between an undulating elastic membrane and the blood, resulting in a pumping mechanism that ejects blood in a physiologic pulsatile regime without exerting high forces on the blood cells [2, 3]. In the left panel of Figure 3.1, we show the cross sectional view of the pump device that works as follows: the electromagnetic actuator at the center drives the oscillatory motion of the mobile magnet ring; the latter transfers the motion to the most external part of the membrane; as a consequence, the motion propagates along the elastic membrane causing a wave-like displacement that propels the blood, coming from the inlet channel, towards the outlet. Thanks to this progressive wave propagation, the membrane can push and transport the blood without causing blood trauma (unlike standard rotary blood pumps). Notice that, being a pump system, the pressure at the outlet is higher than at the inlet; therefore the role of the wave membrane is to win the adverse pressure gradient acting across the pump. The possibility to simulate this complex dynamics inside the pump allows to predict the device performance under different operating conditions (changing parameters like the amplitude or the frequency of the membrane oscillation or the velocity of the blood entering in the device) and in different pump designs (varying geometrical features, like membrane diameter or thickness, or inlet/outlet section). In addition, computational simulations can be used to predict blood adverse events, like hemolysis or thrombosis, and improve the hemocompatibility of the device.

Hence, from the mathematical point of view, the problem at hand is a time-dependant Fluid-Structure Interaction (FSI) problem with two moving structures: the wave membrane and the magnet ring. Therefore, we need to study the mutual interaction dynamics between the blood flow and the two structures. In particular, we are interested to check computationally the membrane wave pumping technology under different working conditions of the pump device. The operating point of the pump is identified by three parameters: (i) the excitation frequency f of the magnet ring and the membrane; (ii) the amplitude Φ of their oscillations; and (iii)



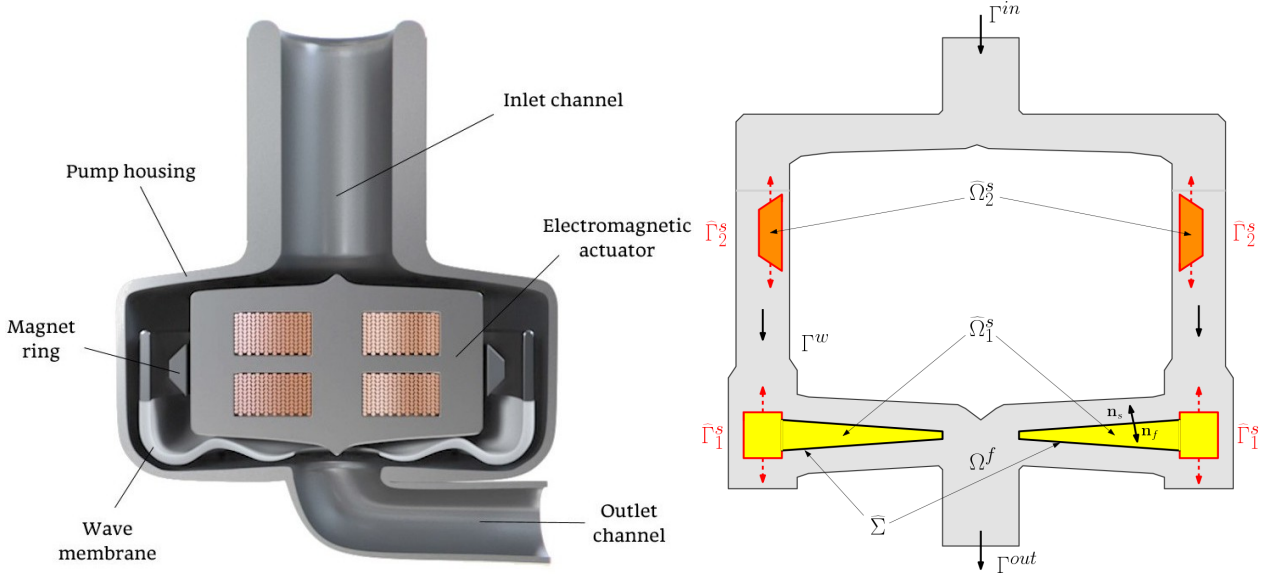


Figure 3.1: Left: Cross section of the implantable progressive wave pump. The blood enters from the superior inlet channel, it flows down along the sides of the central actuator body, it interacts dynamically with the wave silicon membrane and it is ejected into the inferior outlet channel. Right: Sketch of the pump domain.

the pressure difference $\Delta P < 0$ between the pump inlet and the outlet.

However, notice that the solution of this problem can be very tricky. In particular, during the wave motion of the membrane, the structure approaches very closely to the pump housing, potentially reaching a contact configuration with the surrounding pump walls. Therefore, our solution for the benchmark needs to address the possible conditions and the numerical issues arising during the mesh deformation in standard fitted methods for FSI problems. In the next section, we will propose a possible solution to these problems, based on the unfitted Extended Finite Element Method (XFEM).

The benchmark consists in the validation of a numerical method to solve the FSI problem in membrane-based blood pumps against real hydraulic data provided by CorWave Inc. Specifically, the goal is to predict the outflow volume rate Q^{sim} given a certain operating point of the pump $(f, \Phi, \Delta P)$ and compare the numerical result with the measured flow data Q^{data} .

The remaining of the report is structured as follows: in Section 3.2, we describe the FSI model, with a focus on its hierarchical features, and we propose a numerical strategy to solve the benchmark; in Section 3.3, we detail the required numerical libraries and software packages and the installation concept via Docker; in Section 3.4, we report the discuss the technical details to run the benchmark, such as environment settings, hardware and software requirements; and finally in Section 3.5, we present the structure and the format of the experimental data provided by CorWave Inc. (3.5.1), we depict the plan for the benchmark for the model validation (3.5.2). Licences of use and configuration files can be found in the Appendix 3.A.

3.2. Mathematical Formulation

3.2.1. The Fluid-Structure Interaction Model

The intertwined dynamics arising inside a progressive wave pumps can be mathematically described in the framework of FSI modeling, where a system of Partial Differential Equations (PDEs) describes separately the behaviour of the fluid and of the structure in the respective domains, while proper coupling conditions define



their interaction at the interface. In the following, we simplify the formulation of the problem by considering a unique structural domain $\Omega^s = \Omega_1^s \cup \Omega_2^s$, where Ω_1^s is the membrane domain and Ω_2^s is the magnet ring domain.

The FSI problem reads as follows: for each time $t > 0$, find fluid velocity and pressure $(\mathbf{u}(t), p(t))$ in the fluid domain $\Omega^f(t)$ and membrane displacement $\hat{\mathbf{d}}(t)$ in the reference structure domain $\hat{\Omega}^s = \Omega^s(0)$, such that:

$$\rho_f (\partial_t \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u}) - \nabla \cdot \mathbf{T}^f(\mathbf{u}, p) = \mathbf{0} \quad \text{in } \Omega^f(\mathbf{d}^f), \quad (3.1)$$

$$\nabla \cdot \mathbf{u} = 0 \quad \text{in } \Omega^f(\mathbf{d}^f), \quad (3.2)$$

$$\rho_s \partial_{tt} \hat{\mathbf{d}} - \nabla \cdot \hat{\mathbf{T}}^s(\hat{\mathbf{d}}) = \mathbf{0} \quad \text{in } \hat{\Omega}^s, \quad (3.3)$$

$$\mathbf{d}^f = \mathbf{d} \quad \text{on } \Sigma(\mathbf{d}^f), \quad (3.4)$$

$$\mathbf{u} = \partial_t \mathbf{d} \quad \text{on } \Sigma(\mathbf{d}^f), \quad (3.5)$$

$$\mathbf{T}^f(\mathbf{u}, p) \mathbf{n} = \mathbf{T}^s(\mathbf{d}) \mathbf{n} \quad \text{on } \Sigma(\mathbf{d}^f). \quad (3.6)$$

with proper boundary and initial conditions. Equations (3.1) and (3.2) are the Navier-Stokes (NS) equations modeling the conservation of momentum and mass of an incompressible viscous Newtonian fluid. Equation (3.3), written in the Lagrangian formulation with the quantities defined in the reference configuration ($\hat{\cdot}$ supercript), is the elastodynamic equation that models the structure deformation. Equation (3.4) provides the adherence condition between the fluid and the solid domains, and Equations (3.5) and (3.6) guarantee the continuity of velocity and of stresses respectively at the fluid-structure interface Σ . Notice that both the fluid domain Ω^f and the interface Σ depend over time on the fluid displacement \mathbf{d}^f and thus on the structure displacement \mathbf{d} due to the geometric condition (3.4). In particular, the physical quantities presented in the mathematical formulation are:

- ρ_f and ρ_s are the mass densities of fluid and structure;
- \mathbf{n}^f and \mathbf{n}^s are the external normal vectors from fluid and structure domains, satisfying $\mathbf{n}^f = -\mathbf{n}^s = \mathbf{n}$;
- $\mathbf{T}^f(\mathbf{u}, p) = -p\mathbf{I} + 2\mu_f \mathbf{D}(\mathbf{u})$ is the Cauchy stress tensor for a viscous Newtonian fluid with dynamic viscosity μ_f and symmetric operator $\mathbf{D}(\mathbf{w}) = \frac{1}{2}(\nabla \mathbf{w} + \nabla \mathbf{w}^T)$;
- $\hat{\mathbf{T}}^s(\hat{\mathbf{d}})$ is the first Piola-Kirchhoff tensor of the structure, such that $\hat{\mathbf{T}}^s(\hat{\mathbf{d}}) = J \mathbf{T}^s(\mathbf{d}) \mathbf{F}^{-T}$ with \mathbf{T}^s being the solid Cauchy stress tensor depending on the constitutive law of the structure, $\mathbf{F} = \nabla \mathbf{x}$ the gradient of deformation and $J = \det \mathbf{F}$ its determinant. According with previous works [4], we assume linear elasticity for both structures, so that we can use the Hooke Law and write the Cauchy tensor as $\mathbf{T}^s(\mathbf{d}) = \lambda^s (\nabla \cdot \mathbf{d}) \mathbf{I} + 2\mu^s \mathbf{D}(\mathbf{d})$, where λ^s and μ^s are the Lamé parameters for each material.

In the right panel of Figure 3.1, we see the mathematical domain of the blood pump. We specify that the geometrical differences with respect to the view in the right panel of Figure 3.1 are due to the fact that we work on an alternative design of the pump. In addition we have omitted the rigid titanium apparatus that connects the magnet ring to the membrane, because it is not significant for our analysis. Indeed, we apply the oscillatory motion imposed by the actuator directly on the structure boundaries Γ_1^s and Γ_2^s . Hence, the imposed oscillations are modeled by means of a sinusoidal Dirichlet conditions on $\Gamma^s = \Gamma_1^s \cup \Gamma_2^s$ as:

$$\mathbf{d}(t) = \Phi \sin(2\pi f t) \mathbf{e}_z \quad \text{on } \Gamma^s \quad (3.7)$$

where Φ and f are respectively the amplitude and the frequency of the forced oscillation of both the membrane and the magnet ring.

In addition we have the boundary conditions for the fluid subproblem. We use a pair of Neumann conditions to prescribe the pressure conditions at the inlet Γ^{in} and at the outlet Γ^{out} of the pump domain and a non-slip



condition at the pump walls Γ^w . In fact, we have that:

$$\mathbf{T}^f(\mathbf{u}, p) \mathbf{n}^f = \Delta P \mathbf{n}^f \quad \text{on } \Gamma^{in}, \quad (3.8)$$

$$\mathbf{T}^f(\mathbf{u}, p) \mathbf{n}^f = \mathbf{0} \quad \text{on } \Gamma^{out}, \quad (3.9)$$

$$\mathbf{u} = \mathbf{0} \quad \text{on } \Gamma^w, \quad (3.10)$$

where $\Delta P < 0$ is the negative pressure parameter that defines the pressure difference between the inlet and the outlet (having set the zero pressure level at the outlet).

We finally consider null initial conditions for both velocity and displacement variables.

3.2.2. Numerical Method

In our solution of the benchmark, we will try to numerically reproduce the experimental results mimicking the same working conditions of the pump in the experimental setup and solving the mathematical problem above in three dimensions (3D) using the XFEM [5, 6]. XFEM is an unfitted technique which has two main advantages compared with other approaches for FSI problems: i) since the fluid mesh is kept fixed on the background, it avoids the remeshing procedure normally occurring in case of element distortion; ii) the accuracy of the solution is maintained at the interface, thanks to the local enrichment of the functional space of the extended finite elements. However, since the structure mesh moves in the foreground cutting the underlying fluid mesh, XFEM requires to compute the mesh intersections at each time instant to identify the fluid elements that are cut in multiple subportions (called *split elements*), leading to a higher computational cost. For more details on the numerical formulation of the XFEM with a Discontinuous Galerkin (DG) mortaring at the interface, the reader can find an exhaustive explanation in the reference paper [7].

We linearize the fluid subproblem by using the semi-implicit approach for the convective term. In addition, in order to solve efficiently the geometric coupling, we take the first order extrapolation of the fluid domain from previous timestep. We use first order finite elements for all variables and we consider the Continuous Interior Penalty (CIP) stabilization to satisfy the inf-sup condition [8]. We also apply a ghost penalty stabilization to solve the spurious oscillations coming from XFEM, when the size of the split elements becomes too small [9].

The linear system obtained at each timestep after the XFEM-based discretization is solved with a monolithic approach with a GMRES solver preconditioned by a block Gauss-Seidel preconditioner.

3.2.3. Hierarchical Modeling

Due to the multi-component and multiphysics nature of our problem, the strong coupling of the physical system and the computational needs of the unfitted method [10], most of the times the solution of the fluid-structure dynamics can be very tricky and the complexity of the problem can represent a real barrier to in-depth analysis of the system.

However, in our application, there exist several modeling options, both at the geometric and numerical level, that can enable faster computations but at the cost of a decreased accuracy, and viceversa. In fact, the FSI model of our study is actually the result of a sequence of modeling options (e.g. geometry detail, modeling assumptions, numerical approximations, etc.), that act on distinct levels and that contribute in different ways to the global complexity of the model. Hence, the model is considered to be hierarchical. This means that we can tailor the model to fit the needs of our study by acting independently of multiple levels, requiring, for instance, a high level of detail of geometric features but relaxing specific numerical coupling conditions.

In the following, we provide the hierarchical levels that we identified in the FSI problem in progressive wave



blood pumps.

3.2.3.1. Geometry of the domain

The complete domain of the blood pump is composed by several subcomponents: the pump housing, the membrane assembly, the actuator, the magnet ring and the membrane frame (see Figure 3.1). It is clear that model all the components and their interaction can be very complicated and, in most cases, not necessary. Therefore, it is possible to reduce the complexity of the geometry of the domain by omitting the rigid components or either by simplifying the geometrical features of the system.

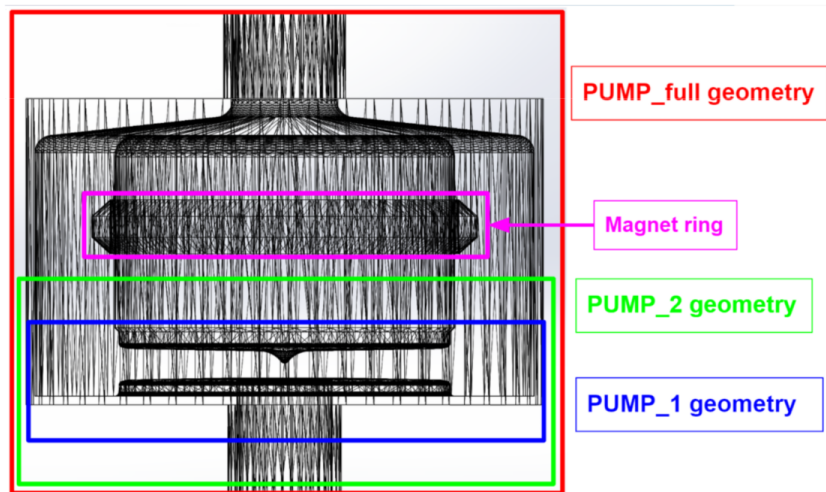


Figure 3.2: Different levels of geometric extensions of the blood pump domain. Blue and green boxes identify reduced pump domains; the red box identifies the full pump geometry, including the inlet and the outlet channels and the magnet ring (violet).

For instance, if we are interested in the study of the fluid-structure dynamics between the blood and the wave membrane, we can limit the computational domain to the so-called pump head region. Therefore, we can simply cut the domain limiting ourselves to the study of the lower part of the pump (see blue square in Figure 3.2). In this way, we heavily reduced the number of nodes and consequently the computational time for the simulations. Alternatively, we can reduce the geometric detail of the pump or the membrane, so that the mesh intersection step in the XFEM is easier and faster to solve.

3.2.3.2. Meshing

Another standard way to modulate the complexity of the problem is by working at the mesh step h of the computational domain. Indeed, you can either choose to have a coarse mesh (large mesh step h) to have fast simulations with small accuracy or finer mesh (small mesh step h) to get accurate results at higher computational cost. In fact, when you decrease the discretization step h , you automatically increase the number of degrees of freedom of the problem and consequently the dimension of the linear system. In case of unfitted methods, the choice of the discretization step is even more important, because it affects as well the number of fluid elements that are split by the structure mesh resulting in a higher complexity of the mesh intersection step.

In case of the multi-component systems like our blood pump, it is possible to make more sophisticated choices of the mesh step depending on the different properties of specific regions. In fact, instead of having a uniform mesh (same h in the whole domain), once can start from a coarse mesh and refine the meshing in the regions where we expect the dynamics are more complex or more interesting for our study - like in the proximity of the membrane or the magnet ring structures - or in regions that present geometrical peculiarities (like narrowing of the flow path or sharp angles), as shown in Figure 3.3.



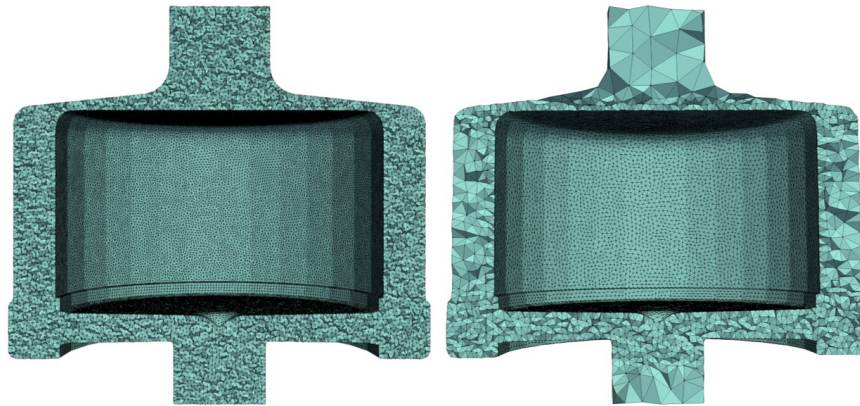


Figure 3.3: Different levels of meshing of the fluid mesh. Left: uniform meshing with small meshsize. Right: coarse fluid mesh with local refinement in the specific regions of interest.

3.2.3.3. Geometric coupling

In FSI problems we have two types of coupling conditions at the interface: physical and geometrical coupling conditions. Physical conditions impose the continuity of the velocities (kinematic condition) and of the normal stresses (dynamic condition) at the fluid-structure interface. The geometric condition requires that the adherence of the fluid and the solid domain is maintained during their motion in time, without the creation of inner holes. The latter coupling condition is normally formulated by asking that the displacement of the structure \mathbf{d} is equal to the fluid domain displacement \mathbf{d}_f in all the points of the interface (see Equation 3.4). We want to remind the reader that in the XFEM framework the fluid mesh is fixed on the background and the fluid domain corresponds to the sole region that is not overlapped by the structure mesh moving on the foreground. Therefore, the fluid domain changes in time according with \mathbf{d}_f , that depends directly on the motion of structure by means of the coupling condition.

In XFEM applications the satisfaction of the geometric adherence condition is actually a source of high computational costs. In fact, the fluid domain displacement \mathbf{d}_f has to be computed from the intersection of the moving structure mesh with the underlying fluid mesh. At each time instant, the structure moves in the space region overlapping the fluid mesh in different positions and thus the mesh intersection step has to be repeated at each iteration. The costs for this operation are definitely not negligible and they increase with the refinement of both meshes.

As a consequence, in our study we propose the so-called *fixed geometry approximation*, for which, in case of small displacement regime (\mathbf{d} small and bounded), we neglect the fluid domain displacement \mathbf{d}_f . This means that the fluid domain is actually not updated at each timestep, but it is approximated to remain fixed to its original configuration. Therefore, the mesh intersection step has to be performed only once during the first time iteration, saving a lot of computational time. We want to emphasize that the displacement of the structure \mathbf{d} is still computed and it is still active for the physical coupling conditions; we only neglect its effect as modifier of the fluid domain. Thus, the fixed geometry approximation can be a smart way in case of small displacement regime, to decrease the computational cost and avoid the additional complications in the splitting of the fluid elements coming from atypical geometrical configurations.

A second level of approximation relies on moving the fluid domain, but in an explicit way, by using extrapolation of information at previous time steps. This strategy will introduce an error with respect to the fully implicit geometric coupling, which is however smaller than that produced by the previous strategy, at the price of an increased computational cost.



3.2.3.4. Modeling the Contact

We consider to have contact when two solid bodies are sufficiently close so that their motions are mutually affected by each other. Therefore it is not strictly necessary that the two bodies touch each other to have contact and under some conditions proper impact cannot even occur. In our application, contact can occur between the moving membrane and the walls of the pump housing. Thus, only one of the two bodies is mobile, simplifying the implementation of the contact condition. Such contact-like situation is actually very important for a correct pumping performance for two reasons: i) near the contact region, the membrane wave generates a pressure gradient that propels the blood towards the outlet, ii) the contact between the membrane and the walls works as a valve that avoids the backflow from the outlet towards the inlet (see Figure 3.4).

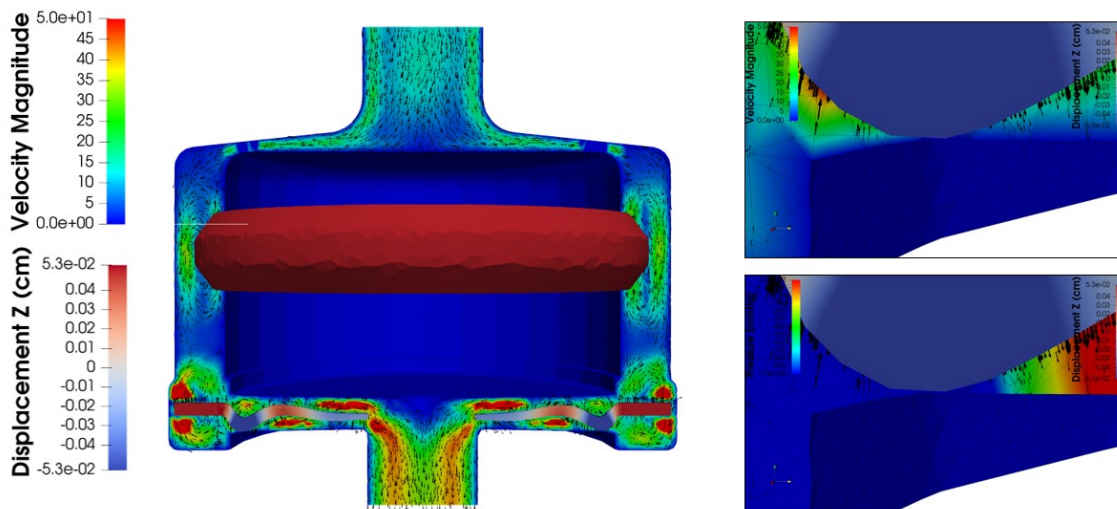


Figure 3.4: Detail of the contact point between the membrane and the lower pump wall. Left: velocity field in the whole pump domain. Right: zoom on the velocity (above) and pressure (below) fields in the contact area.

Therefore, in our study we can add an additional level complexity by investigating the contact between the membrane and the pump wall, finding a way to prohibit the penetration of the former in the latter.

In FSI problems, a possible approach to model the contact derives directly from the combination of non-slip coupling interface conditions and the incompressibility of the fluid. In fact, previous theoretical studies [11] proved that collisions between a body moving in a viscous incompressible fluid and a wall can never occur in finite time. This is justified by the fact that the incompressible fluid opposes a resistance to the motion of the structure with an intensity that increases more and more they get closer. Since there is no possibility to have tangential slip at the interface, then there will always remain a small layer of fluid separating the two bodies. This is confirmed by physical and numerical studies. This approach has the advantage that it is simple to implement, but it requires to have small discretization steps both in space and in time, in order that the model can capture the dynamics generated when the membrane approaches the wall.

Alternatively, we can relax the coupling conditions by allowing the tangential slip at the interface in order to have contact. In case of slip coupling conditions, the kinematic condition requires the continuity of the velocities only in normal direction, without constraints in the tangential one. Under these conditions, contact

can formally occur and we need to add a new term to the weak formulation acting on the portion of the boundary that is proximal to the wall [12]. This additional term represents the contact force exerted on the membrane in case of contact. This more sophisticated approach needs to identify the portion of the membrane boundary where to apply the contact force and calibrate the intensity of such force by tuning a penalty contact parameter.

As a result, we have identified four independent hierarchical levels of our FSI model, that can be set in order to build up simulations with a certain complexity, depending on the degree of accuracy or the computational time of our study. In our solution for this benchmark we decided to consider the full pump geometry because the experimental data referred to the whole pump domain, but we made small changes to the geometry, such as the omission of the rigid holder apparatus and of a small step in the most external region of the membrane. We considered a uniform mesh size for the background mesh, while for the membrane mesh we refined the mesh step h more and more approaching the tip region, where the membrane thickness is smaller. Concerning the geometric coupling we have solved it by taking the first order extrapolation of the fluid domain from the previous timestep. And finally for the contact subproblem, we assumed non-slip condition both at the interface and at the pump walls, in agreement with experimental observations in similar problems [13].

3.3. Implementation

The software for the benchmark will be implemented in L**IB**rary of F**IN**ite E**LE**ments V (LIFEV) (<https://bitbucket.org/lifev-dev/lifev-env.git>) [14], a C++ parallel finite element library for the solution of PDEs. In particular, the library is suitable for solving real problems in the field of cardiovascular applications. The XFEM module requires external libraries to handle the geometric intersections between the fluid and the solid meshes and the treatment of the split elements. Therefore, Triangle (version 1.6) and TetGen (version 1.15.0) [15] are used to mesh the polyhedra generated by the cut of the fluid mesh in 2D and 3D respectively. In particular, the source code of TetGen library has been modified to make it compliant with the LIFEV environment. The licence of use is reported in the Appendix 3.A.1 for each of these libraries.

Despite a release version of LIFEV is available online (accessible with a free bitbucket account), it does not include the module for the XFEM numerical approach that is needed for the benchmark. The branch of the library including the XFEM method will be made public soon, so that the codes will be open access.

LifeV requires some third-part libraries to be built. In particular, the following packages are required:

- cmake (latest version): to configure and create the necessary makefiles for building the source code;
- openblas (v. 0.2.17): low level linear algebra package, providing both BLAS and LAPACK bindings;
- trilinos (v. 11.14.3): parallel linear algebra;
- metis (v. 5): graph partitioning library;
- parmetis (v. 4.0.3): parallel version of metis;
- hdf5 (v. 1.8.16): management of the HDF5 file format for storing data;
- suitesparse (v. 4.5.1): linear algebra library with linear solvers and utilities.

The software will be made publically available soon on Bitbucket. Moreover, in order to simplify the configuration of the system and increase the cross-platform compatibility with other benchmarks in ROMSOC, the installation of the LIFEV environment and the necessary third-part libraries will be set via Docker. Therefore free Docker and Bitbucket accounts will be necessary to run the software to run the benchmark.

Here we present the general idea of the installation procedure, detailing in the following list the steps required for the Docker installation and the configuration procedure:

1. Set the Docker container with the required modules installed by typing in the terminal (it requires docker):

```
docker build -f Dockerfile
```



where the Dockerfile contains the instructions as follows:

```
# Use Ubuntu 16.04 as parent image
FROM ubuntu:xenial

# Install LifeV dependencies
RUN apt-get update && \
apt-get install -y \
g++ \
cmake \
git \
libblacs-mpi-dev \
libscalapack-mpi-dev \
libsuitesparse-dev \
trilinos-all-dev \
libboost-program-options-dev \del
libparmetis-dev \
libmetis-dev \
libhdf5-openmpi-dev \
libmumps-dev \
libsuperlu-dev \
libtbb-dev \
libptscotch-dev \
binutils-dev \
libiberty-dev \
libtriangle-dev && \
groupadd -r lifev && \
useradd -l -m -g lifev lifev
# Set user 'lifev'
USER lifev
# Set working directory
WORKDIR /home/lifev
# Copy the content of the current dir into the WORKDIR
ADD --chown=lifev:lifev . /home/lifev
```

Listing 3.1: Dockerfile

2. Define the paths inside the container as in the *defs.sh* file below:

```
#!/bin/bash
LifeV_DIR=$PWD
LifeV_SRC=$LifeV_DIR/lifev-src
LifeV_BUILD=$LifeV_DIR/lifev-build
LifeV_LIB=$LifeV_DIR/lifev-install
TetGen_DIR=$LifeV_DIR/tetgen1.5.0
mkdir -p $LifeV_SRC
mkdir -p $LifeV_BUILD
mkdir -p $LifeV_LIB
```

Listing 3.2: defs.sh

3. Clone the source codes of LIFEV and TetGen from xx bitbucket repository into the container by executing *./clone.sh* file:

```
#!/bin/bash
source defs.sh
git clone lifev-xfem_REPOSITORY ${LifeV_SRC}
git clone tetgen4lifev_REPOSITORY ${TetGen_DIR}
```

Listing 3.3: clone.sh



Notice that `lifev-xfem_REPOSITORY` and `tetgen4lifev_REPOSITORY` will be the url to the public repository of the LIFEV software and to the modified version of TetGen library.

4. Build TetGen and configure LIFEV typing `./config.sh` from the container:

```
#!/bin/bash
source defs.sh
cd $TetGen_DIR
make tetlib
cd $LifeV_BUILD
cmake \
-D BUILD_SHARED_LIBS:BOOL=OFF \
-D CMAKE_BUILD_TYPE:STRING=RELEASE \
-D CMAKE_INSTALL_PREFIX:PATH=${LifeV_LIB}\
-D CMAKE_C_COMPILER:STRING="mpicc" \
-D CMAKE_CXX_COMPILER:STRING="mpicxx" \
-D CMAKE_CXX_FLAGS:STRING="-O3 -msse3
  -Wno-unused-local-typedefs -Wno-literal-suffix"\
-D CMAKE_C_FLAGS:STRING="-O3 -msse3" \
-D CMAKE_Fortran_COMPILER:STRING="mpif90" \
-D CMAKE_Fortran_FLAGS:STRING="-Og -g" \
-D CMAKE_AR:STRING="ar" \
-D CMAKE_MAKE_PROGRAM:STRING="make" \
[...]
${LifeV_SRC}
cd $LifeV_DIR
```

Listing 3.4: config.sh

Here the `config.sh` is shown in a shorter form with the most important compiling options (`mpicc` compiler for C, `mpicxx` compiler for C++, `O3` optimization). For a complete version with all compiling options and dependencies we refer the reader to Appendix 3.A.2).

5. Build LIFEV from the container by executing `./build.sh`:

```
#!/bin/bash
source defs.sh
cd $LifeV_BUILD
make -j 3
cd $LifeV_DIR
```

Listing 3.5: build.sh

3.4. Computer Requirements

The operative system has to be UNIX-based (like Oracle Solaris, Darwin or macOS), equipped with C++ compiler and all basic system libraries required for software development, like FORTRAN, C++, HDF5 or MPI. It is sufficient to type on the terminal

```
sudo apt-get install build_essentials
```

, to check if your operative system already has such fundamental libraries or, in case they are not present, install them.

The minimum requirements for the hardware are:

- CPU: 1 processor is sufficient to run the benchmark, but multi-thread CPU is needed in case you want to employ parallel computing;
- rigid disk: at least 3 GB of available space in the rigid disk, considering the memory required for building `lifev` and all third-part libraries, as well as the space required by data and results;



- RAM: a minimum of 40 GB of RAM is predicted to be required the benchmark in serial. Be aware that in case of parallel runs, the demand of RAM increases.

3.5. Numerical Example

3.5.1. Experimental Data

The experimental data used for the benchmark are Head pressure-Flow (HQ) curves, which are standard pump performance curves obtained during the *in vitro* testings of the pump system in a static pipe loop machine. Blood is replaced by a mixture of water and glycerin (39% in weight). The bench system is equipped with pressure sensors placed both at the inlet and at the outlet sections, to measure the pressure gradient arisen over the pump, or head pressure, and an ultrasonic flowmeter clamped at the outlet pipe, to measure the outflow volume rate of the pump. The hydraulic resistance in the circuit is varied by means of centrifugal pumps placed in series with the blood pump, that can work in favor or against it. Therefore, the HQ curves are obtained by displaying the different measurements in the pressure-flow plan for the different flow conditions. Figure 3.5 shows the HQ curves of the pump system when the frequency of oscillation of the membrane is fixed to 120 Hz, while the amplitude of oscillation - that is a function of the input voltage V_p of the pump actuator - passes from 0.53 mm (grey triangles), to 0.63 mm (yellow crosses) and finally to 0.73 mm (blue dots).

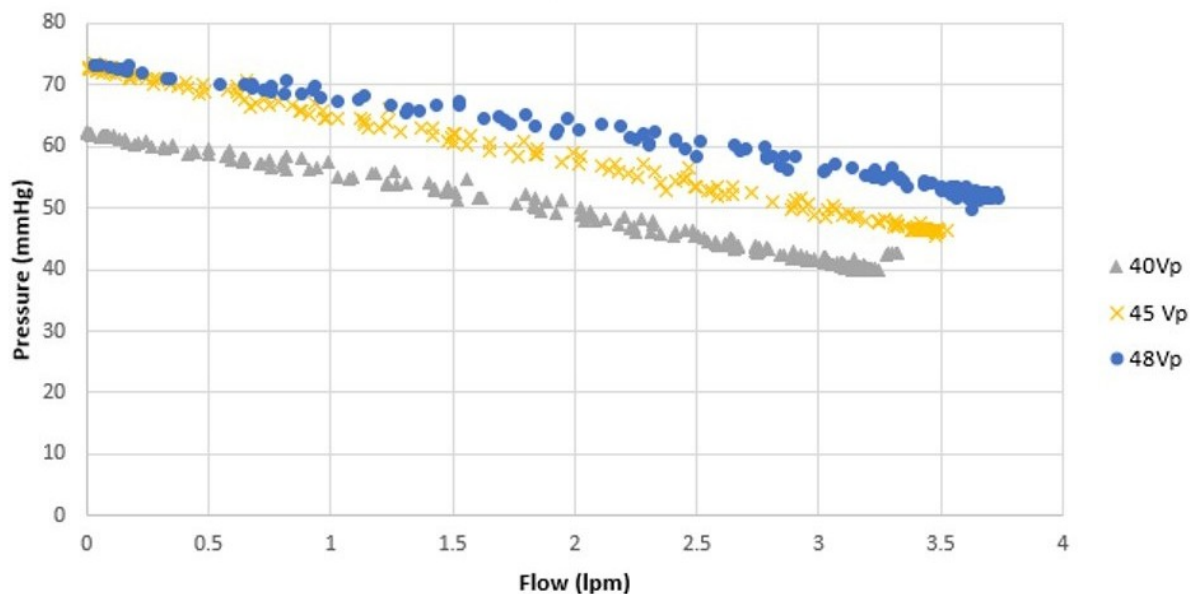


Figure 3.5: HQ experimental curves for three different values of the voltage of the actuator.

Raw data are provided in Excel format for each experiment session and will be made publically available in open-source spreadsheet applications. Each document consists of three separate data columns with the measurements of:

- head pressure H , corresponding to mean pressure difference between the outlet and the inlet, expressed in millimeters of mercury [mmHg] ($1 \text{ mmHg} = 1333.22 \text{ g/cm}^2$).
- outflow volume rates Q , expressed in liters per minute [lpm] ($1 \text{ lpm} = 0.06 \text{ cm}^3/\text{s}$).

The software uses quantities expressed in the *cgs* system.

A separate table explicits the settings of each experimental sessions, including in particular the correspondence between the values of the input voltage of the actuator V_p , expressed in Volt [V], shown in the Figure above, and the respective amplitude of the induced membrane vibrations Φ , expressed in millimeters [mm].



3.5.2. Benchmark Plan

The aim of the proposed benchmark is to validate the FSI model via comparison of the experimental HQ curves with the numerical results. Analogously to the approach for model validation used by Perschal et al. in a similar scenario [4], the pump system is simulated under the same operative conditions of the experimental sessions, so that we can measure the error between the real data and the predicted quantities and quantify the goodness of the model. In particular, we set the pressure difference between the pump outlet and inlet, so that we can predict the corresponding pump outflow rate and compare the numerical result with its experimental counterpart.

```

1: [Q_data, H_data, M_data] = read_data(HQcurves, phys_settings);
2: [Q_Ndata, H_Ndata, M_Ndata] = extract_Ndata(N);           // N ≥ 1 data points
3: for n = 1 : N do
4:   input_pressure = H_Ndata[n];
5:   input_frequency = M_Ndata[n].f;
6:   input_amplitude = M_Ndata[n].phi;
7:   define_bc(input_pressure, input_frequency, input_amplitude); // Boundary conditions
8:   [u0, d0] = initialization()
9:   while t < Tmax do // Time loop
10:    [ut, pt, dt] = solve_FSI(); // Solve the FSI problem
11:    Q_out[t] = f(ut|out); // Outflow computation
12:    t = t + dt;
13:  end while
14:  Q_output[n] = mean(Q|out); // Average in time
15: end for
16: plot(Q_data, H_data; Q_output, H_Ndata); // Validation plot
17: err_Q = norm2(Q_Ndata - Q_output);

```

Algorithm 1: Benchmark structure: Model validation against N data points.

The plan for the model validation is described in the Algorithm 1. Specifically:

1. First, the experimental data coming from the HQ curves are stored in separated variables: H_data collects the data of the head pressure between outlet and inlet, Q_data contains the measurements of the corresponding outflow volume rate, and M_data stores the input settings for the parameters of membrane displacement; moreover, additional physical parameters are contained in the settings variables (line 1).
2. Only N data points are extracted by the curves to be reproduced via simulation and used for the model validation (line 2).
3. Therefore, for each of the N selected cases, the input parameters for the inlet flow (line 4) and for the membrane oscillation (lines 5-6) are used to define the boundary conditions of the problem (line 7).
4. In this way, the pump system is simulated in the time interval $[0, T_{max}]$, so that we obtain the numerical values of the blood velocity \mathbf{u}_t , blood pressure p_t and structure displacement \mathbf{d}_t at each time t ; afterwards, the numerical outflow Q_{out} is computed by taking the integral over the outlet surface of the blood normal velocity (lines 8-13).
5. At the end of each simulation, the pump outflow rate is averaged in time over the last period of membrane oscillation (line 14).
6. Finally, the HQ curves can be plotted together with the numerical output and an error measure can be computed as the 2-norm of the deviation of the numerical results from the experimental data (lines 16-17).

Therefore, the input for the benchmark are:

- the experimental data, i.e. the HQ curves,



- $N \geq 1$: number of data points used for validation (it coincides with the number of simulations),
- `input_pressure`: head pressure values for the n^{th} simulation,
- `input_frequency`: frequency of membrane vibration for the n^{th} simulation,
- `input_amplitude`: amplitude of membrane vibration for the n^{th} simulation,
- physical parameters of the blood pump system, e.g. blood and structure properties;

while the output are:

- `Q_output`: predictions of the outflow rate from the N simulations,
- `err_Q`: 2-norm error between the estimated and the experimental outflow vectors,
- `validation plot`: HQ curve and numerical points.

In conclusion, the proposed benchmark can be used for training in the fields of mathematical modeling of a coupled system, model testing and error estimation.

3.A. Appendix

3.A.1. Licences of Use

3.A.1.1. LIFEV (release version)

Copyright (C) 2004, 2005, 2007 EPFL, Politecnico di Milano, INRIA Copyright (C) 2010 EPFL, Politecnico di Milano, Emory University Copyright (C) 2011,2012,2013 EPFL, Politecnico di Milano, Emory University

This file is part of LifeV.

LifeV is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

LifeV is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with LifeV. If not, see <http://www.gnu.org/licenses/>.

3.A.1.2. TetGen

TetGen is distributed under a dual licensing scheme. You can redistribute it and/or modify it under the terms of the GNU Affero General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version. A copy of the GNU Affero General Public License is reproduced below.

If the terms and conditions of the AGPL v.3. would prevent you from using TetGen, please consider the option to obtain a commercial license for a fee. These licenses are offered by the Weierstrass Institute for Applied Analysis and Stochastics (WIAS). As a rule, licenses are provided "as-is", unlimited in time for a one time fee. Please send corresponding requests to: tetgen@wias-berlin.de. Please do not forget to include some description of your company and the realm of its activities.

3.A.1.3. Triangle

Triangle A Two-Dimensional Quality Mesh Generator and Delaunay Triangulator. Version 1.6



Copyright 1993, 1995, 1997, 1998, 2002, 2005 Jonathan Richard Shewchuk 2360 Woolsey H Berkeley, California 94705-1927 Please send bugs and comments to jrs@cs.berkeley.edu

Created as part of the Quake project (tools for earthquake simulation). Supported in part by NSF Grant CMS-9318163 and an NSERC 1967 Scholarship. There is no warranty whatsoever. Use at your own risk.

These programs may be freely redistributed under the condition that the copyright notices (including the copy of this notice in the code comments and the copyright notice printed when the '-h' switch is selected) are not removed, and no compensation is received. Private, research, and institutional use is free. You may distribute modified versions of this code under the condition that this code and any modifications made of it in the same file remain under Copyright of the original author, both source and object code are made freely available without charge, and clear notice is given of the modifications. Distribution of this code as part of a commercial system is permissible only by direct agreement with the author. (If you are not directly supplying this code to a customer, and you are instead telling them how they can obtain it for free, then you are not required to make any arrangement with me.)

3.A.2. Configuration files

```
#!/bin/bash
source defs.sh

cd $TetGen_DIR

make tetlib

cd $LifeV_BUILD

cmake \
-D BUILD_SHARED_LIBS:BOOL=OFF \
-D CMAKE_BUILD_TYPE:STRING=RELEASE \
-D CMAKE_INSTALL_PREFIX:PATH=${LifeV_LIB} \
-D CMAKE_C_COMPILER:STRING="mpicc" \
-D CMAKE_CXX_COMPILER:STRING="mpicxx" \
-D CMAKE_CXX_FLAGS:STRING="-O3 -msse3 -Wno-unused-local-typedefs -Wno-literal-suffix" \
-D CMAKE_C_FLAGS:STRING="-O3 -msse3" \
-D CMAKE_Fortran_COMPILER:STRING="mpif90" \
-D CMAKE_Fortran_FLAGS:STRING="-Og -g" \
-D CMAKE_AR:STRING="ar" \
-D CMAKE_MAKE_PROGRAM:STRING="make" \
\
-D TPL_ENABLE_AMD=ON \
-D AMD_INCLUDE_DIRS=/usr/include/suitesparse/ \
-D AMD_LIBRARY_DIRS=/usr/lib/x86_64-linux-gnu/ \
-D AMD_LIBRARY_NAMES=amd \
-D TPL_ENABLE_BLACS=ON \
-D BLACS_INCLUDE_DIRS=/usr/include/ \
-D BLACS_LIBRARY_DIRS=/usr/lib/ \
-D BLACS_LIBRARY_NAMES=blacs \
-D TPL_ENABLE_Boost=ON \
-D Boost_INCLUDE_DIRS=/usr/include/boost/ \
-D TPL_ENABLE_BoostLib=ON \
-D Boost_NO_BOOST_CMAKE:BOOL=ON \
-D BoostLib_INCLUDE_DIRS=/usr/include/boost/ \
-D BoostLib_LIBRARY_DIRS=/usr/lib/x86_64-linux-gnu/ \
-D TPL_ENABLE_HDF5=ON \
-D HDF5_INCLUDE_DIRS=/usr/include/hdf5/openmpi/ \
-D HDF5_LIBRARY_DIRS=/usr/lib/x86_64-linux-gnu/ \
-D TPL_ENABLE_SCALAPACK=ON \
-D SCALAPACK_INCLUDE_DIRS= \
```



```

-D SCALAPACK_LIBRARY_DIRS=/usr/lib/ \
-D SCALAPACK_LIBRARY_NAMES=scalapack \
-D TPL_ENABLE_MPI=ON \
-D MPI_BASE_DIR:PATH=/usr/ \
-D ParMETIS_INCLUDE_DIRS=/usr/include/ \
-D ParMETIS_LIBRARY_DIRS=/usr/lib/x86_64-linux-gnu/ \
-D TPL_ENABLE_UMFPACK=ON \
-D UMFPACK_INCLUDE_DIRS=/usr/include/suitesparse/ \
-D UMFPACK_LIBRARY_DIRS=/usr/lib/x86_64-linux-gnu/ \
-D UMFPACK_LIBRARY_NAMES=umfpack \
-D Trilinos_INCLUDE_DIRS:PATH="/usr/include/trilinos/" \
-D Trilinos_LIBRARY_DIRS:PATH="/usr/lib/x86_64-linux-gnu/" \
-D TetGen_INCLUDE_DIRS:PATH="${TetGen_DIR}" \
-D TetGen_LIBRARY_DIRS:PATH="${TetGen_DIR}" \
-D TetGen_LIBRARY_NAMES="tet" \
-D Triangle_INCLUDE_DIRS:PATH="/usr/include" \
-D Triangle_LIBRARY_DIRS:PATH="/usr/lib" \
-D Triangle_LIBRARY_NAMES="triangle" \
\
-D LifeV_ENABLE_DEBUG:BOOL=OFF \
-D LifeV_ENABLE_TESTS:BOOL=ON \
\
-D LifeV_ENABLE_ALL_PACKAGES:BOOL=ON \
-D LifeV_ENABLE_Core:BOOL=ON \
-D LifeV_ENABLE_ETA:BOOL=ON \
-D LifeV_ENABLE_NavierStokes:BOOL=ON \
-D LifeV_ENABLE_BCInterface:BOOL=ON \
-D LifeV_ENABLE_Structure:BOOL=ON \
-D LifeV_ENABLE_ZeroDimensional:BOOL=ON \
-D LifeV_ENABLE_OneDFSI:BOOL=ON \
-D LifeV_ENABLE_LevelSet:BOOL=ON \
-D LifeV_ENABLE_Darcy:BOOL=ON \
-D LifeV_ENABLE_Electrophysiology:BOOL=ON \
-D LifeV_ENABLE_Heart:BOOL=ON \
-D LifeV_ENABLE_FSI:BOOL=ON \
-D LifeV_ENABLE_Multiscale:BOOL=ON \
-D LifeV_ENABLE_XFEM:BOOL=ON \
-D LifeV_ENABLE_Dummy:BOOL=ON \
\
-D LifeV_STRUCTURE_ENABLE_ANISOTROPIC:BOOL=ON \
-D LifeV_STRUCTURE_COLORING_MESH:BOOL=ON \
-D LifeV_STRUCTURE_COMPUTATION_JACOBIAN:BOOL=ON \
-D LifeV_STRUCTURE_EXPORTVECTORS:BOOL=ON \
\
-D BCInterface_ENABLE_TESTS:BOOL=ON \
-D Core_ENABLE_TESTS:BOOL=ON \
-D Darcy_ENABLE_TESTS:BOOL=ON \
-D Dummy_ENABLE_TESTS:BOOL=ON \
-D Electrophysiology_ENABLE_TESTS:BOOL=ON \
-D ETA_ENABLE_TESTS:BOOL=ON \
-D FSI_ENABLE_TESTS:BOOL=ON \
-D Heart_ENABLE_TESTS:BOOL=ON \
-D LevelSet_ENABLE_TESTS:BOOL=ON \
-D Multiscale_ENABLE_TESTS:BOOL=ON \
-D NavierStokes_ENABLE_TESTS:BOOL=ON \
-D OneDFSI_ENABLE_TESTS:BOOL=ON \
-D Structure_ENABLE_TESTS:BOOL=ON \
-D XFEM_ENABLE_TESTS:BOOL=ON \
-D ZeroDimensional_ENABLE_TESTS:BOOL=ON \
\
-D BCInterface_ENABLE_EXAMPLES:BOOL=ON \

```



```

-D Core_ENABLE_EXAMPLES:BOOL=ON \
-D Darcy_ENABLE_EXAMPLES:BOOL=ON \
-D Dummy_ENABLE_EXAMPLES:BOOL=ON \
-D Electrophysiology_ENABLE_EXAMPLES:BOOL=ON \
-D ETA_ENABLE_EXAMPLES:BOOL=ON \
-D FSI_ENABLE_EXAMPLES:BOOL=ON \
-D Heart_ENABLE_EXAMPLES:BOOL=ON \
-D LevelSet_ENABLE_EXAMPLES:BOOL=ON \
-D Multiscale_ENABLE_EXAMPLES:BOOL=ON \
-D NavierStokes_ENABLE_EXAMPLES:BOOL=ON \
-D OneDFSI_ENABLE_EXAMPLES:BOOL=ON \
-D Structure_ENABLE_EXAMPLES:BOOL=ON \
-D XFEM_ENABLE_EXAMPLES:BOOL=ON \
-D ZeroDimensional_ENABLE_EXAMPLES:BOOL=ON \
\
-D LifeV_ENABLE_STRONG_CXX_COMPILE_WARNINGS:BOOL=ON \
${LifeV_SRC}

cd ${LifeV_DIR}

```

Listing 3.6: config.sh

Bibliography

- [1] C. N. Botterbusch, S. Lucquin, P. Monticone, J. B. Drevet, A. Guignabert, and P. Meneroud, “Implantable pump system having an undulating membrane,” May 15 2018, uS Patent 9,968,720.
- [2] C. N. Botterbusch, P. Monticone, E. Illouz, B. Burg, L. Polverelli, and T. Snyder, “Getting past the spin: The CorWave LVAD, a membrane wave pump providing physiologic pulsatility without high shear,” *The Journal of Heart and Lung Transplantation*, vol. 38, no. 4, p. 5345, 2019.
- [3] T. Snyder, A. Bourquin, F. Cornat, B. Burg, J. Biasetti, and C. N. Botterbusch, “CorWave LVAD development update,” *The Journal of Heart and Lung Transplantation*, vol. 38, no. 4, pp. 5341–5342, 2019.
- [4] M. Perschall, J. B. Drevet, T. Schenkel, and H. Oertel, “The progressive wave pump: numerical multiphysics investigation of a novel pump concept with potential to ventricular assist device application,” *Artificial organs*, vol. 36, no. 9, 2012.
- [5] E. Burman and M. A. Fernández, “An unfitted Nitsche method for incompressible fluid–structure interaction using overlapping meshes,” *Computer Methods in Applied Mechanics and Engineering*, vol. 279, pp. 497–514, 2014.
- [6] A. Hansbo and P. Hansbo, “An unfitted finite element method, based on Nitsche’s method, for elliptic interface problems,” *Computer methods in applied mechanics and engineering*, vol. 191, no. 47-48, pp. 5537–5552, 2002.
- [7] S. Zonca, C. Vergara, and L. Formaggia, “An unfitted formulation for the interaction of an incompressible fluid with a thick structure via an XFEM/DG approach,” *SIAM Journal on Scientific Computing*, vol. 40, no. 1, pp. B59–B84, 2018.
- [8] E. Burman, M. A. Fernández, and P. Hansbo, “Continuous interior penalty finite element method for Oseen’s equations,” *SIAM journal on numerical analysis*, vol. 44, no. 3, pp. 1248–1274, 2006.
- [9] E. Burman, “Ghost penalty,” *Comptes Rendus Mathématique*, vol. 348, no. 21-22, pp. 1217–1220, 2010.
- [10] C. Vergara and S. Zonca, *Extended Finite Elements Method for Fluid-Structure Interaction with an Immersed Thick Non-linear Structure*, 01 2018, pp. 209–243.
- [11] M. Hillairet and T. Takahashi, “Collisions in three-dimensional fluid structure interaction problems,” *SIAM Journal on Mathematical Analysis*, vol. 40, no. 6, pp. 2451–2477, 2009.



- [12] E. Burman, M. A. Fernández, and S. Frei, “A Nitsche-based formulation for fluid-structure interactions with contact,” *arXiv preprint arXiv:1808.08758*, 2018.
- [13] E. Lauga, M. P. Brenner, and H. A. Stone, “Microfluidics: the no-slip boundary condition,” *arXiv preprint cond-mat/0501557*, 2005.
- [14] L. Bertagna, S. Deparis, L. Formaggia, D. Forti, and A. Veneziani, “The LifeV library: engineering mathematics beyond the proof of concept,” *arXiv preprint arXiv:1710.06596*, 2017.
- [15] H. Si, “TetGen, a delaunay-based quality tetrahedral mesh generator,” *ACM Transactions on Mathematical Software (TOMS)*, vol. 41, no. 2, p. 11, 2015.



Part II.

Model order reduction methods



4. Model order reduction for parametric high dimensional interest rate models in the analysis of financial risk

Andreas Binder¹, Onkar Jadhav², Volker Mehrmann²

¹*MathConsult*

²*Technische Universität Berlin*

Abstract. This paper presents a model order reduction (MOR) approach for high dimensional problems in the analysis of financial risk. To understand the financial risks and possible outcomes, we have to perform several thousand simulations of the underlying product. These simulations are expensive and create a need for efficient computational performance. Thus, to tackle this problem, we establish a MOR approach based on a proper orthogonal decomposition (POD) method. The study involves the computations of high dimensional parametric convection-diffusion reaction partial differential equations (PDEs). POD requires to solve the high dimensional model at some parameter values to generate a reduced-order basis. We propose an adaptive greedy sampling technique based on surrogate modeling for the selection of the sample parameter set that is analyzed, implemented, and tested on the industrial data. The results obtained for the numerical example of a floater with a cap and floor under the Hull-White model indicate that the MOR approach works well for short-rate models.

Keywords: Financial risk analysis, short-rate models, convection-diffusion-reaction equation, finite difference method, parametric model order reduction, proper orthogonal decomposition, adaptive greedy sampling, packaged retail investment and insurance-based products.

4.1. Introduction

Packaged retail investment and insurance-based products (PRIIPs) are at the essence of the retail investment market. PRIIPs offer considerable benefits for retail investors which make up a market in Europe worth up to €10 trillion. However, the product information provided by financial institutions to investors can be overly complicated and contains confusing legalese. To overcome these shortcomings, the EU has introduced new regulations on PRIIPs (European Parliament Regulation (EU) No 1286/2014) [1]. According to these regulations, a PRIIP manufacturer must provide a *key information document* (KID) for an underlying product that is easy to read and understand. The KID informs about the vital features, such as costs and risks of the investment, before purchasing the product. The PRIIPs include interest rate derivatives such as the interest rate cap and floor [2], interest rate swaps [3], etc.

A key information document includes a section about ‘*what could an investor get in return?*’ for the invested product which requires costly numerical simulations of financial instruments. This paper evaluates interest rate derivatives based on the dynamics of the short-rate models [4]. For the simulations of short-rate models, techniques based on discretized convection-diffusion reaction partial differential equations (PDEs) are very successful [5]. To discretize the PDE, we implemented the finite difference method (FDM) [6]. The FDM has been proven to be efficient for solving the short-rate models [7, 8, 9]. The model parameters are usually calibrated based on market structures like *yield curves*, *cap* volatilities, or *swaption* volatilities [4]. The regulation demands to perform yield curve simulations for at least 10,000 times. A yield curve shows the interest rates varying with respect to 20-30 time points known as *tenor points*. These time points are the contract lengths of an underlying instrument. The calibration based on several thousand simulated yield curves generates a high dimensional model parameter space as a function of these tenor points. The 10,000 different simulated yield curves and the calibrated parameters based on these simulated yield curves can be considered as 10,000 different scenarios. We need to solve a high dimensional model (HDM) obtained by discretizing the short-rate PDE for such scenarios [10]. Furthermore, the results obtained for these several thousand scenarios are used to calculate the possible values for an instrument under favorable, moderate, and unfavorable conditions. The



favorable, moderate, and unfavorable scenario values are the values at 90th percentile, 50th percentile, and 10th percentile of 10,000 values, respectively. However, these evaluations are computationally costly, and additionally, have the disadvantage of being affected by the so-called *curse of dimensionality* [11].

To avoid this problem, we establish a parametric model order reduction (MOR) approach based on a variant of the proper orthogonal decomposition (POD) method [12, 13]. The method is also known as the Karhunen-Loève decomposition [14] or principal component analysis [15] in statistics. The combination of a Galerkin projection approach and POD creates a powerful method for generating a reduced order model (ROM) from the high dimensional model that has a high dimensional parameter space [16]. This approach is computationally feasible as it always looks for low dimensional linear (or affine) subspaces [17, 18]. Also, it is necessary to note that the POD approach considers the nonlinearities of the original system. Thus, the generated reduced order model will be nonlinear if the HDM is nonlinear as well. POD generates an optimally ordered orthonormal basis in the least squares sense for a given set of computational data. Furthermore, the reduced order model is obtained by projecting a high dimensional system onto a low dimensional subspace obtained by truncating the optimal basis called reduced-order basis (ROB). The selection of the data set plays an important role and is most prominently obtained by the method of snapshots introduced in [19]. In this method, the optimal basis is computed based on a set of state solutions. These state solutions are known as snapshots and are calculated by solving the HDM for some pre-selected training parameter values. The quality of the ROM is bounded by the training parameters used to obtain the snapshots. Thus, it is necessary to address the question of how to generate the set of potential parameters which will create the optimal ROB. Some of the previous works implement either some form of fixed sampling or often only uniform sampling techniques [20]. These approaches are straightforward, but they may neglect the vital regions in the case of high dimensional parameter spaces.

In the current work, a greedy sampling algorithm has been implemented to determine the best suitable parameter set [21, 22, 23]. The basic idea is to select the parameters at which the error between the ROM and the HDM is maximal. Further, we compute the snapshots using these parameters and thus obtain the best suitable ROB which will generate a fairly accurate ROM. The calculation of the relative error between the ROM and the HDM is expensive, so instead, we use error estimators like the residual error associated with the ROM [24, 25]. The greedy sampling algorithm picks the optimal parameters which yield the highest values for the error estimator. Furthermore, we use these parameters to construct a snapshot matrix and, consequently, to obtain the desired ROB.

However, it is not reasonable to compute an error estimator for the entire parameter space. The error estimator is based on the norm of the residual, which scales with the size of the HDM. This problem forces us to select a pre-defined parameter set as a subset of the high dimensional parameter space to train the greedy sampling algorithm. We usually select this pre-defined subset randomly. But, a random selection may neglect the crucial parameters within the parameter space. Thus, to surmount this problem, we implemented an adaptive greedy sampling approach. We choose the most suitable parameters adaptively at each greedy iteration using an optimized search based on surrogate modeling. We construct a surrogate model for the error estimator and use it to find the best suitable parameters. The use of the surrogate model avoids the expensive computation of the error estimator over the entire parameter space. The adaptive greedy sampling approach associated with surrogate modeling has been introduced before in [22, 23]. There are several approaches to design a surrogate model like regression analysis techniques [26], response surface models, or Kriging models [27]. The authors of [22] have designed a Kriging based surrogate model to select the most relevant parameters adaptively. However, in our case, due to the high dimensional parameter space, we may face the multicollinearity problem as some variable in the model can be written as a linear combination of the other variables in the model [28]. Also, we need to construct a surrogate model considering the fact that the model parameters are time-dependent. Thus, in this work, we construct a surrogate model based on the principal component regression (PCR) technique [26]. The PCR approach is a dimension reduction technique in which explanatory variables are replaced by few uncorrelated variables known as principal components. It replaces the multivariate problem with a more straightforward low dimensional problem and avoids overfitting.

In the classical greedy sampling approach, the convergence of the algorithm is observed using the error estimator. However, we can use the norm of the residual to estimate the exact error between the HDM and the ROM.



In this work, we establish an error model for an exact error as a function of the error estimator based on the idea presented in [22]. Furthermore, we use this exact error model to observe the convergence of the greedy sampling algorithm.

To summarize, this paper presents an approach to select the most prominent parameters or scenarios for which we solve the HDM and obtain the required ROB. Thus, instead of performing 10,000 expensive computations, we perform very few expensive computations and solve the remaining scenarios with the help of the ROM. The paper illustrates the implementation of numerical algorithms and methods in detail. It is necessary to note that the choice of a short-rate model depends on the underlying financial instrument. In this work, we focus on one-factor short-rate models only. We implement the developed algorithms for the one-factor Hull-White model [29] and present the results with a numerical example of a floater with cap and floor [30]. The current research findings indicate that the MOR approach works well for short-rate models.

The paper is organized as follows. Section 4.2 presents a model hierarchy for the Hull-White model. In Section 4.3.1 a finite difference method for the Hull-White model and the projection-based model reduction technique are presented. The selection of optimal sampling parameters based on the classical greedy approach is presented in Subsection 4.3.2 and based on the adaptive greedy method in Subsections 4.3.3 and 4.3.4. Numerical results for the example of a floater are presented in section 4.4.

4.2. Mathematical Formulation: Model Hierarchy

The management of interest rate risks, i.e., the control of change in future cash flows due to the fluctuations in interest rates is of great importance. Especially, the pricing of products based on the stochastic nature of the interest rate creates the necessity for mathematical models for an underlying financial instrument. There exist several well-known one-factor short-rate models, such as the Vasicek model [31], the Cox-Ingersoll-Ross model [32], or the Hull-White model [29, 33] which is an extension of the Vasicek model. The stochastic differential equation for the Hull-White model is given as

$$dr(t) = (a(t) - b(t)r(t))dt + \sigma(t)dW(t), \quad (4.1)$$

with time-dependent parameters $a(t)$, $b(t)$, and $\sigma(t)$. The term $(a(t) - b(t)r(t))$ is a drift term and $a(t)$ is known as *deterministic drift*. Based on the Ito's lemma, we can define the partial differential equation for the Hull-White model which is given by

$$\frac{\partial V_1}{\partial t} + (a(t) - b(t)r(t))\frac{\partial V_1}{\partial r(t)} + \frac{1}{2}\sigma^2(t)\frac{\partial^2 V_1}{\partial r(t)^2} - r(t)V_1 = 0. \quad (4.2)$$

In [34] it is suggested to use as a simplification that $b(t)$ and $\sigma(t)$ are constant in time. Although this may lead to difficulties, see [35], we follow this suggestion and assume in this paper that $b(t) = b$ and $\sigma(t) = \sigma$ are constant and consider in the remainder of the paper the so called *robust Hull-White model*

$$\frac{\partial V_1}{\partial t} + (a(t) - br(t))\frac{\partial V_1}{\partial r(t)} + \frac{1}{2}\sigma^2\frac{\partial^2 V_1}{\partial r(t)^2} - r(t)V_1 = 0$$

for yield curve simulation and parameter calibration. Our results can, however, be extended to the more general case. Figure 4.1 shows the model hierarchy for the Hull-White model.



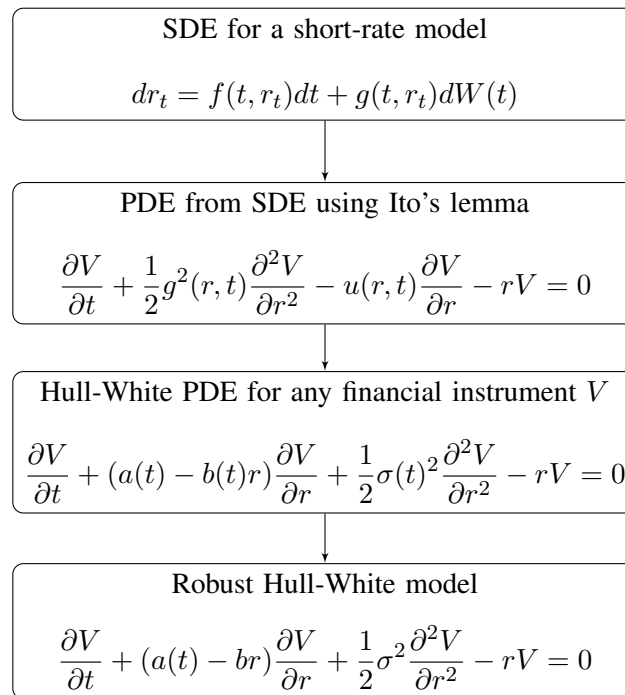


Figure 4.1: A model hierarchy to construct the Hull-White model based on the short-rate r with constant b and σ .

To approximate the time-dependent parameter $a(t)$ one uses yield curves, which determine the average direction in which the short-rate $r(t)$ moves. The PRIIP regulation demands to perform yield curve simulations for at least 10 000 times [1]. The calibration based on several thousand simulated yield curves generates a high dimensional model parameter space as a function of these tenor points. We need to solve a high dimensional model (HDM) obtained by discretizing the short-rate PDE for such a high dimensional parameter space [10]. However, these evaluations are computationally costly, and additionally, have the disadvantage of being affected by the *curse of dimensionality* [11]. To overcome this drawback we establish model order reduction approach.

4.3. Numerical Methods

Figure 4.2 shows the model hierarchy to obtain a reduced-order model for the Hull-White model.

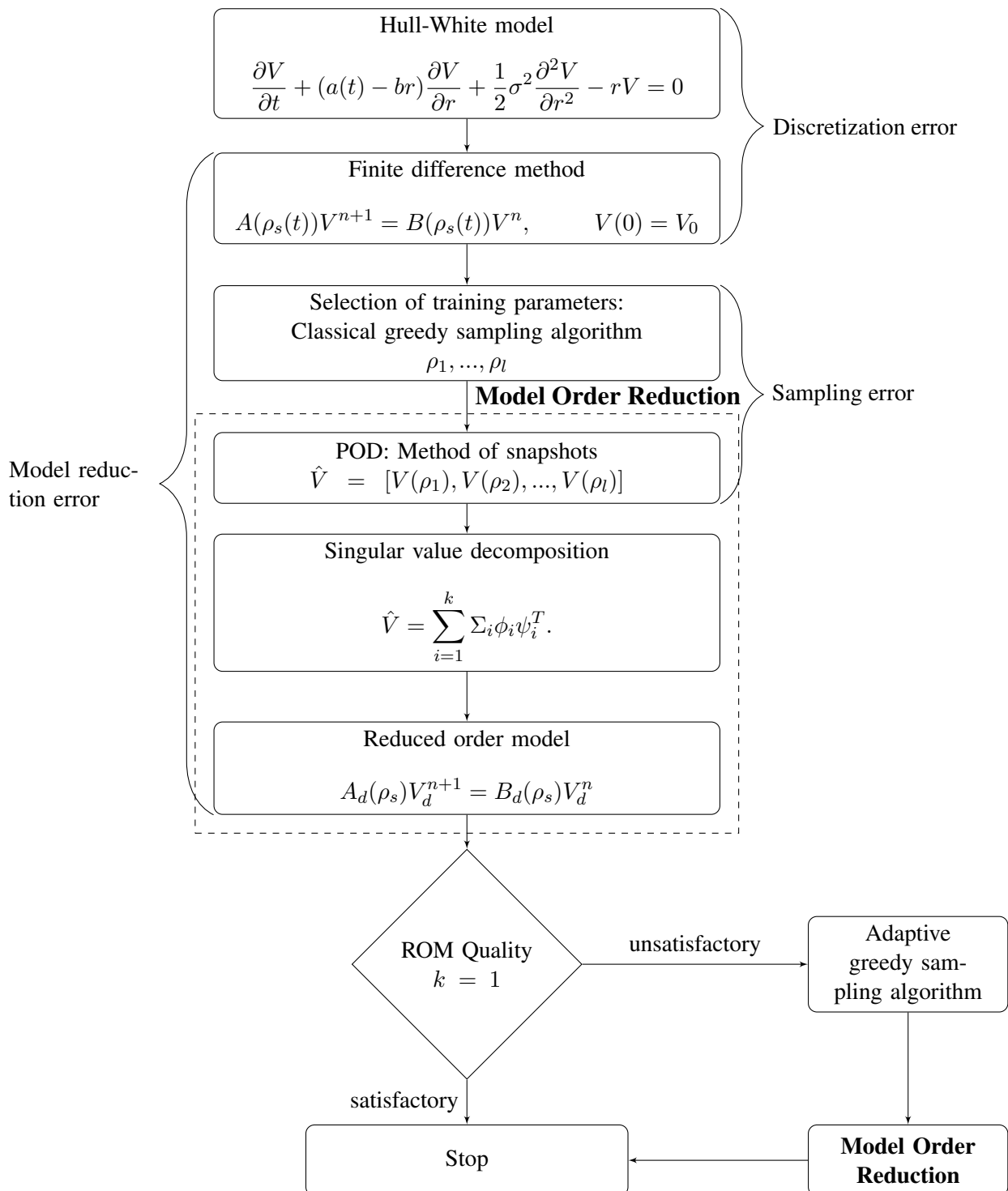


Figure 4.2: Model hierarchy to obtain a reduced order model for the Hull-White model.

We discretize the Hull-White PDE using a finite difference method, which creates a parameter-dependent high dimensional model. Solving the high dimensional model for a large parameter space is computationally costly. Thus, we incorporate the parametric model order reduction approach based on the proper orthogonal decomposition. The POD approach relies on the method of snapshots. The snapshots are nothing but the solutions of the high dimensional model at some parameter values. The idea is to solve the high dimensional model for only a



certain number of training parameters to obtain a reduced-order basis. This reduced-order basis is then used to construct a reduced-order model. Finally, we can solve the reduced-order model cheaply for the large parameter space. The selection of the training parameters is of utmost importance to obtain the optimal reduced-order model. In this work, we have incorporated the classical as well as adaptive greedy sampling approaches for the selection of the training parameter set. Each layer within the model hierarchy associates some error. We have considered the model parameters as constants to obtain the robust model, which leads to the model error. Discretization error occurs due to the implementation of the FDM for simulating the Hull-White model. We have to consider the sampling error associated with the sampling algorithms as well. The total error then can be defined using an inequality as follows

$$\|V_{true} - V_{approx}\| \leq \underbrace{\|V_{model} - V_{Discr}\|}_{\text{Discretization error}} + \underbrace{\|V_{Discr} - V_{MOR}\|}_{\text{MOR error}} + \underbrace{\|E_{sampl.}\|}_{\text{Sampling error}} \quad (4.3)$$

where V_{true} is the true solution for the Hull-White PDE, while V_{approx} is a solution obtained using the model order reduction approach. V_{Discr} is the result of the discretized model based on the finite difference method. V_{MOR} is the result obtained using a reduced order model. $E_{sampl.}$ is the error associated with the sampling techniques. In the following sections, we concentrate on the methods to obtain a suitable ROB using two different sampling approaches.

4.3.1. Parametric Model Order Reduction

The discretization of the Hull-White PDE is a parametric high dimensional model of the following form (4.4)

$$A(\rho_\ell(t))V^{n+1} = B(\rho_\ell(t))V^n, \quad (4.4)$$

with given initial vector V^0 , and matrices $A(\rho_\ell) \in \mathbb{R}^{M \times M}$, and $B(\rho_\ell) \in \mathbb{R}^{M \times M}$. We call this the *full model* for the model reduction procedure in this section. Here again $\ell = 1, \dots, s = 10\,000$, m is the total number of tenor points, and we need to solve this system at each time step n with an appropriate boundary condition and a known initial value of the underlying instrument. Altogether we have a parameter space \mathcal{P} of size $10\,000 \times m$ to which we now apply model reduction.

To perform the parametric model reduction for system (4.4) we employ Galerkin projection onto a low dimensional subspace via

$$\bar{V}^n = QV_d^n, \quad (4.5)$$

where the columns of $Q \in \mathbb{R}^{M \times d}$ represent the reduced-order basis with $d \ll M$, V_d^n is a vector of reduced coordinates, and $\bar{V}^n \in \mathbb{R}^M$ is the solution in the n th time step obtained using the reduced order model. For the Galerkin projection we require that the residual of the reduced state

$$p^n(V_d^n, \rho_\ell) = A(\rho_\ell)QV_d^{n+1} - B(\rho_\ell)QV_d^n \quad (4.6)$$

is orthogonal to the reduced basis matrix Q , i.e.,

$$Q^T p^n(V_d^n, \rho_\ell) = 0, \quad (4.7)$$

so that by multiplying (4.7) with Q^T , we get

$$\begin{aligned} Q^T A(\rho_\ell)QV_d^{n+1} &= Q^T B(\rho_\ell)QV_d^n, \\ A_d(\rho_\ell)V_d^{n+1} &= B_d(\rho_\ell)V_d^n, \end{aligned} \quad (4.8)$$

where $A_d(\rho_\ell) \in \mathbb{R}^{d \times d}$ and $B_d(\rho_\ell) \in \mathbb{R}^{d \times d}$ are the parameter dependent reduced matrices.

We obtain the Galerkin projection matrix Q (4.7) based on a proper orthogonal decomposition (POD) approach, which generates an optimal order orthonormal basis Q in the least square sense that is independent of the



parameter space \mathcal{P} and we do this by the method of snapshots. The snapshots are nothing but the state solutions obtained by simulating the full model for selected parameter groups. We assume that we have a training set of parameter groups $\rho_1, \dots, \rho_k \in [\rho_1, \rho_s]$. We compute the solutions of the full model for this training set and combine them in a snapshot matrix $\hat{V} = [V(\rho_1), V(\rho_2), \dots, V(\rho_k)]$. The POD method solves

$$\text{POD}(\hat{V}) := \underset{Q}{\operatorname{argmin}} \frac{1}{k} \sum_{i=1}^k \|V_i - QQ^T V_i\|^2, \quad (4.9)$$

for an orthogonal matrix $Q \in \mathbb{R}^{M \times d}$ via a *truncated SVD*, see [36],

$$\hat{V} = \Phi \Sigma \Psi^T = \sum_{i=1}^k \Sigma_i \phi_i \psi_i^T, \quad (4.10)$$

where ϕ_i and ψ_i are the left and right singular vectors of the matrix \hat{V} respectively, and Σ_i are the singular values. The truncated SVD computes only the first k columns of the matrix Φ . The optimal projection subspace Q then consists of d left singular vectors ϕ_i known as *POD modes*. The dimension d of the subspace Q is chosen such that we get a good approximation of the snapshot matrix. According to [17], large singular values correspond to the main characteristics of the system, while small singular values give only small perturbations of the overall dynamics. The relative importance of the i th POD mode of the matrix \hat{V} is determined by the *relative energy* Ξ_i of that mode

$$\Xi_i = \frac{\Sigma_i}{\sum_{i=1}^k \Sigma_i} \quad (4.11)$$

If the sum of the energies of the generated modes is 1, then these modes can be used to reconstruct a snapshot matrix completely [37]. In general, the number of modes required to generate the complete data set is significantly less than the total number of POD modes [38]. Thus, a matrix \hat{V} can be accurately approximated by using POD modes whose corresponding energies sum to almost all of the total energy. Thus, we choose only d out of k POD modes to construct $Q = [\phi_1 \cdots \phi_d]$ which is a parameter independent projection space based on (4.11).

We summarize the procedure in Algorithm 1, where we denote by $V(\rho_i)$ the solution of the full model for a parameter set ρ_i , by \hat{V} the snapshot matrix, by Φ (Ψ) the matrix of left (right) singular vectors, associated with the diagonal matrix of singular values Σ , and by Ξ_j the relative energy of the j th POD mode.

Algorithm 1 (Parametric POD).

Input: Parameter group $a(t)$, b , σ , energy level EL , number of samples k .

Output: Projection matrix Q .

Choose sample parameter groups ρ_1, \dots, ρ_k .

FOR $i = 1$ to k .

Solve the full order model for the parameter group ρ_i and determine $V(\rho_i)$.

END

Construct snapshot matrix $\hat{V} = [V(\rho_1), \dots, V(\rho_k)]$.

Compute leading singular values and vectors of \hat{V} using truncated SVD $\hat{V} = \Phi \Sigma \Psi^T$.

Set $\Xi = \operatorname{diag}(\Sigma) / \operatorname{sum}(\operatorname{diag}(\Sigma))$.

FOR $j = 1$ to $\operatorname{length}(\Sigma)$.

$\bar{\Xi} = \operatorname{sum}(\Xi(1:j)) \times 100$.

IF $\bar{\Xi} > EL$ then set $d = j$.

END

$Q = [\phi_1 \cdots \phi_d]$.



It is evident that the quality of the reduced model strongly depends on the selection of parameter groups ρ_1, \dots, ρ_k that are used to compute the snapshots. Hence it is essential to introduce an efficient sampling technique for the parameter space. We could consider the standard sampling techniques, like uniform sampling or random sampling [28]. However, these techniques may neglect vital regions within the parameter space. As an alternative, a greedy sampling method has been suggested in the framework of model order reduction, see [23, 21, 22].

4.3.2. Greedy Sampling Method

The greedy sampling technique selects the parameter groups at which the error between the reduced model and the full model is maximal. We compute the snapshots using these parameter groups in such a way that we obtain an optimal reduced basis Q . Let $\|e\| = \frac{\|V - \hat{V}\|}{\|V\|}$ be the relative error between the reduced and full model and set

$$\rho_I = \operatorname{argmax}_{\rho \in \mathcal{P}} \|e\|. \quad (4.12)$$

At each greedy iteration $i = 1, \dots, I_{max}$, the greedy sampling algorithm selects the optimal parameter group ρ_I that maximizes the relative error $\|e\|$. However, the computation of the relative error $\|e\|$ is computationally costly as it entails the solution of the full model. Thus, usually, the relative error is replaced by error bounds or the residual $\|p\|$. However, in some cases, it is not possible to obtain error bounds, see [24, 25, 22]. Let ε be the error estimator, i.e., in our case the norm of the residual. At each iteration $i = 1, \dots, I_{max}$, the classical greedy sampling algorithm chooses the parameter group as the maximizer

$$\rho_I = \operatorname{argmax}_{\rho \in \mathcal{P}} \varepsilon(\rho). \quad (4.13)$$

The algorithm is initiated by selecting a parameter group ρ_1 from the parameter set \mathcal{P} and computing a reduced basis Q_1 as in section 4.3.1. Choosing a pre-defined parameter set $\hat{\mathcal{P}}$ of cardinality c randomly from the set \mathcal{P} , at each point of $\hat{\mathcal{P}}$, the algorithm determines a reduced order model with reduced basis Q_1 and then computes error estimator values, $\varepsilon(\rho_j)_{j=1}^c$. The parameter group in $\hat{\mathcal{P}}$ at which the error estimator is maximal is then selected as the optimal parameter group ρ_I . Then the full order model is solved for this parameter group and the snapshot matrix \hat{V} is updated. Finally, a new reduced basis is obtained by computing a truncated singular value decomposition of the updated snapshot matrix, as in Algorithm 1. These steps are then repeated for I_{max} iterations or until the maximum value of the error estimator is lower than the specified tolerance ε_{tol} . The procedure for the classical greedy approach is summarized in Algorithm 2.

Algorithm 2 (Classical greedy sampling).

Input: Maximum number of iterations I_{max} , maximal number of parameter groups c , parameter space \mathcal{P} , tolerance ε_{tol} .

Output: Galerkin projection matrix Q .

Choose first parameter group $\rho_1 = [(a_{11}, \dots, a_{1m}), b, \sigma]$ from \mathcal{P} .

Solve full order model for parameter group ρ_1 and store the results in V_1 .

Compute truncated SVD of the matrix V_1 and construct Q_1 .

Randomly select a set of c parameter groups $\hat{\mathcal{P}} = \{\rho_1, \rho_2, \dots, \rho_c\} \subset \mathcal{P}$.

FOR $i = 2$ to I_{max}

FOR $j = 1$ to c

 Compute reduced model for parameter group ρ_j with reduced basis Q_{i-1} .

 Compute error estimator $\varepsilon(\rho_j)$.

END

Compute $\rho_I = \operatorname{argmax}_{\rho \in \hat{\mathcal{P}}} \varepsilon(\rho)$.



IF $\varepsilon(\rho_I) \leq \varepsilon_{tol}$, then $Q = Q_{i-1}$, STOP.

Simulate the full model for the parameter group ρ_I and store the result in V_i .

Construct snapshot matrix \hat{V} by concatenating the solutions V_ℓ for $\ell = 1, \dots, i$.

Compute a truncated SVD of the matrix \hat{V} and construct Q_i .

END

The classical greedy sampling method computes an inexpensive *a posteriori* error estimator for the reduced model. However, it is not feasible to calculate the error estimator values for the entire parameter space \mathcal{P} . To address this, the classical greedy sampling technique chooses the pre-defined parameter set $\hat{\mathcal{P}}$ randomly as a subset of \mathcal{P} . Random sampling is designed to represent the whole parameter space \mathcal{P} , but there is no guarantee that $\hat{\mathcal{P}}$ will reflect the complete space \mathcal{P} , since the random selection may neglect parameter groups corresponding to the most significant error. These observations motivate us to design a new criterion for the selection of the subset $\hat{\mathcal{P}}$.

Another drawback of the classical greedy sampling technique is that we have to specify the maximum error estimator tolerance ε_{tol} . The error estimator usually depends on some error bound, which may not be tight or may not exist. To overcome this drawback, following ideas in [22], we establish a strategy to construct a surrogate model for the error as a function of the error estimator and we use this error model to control the convergence of the greedy sampling algorithm.

4.3.3. Adaptive Greedy Sampling Method

To overcome drawbacks of the classical greedy sampling approach, we implement adaptive sampling approach which selects the parameter groups adaptively at each iteration of the greedy procedure, using an optimized search based on surrogate modeling. We construct a surrogate model of the error estimator $\bar{\varepsilon}$ to approximate the error estimator ε over the entire parameter space. Further, we use this surrogate model to select the parameter groups $\hat{\mathcal{P}}_k = \{\rho_1, \dots, \rho_{c_k}\}$ with $c_k < c$, where the values of the error estimator are highest. For each parameter group within the parameter set $\hat{\mathcal{P}}_k$, we determine a reduced model and compute the values of the error estimator. Then the process repeats itself until the total number of parameter groups reaches c , resulting in the desired parameter set $\hat{\mathcal{P}}$.

The first stage of the adaptive greedy sampling algorithm computes the error estimator over a randomly selected parameter set $\hat{\mathcal{P}}_0$ of cardinality c_0 . The algorithm uses these error estimator values $\{\varepsilon_i\}_{i=1}^{c_0}$ to build a surrogate model $\bar{\varepsilon}_0$ and locates the c_k parameter groups corresponding to the c_k maximal values of the surrogate model. This process repeats itself for $k = 1, \dots, K$ iterations until the total number of parameter groups reaches c . Finally, the optimal parameter group ρ_I is the one that maximizes the error estimator within the parameter set

$$\hat{\mathcal{P}} = \hat{\mathcal{P}}_0 \cup \hat{\mathcal{P}}_1 \cup \hat{\mathcal{P}}_2 \cup \dots \cup \hat{\mathcal{P}}_K.$$

Thus, at the k th iteration, a surrogate model $\bar{\varepsilon}_k$ is constructed that approximates the error estimator over the entire parameter space \mathcal{P} . There are different choices to build a surrogate model [28]. In this paper, we use the principal component regression (PCR) technique. Suppose $\hat{\varepsilon} = (\varepsilon_1, \dots, \varepsilon_{c_k}) \in \mathbb{R}^{c_k \times 1}$ is the vector of error estimator values at the k th iteration. Since the parameters b and σ are constant, we build a surrogate model with the parameter $a(t)$ only. Let $\hat{\mathcal{P}}_k = [\rho_1, \dots, \rho_{c_k}] \in \mathbb{R}^{c_k \times m}$ be the matrix composed of c_k parameter groups at the k th iteration. The rows of the matrix $\hat{\mathcal{P}}_k$ represent c_k parameter vectors, while the m columns represent m tensor points for the parameter vector $a(t)$. We can fit a simple multiple regression model as

$$\hat{\varepsilon} = \hat{\mathcal{P}}_k \cdot \eta + err, \quad (4.14)$$

where $\eta = [\eta_1, \dots, \eta_m]$ is an array containing regression coefficients and err is an array of residuals. The least



square estimate of η is obtained as

$$\hat{\eta} = \underset{\eta}{\operatorname{argmin}} \|\hat{\varepsilon} - \hat{\mathcal{P}}_k \cdot \eta\|_2^2 = \underset{\eta}{\operatorname{argmin}} \|\hat{\varepsilon} - \sum_{i=1}^m \rho_i \eta_i\|_2^2.$$

If c_k is not much larger than m , then the model may give weak predictions due to the risk of over-fitting for the parameter groups which are not used in model training. Also, if c_k is smaller than m , then the least square approach cannot produce a unique solution, restricting the use of the simple linear regression model. We may face this problem during the first few iterations of the adaptive greedy sampling algorithm, as we will have few error estimator values to build a reasonably accurate model. The principal component regression (PCR) technique is a dimension reduction technique in which m explanatory variables are replaced by p linearly uncorrelated variables called principal components. The dimension reduction is achieved by considering only a few relevant principal components. The PCR approach helps to reduce the problem of estimating m coefficients to the more simpler problem of determining p coefficients. In the following, we describe the method to construct a surrogate model at the k th iteration in detail.

Before performing a principal component analysis (PCA), we center both the response vector $\hat{\varepsilon}$ and the data matrix $\hat{\mathcal{P}}_k$. The PCR starts by performing a PCA of the matrix $\hat{\mathcal{P}}_k$. For this, we compute an SVD

$$\hat{\mathcal{P}}_k = \hat{\Phi} \hat{\Sigma} \hat{\Psi},$$

and then the principal components are the columns of the matrix $\hat{\mathcal{P}}_k \hat{\Psi}$. For dimension reduction, we select only p columns of the matrix $\hat{\Psi}$ to construct a fairly accurate reduced model. In [39] it is suggested that the first three or four principal components are enough to analyze the yield curve changes. Let $Z = \hat{\mathcal{P}}_k \hat{\Psi}_p = [\hat{\mathcal{P}}_k \hat{\psi}_1, \dots, \hat{\mathcal{P}}_k \hat{\psi}_p]$ be the matrix containing first p principal components. We regress $\hat{\varepsilon}$ on these principal components via

$$\hat{\varepsilon} = Z\Omega + \text{err}, \quad (4.15)$$

where $\Omega = [\omega_1, \dots, \omega_p]$ is the vector containing the regression coefficients obtained using the principal components. The least square estimate for Ω is given as

$$\hat{\Omega} = \underset{\Omega}{\operatorname{argmin}} \|\hat{\varepsilon} - Z\Omega\|_2^2 = \underset{\omega}{\operatorname{argmin}} \|\hat{\varepsilon} - \sum_{i=1}^p z_i \omega_i\|_2^2.$$

We obtain the PCR estimate $\eta_{PCR} \in \mathbb{R}^m$ of the regression coefficients η as

$$\eta_{PCR} = \hat{\Psi}_p \hat{\Omega} \quad (4.16)$$

Finally, the value of the surrogate model for any parameter vector $a_\ell = [a_{\ell,1}, \dots, a_{\ell,m}]$ is

$$\bar{\varepsilon}(\rho_\ell) = \eta_1 a_{\ell,1} + \dots + \eta_m a_{\ell,m}. \quad (4.17)$$

Algorithm 3 (Surrogate error model using PCR).

Input: Vector $\hat{\varepsilon} = [\varepsilon_1, \dots, \varepsilon_{c_k}]$, $\hat{\mathcal{P}}_k = [\rho_1, \dots, \rho_{c_k}] \in \mathbb{R}^{c_k \times \bar{m}}$, number of principal components p .

Output: Vector of regression coefficients η .

- Standardize $\hat{\mathcal{P}}_k$ and $\hat{\varepsilon}$ with zero mean and variance one.
- Compute SVD $\hat{\mathcal{P}}_k = \hat{\Phi} \hat{\Sigma} \hat{\Psi}$.
- Construct matrix $Z = \hat{\mathcal{P}}_k \hat{\Psi}_p = [\hat{\mathcal{P}}_k \hat{\psi}_1, \dots, \hat{\mathcal{P}}_k \hat{\psi}_p]$ composed of principal components.
- Compute the least square regression using the principal components as independent variables: $\hat{\Omega} = \underset{\Omega}{\operatorname{argmin}} \|\hat{\varepsilon} - Z\Omega\|_2^2$.
- Compute the PCR estimate $\eta_{PCR} = \hat{\Psi}_p \hat{\Omega}$ of the regression coefficients η .



We will use the error model in the construction of an adaptive greedy sampling method.

4.3.4. Adaptive Greedy Sampling Algorithm

The adaptive greedy sampling algorithm utilizes the designed surrogate model to locate optimal parameter groups adaptively at each greedy iteration $i = 1, \dots, I_{max}$. The first few steps of the algorithm resemble the classical greedy sampling approach. It selects the first parameter group ρ_1 from the parameter space \mathcal{P} and computes the reduced basis Q_1 . Furthermore, the algorithm randomly selects c_0 parameter groups and constructs a temporary parameter set $\hat{\mathcal{P}}_0 = \{\rho_1, \dots, \rho_{c_0}\}$. For each parameter group in the parameter set $\hat{\mathcal{P}}_0$, the algorithm determines a reduced order model and computes an array of residual errors $\varepsilon^0 = \{\varepsilon(\rho_1), \dots, \varepsilon(\rho_{c_0})\}$ obtained for the parameter set $\hat{\mathcal{P}}_0$. Then a surrogate model for the error estimator $\bar{\varepsilon}$ is constructed based on the estimator values $\{\varepsilon(\rho_j)\}_{j=1}^{c_0}$, as discussed in subsection 4.3.3. The obtained surrogate model is then simulated for the entire parameter space \mathcal{P} . Then we locate c_k parameter groups corresponding to the first c_k maximal values of the surrogate model. We then construct a new parameter set $\hat{\mathcal{P}}_k = \{\rho_1, \dots, \rho_k\}$ composed of these c_k parameter groups.

The algorithm determines a reduced model for each parameter group within the parameter set $\hat{\mathcal{P}}_k$ and obtains an array of error estimator values $\varepsilon^k = \{\varepsilon(\rho_1), \dots, \varepsilon(\rho_{c_k})\}$ for the parameter set $\hat{\mathcal{P}}_k$. Furthermore, we concatenate the set $\hat{\mathcal{P}}_k$ and the set $\hat{\mathcal{P}}_0$ to form a new parameter set $\hat{\mathcal{P}} = \hat{\mathcal{P}}_k \cup \hat{\mathcal{P}}_0$. Let $e_{sg} = \varepsilon^0 \cup \dots \cup \varepsilon^k$ be the set composed of all the error estimator values available at the k th iteration. The algorithm then uses this error estimator set e_{sg} to build a new surrogate model. The quality of the surrogate model increases with each iteration as we get more error estimator values. This process is repeated until the cardinality of the set $\hat{\mathcal{P}}$ reaches c , giving

$$\hat{\mathcal{P}} = \hat{\mathcal{P}}_0 \cup \hat{\mathcal{P}}_1 \cup \hat{\mathcal{P}}_2 \cup \dots \cup \hat{\mathcal{P}}_K.$$

Finally, the optimal parameter group ρ_I which maximizes the error estimator (4.13) is extracted from the parameter set $\hat{\mathcal{P}}$. Note that typically it is not necessary to obtain a very accurate sampling using the designed surrogate model. Sampling the full model in the neighborhood of the parameter group with maximal error is sufficient to obtain good results. In the adaptive greedy sampling algorithm, we then use an approximate error \bar{e} determined from the surrogate error model instead of the exact error.

To build an approximate error model, we simulate one full model at each greedy iteration for the optimal parameter group ρ_I , and update the snapshot matrix \hat{V} . A new reduced basis Q is then obtained by computing the truncated singular value decomposition of the updated snapshot matrix as explained in Section 4.3.1. Furthermore, we solve the reduced model for the optimal parameter group before and after updating the reduced basis and obtain the respective error estimator values $\varepsilon^{bef}(\rho_I)$, and $\varepsilon^{aft}(\rho_I)$. Then we compute the relative errors e^{bef}, e^{aft} between the full and reduced model constructed before and after updating the reduced basis.

In this way, at each greedy iteration, we get a set of error values E_p that we use to construct a linear approximate error model for the exact error e based on the error estimator ε .

$$e_p = \{(e_1^{bef}, \varepsilon_1^{bef}) \cup (e_1^{aft}, \varepsilon_1^{aft}), \dots, (e_i^{bef}, \varepsilon_i^{bef}) \cup (e_i^{aft}, \varepsilon_i^{aft})\} \quad (4.18)$$

as

$$\log(\bar{e}_i) = \gamma_i \log(\varepsilon) + \log \tau. \quad (4.19)$$

Setting $\mathcal{Y} = \log(\bar{e})$, $\mathcal{X} = \log(\varepsilon)$ and $\hat{\tau} = \log(\tau)$ we get

$$\mathcal{Y} = \gamma_e \mathcal{X} + \hat{\tau},$$

where γ_e is the slope of the linear model and $\hat{\tau}$ is the intersection with the logarithmic axis $\log(y)$.

After each greedy iteration, we get more data points in the error set E_p , which increases the accuracy of the error model. In Section 4.4, we illustrate that this linear model is sufficient to achieve an accurate error model. The adaptive greedy sampling approach is summarized in Algorithm 4.



Algorithm 4 (Adaptive greedy sampling algorithm).

Input: Maximal number of iterations I_{max} , maximal number of parameter groups c , number of adaptive candidates c_k , parameter space \mathcal{P} , tolerance e_{tol}^{max} .

Output: Q .

Choose first parameter group $\rho_1 = [[a_{11}, \dots, a_{1m}], b, \sigma]$ from \mathcal{P} .

Simulate the full model for parameter group ρ_1 and store the results in V_1 .

Compute a truncated SVD of the matrix V_1 and construct Q_1 .

FOR $i = 2, \dots, I_{max}$

Randomly select a set of parameter groups $\hat{\mathcal{P}}_0 = \{\rho_1, \rho_2, \dots, \rho_{c_0}\} \subset \mathcal{P}$.

FOR $j = 1, \dots, c_0$

determine reduced model for parameter group ρ_j with reduced basis Q_{i-1} .

Compute error estimator $\varepsilon(\rho_j)$.

END

Let $\varepsilon^0 = \{\varepsilon(\rho_1), \dots, \varepsilon(\rho_{c_0})\}$ be error estimators for $\hat{\mathcal{P}}_0$.

Let $k = 1$ and $e_{sg} = \varepsilon^0$.

WHILE $n(\hat{\mathcal{P}}) < c$

Construct a surrogate model $\bar{\varepsilon}(\rho)$ using the values e_{sg} .

Compute the values $\bar{\varepsilon}(\rho)$ of the surrogate model over \mathcal{P} .

Determine first c_k maximum values of $\bar{\varepsilon}(\rho)$ and parameter groups $\hat{\mathcal{P}}_k = \{\rho_1, \dots, \rho_{c_k}\}$.

For $x = 1, \dots, n(\hat{\mathcal{P}}_k)$

Compute reduced basis Q_{i-1} for parameter group ρ_x .

Compute error estimator $\varepsilon(\rho_x)$.

END

Let $\varepsilon^k = \{\varepsilon(\rho_1), \dots, \varepsilon(\rho_{c_k})\}$ be the error estimators for $\hat{\mathcal{P}}_k$.

Update $e_{sg} = \{\varepsilon^0 \cup \dots \cup \varepsilon^k\}$.

Construct new parameter set $\hat{\mathcal{P}} = \hat{\mathcal{P}}_0 \cup \hat{\mathcal{P}}_k$.

$k = k + 1$.

END

Find $\rho_I = \operatorname{argmax}_{\rho \in \hat{\mathcal{P}}} \varepsilon(\rho)$.

IF $i > 2$ and $\bar{e}_i \leq e_{tol}^{max}$, set $Q = Q_{i-1}$, STOP.

Solve full model for the parameter group ρ_I and store result in V_i .

Solve reduced model for the parameter group ρ_I using Q_{i-1} and store result in \bar{V}_i .

Compute relative error e_i^{bef} and error estimator ε_i^{bef} using reduced model obtained with Q_{i-1} , before updating reduced basis. Set $e_i^{bef} = \|V_i(\rho_I) - \bar{V}_i(\rho_I)\| / \|V_i(\rho_I)\|$.

Construct a snapshot matrix \hat{V} by concatenating the solutions V_ℓ for $\ell = 1, \dots, i$.

Compute an SVD of the matrix \hat{V} and construct Q_i .

Simulate reduced model for parameter group ρ_I using Q_i and store result in \bar{V}_{i+1} .

Compute relative error e_i^{aft} and error estimator ε_i^{aft} using the reduced model obtained with Q_i (after updating the reduced basis). Set $e_i^{aft} = \|V_i(\rho_I) - \bar{V}_{i+1}(\rho_I)\| / \|V_i(\rho_I)\|$.

Construct error set $E_p = \{(e_1^{bef}, \varepsilon_1^{bef}) \cup (e_1^{aft}, \varepsilon_1^{aft}), \dots, (e_i^{bef}, \varepsilon_i^{bef}) \cup (e_i^{aft}, \varepsilon_i^{aft})\}$.

Construct model for error \bar{e} using error set E_p : $\log(\bar{e}_i) = \gamma_i \log(\varepsilon) + \log \tau$.



END

4.4. Numerical Example

A numerical example of a floater with cap and floor [30] is used to test the developed algorithms and methods. We model the floater instrument using the Hull-White model and compare the results of the PDE model by discretizing as in subsection 4.3.1 with the results of the reduced model. The reduced model is generated by implementing the POD method along with the classical and the adaptive greedy sampling techniques. The characteristics of the floater instrument are as shown in Table 4.1. The interest rates are capped at $C_R = 2.25\%$

Table 4.1: Numerical Example of a floater with cap and floor.

Coupon frequency	quarterly
Cap rate, C_R	2.25 % p.a.
Floor rate, F_R	0.5 % p.a.
Currency	EURO
Maturity	10 years
Nominal amount	1.0
b	0.015
σ	0.006

p.a. and floored at $C_F = 0.5\%$ p.a. with the reference rate as Euribor3M. The coupon rates can be written as

$$coup = \min(2.25\%, \max(0.5\%, \text{Euribor3M})) \quad (4.20)$$

Note that the coupon rate $coup^{(n)}$ at time t_n is set in advance as the coupon rate at t_{n-1} .

Computer Requirements: All computations are carried out on a PC with 4 cores and 8 logical processors at 2.90 GHz (Intel i7 7th generation). We used MATLAB R2018a for the yield curve simulations. The numerical method for the yield curve simulations is tested with real market based historical data from the State-of-the-art stock exchange information system, "Thomson Reuters EIKON [40]". The daily interest rate data are collected at 26 tenor points in time over the past 5 years, where each year has 260 working days, so there are 1300 observation periods. We have used the inbuilt UnRisk tool for the parameter calibration, which is well integrated with Mathematica (version used: Mathematica 11.3). Further, we used calibrated parameters for the construction of a Hull-White model. We implemented the finite difference method and the model reduction approach for the solution of the Hull-White model in MATLAB R2018a.

The collected historical data has 21 tenor points and 1306 observation periods as follows (D: Day, M: Month, Y: Year):

$$m =: \{1D, 3M, 6M, 1Y, 2Y, 3Y, \dots, 10Y, 12Y, 15Y, 20Y, 25Y, 30Y, 40Y, 50Y\}$$

$$n =: \{1306 \text{ daily interest rates at each tenor point}\}$$

The 10 000 simulated yield curves in 10 years in the future are presented in Fig. 4.1. For the floater example, we need parameter values only until the 10Y tenor point (maturity of the floater). Henceforth, we consider the simulated yield curves with only the first 13 tenor points. The calibration generates the real parameter space of dimension $\mathbb{R}^{10000 \times 13}$ for the parameter $a(t)$. We considered the constant volatility $\sigma = 0.006$ and the constant mean reversion $b = 0.015$. All variable parameters are assumed to be piecewise constants between the tenor points ($0 - 3M, 3M - 6M, 6M - 1Y, 1Y - 2Y, 2Y - 3Y, \dots, 9Y - 10Y$). Figure 4.2 shows 10 000 different piecewise constant parameters $a(t)$.



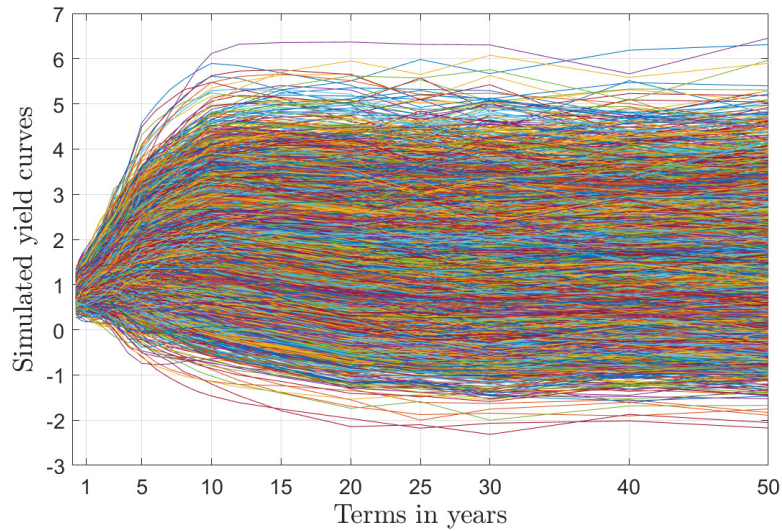


Figure 4.3: 10,000 simulated yield curves obtained by bootstrapping for next 10 years.

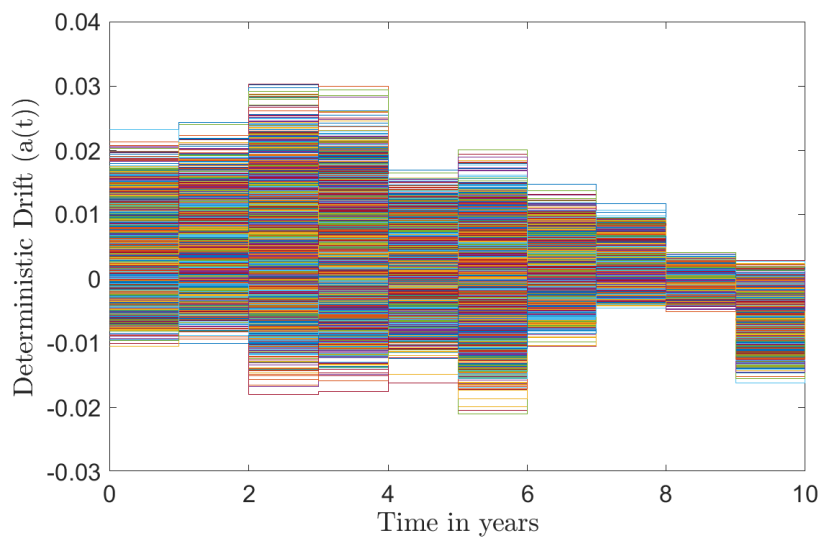


Figure 4.4: 10,000 parameter vectors $a(t)$ as a piecewise function of time.

The computational domain for the interest rate $r(t)$ with an interval $[r_{low}, r_{up}]$, according to [5] is given by

$$r_{low} = r(T) - 7\sigma\sqrt{T}, \quad r_{up} = r(T) + 7\sigma\sqrt{T}, \quad (4.21)$$

where $r(T)$ is the yield at the maturity T also known as a *spot rate*. Here, $r_{low} = -0.1$ and $r_{up} = 0.1$. We applied homogeneous Neumann boundary conditions of the form

$$\frac{\partial V}{\partial r} \Big|_{r=u} = 0, \quad \frac{\partial V}{\partial r} \Big|_{r=v} = 0. \quad (4.22)$$

We divided the spatial domain into $M = 600$ equidistant grid points $\{r(1), r(2), \dots, r(M)\}$ and used the N time points (measured in days) starting from $t = 0$ until maturity T , i.e., in our case, the number of days until



maturity are assumed to be $3600 \approx 10Y$ with an interval $\tau = 1$. Rewriting (4.4), we obtain

$$A(\rho_\ell(t))V^{n+1} = B(\rho_\ell(t))V^n, \quad V(0) = V_0.$$

We can apply the first boundary condition in (4.22) by updating the first and the last rows (A_1 and A_M) of the matrix $A(\rho_\ell)$ which yields

$$A_1 = (-1, 1, 0, \dots, 0) \quad \text{and} \quad A_M = (0, \dots, 0, 1, -1).$$

The second Neumann boundary condition can be applied by changing the last entry of the vector BV^n to zero. Starting at $t = 0$ with the known initial condition $V(0)$ as the principal amount, at each time step, we solve the system of linear equations (4.4).

Note that we need to update the value of the grid point $r(i)$ every three months as the coupon frequency is quarterly by adding coupon f^n based on the coupon rate given by (4.20). The value at the intermediate holding period is calculated by accreting the coupons in between today and the intermediate holding period to the fair value of the instrument.

We have implemented the parametric model reduction approach for the floater example, as discussed in Section 4.3.1. using both the classical and the adaptive greedy sampling algorithm.

At each iteration of the classical greedy sampling approach, the algorithm constructs a reduced basis via Algorithm 1. We have specified a maximum number of 40 pre-defined candidates to construct a set $\hat{\mathcal{P}}$ and a maximum number of iterations $I_{max} = 10$. The progression of the maximal and average residuals with each it-

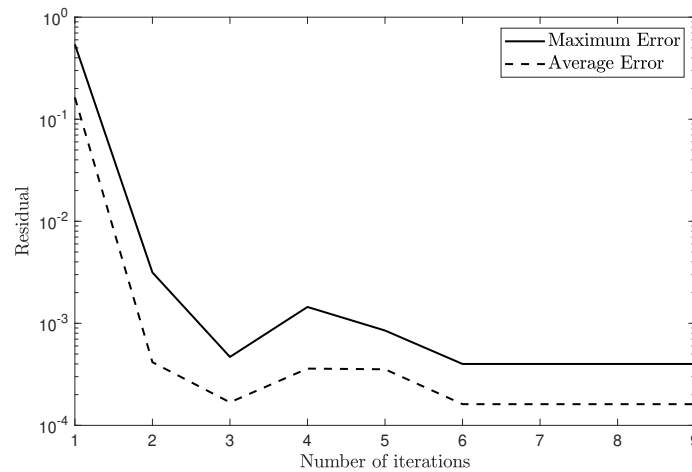


Figure 4.5: Evolution of maximal and average residuals in each iteration of classical greedy algorithm.

eration of the greedy algorithm is presented in Fig. 4.5. It is observed that the maximal residual error typically decreases in the process and the proposed greedy algorithm efficiently locates the optimal parameter groups and constructs the desired reduced basis Q . Furthermore, we tested the effect of change in the cardinalities of the set $\hat{\mathcal{P}}$. The proposed algorithm is applied with three different cardinalities of $\hat{\mathcal{P}}$: $|\hat{\mathcal{P}}_1| = 20$, $|\hat{\mathcal{P}}_2| = 30$, $|\hat{\mathcal{P}}_3| = 40$. Note that we have constructed $\hat{\mathcal{P}}$ by randomly selecting the parameter groups from the parameter space \mathcal{P} .

Figure 4.6 shows the plot of the maximal residual against the number of iterations for three different cardinalities.



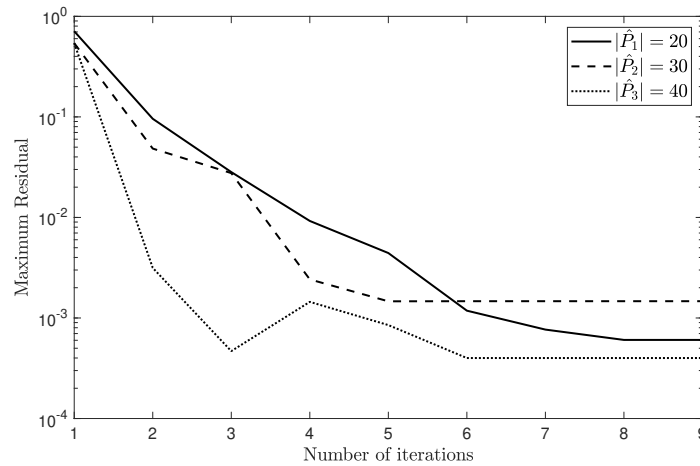


Figure 4.6: Evolution of the maximum residual error for three different cardinalities of set \hat{P} .

It is evident that with an increasing number of candidates, the maximal residual error decreases and the decrease is sufficient for a cardinality 20 of \hat{P} .

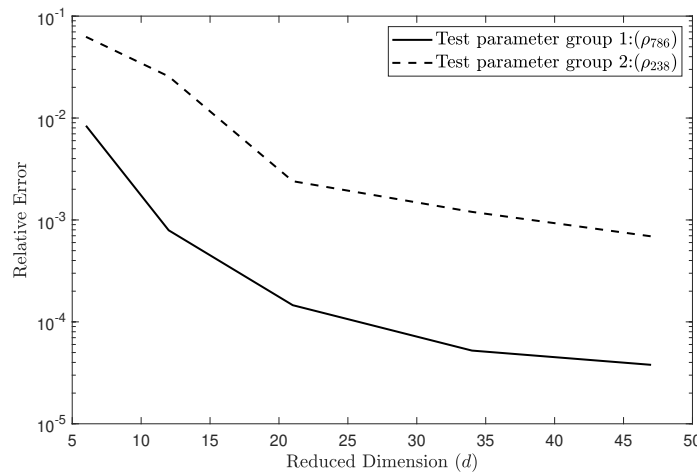


Figure 4.7: Relative error between full and reduced model for two different parameter groups.

During our tests we noticed that there are some parameter groups (e.g., ρ_{238}) for which the reduced model gives unsatisfactory results. Figure 4.7 illustrates the relative error between the full model and the reduced-order model for two different parameter groups. One can observe that the reduced model for the parameter group ρ_{238} (dashed line) shows inferior results as compared to the reduced model for the parameter group ρ_{786} (solid line). Even an increase in the reduced dimension d does not improve the quality of the result substantially. This reveals that the selection of trial candidates by random sampling may neglect parameter groups corresponding to the significant error.

To overcome this drawback, we have also implemented the adaptive greedy sampling approach for the floater example. At each greedy iteration, the algorithm locates $c_k = 10$ parameter groups adaptively using the surrogate modeling technique, as described in Subsection 4.3.3. We have fixed the maximum number of elements within the parameter set \hat{P} to 40. Furthermore, the adaptively obtained parameter set \hat{P} has been used to locate the optimal parameter group ρ_I . These steps are repeated for a maximum of $I_{max} = 10$ iterations or until convergence. The algorithm has been initiated by selecting $c_0 = 20$ random parameter groups.

The optimal parameter group updates the snapshot matrix, and consecutively the algorithm generates a new reduced basis at each greedy iteration. Figure 4.8 shows the evolution of maximum and average residual errors



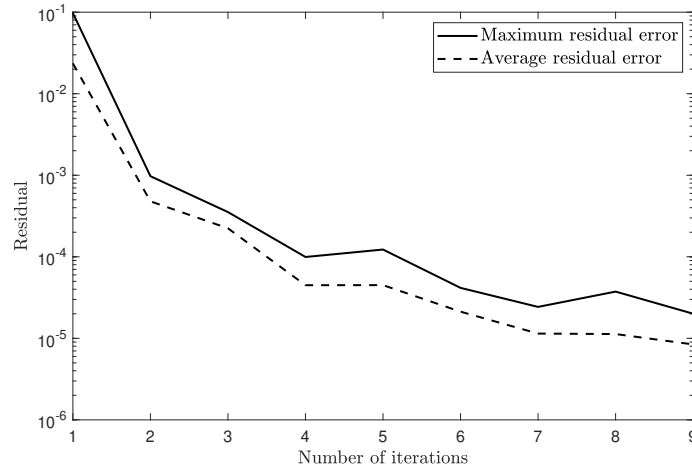


Figure 4.8: Maximal and average residuals per iteration of adaptive greedy algorithm.

with each iteration of the adaptive greedy algorithm. The residual error decreases with each incrementing iteration and hence the algorithm succeeded in locating the optimal parameter group efficiently.

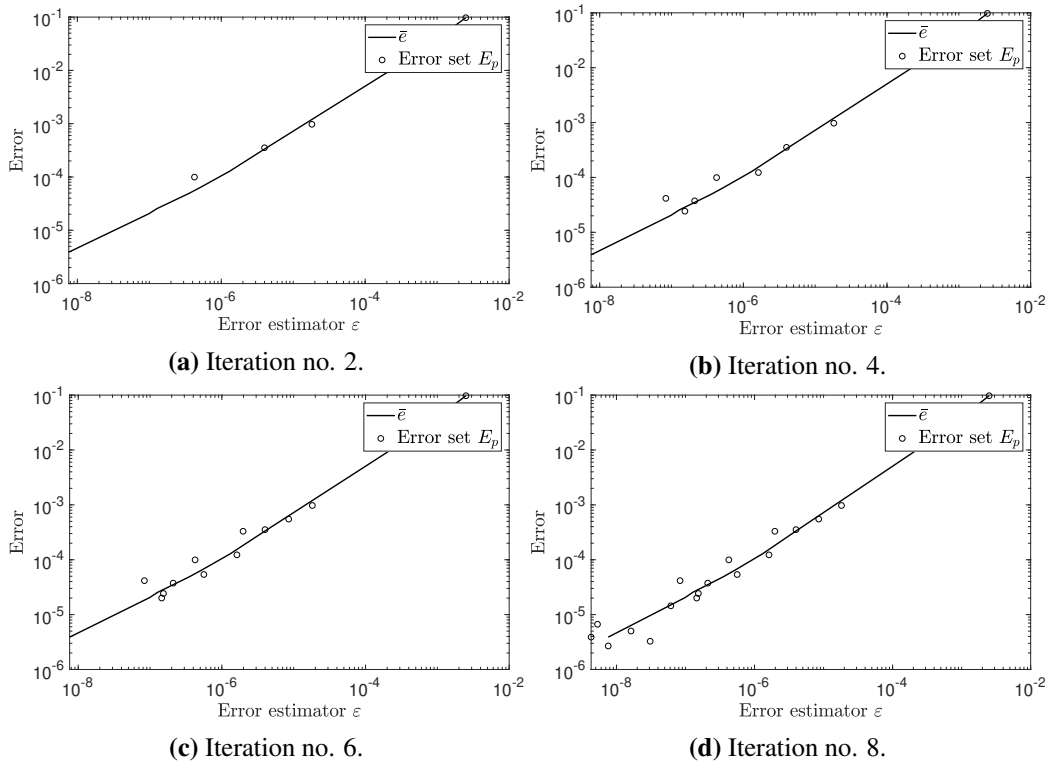


Figure 4.9: Error model \bar{e} based on error set E_p for four different greedy iterations.

To monitor the convergence of the adaptive greedy algorithm, we have designed an error model \bar{e} for the relative error e as a function of the residual error ϵ . Figure 4.9 shows the designed error model based compared to the available error set E_p for four different greedy iterations. The error plot exhibits a strong correlation between the relative error and the residual error. The results indicate that a linear error model is satisfactory to capture the overall behavior of the exact error as a function of the residual error.

We have used the reduced basis obtained from the adaptive greedy sampling procedure to design the reduced model. Figure 4.10 presents the relative error plot for the parameter groups ρ_{238} , and ρ_{786} . We see that the



adaptive greedy approach gives better results than the classical greedy method. With a reduced dimension of $d = 6$, we obtained an excellent result as the relative error is less than 10^{-3} .

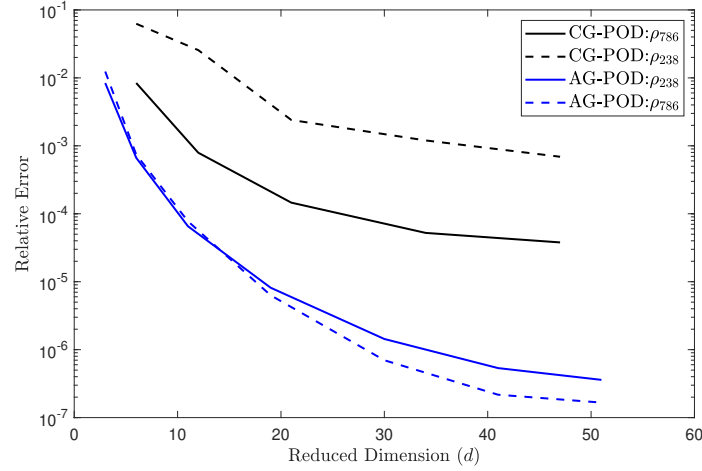


Figure 4.10: Comparison of classical (CG) and adaptive (AG) greedy sampling approach.

4.4.1. Computational cost

In the case of the classical greedy sampling approach, the algorithm solves c reduced models and one full model at each greedy iteration. It also computes a truncated SVD of the updated snapshot matrix with each proceeding iteration. Let t_{RM} be the time required to solve one reduced model, t_{FM} the computational time required for one full model, and t_{SVD} the time required to obtain a truncated SVD of the snapshot matrix. The total computational time T_Q^{CG} required to obtain the reduced basis in the case of the classical greedy sampling approach can be given as

$$T_Q^{CG} \approx \left[c \times t_{ROM} + (t_{HDM} + t_{SVD}) \right] \times i.$$

Similarly, in the case of the adaptive greedy approach, the total computational time T_Q^{AG} can be given as

$$T_Q^{AG} \approx \left[c_0 \times t_{ROM} + \underbrace{k(C_k \times t_{ROM} + t_{SM} + t_{SM}^{ev})}_{t_{\rho_I}} + t_{HDM} + t_{SVD} + 2t_{ROM}^{af,bf} + t_{EM} \right] \times i,$$

where t_{SM} and t_{SM}^{ev} denote the computational times required to build and evaluate a surrogate model for the entire parameter space respectively. t_{EM} is the time required to build an error model. The term $2t_{ROM}^{af,bf}$ shows the computational time needed to solve the reduced model after and before updating the reduced basis. Table

Table 4.2: Computation time/ reduction time (T_Q) to generate projection subspace.

Algorithm	Cardinality $ \hat{P} $	Max. no. iterations I_{max}	Computational time
Classical greedy sampling	20	10	56.82 s
Classical greedy sampling	30	10	82.54 s
Classical greedy sampling	40	10	95.04 s
Adaptive greedy sampling	40	10	183.21 s

4.2 compares the computational times required to generate the reduced basis Q for different sets of \hat{P} in case



of the classical greedy sampling approach with that needed to obtain the reduced basis in case of the adaptive greedy sampling approach.

Table 4.3: Evaluation time.

Algorithm	Model	Eva. time single ρ_s	Total Eva. time (T_{eva})	Total time $T_Q + T_{\text{eva}}$
	FM, $M = 600$	0.2193 s	2193.72 s	2193.72 s
Classical greedy sampling	RM, $d = 5$	0.0102 s	102.56 s	197.60 s
Classical greedy sampling	RM, $d = 10$	0.0125 s	125.48 s	220.52 s
Adaptive greedy sampling	RM, $d = 6$	0.0104 s	104.38 s	287.59 s
Adaptive greedy sampling	RM, $d = 10$	0.124 s	124.32 s	307.53 s

The computational time t_{ρ_I} required to locate the optimal parameter group by constructing a surrogate model for one greedy iteration is approximately 8 seconds. t_{ρ_I} is nothing but the time required to complete a while loop outlined in Algorithm 4 for a single greedy iteration. Thus, the total time contributed to generate the reduced basis via surrogate modeling is $I_{\text{max}} \times t_{\rho_I} = 78.56$ s, considering the adaptive greedy algorithm runs for I_{max} iterations. Figure 4.10 shows that we can truncate the algorithm after 4 or 5 iterations as the residual error falls below 10^{-4} .

The computational times required to simulate reduced models and full models are presented in Table 4.3. The time required to solve the complete system with a parameter space of $10000 \times m$ for both a full and reduced model is given in the total time column.

We can see that the evaluation time required for the reduced model is at least 18 – 20 times less than that of the full model. However, there is a slight increase in total time due to the addition of the reduction time T_Q . One can also observe that with an increase in the dimension d of the reduced model, the evaluation time increases significantly. The reduced model with the classical greedy sampling approach is at least 10 – 11 times faster than that for the full model. The time required to simulate the reduced model with the adaptive greedy sampling approach is a bit higher due to the time invested in building surrogate and error models. Despite of that, the reduced model is at least 8 – 9 times faster than the full model. The computational time presented in both tables considers that the greedy algorithms run for the maximal number of iterations I_{max} . However, we can truncate the algorithms after 4 or 5 iterations, i.e., we can practically achieve even more speedup than discussed here.

4.4.1.1. Floater Scenario Values

Table 4.4: Results for a floater with cap and floor.

Scenario	5 years	10 years
Favorable (90th percentile)	1.0759	1.0829
Moderate (50th percentile)	1.0578	1.0615
Unfavorable (10th percentile)	1.0183	1.0254

To design a key information document, we need the values of the floater at different spot rates. The spot rate r_{sp} is the yield rate at the first tenor point from the simulated yield curve. The value of a floater at the spot rate

r_{sp} is nothing but the value at that short rate $r = r_{sp}$. For 10 000 simulated yield curves, we get 10 000 different spot rates and the corresponding values for the floater. These values are further used to calculate three different scenarios: (i) favorable scenario, (ii) moderate scenario, (iii) unfavorable scenario which are the values at 90th percentile, 50th percentile and 10th percentile of 10 000 values, respectively.

4.5. Conclusion

This paper presents a parametric model reduction approach for a discretized convection-diffusion-reaction PDE that arises in the analysis of financial risk. The parameter space with time-dependent parameters is generated via the calibration of financial models based on market structures. A finite difference method has been implemented to solve this PDE. The selection of parameters to obtain the reduced basis is of utmost importance. We have established a greedy approach for parameter sampling, and we noticed that there are some parameter groups for which the classical greedy sampling approach gave unsatisfactory results. To overcome this drawback, we have applied an adaptive greedy sampling method using a surrogate model for the error estimator that is constructed for the entire parameter space and further used to locate the parameters which most likely maximize the error estimator. The surrogate model is built using the principal component regression technique. We tested the designed algorithms for a numerical example of a floater with cap and floor solved using the Hull-White model. The results indicate the computational advantage of the parametric model reduction technique for the short-rate models. A reduced model of dimension $d = 6$ was enough to reach an accuracy of 0.01%. The reduced model was at least 10 – 12 times faster than that of the full model. The developed model order reduction approach shows potential applications in the historical or Monte Carlo value at risk calculations as well, where a large number of simulations need to be performed for the underlying instrument.

4.A. Relation between a singular value decomposition and a principal component analysis

Consider a data matrix X of size $n \times m$, where n is the number of samples and m is the number of variables. Let us also assume that the data matrix X is centered. We can decompose the matrix X using singular value decomposition (SVD) as

$$X = \Phi \Sigma \Psi^T, \quad (4.23)$$

where the matrices Φ and Ψ contain left and right singular vectors of the matrix X respectively. The matrix Σ is a diagonal matrix having singular values Σ_i arranged in descending order along the diagonal. Consider a covariance matrix \mathcal{C} of order $m \times m$ as follows:

$$\mathcal{C} = X^T X. \quad (4.24)$$

We now compute an eigendecomposition, also known as spectral decomposition of the covariance matrix as

$$\mathcal{C} = X^T X = \Psi \bar{\Sigma} \Psi^T, \quad (4.25)$$

where Ψ is the matrix of eigenvectors and $\bar{\Sigma}$ is a diagonal matrix having eigenvalues λ_i arranged in descending order along the diagonal. In the principal component analysis (PCA), the columns of a matrix $X \Psi$ are known as the principal components, while the columns of the matrix Ψ are known as principal directions or PCA loading. Furthermore, one can easily see the similarities between the SVD and the PCA. We can write a covariance matrix as follows

$$\begin{aligned} X^T X &= (\Phi \Sigma \Psi^T)^T \Phi \Sigma \Psi^T, \\ X^T X &= \Psi \Sigma \Phi^T \Phi \Sigma \Psi^T, \\ X^T X &= \Psi \Sigma^2 \Psi^T, \end{aligned} \quad (4.26)$$

where $\Phi^T \Phi = I$. Now we can see that the right singular vectors of the matrix X are simply the eigenvectors of the matrix \mathcal{C} , and the singular values of X are related to the eigenvalues of \mathcal{C} as $\lambda_i = \Sigma_i^2$.



Glossary

Packaged retail investment and insurance-based products Packaged retail investment and insurance-based products are a broad class of financial instruments that are provided to customers in the EU through banks or financial institutions, which include stocks, bonds, insurance policies, structured funds, structured deposits, and structured products. 46

key information document A key information document is a 3-page document provided to the customers by financial institutions for the invested product. It includes the risk and reward profile of the product, the cost of the product, the recommended holding period, possible outcomes, etc., so that investors can understand the product easily.. 46

yield curve The yield curve is the curve showing interest rates plotted against different maturities for the same financial instrument. 46

cap An upper limit on the interest rate for a floating interest rate instrument. 46

swaption a. Swap: A swap is an agreement between two parties to exchange financial instruments for a certain time.

b. Swaption: A swaption is an option to enter into a swap. 46

tenor Tenor refers to the amount of time left until the financial instrument expires. The maturity time points on a yield curve are also known as tenor points. 46

Bibliography

- [1] European Commission, “Commission delegated regulation (EU) 2017/653 OJ L 100,” *Off. J. EU*, vol. 1, pp. 1–52, 2017.
- [2] A. Gupta and M. Subrahmanyam, “Pricing and hedging interest rate options: Evidence from cap-floor markets,” *J. Bank. Finance*, vol. 29, pp. 701–733, 2005.
- [3] J. Bicksler and A. Chen, “An economic analysis of interest rate swaps,” *J. Finance*, vol. 3, pp. 645–655, 1986.
- [4] D. Brigo and F. Mercurio, *Interest Rate Models - Theory and Practice*, 2nd ed. Berlin: Springer-Verlag, 2006.
- [5] M. Aichinger and A. Binder, *A Workout in Computational Finance*, 1st ed. West Sussex, UK: John Wiley and Sons Inc., 2013.
- [6] E. Ekström, P. Lötstedt, and J. Tysk, “Boundary values and finite difference methods for the single factor term structure equation,” *J. Appl. Math. Finance*, vol. 16, pp. 253–259, 2009.
- [7] T. Haentjens and K. I. Hout, “Alternating direction implicit finite difference schemes for the Heston-Hull-White partial differential equation,” *J. Comput. Finance*, vol. 16, pp. 83–110, 2012.
- [8] A. Falcó, L. Navarro, and C. Cendón, “Finite difference methods for hull-white pricing of interest rate derivatives with dynamical consistent curves,” *Soc. Sci. Res. Net. Elec. J.*, 2014.
- [9] M. Briani, L. Caramellino, and A. Zanette, “A hybrid tree/finite-difference approach for Heston–Hull–White-type models,” *J. Comput. Finance*, vol. 21, no. 3, 2017.
- [10] A. Cohen and R. DeVore, “Approximation of high-dimensional parametric PDEs,” *Acta Numerica*, vol. 24, pp. 1–159, 2015.
- [11] I. Piotr and R. Motwani, “Approximate nearest neighbors: Towards removing the curse of dimensionality,”



- in *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*. New York, NY, USA: ACM Press, 1998, pp. 604–613.
- [12] A. Chatterjee, “An introduction to the proper orthogonal decomposition,” *Curr. Sci.*, vol. 78, pp. 808–817, 2000.
- [13] G. Berkooz, P. Holmes, and J. Lumley, “The proper orthogonal decomposition in the analysis of turbulent flows,” *Annu. Rev. Fluid Mech.*, vol. 25, no. 1, pp. 539–575, 1993.
- [14] M. Graham and I. Kevrekidis, “Alternative approaches to the Karhunen-Loève decomposition for model reduction and data analysis,” *Comput. Chem. Eng.*, vol. 20, no. 5, pp. 495–506, 1996.
- [15] I. Jolliffe, *Principal Component Analysis*, 1st ed. Berlin: Springer-Verlag, 2014.
- [16] M. Graham and I. Kevrekidis, “Proper orthogonal decomposition and its applications part I: Theory,” *J. Sound Vib.*, vol. 252, no. 3, pp. 527–544, 2002.
- [17] M. Rathinam and L. Petzold, “A new look at proper orthogonal decomposition,” *SIAM J. Numer. Anal.*, vol. 41, no. 5, pp. 1893–1925, 2003.
- [18] A. Vidal and D. Sakrison, “On the optimality of the Karhunen-Loève expansion (corresp.),” *IEEE Trans. info. theory*, vol. 15, no. 2, pp. 319–321, 1969.
- [19] L. Sirovich, “Turbulence and the dynamics of coherent structures. Part I: coherent structures,” *Quart. Appl. Math.*, vol. 45, no. 3, pp. 561–571, 1987.
- [20] J. Lucia, P. Beran, and W. Silva, “Reduced order modeling: new approaches for computational physics,” *Prog. Aerosp. Sci.*, vol. 40, no. 1-2, pp. 51–117, 2004.
- [21] M. Grepl and A. Patera, “A posteriori error bounds for reduced-basis approximations of parametrized parabolic partial differential equations,” *M2AN, Math. Model. Numer. Anal.*, vol. 39, pp. 157–181, 2005.
- [22] A. Paul-Dubois-Taine and D. Amsallem, “An adaptive and efficient greedy procedure for the optimal training of parametric reduced-order models,” *Int. J. Numer. Meth. Engng.*, vol. 102, pp. 1262–1292, 2014.
- [23] D. Amsallem, M. Zahr, and Y. Choi, “Design optimization using hyper-reduced-order models,” *Struct. Multidisc. Optim.*, vol. 51, no. 4, pp. 919–940, 2015.
- [24] T. Bui-Thanh, K. Willcox, and O. Ghattas, “Model reduction for large-scale systems with high-dimensional parametric input space,” *SIAM J. Sci. Comput.*, vol. 30, no. 6, pp. 3270–3288, 2008.
- [25] K. Veroy and A. Patera, “Certified real-time solution of the parametrized steady incompressible Navier–Stokes equations: rigorous reduced-basis a posteriori error bounds,” *Int. J. Numer. Meth. Fluids*, vol. 47, pp. 773–788, 2005.
- [26] H. Lee, Y. Park, and S. Lee, “Principal component regression by principal component selection,” *Commun. Stat. Appl. Methods*, vol. 22, no. 2, pp. 173–180, 2015.
- [27] D. Jones, “A taxonomy of global optimization methods based on response surfaces,” *J. Global Optim.*, vol. 21, no. 4, pp. 345–383, 2001.
- [28] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning*, 1st ed. New York, NY: Springer-Verlag, 2013.
- [29] J. Hull and A. White, “Pricing interest-rate-derivative securities,” *Rev. Financial Stud.*, vol. 3, no. 4, pp. 573–592, 1990.
- [30] F. Fabozzi, *Valuation of Fixed Income Securities and Derivatives*, 3rd ed. John Wiley and Sons Inc., 1998.
- [31] O. Vasicek, “An equilibrium characterization of the term structure,” *J. Financial Econ.*, vol. 5, no. 2, pp. 177–188, 1977.
- [32] J. Cox, J. Ingersoll Jr., and S. Ross, “A theory of the term structure of interest rates,” *Econometrica*, vol. 53, no. 2, pp. 385–407, 1985.



-
- [33] J. Hull and A. White, “One-factor interest-rate models and the valuation of interest-rate derivative securities,” *J. Financ. Quant. Anal.*, vol. 28, no. 2, pp. 235–254, 1993.
- [34] —, “Numerical procedures for implementing term structure models I: single-factor models,” *J. Deriv.*, vol. 2, pp. 7–16, 1994.
- [35] G. Darbellay, “A note on the volatility term structure in short rate models,” *J. Appl. Math. Mech.*, vol. 78, pp. 885–886, 1998.
- [36] G. Golub and C. Reinsch, “Singular value decomposition and least squares solutions,” *Numer. Math.*, vol. 24, pp. 403–420, 1970.
- [37] M. Williams, P. Schmid, and J. Kutz, “Hybrid reduced-order integration with proper orthogonal decomposition and dynamic mode decomposition,” *SIAM J. Multiscale Model. Simul.*, vol. 11, no. 2, pp. 522–544, 2013.
- [38] R. Pinnau, “Model reduction via proper orthogonal decomposition,” in *Model Order Reduction: Theory, Research Aspects and Applications*, W. Schilders, H. van der Vorst, and J. Rommes, Eds. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 95–109.
- [39] A. Binder, “A clever handful is enough,” *Wilmott*, pp. 10–14, 2007.
- [40] MathConsult, “Calibration of interest rate models,” MathConsult GmbH, Linz, Austria, Report, 2009.

5. Software-based representation of an inverse heat conduction problem

Patricia Barral¹, Federico Bianco², Riccardo Conte², Umberto Emil Morelli³, Peregrina Quintela¹, Gianluigi Rozza⁴, Giovanni Stabile⁴

¹*Instituto Tecnológico de Matemática Industrial, Universidade de Santiago de Compostela*

²*Danieli*

³*Instituto Tecnológico de Matemática Industrial*

⁴*Scuola Internazionale Superiore di Studi Avanzati*

Abstract. In continuous casting molds, it is important to know the steel-mold heat flux in real time for a proper control of the process. This heat flux can be computed using thermocouples measurements inside the mold and solving an inverse problem. To validate, analyze and compare different techniques for the solution of this inverse problem, we design a benchmark case. In this document, we provide a description of the mathematical formulation of the problem and a methodology for computing its numerical solution. Then, we discuss its software implementation. Finally, we present the benchmark case with the related numerical results and all the required data for its reproducibility.

Keywords: Inverse problem, heat transfer, continuous casting mold, boundary flux estimation

5.1. Introduction

Continuous casting of steel is presently the most used process to produce steel worldwide. Figure 5.1(a) provides a schematic of the process. These casters are complex machinery, the most critical part for the process being the mold. Here the steel undergoes to its primary solidification. The mold extracts heat from the liquid steel thanks to a liquid cooling system. This system is composed of drilled channels in which water flows at high flow rate and pressure (see Figure 5.1(b)).

For safety and quality reasons, it is important to control the heat extraction from the steel during the casting. For example, if the heat extraction is too little the steel solid skin is thin at its exit can brake. For this and other reasons, it is essential to know the real time behavior of the mold to properly control the casting process.

One way to compute this heat flux could be to simulate all the phenomena happening inside the mold: from the tapping of the molten steel to the secondary cooling region (e.g. multiphase flow, heat transport, solidification, thermodynamic reactions etc.). However, the resulting model would be quite complex and computationally expensive to deal with, especially for real time applications. Then, this option was discarded.

Also the fully experimental approach is no feasible since it is not possible to make direct measurements in the solidification region. The only measurements available are made by thermocouples that are buried inside the mold plates. They provide temperature measurements few centimeters into the mold. Then, our approach to study the real time behavior of continuous casting molds and the mold-slab heat flux is to solve an inverse problem having as data the thermocouples measurements.

In modeling the thermal behavior of the mold, we consider the following well established assumptions:

- The copper mold is assumed an homogeneous and isotropic solid material;
- The cooling water is a isotropic, Newtonian and incompressible fluid;
- The thermal expansion of the mold and its mechanical distortion are negligible;
- The material properties are assumed constant;
- The boundaries in contact with air are assumed adiabatic;
- No boiling in the water is assumed;



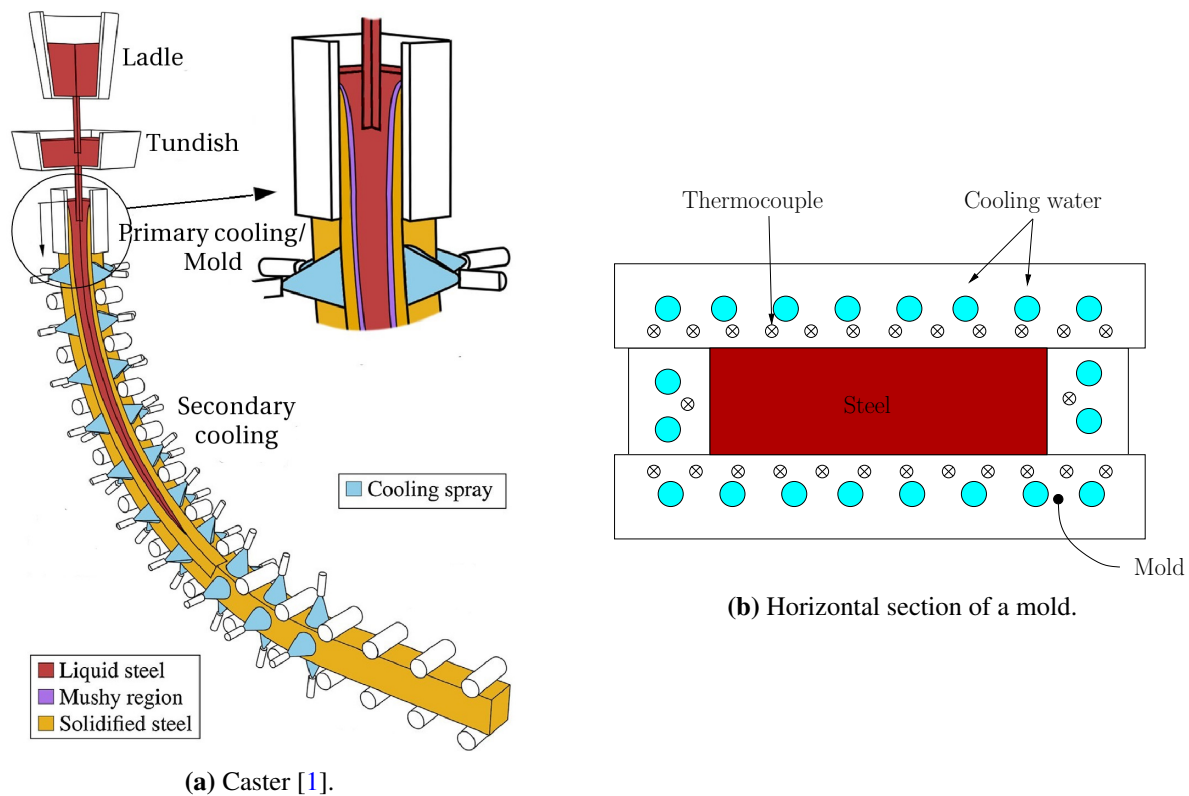


Figure 5.1: Schematic of a continuous caster (a) and of a cross section of a mold (b).

- The heat transmitted by radiation is neglected.

Notice that the running parameters of the cooling system and its geometry ensure a fully developed turbulent flow. In fact, these molds are equipped with a closed loop cooling system, the water is pumped at a high pressure and the average velocity in each cooling channel is approximately 10 m/s, the diameter being approx. 10 mm. Thus, the Reynolds number in the cooling system is around 10^5 , which ensures a turbulent flow. Since we want to have solution in real-time (e.g. at each second) and the casting speed is of few meter per minute, we consider steady-state models. Moreover, we only consider 3D mold models because we are interested in the heat flux in all the mold-slab interface. By adding specific assumptions at each simplification step, we propose now the hierarchy of mold thermal models of Figure 5.2.

According to the mentioned assumptions, we consider the following physical problems:

(M1): The domain is composed of the (solid) copper mold and (liquid) cooling water. Then, a liquid-solid steady-state three-dimensional heat transfer model one-way coupled with a turbulent fluid flow model for the fluid dynamics of the cooling water is considered.

The one-way coupling is in the sense that we neglect the effects that changes in the fluid temperature have on the fluid dynamics (i.e. buoyancy).

Notice that in the mold, we consider the cooling water flowing in circular channels with constant cross section. Then, we can consider simplified 1D models for the turbulent flow in pipes (see e.g. [2], Chapter 7). These models give the shape of the averaged boundary layer in a pipe and the velocity has only one component, the vertical component along the axis of the pipe. Thus, we must pay attention in making this simplification for the following model since we neglect the heat transport due to turbulence eddies and mixing. Therefore, we derive the following simplified version of the previous model:

(M2): The domain is composed of the (solid) copper mold and (liquid) cooling water. Then, a liquid-solid steady-state three-dimensional heat transfer model one-way coupled with a simplified pipe flow model



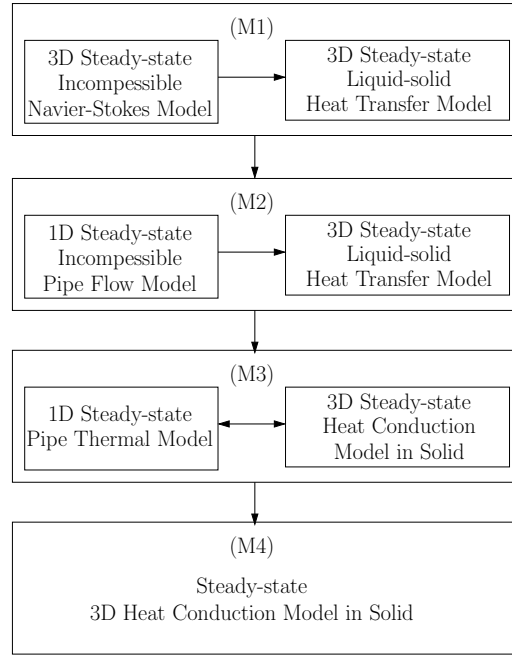


Figure 5.2: Scheme of the models hierarchy for the continuous casting mold.

for the fluid dynamics of the cooling water is considered.

Thanks to the high Reynolds number of the flow, we can further assume that the cooler and hotter water molecules are well mixed. Such that, the temperature in each section of the cooling channel is approximately constant. Moreover the water is pumped in a closed circuit, so we can assume that the water flow rate is constant. In turn, since the channels have constant section, the velocity of the fluid is also uniform and constant (plug flow). Making these assumptions, we end up with the following model which is a three dimensional version of the 2D model proposed by Samarasekera and Brimacombe [3]:

(M3): The domain is composed of the (solid) copper mold and (liquid) cooling water. A steady-state three-dimensional heat conduction model with a convective BC in the portion of the boundary in contact with the cooling water is considered in the solid domain. Assuming constant temperature in each transversal section of the cooling system, the 1D water thermal model is given by a cross sectional heat balance. In this case the two models are two-way coupled by the boundary condition.

Finally, considering that the temperature increase of the cooling water is of only few degrees, we can assume that the water temperature is known. Then, we consider the following simplification of the previous model:

(M4): The domain is composed of the (solid) copper mold. A steady-state three-dimensional heat conduction model with a convective BC in the portion of the boundary in contact with the cooling water is considered. Here, the water temperature is assumed to be known.

In the following, we provide the mathematical formulation of model (M4) and its numerical solution. Then, we formulate the respective inverse problem discussing the methodology for its solution. The core of this document is then the discussion on the implementation of the direct and inverse models in a software and their application to the proposed benchmark case.

5.2. Mathematical Formulation

5.2.1. Computational Domain and Notation

Consider a solid domain, Ω , which is an open Lipschitz bounded subset of \mathbb{R}^3 , with smooth boundary Γ (see Figure 5.3). Let $\Gamma = \Gamma_{in} \cup \Gamma_I \cup \Gamma_{II} \cup \Gamma_{III} \cup \Gamma_{IV} \cup \Gamma_{sf}$ where $\dot{\Gamma}_{in}$, $\dot{\Gamma}_I$, $\dot{\Gamma}_{II}$, $\dot{\Gamma}_{III}$, $\dot{\Gamma}_{IV}$ and $\dot{\Gamma}_{sf}$ are



two dimensional open disjoint sets. The Eulerian Cartesian coordinate vector is denoted by $\mathbf{x} \in \Omega$ and $\mathbf{n}(\mathbf{x})$ represents the unit normal vector that is directed outwards from Ω .

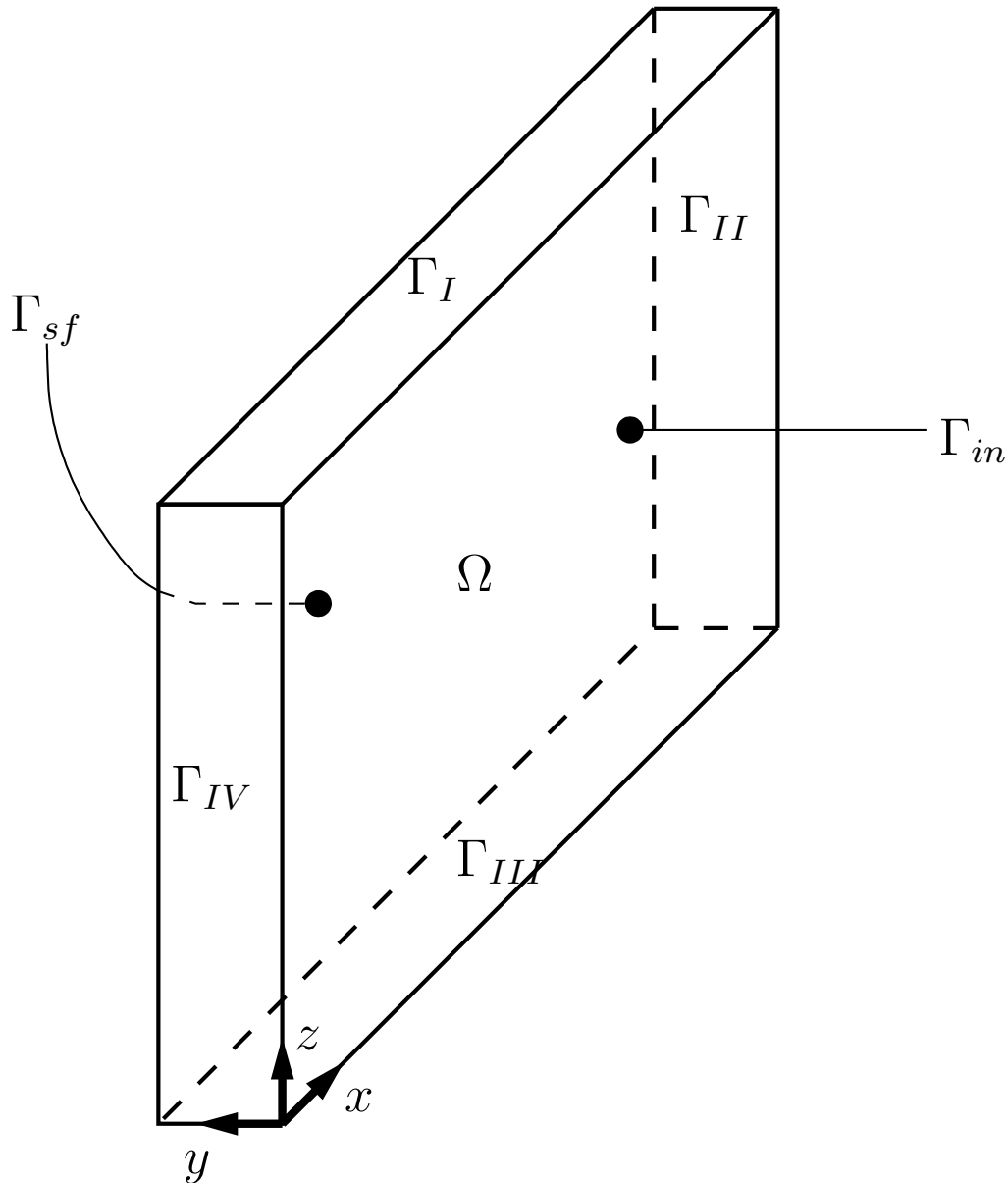


Figure 5.3: Schematic of the solid domain.

In this setting, Ω corresponds to the region of the space occupied by the solid part of the mold. The boundary Γ_{sf} is the interface between the mold and the cooling system. The boundaries Γ_I , Γ_{II} , Γ_{III} and Γ_{IV} are in contact with the air surrounding the mold and Γ_{in} is the portion of the mold boundary in contact with the solidifying steel.

5.2.2. Direct Heat Transfer Problem

The objective of the direct problem is to compute the absolute temperature in Ω assuming known the heat flux through the surface of the mold in contact with the hot steel, Γ_{in} . In what follows, we denote the absolute temperature by T . We shall assume all along the following assumptions on the data:

(H1): The thermal conductivity is constant and strictly positive: $k \in \mathbb{R}^+$.



(H2): The heat fluxes, q_I , q_{II} , q_{III} and q_{IV} , belong to $L^2(\Gamma_I)$, $L^2(\Gamma_{II})$, $L^2(\Gamma_{III})$ and $L^2(\Gamma_{IV})$ respectively.

(H3): There are not heat sources inside the mold.

(H4): The heat transfer coefficient is constant and strictly positive: $h \in \mathbb{R}^+$.

(H5): The cooling water temperature, T_f , is known and belongs to $L^2(\Gamma_{sf})$.

(H6): The heat flux at the mold-steel interface, g , belongs to $L^2(\Gamma_{in})$.

According to these assumptions, we model the mold thermal behavior as a diffusion problem. Moreover, we model the cooling due to the water with a convection condition at the boundary Γ_{sf} . Then, we write our mold thermal model as

Problem 1. Find T such that

$$-k\Delta T = 0, \text{ in } \Omega, \quad (5.1)$$

with BCs

$$\left\{ \begin{array}{ll} -k\nabla T \cdot \mathbf{n} = g & \text{on } \Gamma_{in}, \\ -k\nabla T \cdot \mathbf{n} = q_I & \text{on } \Gamma_I, \\ -k\nabla T \cdot \mathbf{n} = q_{II} & \text{on } \Gamma_{II}, \\ -k\nabla T \cdot \mathbf{n} = q_{III} & \text{on } \Gamma_{III}, \\ -k\nabla T \cdot \mathbf{n} = q_{IV} & \text{on } \Gamma_{IV}, \\ -k\nabla T \cdot \mathbf{n} = h(T - T_f) & \text{on } \Gamma_{sf}. \end{array} \right. \quad (5.2)$$

$$-k\nabla T \cdot \mathbf{n} = q_I \quad \text{on } \Gamma_I, \quad (5.3)$$

$$-k\nabla T \cdot \mathbf{n} = q_{II} \quad \text{on } \Gamma_{II}, \quad (5.4)$$

$$-k\nabla T \cdot \mathbf{n} = q_{III} \quad \text{on } \Gamma_{III}, \quad (5.5)$$

$$-k\nabla T \cdot \mathbf{n} = q_{IV} \quad \text{on } \Gamma_{IV}, \quad (5.6)$$

$$-k\nabla T \cdot \mathbf{n} = h(T - T_f) \quad \text{on } \Gamma_{sf}. \quad (5.7)$$

We recall that for this problem the following result is well established (see [4], Theorem 3.14 and [5], Theorem 3.3.5):

Theorem 1. Under assumptions (H1)-(H6), the solution to Problem 1 exists and is unique in $H^1(\Omega)$. Moreover, there exists a $\gamma > 0$ such that the solution to Problem 1 belongs to $C^{0,\gamma}(\Omega)$.

As a final remark, we recall (see [5], Theorem 3.3.5),

Theorem 2. If g , q_I , q_{II} , q_{III} , q_{IV} and T_f belong to $L^s(\Gamma_{in})$, $L^s(\Gamma_I)$, $L^s(\Gamma_{II})$, $L^s(\Gamma_{III})$, $L^s(\Gamma_{IV})$ and $L^s(\Gamma_{sf})$ respectively, with $s > 2$, then the solution T to Problem 1 belongs to $C(\bar{\Omega})$.

Regarding the numerical solution of Problem 1, we use the finite volume method for its discretization. Given a discretization of \mathcal{T} of the domain, Ω , we write the discrete unknown $(T_C)_{C \in \mathcal{T}}$ into the real vector \mathbf{T} , belonging to \mathbb{R}^{N_h} with $N_h = \text{size}(\mathcal{T})$. Then, we write the discretized problem as the linear system.

$$A\mathbf{T} = \mathbf{b}, \quad (5.8)$$

where A is the stiffness matrix and \mathbf{b} the source term. The value of each element of A and \mathbf{b} depends on the particular finite volume scheme for the discretization and the mesh used. Since our problem is a classic diffusion problem, we refer for further details regarding the finite volume discretization to the Eymard's monograph [6].

5.2.3. Inverse Problem

We state now the inverse problem corresponding to Problem 1. In particular, we want to estimate the heat flux g capable of reproducing the measured temperatures at the thermocouples points. This can be stated as an optimal control problem with pointwise observations.

We introduce the following notation. Let $\Psi := \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_M\}$ be a collection of points in Ω . We define the application $\mathbf{x}_i \in \Psi \rightarrow \hat{T}(\mathbf{x}_i) \in \mathbb{R}^+$, $\hat{T}(\mathbf{x}_i)$ being the experimentally measured temperature at $\mathbf{x}_i \in \Psi$. Moreover, let G_{ad} be a bounded set in $L^2(\Gamma_{in})$.

Using a least square, deterministic approach, we state the inverse problem as



Problem 2. (Inverse) Given $\{\hat{T}(\mathbf{x}_i)\}_{i=1}^M$ a collection of values in \mathbb{R}^+ , find the heat flux $g \in G_{ad}$ that minimizes the functional $J_1 : L^2(\Gamma_{in}) \rightarrow \mathbb{R}^+$,

$$J_1[g] := \frac{1}{2} \sum_{i=1}^M [T[g](\mathbf{x}_i) - \hat{T}(\mathbf{x}_i)]^2, \quad (5.9)$$

where $T[g](\mathbf{x}_i)$ is the solution of Problem 1 at points \mathbf{x}_i , for all $i = 1, 2, \dots, M$.

We notice that, thanks to Theorem 1 the state variable T is continuous in Ω , then its value at pointwise observations are well-defined.

The adjoint equation for Problem 2 is (see [5], Section 3.5)

Problem 3. (Adjoint) Find $\lambda[g]$ such that

$$\frac{1}{k} \Delta \lambda[g] = \sum_{i=1}^M (T[g](\mathbf{x}_i) - \hat{T}(\mathbf{x}_i)) \delta(\mathbf{x} - \mathbf{x}_i), \quad \text{in } \Omega, \quad (5.10)$$

with BCs

$$\begin{cases} \nabla \lambda[g] \cdot \mathbf{n} = 0 & \text{on } \Gamma_{in} \cup \Gamma_I \cup \Gamma_{II} \cup \Gamma_{III} \cup \Gamma_{IV}, \\ \nabla \lambda[g] \cdot \mathbf{n} + \frac{h}{k} \lambda[g] = 0 & \text{on } \Gamma_{sf}, \end{cases} \quad (5.11)$$

$$(5.12)$$

$\delta(\mathbf{x} - \mathbf{x}_i)$ being the Dirac's delta function centered at \mathbf{x}_i .

We recall that (see [5], Theorem 3.5.1),

Theorem 3. Under the assumptions (H1)-(H5), Problem 3 admits a unique solution in $L^q(\Omega)$ for all $q < 3$, and this solution belongs to $W^{1,\tau}(\Omega)$ for all $\tau < \frac{3}{2}$.

Let $\delta T[g]$ be the solution to

Problem 4. (Sensitivity) Find δT such that

$$-k \Delta \delta T[\delta g] = 0, \quad \text{in } \Omega, \quad (5.13)$$

with BCs

$$\begin{cases} -k \nabla \delta T[\delta g] \cdot \mathbf{n} = \delta g & \text{on } \Gamma_{in}, \\ -k \nabla \delta T[\delta g] \cdot \mathbf{n} = 0 & \text{on } \Gamma_I \cup \Gamma_{II} \cup \Gamma_{III} \cup \Gamma_{IV}, \\ -k \nabla \delta T[\delta g] \cdot \mathbf{n} = h(\delta T) & \text{on } \Gamma_{sf}. \end{cases} \quad (5.14)$$

$$(5.15)$$

$$(5.16)$$

Denoting by dJ_1 and J'_1 the Frechet and Gâteaux derivatives of J_1 , respectively, we have,

$$dJ_1(g_{an})g = \sum_{i=1}^M (T[g_{an}](\mathbf{x}_i) - \hat{T}(\mathbf{x}_i)) \delta T[g](\mathbf{x}_i) = \frac{1}{k^2} \int_{\Gamma_{in}} \lambda g. \quad (5.17)$$

Then, from (5.17) and the definition of the Gâteaux derivative of a functional, we have

$$J'_1[g] = \frac{\lambda[g]}{k^2} \text{ in } L^2(\Gamma_{in}). \quad (5.18)$$



Different methods can be used for the solution of this minimization problem. Here, we discuss its solution by Alifanov's regularization method (see [7]).

5.2.3.1. Alifanov's Regularization

The Alifanov's regularization method is a conjugate gradient method applied on the adjoint equation (see [8]).

We consider the following iterative procedure for the estimation of the function g that minimizes the functional (5.9). Given an initial estimation $g^0 \in L^2(\Gamma_{in})$, for $n > 0$ a new iterant is computed as:

$$g^{n+1} = g^n - \beta^n P^n, \quad n = 0, 1, 2, \dots \quad (5.19)$$

where n is the iteration counter, β is the stepsize in the search direction P^n given by

$$P^0 = J'_1[g^0], \quad P^{n+1} = J'_1[g^{n+1}] + \gamma^{n+1} P^n \text{ for } n \geq 1, \quad (5.20)$$

γ^n being the conjugate coefficient.

The stepsize β^n in (5.19) is obtained by minimizing the functional $J_1[g^n - \beta P^n]$ with respect to β . Therefore, β^n is the solution of the critical point equation of the functional J , restricted to a line passing through g^n in the direction defined by P^n , i.e. β^n is the critical point of $J_1[g^n - \beta P^n]$ which then satisfies

$$J_1[g^n - \beta^n P^n] = \min_{\beta} \left\{ \frac{1}{2} \sum_{i=1}^M \{T[g^n - \beta P^n](\mathbf{x}_i) - \hat{T}(\mathbf{x}_i)\}^2 \right\}. \quad (5.21)$$

Recalling Problem 4,

$$\begin{aligned} J_1[g^n - \beta P^n] &= \frac{1}{2} \sum_{i=1}^M [T[g^n - \beta P^n](\mathbf{x}_i) - \hat{T}(\mathbf{x}_i)]^2 \\ &= \frac{1}{2} \sum_{i=1}^M [(T[g^n] - \beta \delta T[P^n])(\mathbf{x}_i) - \hat{T}(\mathbf{x}_i)]^2. \end{aligned} \quad (5.22)$$

Differentiating with respect to β , we obtain the critical point equation

$$\frac{dJ_1[g^n - \beta P^n]}{d\beta} = \sum_{i=1}^M [(T[g^n] - \beta \delta T[P^n])(\mathbf{x}_i) - \hat{T}(\mathbf{x}_i)](-\delta T[P^n](\mathbf{x}_i)) = 0. \quad (5.23)$$

Finally, we have

$$\beta^n = \frac{\sum_{i=1}^M \{T[g^n](\mathbf{x}_i) - \hat{T}(\mathbf{x}_i)\} \delta T[P^n](\mathbf{x}_i)}{\sum_{i=1}^M (\delta T[P^n](\mathbf{x}_i))^2}. \quad (5.24)$$

With respect to the conjugate coefficient, γ , its value is zero for the first iteration and for other iterations it can be calculated using Fletcher-Reeves expression as follows [9]

$$\gamma^n = \frac{\|J'_1[g^n]\|_{L^2(\Gamma_{in})}^2}{\|J'_1[g^{n-1}]\|_{L^2(\Gamma_{in})}^2}. \quad (5.25)$$

Notice that, to use this iterative procedure, we have to compute at each iteration the Gâteaux derivative $J'_{1_g}(\mathbf{x})$ which is given by (5.18). Thus, we must solve the adjoint problem to compute it.

Alifanov's regularization algorithm is summarized in Algorithm 2.



```

Input:  $g^0$ 
Set  $n = 0$ ;
while  $n < n_{max}$  do
    Compute  $T[g^n]$  by solving Problem 1;
    Compute  $J_1[g^n]$  by (5.9);
    if  $J_1[g^n] < J_{1_{tol}}$  then
        | Stop;
    end
    Compute  $\lambda[g^n]$  by solving Problem 3;
    Compute  $J'_1[g^n]$  by (5.18);
    if  $n \geq 1$  then
        | Compute the conjugate coefficient,  $\gamma^n$ , by (5.25);
        | Compute the search direction,  $P^n(\mathbf{x})$ , by (5.20);
    else
        |  $P^0 = J'_1[g^0]$ ;
    end
    Compute  $\delta T[P^n]$  by solving Problem 4 with  $\delta g = P^n$ ;
    Compute the stepsize in the search direction,  $\beta^n$ , by (5.24);
    Update heat flux  $g^n$  by (5.19);
     $n = n + 1$ ;
end

```

Algorithm 2: Alifanov's regularization.

5.3. Implementation

The implementation of the inverse problem solution methodology can be divided into four stages: geometry and mesh generation, solving direct heat transfer problems, solving inverse problem and visualizing numerical solutions. In this section, we discuss the implementation of the four stages separately.

For the numerical solution of Problem 2, we use ITHACA-FV [10] which is freely available under the GPL 3 License. ITHACA-FV is an open-source C++ library based on the finite volume solver OpenFOAM [11]. Then, for the understanding of the following implementation, some basic knowledge of the OpenFOAM and C++ environment may be required (e.g. setting up an OpenFOAM case, object oriented programming etc.). We refer to the OpenFOAM and ITHACA-FV documentation for all the details. All the updated information regarding installation and usage are available at <https://mathlab.sissa.it/ithaca-fv> together with usefull examples.

5.3.1. Geometry and Meshing

The digital representation of the setup is first done by drawing the CAD of the geometry and then discretizing it to form a mesh. While several tools and techniques are available for this, we propose the use of two open-source tools:

- (i) The open-source software SALOME. It provides capabilities for interfacing with various numerical simulation tools and has a flexible cross-platform architecture made of reusable components, allowing for customized integration and handling of complex geometrical objects. Moreover, it allows for creation of geometry and meshes using a graphical and a text user interface.
- (ii) The `blockMesh` utility supplied with OpenFOAM [11]. The `blockMesh` utility creates parametric, structured, hexahedral meshes with grading and curved edges. For the generation of the geometry and mesh, the user must use a dictionary text file.

The decision on when to use which is fairly simple since `blockMesh` only allows the meshing of simple



geometries, it could be preferred in these cases. In fact, it may be not very intuitive to use but provides the user a mesh ready to be used in OpenFOAM in few seconds. For any complex geometry, the choice must be to use SALOME.

In this work, we present a simple academic benchmark with a trivial geometry. Then, we use `blockMesh`. The principle behind `blockMesh` is to decompose the domain geometry into a set of 1 or more three dimensional, hexahedral blocks. Each block of the geometry is defined by 8 vertices, one at each corner of a hexahedron. Edges of the blocks can be straight lines, arcs or splines. The mesh is ostensibly specified as a number of cells in each direction of the block, sufficient information for `blockMesh` to generate the mesh data.

5.3.2. Direct Problem Solver

For the solution of the direct heat transfer problems, we use OpenFOAM, a popular open-source finite volume library for solving Partial Differential Equations (PDEs). Different versions of OpenFOAM are available. We use OpenFOAM v1812.

This C++ library, offer a reach interface for solving PDEs in with the finite volume method. One of its key features is the easy implementation of the PDEs to be solved. In fact, it has `fvScalarMatrix` and `fvVectorMatrix` classes in which we can directly write the equations to be solved for a scalar and a vector state variable, respectively. Then, the software takes care of the finite volume discretization. The scheme used for the discretization of each term in the equation are specified by the user in the `fvScheme` dictionary.

As an example of using the `fvScalarMatrix` class, Problem 1 is implemented and solved in OpenFOAM as

```
while (simple.loop())
{
    while (simple.correctNonOrthogonal())
    {
        fvScalarMatrix TEqn
        (
            fvm::laplacian(DT, T)
        );
        TEqn.solve();
    }
}
```

where the BCs are assigned before this piece of code using the ITHACA-FV command `ITHACAUtilities::assignBC`. A detailed documentation along with numerous examples are offered by Jasak [11]. Moreover, we refer the the OpenFOAM official guide for any detail [12].

5.3.3. Inverse Problem Solver

The implementation of the inverse problem solver (i.e. the Alifanov's regularization) was entirely made in the ITHACA-FV framework. As previously mentioned, it is a C++ library based on the finite volume solver OpenFOAM. It consists of the implementation of several reduced order modeling techniques for parameterized problems. One of the advantages of this library is that linear and non-linear algebra operations which are not already implemented in OpenFOAM are performed with the external library Eigen, the source code of Eigen 3.3.7 [13] being provided together with ITHACA-FV.

All the input parameter for the inverse problem solver are provided by the user through the dictionary `ITHACAdict`. An example of `ITHACAdict` is

```
FoamFile
{
    version      2.0;
    format       ascii;
    class        dictionary;
```



```

        location    "system";
        object      ITHACAdict;
    }

    cgIterMax      200;
    Jtolerance     1e-3;

    thermocouplesNumber 16;
    thermalConductivity 1.0;
    heatTransferCoeff 1.0;

```

where `cgIterMax` is the maximum number of iterations for the Alifanov's regularization, `Jtolerance` the J_{1tol} of Algorithm 2, `thermocouplesNumber` the number of thermocouples, `thermalConductivity` the mold thermal conductivity k in $\frac{W}{mK}$ and `heatTransferCoeff` the heat transfer coefficient h in $\frac{W}{m^2K}$.

5.3.4. Post-processing and Visualization

Visualization and post-processing of the obtained data is critical both in the development phase and to present the results. Implementing and representing these data in a simple and effective manner is extremely useful for deriving information, presenting results, and also in testing and debugging. The post-processing operations are very problem specific and depend also on the user taste. In our benchmark case, the analysis of the solutions include visualization of thermal fields and boundary heat fluxes, computation of error fields, computation of norms and many others.

In general, the post-processing can be included inside the solver or can be made during the visualization phase. Within the solver phase, these could just be operations on the solution data done using ITHACA-FV and plotted using some common user preferred graphing libraries like Matplotlib [14].

For the visualization of the solution fields, we use the dedicated visualization tool ParaView [15]. It is a widely used open-source visualization tool for plotting and viewing solutions and graphs. Moreover, it is able to directly read the OpenFOAM outputs without need of converting them to other formats.

ParaView comes with a wide range of filters for analysis and visualization including plotting graphs, contours, surface plots, vector field plots etc. In addition, it is also possible to define user-defined filters to perform customized operations. For further information, we refer to the ParaView user guide [15].

5.4. Computer Requirements

The software tools used in the present study require a minimal UNIX system with at least 1GB of memory and about 500MB of disk space (swap) for execution. For the installation of OpenFOAM and ParaView the following minimal versions are required:

- gcc: 4.8.5;
- cmake: 3.3 (required for ParaView and CGAL build);
- boost: 1.48 (required for CGAL build);
- fftw: 3.3.7 (optional - required for FFT-related functionality);
- Qt: 4.8 (optional - required for ParaView build).

To handle larger problems it is recommended to have some higher configuration.

5.5. Benchmark Case

The benchmark presented in this section is an academic case. It is a steady-state heat conduction problem in an homogeneous isotropic solid in a rectangular parallelepiped domain. By carefully selecting the BC on the faces of the parallelepiped, we are able to compute the analytical solution of the heat conduction problem. Then, we



use this academic test to validate the numerical solution of the direct problem. Moreover, by arbitrarily selecting some temperature measurements points, we test the inverse problem solver discussed in Section 5.2.3.1.

Let the domain be $\Omega = (0, L) \times (0, W) \times (0, H)$ as in Figure 5.3 with positive real constants L, W and H . Then, the boundaries of the domain are

$$\begin{aligned}
 \Gamma_{sf} &:= \{\mathbf{x} \in \Gamma \mid \mathbf{x} = (x, W, z)\}, \\
 \Gamma_{in} &:= \{\mathbf{x} \in \Gamma \mid \mathbf{x} = (x, 0, z)\}, \\
 \Gamma_I &:= \{\mathbf{x} \in \Gamma \mid \mathbf{x} = (x, y, H)\}, \\
 \Gamma_{II} &:= \{\mathbf{x} \in \Gamma \mid \mathbf{x} = (L, y, z)\}, \\
 \Gamma_{III} &:= \{\mathbf{x} \in \Gamma \mid \mathbf{x} = (x, y, 0)\}, \\
 \Gamma_{IV} &:= \{\mathbf{x} \in \Gamma \mid \mathbf{x} = (0, y, z)\}.
 \end{aligned} \tag{5.26}$$

Let a, b, c be real constants. To have an analytical solution T in Ω , we consider the following BCs for Problem 1,

$$\begin{aligned}
 g &= kbx + c, \\
 q_I &= 2kaH, \\
 q_{II} &= -k(2aL + by), \\
 q_{III} &= 0, \\
 q_{IV} &= kby, \\
 T_f &= \frac{kbx + c}{h} + ax^2 + cy - az^2 + bxW + c.
 \end{aligned} \tag{5.27}$$

Then,

$$T_{an} = ax^2 + bxy + cy - az^2 + c, \tag{5.28}$$

is the solution to Problem 1. In the following, we use the input parameters of Table 5.1.

Table 5.1: Parameters used in the numerical example.

Parameter	Value
Thermal conductivity, k	$3 \text{ W}/(mK)$
Heat transfer coefficient, h	$5 \text{ W}/(m^2K)$
L	1 m
W	1 m
H	1 m
a	5
b	10
c	15

5.5.1. Direct Problem

Due to its simplicity, the domain Ω is discretized by uniform, structured, orthogonal, hexahedral meshes. To study the convergence of the numerical solution to the analytical one, we use four different grids. They have 20, 30, 40 and 50 grid elements per axis. Since we use the same number of elements for the three axes, the grids have $8e3$, $27e3$, $64e3$ and $125e3$ elements respectively.

Using `blockMesh`, the $125e3$ elements mesh is generated by writing the `blockMeshDict` as follow

```
scale 1;
vertices
```



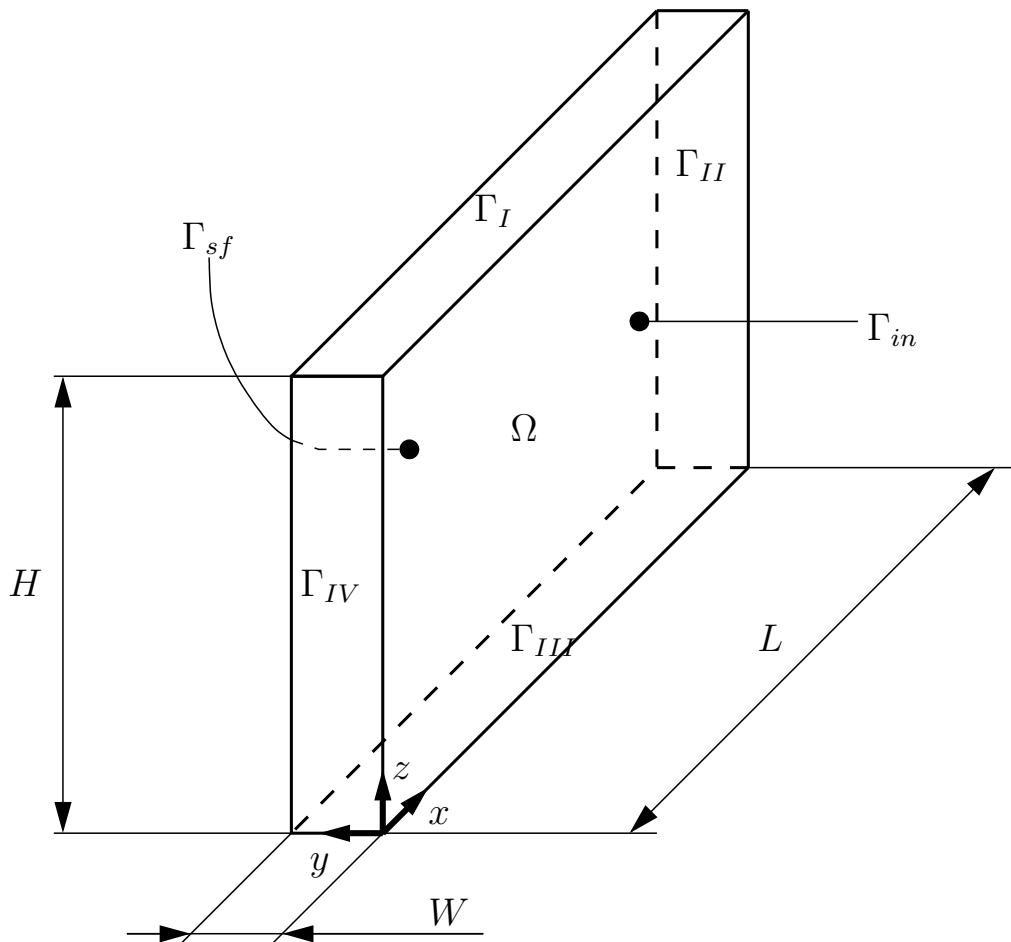


Figure 5.4: Schematic of the solid rectangular parallelepiped domain.

```
(
  (0 0 0)
  (1 0 0)
  (1 1 0)
  (0 1 0)

  (0 0 1)
  (1 0 1)
  (1 1 1)
  (0 1 1)
);
blocks
(
  hex (0 1 2 3 4 5 6 7) (50 50 50) simpleGrading (1 1 1)
);
edges
(
);
boundary
(
  hotSide
  {
```



```

    type patch;
    faces
    (
        (0 1 5 4)
    );
}
coldSide
{
    type patch;
    faces
    (
        (3 2 6 7)
    );
}
gammaEx1
{
    type patch;
    faces
    (
        (4 5 6 7)
    );
}
gammaEx2
{
    type patch;
    faces
    (
        (1 5 6 2)
    );
}
gammaEx3
{
    type patch;
    faces
    (
        (0 1 2 3)
    );
}
gammaEx4
{
    type patch;
    faces
    (
        (0 4 7 3)
    );
}
};

```

Remember that the default length unit in OpenFOAM is meter.

As previously discussed, we use the FVM for the discretization of Problem 1. For the discretization of the laplacian, we use the Gaussian integration. Since we have a structured orthogonal grid, no correction is needed when computing the gradient normal to the cells faces. Moreover, we use linear interpolation to interpolate the values from cell centers to face centers. The resulting scheme is second order accurate.

From the discretization of Problem 1, we obtain a linear system. We solve it by using the Preconditioned Conjugate Gradient solver (PCG) with Diagonal Incomplete Cholesky (DIC) preconditioning. The tolerance used for the linear system solver is 10^{-6} . All the computations are performed in OpenFOAM [12].



To evaluate the accuracy of the numerical solutions, we compute the error ϵ as

$$\epsilon_K := T_K - T_{an}(\mathbf{x}_K) \quad \forall K \in \mathcal{T}. \quad (5.29)$$

Figure 5.5 shows the decay of the L^2 - and L^∞ -norm of ϵ with the grid refinement. We conclude that Problem 1 is well numerically solved.

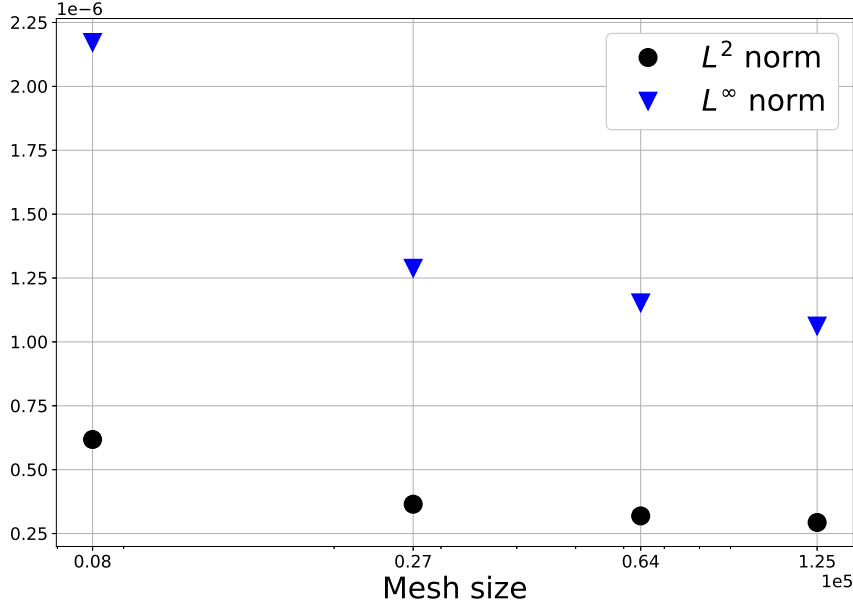


Figure 5.5: Decay of the L^2 - and L^∞ -norm of ϵ with the mesh refinement.

5.5.2. Inverse Problem

We now consider the numerical solution of Problem 2 where $T[g](\mathbf{x}_i)$ is the solution of Problem 1 assuming:

(HB1): Known temperatures at thermocouples points, $\{\hat{T}(\mathbf{x}_i)\}_{i=1}^M$;

(HB2): The remaining data (except the unknown g) corresponding to the academic direct problem of Section 5.5.1.

So the analytical solution of Problem 2 is:

$$g_{an} = kbx + c. \quad (5.30)$$

To numerically analyze the performances of the inverse solvers, we design a numerical test. The idea is to select arbitrarily the measurement points $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_M$. Then, we use the analytical solution of Problem 1, (5.28), at the virtual thermocouples locations as temperature measurements, i.e.

$$\hat{T}(\mathbf{x}_i) = T_{an}(\mathbf{x}_i), \quad i = 1, 2, \dots, M. \quad (5.31)$$

Finally with the measurements $\hat{T}(\mathbf{x}_i)$, we solve Problem 2.

The thermocouples are assumed to be located in the plane $y = 0.2$ m. The (x, z) coordinates of the thermocouples are shown in Figure 5.6. The parameters used for the computations are summarized in Table 5.2. The following tests are all performed on the $123e3$ elements grid.

Figure 5.7(a) and (b) illustrate the convergence of the functional J and of the L^2 - and L^∞ -norm of the relative



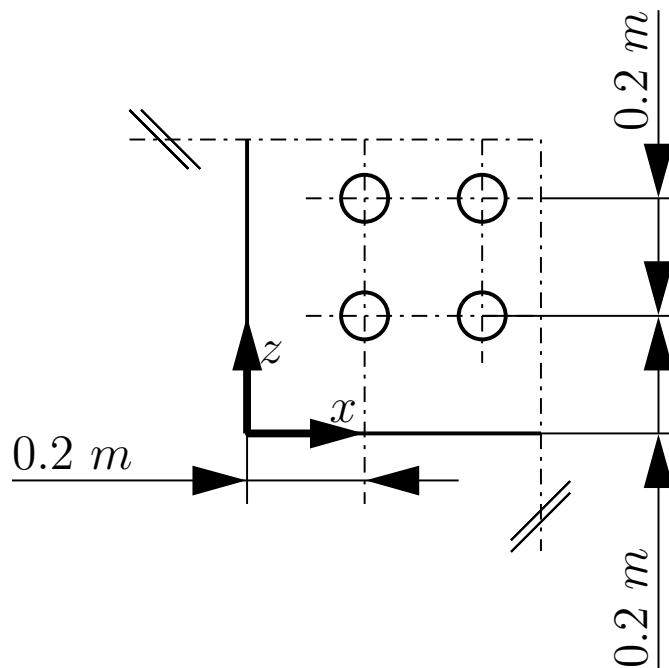


Figure 5.6: Positions of the thermocouples.

Table 5.2: Parameters used for the test of inverse problem solver.

Parameter	Value
N. of thermocouples	16
Thermocouples plane	$y = 0.2 \text{ m}$
CG stop criterion	$J^n < 10^{-6} K^2$
g^0	0 W/m^2

error, respectively. The relative error norms are computed as

$$\|\epsilon_{rel}\|_{L^2(\Gamma_{in})} = \left\| \frac{g - g_{an}}{g_{an}} \right\|_{L^2(\Gamma_{in})}, \quad \|\epsilon_{rel}\|_{L^\infty(\Gamma_{in})} = \left\| \frac{g - g_{an}}{g_{an}} \right\|_{L^\infty(\Gamma_{in})}. \quad (5.32)$$

The relative error at the last iteration is 0.017 and 0.055 in the L^2 - and L^∞ -norm respectively. By looking at Figure 5.8, we see that also qualitatively the heat flux is well estimated.

Bibliography

- [1] L. Klimeš and J. Štětina, “A rapid gpu-based heat transfer and solidification model for dynamic computer simulations of continuous steel casting,” *Journal of Materials Processing Technology*, vol. 226, pp. 1–14, 2015.
- [2] F. M. White, *Fluid Mechanics*. McGraw Hill, 2011.
- [3] I. Samarasekera and J. Brimacombe, “The influence of mold behavior on the production of continuously cast steel billets,” *Metallurgical Transactions B*, vol. 13, no. 1, pp. 105–116, 1982.
- [4] R. Nittka, “Regularity of solutions of linear second order elliptic and parabolic boundary value problems on lipschitz domains,” *Journal of Differential Equations*, vol. 251, no. 4, pp. 860 – 880, 2011.
- [5] J.-P. Raymond, “Optimal control of partial differential equations,” *Université Paul Sabatier, Internet*,



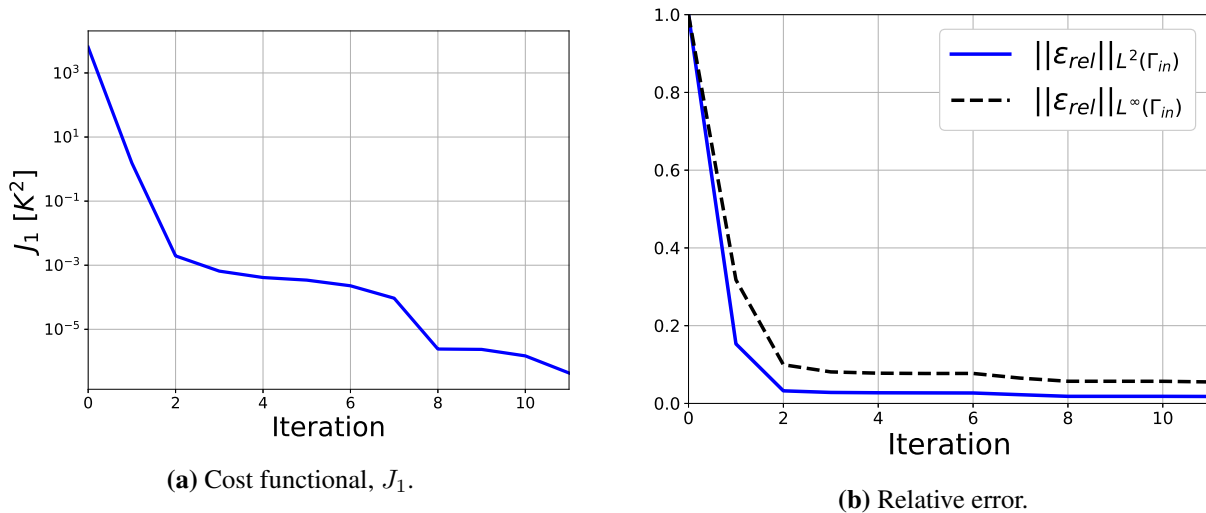


Figure 5.7: Convergence of the Alifanov's regularization with thermocouples at $y = 0.2 m$.

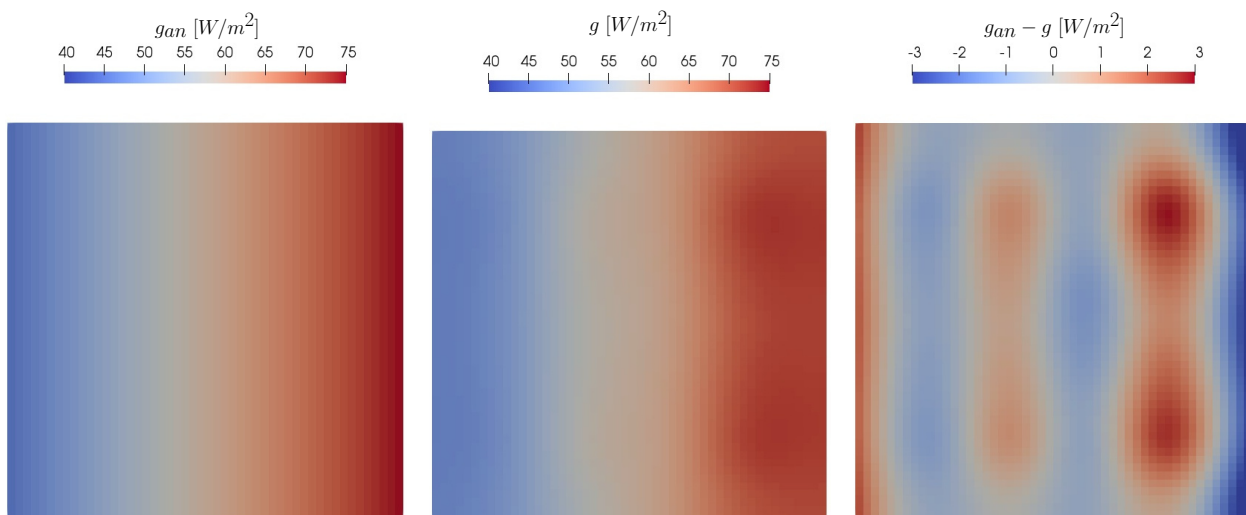


Figure 5.8: Comparison of the estimated and real boundary heat flux by the Alifanov's regularization with thermocouples at $y = 0.2 m$.

2013.

- [6] R. Eymard, T. Gallouet, and R. Herbin, "Finite volume methods," in *Solution of Equation in R^n (Part 3), Techniques of Scientific Computing (Part 3)*, ser. Handbook of Numerical Analysis. Elsevier, 2000, vol. 7, pp. 713 – 1018.
- [7] O. Alifanov, *Inverse Heat Transfer Problems*, 1st ed. Moscow Izdatel Mashinostroenie, 1988.
- [8] F. D. Moura Neto and A. J. da Silva Neto, *An Introduction to Inverse Problems with Applications*. Springer Publishing Company, Incorporated, 2012.
- [9] R. Fletcher and C. M. Reeves, "Function minimization by conjugate gradients," *The Computer Journal*, vol. 7, no. 2, pp. 149–154, 01 1964.
- [10] G. Stabile, S. Hijazi, A. Mola, S. Lorenzi, and G. Rozza, "POD-Galerkin reduced order methods for cfd using finite volume discretisation: vortex shedding around a circular cylinder," *Communications in Applied and Industrial Mathematics*, vol. 8, no. 1, pp. 210 – 236, 2017.
- [11] H. Jasak, "Openfoam: Open source cfd in research and industry," *International Journal of Naval Archi-*



tecture and Ocean Engineering, vol. 1, no. 2, pp. 89 – 94, 2009.

- [12] OpenCFD, *OpenFOAM - The Open Source CFD Toolbox - User's Guide*, 1st ed., OpenCFD Ltd., United Kingdom, 11 2007. [Online]. Available: <https://cfd.direct/openfoam/user-guide/>
- [13] G. Guennebaud, B. Jacob *et al.*, “Eigen v3,” <http://eigen.tuxfamily.org>, 2010.
- [14] J. D. Hunter, “Matplotlib: A 2d graphics environment,” *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.
- [15] U. Ayachit, *The ParaView Guide: A Parallel Visualization Application*. Clifton Park, NY, USA: Kitware, Inc., 2015.



6. Coupled parameterized reduced order modelling of thermo-mechanical phenomena arising in blast furnaces

Nirav Shah¹, Patricia Barral², Michele Girfoglio¹, Alejandro Lengomin³, Peregrina Quintela², Gianluigi Rozza¹

¹*Scuola Internazionale Superiore di Studi Avanzati*

²*Instituto Tecnológico de Matemática Industrial, Universidade de Santiago de Compostela*

³*Global Research and Development Center, AMIII*

Abstract. Blast furnace operations are subjected to temperatures up to 1500°C, producing very high thermal stresses inside the blast furnace hearth walls. An optimum design is necessary to enlarge the hearth lifetime and, consequently, each blast furnace campaign. The thermal stresses are evaluated by finite element formulation of a one-way coupled axisymmetric thermo-mechanical model. The numerical implementation of the discretized model is validated performing proper benchmark tests. The parameters directly affecting the thermal stresses are physical properties of the material, process conditions and geometric dimensions of the hearth. In this framework, we apply a parametric reduced order model procedure.

Keywords: Blast furnace hearth, thermo-mechanical axisymmetric model, finite element method, benchmark verification, model order reduction, proper orthogonal decomposition.

6.1. Introduction

6.1.1. Conceptual model

Steel making is a very old process and has contributed to the development of technological societies since ancient times. The previous stage to the steelmaking is the ironmaking process, which is performed inside a blast furnace. Its general layout is shown in Figure 6.1. It is a metallurgical reactor used to produce hot metal (95 % Fe, 4.5% C) from iron ore. The burden contains iron ore, fluxes and coke. The process involves exothermic and endothermic reactions.

High temperature inside the blast furnace is required to increase the gasification of carbon. The temperature in the hearth can be as high as 1500° C. The thermal stresses induced by high temperature inside the blast furnace hearth limits the overall blast furnace campaign period. The numerical computation of thermal stresses requires coupling of thermal and mechanical models and solve the associated coupled system. The hearth is made up of several refractory zones, with each zone having specific functions and, accordingly, design requirements.

The proper materials selection ensures less thermal stress, which causes mechanical design challenge. The heat transfer through the furnace wall occurs by conduction. Additionally, convection occurs at the surface. Wear of carbon refractory limits the hearth lifetime, which in turn restricts lifetime of blast furnace. An optimum design enlarges the hearth lifetime and consequently, the blast furnace campaign.

In this document, we focus on some parameters affecting the hearth design such as material properties and geometric dimensions. In particular, concerning the parameters related to material properties, the effects of thermal conductivity, thermal expansion coefficient and Lamé parameters (or Young's modulus and Poisson's ratio) are of great relevance. The geometric parameters characterizing the hearth walls are the thickness of hearth blocks, number of blocks, inner diameter of the hearth at each transversal section, outer diameter of the hearth, and volume of the hot liquids that can be contained.

The blast furnace operates under different conditions, each of which is governed by different mathematical models. Considering the objectives of the present work, below simplifications are considered.

- Taphole operation is not part of this study. The perforation action of the taphole and the important



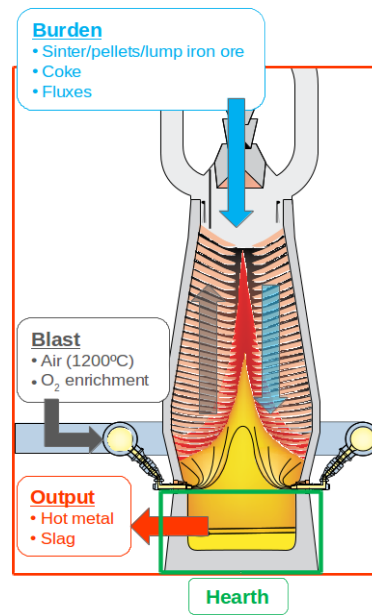


Figure 6.1: Blast furnace [Courtesy:ArcelorMittal]

pressures in the draining of the hot metal and slag produce important mechanical stresses located in the area that require a deeper analysis and that is not part of this work. So, in this document, the detail of the tap on the furnace walls is omitted.

- Since the objective is to be able to calculate in real time the effects of wall design on blast furnace operation, we focus on the steady state operations.
- We assume that the hearth is made up of a single homogeneous, elastic and isotropic material with temperature-independent material properties.
- Heat transfer only by conduction within hearth walls will be considered. The temperature of the molten metal inside the hearth is assumed constant and known. Therefore, the fluid region will not be part of the problem.

6.1.2. Mathematical problem and Benchmark cases

As explained in the subsection 6.1.1, the problem consists of a thermal and mechanical system. The governing equations are the momentum conservation for small displacements and the energy conservation (see [1], [2]). The energy conservation equation is solved first and the computed temperature field is used in the momentum conservation equation as a data.

We consider the hearth domain Ω (Figure 6.2). Assuming known the heat source term, Q , and the body force term, \vec{f}_0 , the conservation equations to solve are:

$$- \text{Div}(k\nabla T) = Q, \text{ in } \Omega, \quad (6.1)$$

$$- \text{Div}(\boldsymbol{\sigma}) = \vec{f}_0, \text{ in } \Omega, \quad (6.2)$$

where k is the thermal conductivity, and T is the temperature scalar field. Let us denote the displacement vector field as \vec{u} . The thermo-mechanical stress tensor $\boldsymbol{\sigma}$ is related to the strain tensor through the Hooke's law:

$$\boldsymbol{\sigma}(\vec{u})[T] = \lambda \text{Tr}(\boldsymbol{\varepsilon}(\vec{u}))\mathbf{I} + 2\mu\boldsymbol{\varepsilon}(\vec{u}) - (2\mu + 3\lambda)\alpha(T - T_0)\mathbf{I}, \quad (6.3)$$



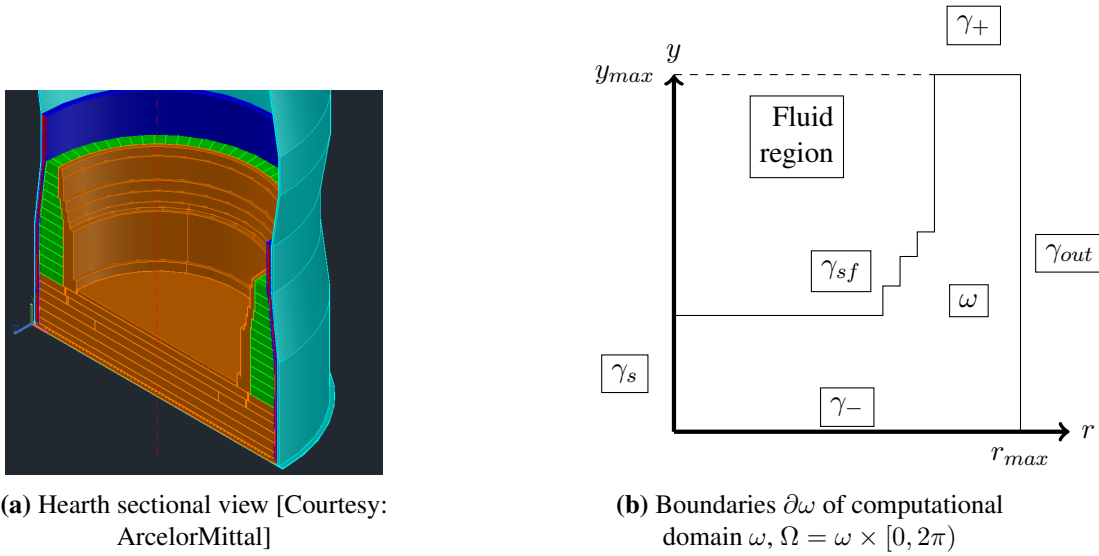


Figure 6.2: Hearth domain and its boundaries

where \mathbf{I} refers to the identity tensor, $\varepsilon(\vec{u})$ is the strain tensor defined as,

$$\varepsilon(\vec{u}) = \frac{1}{2}(\nabla \vec{u} + \nabla \vec{u}^T). \quad (6.4)$$

In addition, T_0 is the reference temperature, α is the thermal expansion coefficient, and λ and μ are the Lamé parameters of the material. These latter can be expressed in terms of Young modulus, E , and the Poisson ratio, ν , as:

$$\mu = \frac{E}{2(1 + \nu)}, \quad \lambda = \frac{E\nu}{(1 - 2\nu)(1 + \nu)}. \quad (6.5)$$

In further analysis we use the terms normal force, σ_n , and tangential force, $\vec{\sigma}_t$, defined by (\vec{n} is the unit outer normal vector):

$$\begin{aligned} \sigma_n &= (\boldsymbol{\sigma} \vec{n}) \cdot \vec{n}, \\ \vec{\sigma}_t &= \boldsymbol{\sigma} \vec{n} - \sigma_n \vec{n}. \end{aligned}$$

We consider the following boundary conditions.

- On the upper boundary, γ_+ , the applied force, \vec{g}_+ , and the density of heat flux, q_+ , are known.
- On the bottom boundary, γ_- , the convection heat transfer with heat exchanger at temperature T_- and heat transfer coefficient $h_{c,-}$ occurs. The normal displacement is null and shear forces are assumed to be \vec{g}_- .
- On the inner boundary, γ_{sf} , convection heat transfer with the fluid phase occurs and hydrostatic pressure is acting. The fluid temperature T_f is assumed to be known and constant at the steady state, $h_{c,f}$ is the convective heat transfer coefficient on γ_{sf} , and \vec{g}_{sf} is the applied force. In the blast furnace, \vec{g}_{sf} is related to the hydrostatic pressure by the expression: $\vec{g}_{sf} = -p_h \vec{n}$, p_h being the hydrostatic pressure.
- On the outer boundary, γ_{out} , a convective heat flux and known applied force \vec{g}_{out} are assumed. $h_{c,out}$ is the convective heat transfer coefficient on γ_{out} , and T_{out} the ambient temperature.
- On the symmetry boundary, γ_s , symmetry conditions are assumed as explained in the next section.

Considering that the different physical phenomena are under interaction, the coupled system is split into 3



subsystems : Thermal model, Mechanical model and Coupling. Separate benchmark cases for each individual subsystem are proposed. This helps for step by step error resolution.

6.1.2.1. Benchmark for the thermal model

A known analytical temperature function, corresponding source term and boundary conditions functions will be applied. The computed temperature field will be compared with applied analytical temperature field.

6.1.2.2. Benchmark for the mechanical model

Similar procedure to benchmark test of energy equation is followed. A known analytical displacement field, corresponding source term and boundary conditions will be applied. The computed displacement field will be compared with the analytical displacement field.

6.1.2.3. Benchmark for the coupling

We apply known displacement and temperature fields in the domain and compute the corresponding displacements and thermomechanical stresses. The computed fields are compared with the analytical ones. To complete the test, a separate analysis of the effective thermal stresses will be made analysing the hydrostatic part of thermomechanical stresses.

6.2. Mathematical formulation

In addition to the simplifications indicated at the end of Section 6.1.1, in the context of blast furnace application, the axisymmetric hypothesis is applicable. In general, a hierarchy of the main models helps to understand simplifications and assumptions used to arrive at a specific model (see appendices 6.3 and 6.4).

6.2.1. Axisymmetric model

We express now the governing equations (6.2), (6.1) and boundary conditions in cylindrical coordinate system (r, y, θ) having corresponding unit vectors $(\vec{e}_r, \vec{e}_y, \vec{e}_\theta)$. The body force density term \vec{f}_0 can be expressed as,

$$\vec{f}_0 = f_{0,r}(r, y)\vec{e}_r + f_{0,y}(r, y)\vec{e}_y, \quad (6.6)$$

and it depends only on (r, y) coordinates. Similarly, applied surface forces have zero component in \vec{e}_θ direction and they do not depend on θ . In the axisymmetric system, we represent the displacement \vec{u} and temperature T , both independent of θ , as

$$\begin{aligned} \vec{u} &= u_r(r, y)\vec{e}_r + u_y(r, y)\vec{e}_y, \\ T &= T(r, y). \end{aligned} \quad (6.7)$$

Under the assumption of axisymmetry, the strain and stress tensors in cylindrical coordinate system (r, y, θ) respectively, can be reduced to,

$$\varepsilon(\vec{u}) = \begin{bmatrix} \frac{\partial u_r}{\partial r} & \frac{1}{2} \left(\frac{\partial u_r}{\partial y} + \frac{\partial u_y}{\partial r} \right) & 0 \\ \frac{1}{2} \left(\frac{\partial u_r}{\partial y} + \frac{\partial u_y}{\partial r} \right) & \frac{\partial u_y}{\partial y} & 0 \\ 0 & 0 & \frac{u_r}{r} \end{bmatrix}, \quad (6.8)$$



$$\boldsymbol{\sigma}(\vec{u})[T] = \begin{bmatrix} \sigma_{rr} & \sigma_{ry} & 0 \\ \sigma_{ry} & \sigma_{yy} & 0 \\ 0 & 0 & \sigma_{\theta\theta} \end{bmatrix}. \quad (6.9)$$

In vector notation, axisymmetric stress-strain relationship can be expressed as,

$$\{\boldsymbol{\sigma}(\vec{u})[T]\} = \mathbf{A}\{\boldsymbol{\varepsilon}(\vec{u})\} - (2\mu + 3\lambda)\alpha(T - T_0)\{\mathbf{I}\},$$

$$\mathbf{A} = \frac{E}{(1 - 2\nu)(1 + \nu)} \begin{bmatrix} 1 - \nu & \nu & \nu & 0 \\ \nu & 1 - \nu & \nu & 0 \\ \nu & \nu & 1 - \nu & 0 \\ 0 & 0 & 0 & \frac{1 - 2\nu}{2} \end{bmatrix}, \quad (6.10)$$

$$\{\boldsymbol{\sigma}\} = \{\sigma_{rr} \quad \sigma_{yy} \quad \sigma_{\theta\theta} \quad \sigma_{ry}\}^T,$$

$$\{\boldsymbol{\varepsilon}\} = \{\varepsilon_{rr} \quad \varepsilon_{yy} \quad \varepsilon_{\theta\theta} \quad 2\varepsilon_{ry}\}^T,$$

$$\{\mathbf{I}\} = \{1 \quad 1 \quad 1 \quad 0\}^T.$$

Taking into account the expression of the unit normal vector at different boundaries as (see Figure 6.2b),

$$\begin{aligned} \text{on } \gamma_+ : n_r &= 0, n_y = 1, \vec{n} = \vec{e}_y, \\ \text{on } \gamma_- : n_r &= 0, n_y = -1, \vec{n} = -\vec{e}_y, \\ \text{on } \gamma_{sf} : \vec{n} &= n_r \vec{e}_r + n_y \vec{e}_y, \\ \text{on } \gamma_{out} : n_r &= 1, n_y = 0, \vec{n} = \vec{e}_r, \\ \text{on } \gamma_s : n_r &= -1, n_y = 0, \vec{n} = -\vec{e}_r, \end{aligned} \quad (6.11)$$

the axisymmetric thermomechanical model considered can be summarized as :

- Thermal model (T1) :

$$-\frac{1}{r} \frac{\partial}{\partial r} \left(rk \frac{\partial T}{\partial r} \right) - \frac{\partial}{\partial y} \left(k \frac{\partial T}{\partial y} \right) = Q, \text{ in } \omega. \quad (6.12)$$

Boundary conditions:

$$\begin{aligned} \text{on } \gamma_+ : -k \frac{\partial T}{\partial y} &= q_+, \\ \text{on } \gamma_- : k \frac{\partial T}{\partial y} &= h_{c,-}(T - T_-), \\ \text{on } \gamma_{sf} : -k \frac{\partial T}{\partial r} n_r - k \frac{\partial T}{\partial y} n_y &= h_{c,f}(T - T_f), \\ \text{on } \gamma_{out} : -k \frac{\partial T}{\partial r} &= h_{c,out}(T - T_{out}), \\ \text{on } \gamma_s : \frac{\partial T}{\partial r} &= 0. \end{aligned} \quad (6.13)$$

- Mechanical model (M1) :

$$\begin{aligned} \frac{\partial \sigma_{rr}}{\partial r} + \frac{\partial \sigma_{ry}}{\partial y} + \frac{\sigma_{rr} - \sigma_{\theta\theta}}{r} + f_{0,r} &= 0, \text{ in } \omega, \\ \frac{\partial \sigma_{ry}}{\partial r} + \frac{\partial \sigma_{yy}}{\partial y} + \frac{\sigma_{ry}}{r} + f_{0,y} &= 0, \text{ in } \omega. \end{aligned} \quad (6.14)$$

Boundary conditions :



$$\begin{aligned}
\text{on } \gamma_+ & : \sigma_{ry} = g_{+,r} , \sigma_{yy} = g_{+,y} , \\
\text{on } \gamma_- & : u_y = 0 , \sigma_{ry} = -g_{-,r} , \\
\text{on } \gamma_{sf} & : \sigma_{rr}n_r + \sigma_{ry}n_y = g_{sf,r} , \\
& \sigma_{ry}n_r + \sigma_{yy}n_y = g_{sf,y} , \\
\text{on } \gamma_{out} & : \sigma_{rr} = g_{out,r} , \sigma_{ry} = g_{out,y} , \\
\text{on } \gamma_s & : u_r = 0 , \sigma_{ry} = 0 .
\end{aligned} \tag{6.15}$$

6.2.2. Weak formulation for thermal model

We introduce weighted Sobolev spaces [3], $L_r^2(\omega)$ and $H_r^1(\omega)$ with respective norms $\|\cdot\|_{L_r^2(\omega)}$ and $\|\cdot\|_{H_r^1(\omega)}$ as,

$$\begin{aligned}
L_r^2(\omega) & = \left\{ \psi : \omega \mapsto \mathbb{R} , \int_{\omega} \psi^2 r dr dy < \infty \right\} , \\
& \|\psi\|_{L_r^2(\omega)}^2 = \int_{\omega} \psi^2 r dr dy , \\
H_r^1(\omega) & = \left\{ \psi : \omega \mapsto \mathbb{R} , \int_{\omega} \left(\psi^2 + \left(\frac{\partial \psi}{\partial r} \right)^2 + \left(\frac{\partial \psi}{\partial y} \right)^2 \right) r dr dy < \infty \right\} , \\
& \|\psi\|_{H_r^1(\omega)}^2 = \int_{\omega} \left(\psi^2 + \left(\frac{\partial \psi}{\partial r} \right)^2 + \left(\frac{\partial \psi}{\partial y} \right)^2 \right) r dr dy .
\end{aligned} \tag{6.16}$$

Before discussing the weak formulation of the thermal model (T1), we assume the following hypotheses on the data:

(TH1) The heat source term, Q , verifies

$$Q \in L_r^2(\omega) .$$

(TH2) The convection temperatures belong to the spaces,

$$T_{sf} \in L_r^2(\gamma_{sf}) , T_- \in L_r^2(\gamma_-) , T_{out} \in L_r^2(\gamma_{out}) .$$

(TH3) The boundary heat flux verifies

$$q_+ \in L_r^2(\gamma_+) .$$

(TH4) There exists a constant $k_0 > 0$, such that the thermal conductivity, $k(r, y)$ satisfies,

$$k(r, y) \in L^\infty(\omega) , k(r, y) > k_0 , \text{ in } \omega .$$

Also, there exist constants $h_{c,f,0} > 0, h_{c,out,0} > 0, h_{c,-,0} > 0$ such that,

$$\begin{aligned}
h_{c,f}(r, y) & \in L^\infty(\gamma_{sf}) , h_{c,f}(r, y) > h_{c,f,0} , \text{ on } \gamma_{sf} , \\
h_{c,out}(r, y) & \in L^\infty(\gamma_{out}) , h_{c,out}(r, y) > h_{c,out,0} \text{ on } \gamma_{out} , \\
h_{c,-}(r, y) & \in L^\infty(\gamma_-) , h_{c,-}(r, y) > h_{c,-,0} , \text{ on } \gamma_- .
\end{aligned}$$

In order to propose a weak formulation for the thermal model (T1), in the following we assume sufficient regularity to perform the following calculations. We multiply the energy equation (6.12) by $r\psi(r, y)$ and integrate over the axisymmetric domain ω with respect to (r, y) variables, so we obtain:

$$-\int_{\omega} \frac{\psi}{r} \frac{\partial}{\partial r} \left(r k \frac{\partial T}{\partial r} \right) r dr dy - \int_{\omega} \psi \frac{\partial}{\partial y} \left(k \frac{\partial T}{\partial y} \right) r dr dy = \int_{\omega} \psi Q r dr dy . \tag{6.17}$$



By applying Gauss divergence theorem, we deduce,

$$\int_{\omega} rk \left(\frac{\partial T}{\partial y} \frac{\partial \psi}{\partial y} + \frac{\partial T}{\partial r} \frac{\partial \psi}{\partial r} \right) drdy = \int_{\omega} \psi Q r drdy + \int_{\partial\omega} r \psi k \left(\frac{\partial T}{\partial y} n_y + \frac{\partial T}{\partial r} n_r \right) d\gamma, \quad (6.18)$$

where $\vec{n} = (n_r, n_y)$ is the unit outer normal vector to the boundaries of ω . Using boundary conditions from equation (6.13), expression (6.18) can be written as

$$\begin{aligned} \int_{\omega} rk \left(\frac{\partial T}{\partial y} \frac{\partial \psi}{\partial y} + \frac{\partial T}{\partial r} \frac{\partial \psi}{\partial r} \right) drdy + \int_{\gamma_{sf}} \psi h_{c,f} T r d\gamma + \int_{\gamma_{out}} \psi h_{c,out} T r d\gamma + \\ \int_{\gamma_-} \psi h_{c,-} T r d\gamma = \int_{\omega} \psi Q r drdy + \int_{\gamma_{sf}} \psi h_{c,f} T_f r d\gamma + \\ \int_{\gamma_{out}} \psi h_{c,out} T_{out} r d\gamma + \int_{\gamma_-} \psi h_{c,-} T_- r d\gamma - \int_{\gamma_+} \psi q^+ r d\gamma. \end{aligned} \quad (6.19)$$

Therefore, we propose the following weak formulation for thermal problem (T1):

Problem 5. Weak thermal formulation (WT1) : Under the assumptions (TH1)-(TH4), find $T \in H_r^1(\omega)$, such that equality (6.19) is verified for all $\psi \in H_r^1(\omega)$.

It is to be noted that under assumptions (TH1)-(TH4) all integrals of the proposed weak formulation are well defined. The left hand side of equation (6.19) is bilinear and symmetric. So, we define in $H_r^1(\omega) \times H_r^1(\omega)$ the operator:

$$a_T(T, \psi) = \int_{\omega} rk \left(\frac{\partial T}{\partial y} \frac{\partial \psi}{\partial y} + \frac{\partial T}{\partial r} \frac{\partial \psi}{\partial r} \right) drdy + \int_{\gamma_{sf}} \psi h_{c,f} T r d\gamma + \int_{\gamma_{out}} \psi h_{c,out} T r d\gamma + \int_{\gamma_-} \psi h_{c,-} T r d\gamma. \quad (6.20)$$

The right hand side of equation (6.19) is linear and the following linear operator defined on $H_r^1(\omega)$ is introduced:

$$l_T(\psi) = \int_{\omega} \psi Q r drdy + \int_{\gamma_{sf}} \psi h_{c,f} T_f r d\gamma + \int_{\gamma_{out}} \psi h_{c,out} T_{out} r d\gamma + \int_{\gamma_-} \psi h_{c,-} T_- r d\gamma - \int_{\gamma_+} \psi q^+ r d\gamma. \quad (6.21)$$

6.2.3. Weak formulation of the mechanical model

Before discussing the weak formulation for the mechanical model (M1) (see [4],[5]), the following space \mathbb{V} for the displacements is considered:

$$\mathbb{V} = (H_r^1(\omega) \cap L_{1/r}^2(\omega)) \times H_r^1(\omega).$$

It will be equipped with the inner product,

$$\begin{aligned} \langle \vec{\phi}, \vec{u} \rangle_{\mathbb{V}} = \int_{\omega} \left(\phi_r u_r + \phi_y u_y + \frac{\partial u_r}{\partial r} \frac{\partial \phi_r}{\partial r} + \frac{\partial u_r}{\partial y} \frac{\partial \phi_r}{\partial y} + \frac{u_r}{r} \frac{\phi_r}{r} + \frac{\partial u_y}{\partial r} \frac{\partial \phi_y}{\partial r} + \frac{\partial u_y}{\partial y} \frac{\partial \phi_y}{\partial y} + \frac{\partial u_r}{\partial y} \frac{\partial \phi_y}{\partial r} + \frac{\partial u_y}{\partial r} \frac{\partial \phi_r}{\partial y} \right) r drdy, \end{aligned} \quad (6.22)$$



and with the norm,

$$\|\vec{\phi}\|_{\mathbb{V}}^2 = \langle \vec{\phi}, \vec{\phi} \rangle_{\mathbb{V}} . \quad (6.23)$$

Its closed and convex subspace

$$\mathbb{U} = \{ \vec{\phi} = (\phi_r, \phi_y) \in \mathbb{V}, \phi_y = 0 \text{ on } \gamma_-, \phi_r = 0 \text{ on } \gamma_s \},$$

will be the set of admissible displacements.

The function space for stress tensor is defined as,

$$\mathbb{S} = \{ \boldsymbol{\sigma} = [\sigma_{ij}] \in [L_r^2(\omega)]^{3 \times 3}, \sigma_{ij} = \sigma_{ji}, \sigma_{\alpha 3} = 0 \} .$$

We assume the following hypotheses on the mechanical data:

(MH1) The body force density, \vec{f}_0 , is such that:

$$\vec{f}_0 \in [L_r^2(\omega)]^2 .$$

(MH2) The boundary forces verify the following regularity assumptions:

$$\begin{aligned} \vec{g}_+ &\in [L_r^2(\gamma_+)]^2, \vec{g}_{sf} \in [L_r^2(\gamma_{sf})]^2, \vec{g}_{out} \in [L_r^2(\gamma_{out})]^2, \\ \vec{g}_- &\in [L_r^2(\gamma_-)]^2, \text{ and } \vec{n} \cdot \vec{g}_- = 0 \text{ on } \gamma_- . \end{aligned}$$

(MH3) There exist constants $E_0 > 0$ and $\alpha_0 > 0$ such that, the Young's modulus, $E(r, y)$, and the coefficient of thermal expansion, $\alpha(r, y)$ satisfy:

$$\begin{aligned} E(r, y) &\in L^\infty(\omega), E > E_0, \\ \alpha(r, y) &\in L^\infty(\omega), \alpha > \alpha_0, \text{ in } \omega . \end{aligned}$$

(MH4) There exist constants $\nu_0 > 0, \nu_1 < 0.5$ such that, the Poisson's ratio, $\nu(r, y)$, satisfies:

$$\nu(r, y) \in L^\infty(\omega), \nu_0 < \nu < \nu_1, \text{ in } \omega .$$

It can be seen that in case $T = T_0$ in equation (6.10), the stress tensor $\boldsymbol{\sigma}(\vec{\phi})$ belongs to the space \mathbb{S} . Besides, if T and T_0 belong to $H_r^1(\omega)$, the stresses generated belong to \mathbb{S} too. In other words, the stresses generated due to mechanical effects lies also in the same space as that of stresses generated due to thermo-mechanical effects.

In order to propose a weak formulation of the mechanical model (6.14) - (6.15), we assume that all functions have sufficiently regularity as necessary for the following calculations. Given a function $\vec{\phi} = (\phi_r, \phi_y)$, we multiply the first equation of (6.14) by $r(\phi_r(r, y))$, the second one by $r\phi_y(r, y)$, we sum both, and integrate over ω to obtain

$$\begin{aligned} \int_{\omega} \left(\phi_r \frac{\partial \sigma_{rr}}{\partial r} + \phi_r \frac{\partial \sigma_{ry}}{\partial y} + \frac{\phi_r}{r} (\sigma_{rr} - \sigma_{\theta\theta}) + \phi_y \frac{\partial \sigma_{ry}}{\partial r} + \phi_y \frac{\partial \sigma_{yy}}{\partial y} + \phi_y \frac{\sigma_{ry}}{r} \right) r dr dy + \\ \int_{\omega} (\phi_r f_{0,r} + \phi_y f_{0,y}) r dr dy = 0 . \end{aligned}$$



So,

$$\int_{\omega} \left(\phi_r \frac{\partial \sigma_{rr}}{\partial r} + \phi_y \frac{\partial \sigma_{ry}}{\partial r} \right) r dr dy + \int_{\omega} \left(\phi_r \frac{\partial \sigma_{ry}}{\partial y} + \phi_y \frac{\partial \sigma_{yy}}{\partial y} \right) r dr dy + \int_{\omega} (\phi_r (\sigma_{rr} - \sigma_{\theta\theta}) + \phi_y \sigma_{ry}) dr dy + \int_{\omega} (\phi_r f_{0,r} + \phi_y f_{0,y}) r dr dy = 0 .$$

Then, applying Green formula, using boundary conditions given by equations in (6.15), taking into account (6.8) and (6.9), assuming that the normal component of $\vec{\phi}$ is null on $\gamma_- \cup \gamma_s$, we obtain,

$$\begin{aligned} \int_{\omega} \mathbf{A}\{\varepsilon(\vec{u})\} \cdot \{\varepsilon(\vec{\phi})\} r dr dy &= \int_{\omega} (2\mu + 3\lambda)\alpha(T - T_0)\{\mathbf{I}\} \cdot \{\varepsilon(\vec{\phi})\} r dr dy + \\ &\int_{\omega} (\phi_r f_{0,r} + \phi_y f_{0,y}) r dr dy + \int_{\gamma_{sf}} \vec{\phi} \cdot \vec{g}_{sf} r d\gamma + \int_{\gamma_{out}} \vec{\phi} \cdot \vec{g}_{out} r d\gamma + \\ &\int_{\gamma_-} \vec{\phi} \cdot \vec{g}_- r d\gamma + \int_{\gamma_+} \vec{\phi} \cdot \vec{g}_+ r d\gamma , \end{aligned} \quad (6.24)$$

where T is assumed to be the solution of the weak thermal model (WT1).

Firstly, notice that under assumptions (MH1)-(MH4), and since $T \in H_r^1(\omega)$, all integrals in (6.24) are well defined. Therefore, we propose the following weak formulation for the mechanical model:

Problem 6. Weak mechanical formulation (WM1) : Let $T \in H_r^1(\omega)$ be the solution of the weak thermal model (WT1). Under assumptions (MH1)-(MH4), find $\vec{u} \in \mathbb{U}$, such that equality (6.24) is verified for all $\vec{\phi} \in \mathbb{U}$.

The left hand side of equation (6.24) is bilinear in $\mathbb{V} \times \mathbb{V}$,

$$a_M(\vec{u}, \vec{\phi}) = \int_{\omega} \mathbf{A}\{\varepsilon(\vec{u})\} \cdot \{\varepsilon(\vec{\phi})\} r dr dy , \quad (6.25)$$

while the right hand side of the equation is linear in \mathbb{V} ,

$$\begin{aligned} l_M[T](\vec{\phi}) &= \int_{\omega} (2\mu + 3\lambda)\alpha(T - T_0)\{\mathbf{I}\} \cdot \{\varepsilon(\vec{\phi})\} r dr dy + \int_{\omega} (\phi_r f_{0,r} + \phi_y f_{0,y}) r dr dy + \\ &\int_{\gamma_{sf}} \vec{\phi} \cdot \vec{g}_{sf} r d\gamma + \int_{\gamma_{out}} \vec{\phi} \cdot \vec{g}_{out} r d\gamma + \int_{\gamma_-} \vec{\phi} \cdot \vec{g}_- r d\gamma + \int_{\gamma_+} \vec{\phi} \cdot \vec{g}_+ r d\gamma . \end{aligned} \quad (6.26)$$

One can also use the principle of superposition here. In other words, the net displacement at any point in the domain is the sum of the displacement due to purely mechanical effects $\vec{u}_M \in \mathbb{U}$ and the displacement due to purely thermal effects $\vec{u}_T \in \mathbb{U}$.

$$\begin{aligned} a_M(\vec{u}_M, \vec{\phi}) &= l_M[T_0](\vec{\phi}) , \quad \forall \vec{\phi} \in \mathbb{U} , \\ a_M(\vec{u}_T, \vec{\phi}) &= l_M[T](\vec{\phi}) - l_M[T_0](\vec{\phi}) , \quad \forall \vec{\phi} \in \mathbb{U} , \\ \vec{u} &= \vec{u}_M + \vec{u}_T . \end{aligned} \quad (6.27)$$

6.2.4. Finite element method

In the course of finite element analysis [6], the weak formulations are projected on the finite dimensional subspaces. For sake of brevity, the details about finite element analysis are only briefly described.

- We use the Galerkin method of weighted residuals.



- The Lagrange basis functions are used. The basis functions for displacement and temperature are piecewise polynomials of degree $p \geq 1$.
- Since, the bilinear forms $a_T(T, \psi)$ and $a_M(\vec{u}, \vec{\phi})$ are coercive, the matrices associated with the corresponding linear system are positive definite.
- Also, since the piecewise polynomials have compact support in the domain, the system matrices of the corresponding linear systems are sparse matrices.
- Since Galerkin weighted residual method is used, the symmetry of bilinear forms $a_T(T, \psi)$ and $a_M(\vec{u}, \vec{\phi})$ is maintained.

6.2.5. Parameter space

In the context of geometric parametrization [7, 8, 9], domain ω is characterized by some geometric parameters. The geometric parameters relevant in our analysis are the diameter of each section of the hearth D_0, D_1, D_2, D_3, D_4 , the volume of the hearth and the height of each section of the hearth t_0, t_1, t_2, t_3, t_4 (see Figure 6.3). The relevant material parameters in our analysis are thermal conductivity of the material, k , thermal expansion coefficient, α , and Lamé parameters, μ, λ . Let Ξ be the tuple of parameters and \mathbb{P} be the parameters space. Let $d_p = 14$ be the dimensionality of the parameter space. Therefore,

$$\Xi \in \mathbb{P} \subset \mathbb{R}^{d_p} .$$

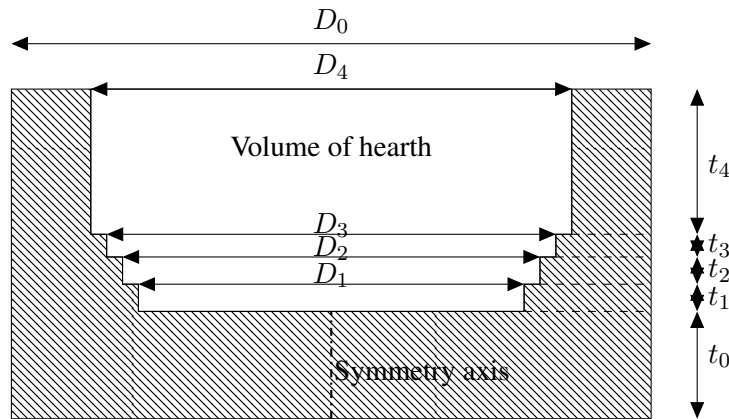


Figure 6.3: Hearth geometric parameters

Given a tuple $\Xi \in \mathbb{P}$, let T and \vec{u} be the solutions of problems (WT1) and (WM1), respectively. We assume the following affine parametric dependence i.e.,

$$\begin{aligned}
 a_T(T, \psi; \Xi) &= \sum_{i=1}^{n_{a_T}} \theta_{a_T,i}(\Xi) a_{T,i}(T, \psi) , \\
 l_T(\psi; \Xi) &= \sum_{i=1}^{n_{l_T}} \theta_{l_T,i}(\Xi) l_{T,i}(\psi) .
 \end{aligned}
 \tag{6.28}$$



and

$$\begin{aligned}
 a_M(\vec{u}, \vec{\phi}; \Xi) &= \sum_{i=1}^{n_{a_M}} \theta_{a_{M,i}}(\Xi) a_{M,i}(\vec{u}, \vec{\phi}), \\
 l_M[T](\vec{\phi}; \Xi) &= \sum_{i=1}^{n_{l_M}} \theta_{l_{M,i}}(\Xi) l_{M,i}[T](\vec{\phi}).
 \end{aligned} \tag{6.29}$$

The bilinear forms $a_T(T, \psi; \Xi)$, $a_M(\vec{u}, \vec{\phi}; \Xi)$ and the linear forms $l_T(\psi; \Xi)$, $l_M[T](\vec{\phi}; \Xi)$ are decomposed into n_{a_T} , n_{a_M} bilinear forms and n_{l_T} , n_{l_M} linear forms respectively and the latter are evaluated once in offline phase. In case new parameter tuple is given, instead of evaluating bilinear and linear forms, scalar parameters $\theta_{a_{T,i}}$, $\theta_{l_{T,i}}$, $\theta_{a_{M,i}}$, $\theta_{l_{M,i}}$ are evaluated during online phase. The affine expansion of operators is essentially a change of variables and helps to achieve offline-online decomposition of computations. The details of the affine expansion and offline-online decomposition has been explained in the literature such as [7].

6.2.6. Model order reduction

Various methods have been used for model order reduction of parametric systems with physical and/or geometric parameters (see [7], [10], [11]). The snapshot Proper Orthogonal Decomposition (POD) method with Galerkin projection is used in the present work. For description of POD-Galerkin method we refer to the section 3.2.1 of [7].

A coupled system is characterized by different subsystems interacting with each other through coupling. The coupled systems require special considerations while applying model order reduction techniques (see [12], [13]). In our approach, the coupled system is divided into 3 parts: thermal system, mechanical system and coupling system. Each individual subsystem can be written as below.

Thermal model:

$$a_T(T, \psi) = l_T(\psi), \quad \forall \psi \in H_r^1(\omega). \tag{6.30}$$

Mechanical model:

$$a_M(\vec{u}_M, \vec{\phi}) = l_M[T_0](\vec{\phi}), \quad \forall \vec{\phi} \in \mathbb{U}. \tag{6.31}$$

Coupling thermo-mechanical model:

$$a_M(\vec{u}_T, \vec{\phi}) = l_M[T](\vec{\phi}) - l_M[T_0](\vec{\phi}), \quad \forall \vec{\phi} \in \mathbb{U}. \tag{6.32}$$

The total displacement is then given by,

$$\vec{u} = \vec{u}_M + \vec{u}_T. \tag{6.33}$$

Generation of the reduced basis space and the Galerkin projection are steps performed during the offline phase. Solution of the smaller system of equations to compute the reduced basis approximations are steps performed during the online phase. It is to be noted that, during the online phase the matrices corresponding to the bilinear forms and the linear forms are assembled using the affine decomposition.

6.3. Implementation

All benchmark tests are performed in FEniCS [14] and/or RBniCS [7], whose licensing and the software version are mentioned below.

- FEniCS
 1. Website : <https://fenicsproject.org/>
 2. Download and installation : <https://fenicsproject.org/download/>



3. Licensing : The FEniCS Project is developed and maintained as a freely available, open-source project by a global community of scientists and software developers. The project is developed by the FEniCS Community, is governed by the FEniCS Steering Council and is overseen by the FEniCS Advisory Board.
 4. Version : 2019.1.0
- RBniCS
 1. Website : <https://mathlab.sissa.it/rbnics>
 2. Download and installation : <https://gitlab.com/RBniCS/RBniCS/>
 3. Licensing : RBniCS is freely available under the GNU LGPL, version 3.
 4. Version : 0.1.0

The mesh was created using Gmsh [15].

- Gmsh
 1. Website : <https://gmsh.info/>
 2. Download and installation : <https://gmsh.info/#Download>
 3. Licensing : <https://gmsh.info/#Licensing>
 4. Version : 4.0.0

For the post-processing of the results, the following visualization libraries are used.

- Paraview
 1. Website : <https://www.paraview.org/>
 2. Download and installation : <https://www.paraview.org/download/>
 3. Licensing : <https://www.paraview.org/paraview-license/>
 4. Version : 5.4.0
- Matplotlib
 1. Website : <https://matplotlib.org/>
 2. Download and installation : <https://matplotlib.org/users/installing.html>
 3. Licensing : <https://matplotlib.org/users/license.html>
 4. Version : 1.5.1

6.4. Computer requirements

The operating system used is Ubuntu 16.04.5 LTS and python version used is 3.5.2. Additionally, FEniCS is available on platforms Linux, Mac, Windows and also as docker image. RBniCS is available additionally as docker image (<https://hub.docker.com/r/rbnics/rbnics/>), on Google Colab (<https://colab.research.google.com/notebooks/intro.ipynb>) using Jupyter notebooks (<https://gitlab.com/RBniCS/RBniCS-jupyter>) and on ARGOS, the Advanced Reduced Groupware On-line Simulation platform (<https://argos.sissa.it/tutorials>). Gmsh runs on Windows, Mac OS X, Linux and most Unix variants.

6.5. Numerical examples

In this chapter we provide results for the benchmark tests to verify the numerical implementations (Section 6.5.1) including the simulation for actual problem (Section 6.5.1.4) and test problem for reduced basis method (Section 6.5.2).



6.5.1. Benchmark tests

In this chapter we provide results for the benchmark tests to verify the numerical implementations. Python and Gmsh codes that have been developed and used for this benchmark. The domain considered in this analysis corresponds to a real furnace and is a polygon in the $r - y$ plane with vertices (see Figure 6.2):

Dimension	Value (m)
Vertices (r, y)	(0, 0), (7.05, 0), (7.05, 7.265), (5.3, 7.265), (5.3, 4.065), (4.95, 4.065), (4.95, 3.565), (4.6, 3.565), (4.6, 2.965), (4.25, 2.965), (4.25, 2.365), (0, 2.365)
r_{max}	7.05
y_{max}	7.265

The two dimensional domain considered in the analysis was divided into 30 triangular subdomains. The division of domain ω into the triangular subdomains verifies the assumption of affine parameter dependence. The mesh considered is compatible with the triangular subdomains and it contains 8887 triangular elements and 4608 vertices. The mesh condition number is also presented (see Figure 6.4). The decrease in condition number of an element increases its distance from the set of degenerate elements. The condition number of an element ranges from 1 to ∞ , with 1 being a perfectly shaped element.

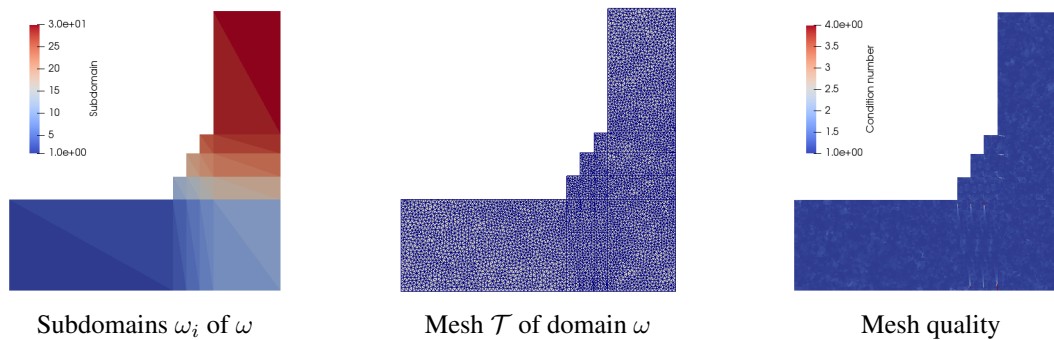


Figure 6.4: Discretization of domain ω

We also assess quality, q_e , of each element of mesh by the following formula (see [16]):

$$q_e = \frac{4\sqrt{3}A}{l_1^2 + l_2^2 + l_3^2}, \quad (6.34)$$

where, A is the area of an triangular element, and l_1, l_2 and l_3 are the lengths of each edge of the triangle. The minimum quality over all elements was 0.25 and the average quality over all elements was 0.94.

Following material properties are considered for all benchmark cases presented here, unless otherwise stated:

Material property	Value
Thermal conductivity	$k = 10 \frac{W}{mK}$
Convection coefficients	$h_{c,-} = 2000 \frac{W}{m^2K}$ $h_{c,f} = 200 \frac{W}{m^2K}$ $h_{c,out} = 2000 \frac{W}{m^2K}$
Young's modulus	$E = 5e9 Pa$
Poisson's ratio	$\nu = 0.2$
Thermal expansion coefficient	$\alpha = 10^{-6} / ^\circ K$
Density of Molten metal	$\rho_m = 7460 \frac{Kg}{m^3}$
Gravitation acceleration	$g = 10 \frac{m}{s^2}$

We also introduce some definitions, which will be used in the subsequent sections.

- Hydrostatic stress σ_m :

$$\sigma_m = \frac{1}{3} Tr(\boldsymbol{\sigma}) . \quad (6.35)$$

- Deviatoric part of the stress tensor $\boldsymbol{\sigma}_d$:

$$\boldsymbol{\sigma}_d = \boldsymbol{\sigma} - \frac{1}{3} Tr(\boldsymbol{\sigma}) \mathbf{I} = \boldsymbol{\sigma} - \sigma_m \mathbf{I} . \quad (6.36)$$

- Von Mises effective stress σ_{vm} :

$$\sigma_{vm} = \sqrt{\frac{3}{2} \boldsymbol{\sigma}_d : \boldsymbol{\sigma}_d} . \quad (6.37)$$

6.5.1.1. Thermal model

We consider as analytical temperature, the known temperature,

$$T_a = C' r^2 y, \quad C' = 1K/m^3 , \quad (6.38)$$

and the corresponding data for the thermal model are calculated. Then, using the obtained data, the computed temperature, solving (*WT1*) model, is compared with the known analytical temperature. It can be clearly verified that $\frac{\partial T_a}{\partial r} = 0$ on the symmetry boundary, γ_s .

From thermal model (*T1*) following data can be derived to ensure that T is the strong solution to the problem (*T1*).

The corresponding source term Q is given by,

$$Q(r, y) = -k \frac{\partial^2 T_a}{\partial r^2} - k \frac{\partial^2 T_a}{\partial y^2} - \frac{k}{r} \frac{\partial T_a}{\partial r} = -4C' k y . \quad (6.39)$$

Heat flux on γ_+ , that is q_+ , is given by,

$$q_+(r, y) = -k \frac{\partial T_a}{\partial r} n_r - k \frac{\partial T_a}{\partial y} n_y = -C' k r^2 . \quad (6.40)$$



The boundary temperatures T_f, T_{out}, T_- are given by,

$$T_f = T_a + \frac{k}{h_{c,f}} \nabla T_a \cdot \vec{n} = C' r^2 y + C' \frac{k}{h_{c,f}} (2ry n_r + r^2 n_y), \text{ on } \gamma_{sf}, \quad (6.41a)$$

$$T_{out} = T_a + \frac{k}{h_{c,out}} \nabla T_a \cdot \vec{n} = C' r^2 y + C' \frac{2ryk}{h_{c,out}}, \text{ on } \gamma_{out}, \quad (6.41b)$$

$$T_- = T_a + \frac{k}{h_{c,-}} \nabla T_a \cdot \vec{n} = C' r^2 y - C' \frac{r^2 k}{h_{c,-}}, \text{ on } \gamma_-. \quad (6.41c)$$

Therefore T_a is the analytical solution of the thermal problem (T1) for the data $Q, q_+, T_f, T_{out}, T_-$ given by expressions (6.39)-(6.41c). The analytical temperature with the temperature computed using an discretized space of polynomial of degree 3 is compared in Figure 6.5. For a more quantitative comparison, we computed the relative error in H_r^1 -norm (equation (6.16)), that was $2.4022e - 11$. We also assess p -convergence behavior and h -convergence behavior, using an discretized space of polynomial of degree 1, of the computed temperature field with the analytical temperature field in Figure 6.6.

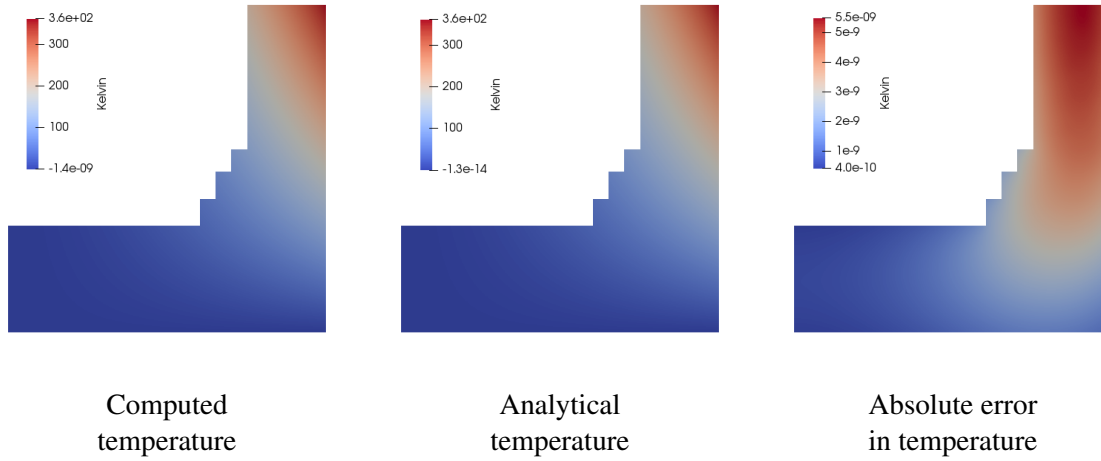


Figure 6.5: Benchmark tests for energy equation

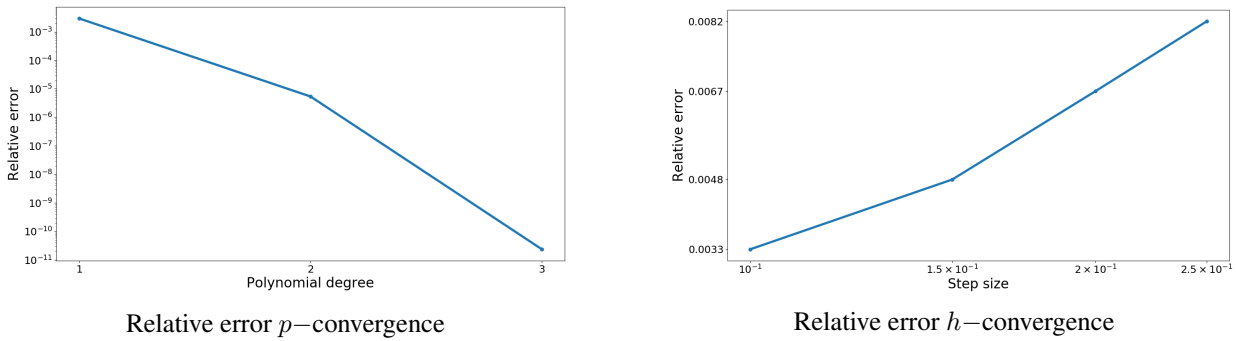


Figure 6.6: Convergence behavior for thermal model

6.5.1.2. Mechanical model

In this section, we consider that the body is at reference temperature i.e. thermal stresses are not present. We postpone the discussion of considering thermal stresses to benchmark test for coupling.



We consider a known displacement function

$$\vec{u}_a = C(ry^2, r^2y), \quad C = 1e - 4/m^2, \quad (6.42)$$

to mechanical problem, without considering thermal stresses, along with corresponding body force term. Clearly $\vec{u}_a \cdot \vec{n} = 0$ on $\gamma_- \cup \gamma_s$.

In order for \vec{u}_a to be a solution of the mechanical model (M1) with $T = T_0$ the following data must be considered:

The source term $\vec{f}_0 = [f_{0,r} \ f_{0,y}]$ is given by,

$$f_{0,r} = - \left(\frac{\partial \sigma_{rr}}{\partial r} + \frac{\partial \sigma_{ry}}{\partial y} + \frac{\sigma_{rr} - \sigma_{\theta\theta}}{r} \right) = - \left(\frac{2E\nu Cr}{(1-2\nu)(1+\nu)} + \frac{2ECr}{(1+\nu)} \right), \quad (6.43a)$$

$$f_{0,y} = - \left(\frac{\partial \sigma_{ry}}{\partial r} + \frac{\partial \sigma_{yy}}{\partial y} + \frac{\sigma_{ry}}{r} \right) = - \left(\frac{4ECy}{(1+\nu)} + \frac{4E\nu Cy}{(1-2\nu)(1+\nu)} \right). \quad (6.43b)$$

The boundary tractions are given by,

$$\text{on } \gamma_+ : g_{+,r} = \frac{2ECry}{(1+\nu)}, \quad g_{+,y} = \frac{E}{(1-2\nu)(1+\nu)} (2\nu Cy^2 + (1-\nu)Cr^2), \quad (6.44a)$$

$$\text{on } \gamma_- : u_y = 0, \quad g_{-,r} = -\frac{2ECry}{(1+\nu)}, \quad (6.44b)$$

$$\text{on } \gamma_{sf} : g_{sf,r} = \frac{E}{(1-2\nu)(1+\nu)} (Cy^2 + \nu Cr^2) n_r + \frac{2ECry}{(1+\nu)} n_y, \quad (6.44c)$$

$$g_{sf,y} = \frac{2ECry}{(1+\nu)} n_r + \frac{E}{(1-2\nu)(1+\nu)} (2\nu Cy^2 + (1-\nu)Cr^2) n_y, \quad (6.44d)$$

$$\text{on } \gamma_{out} : g_{out,r} = \frac{E}{(1-2\nu)(1+\nu)} (Cy^2 + \nu Cr^2), \quad g_{out,y} = \frac{2ECry}{(1+\nu)}. \quad (6.44e)$$

Thus, the \vec{u}_a is the displacement solution of mechanical model (M1) for the data given by expressions (6.43a) - (6.44e) and $T = T_0$. The comparison between the known analytical displacement with computed displacement is given in Figure 6.7. The comparison between the Von Mises stress (equation (6.37)) calculated from analytical displacement and the Von Mises stress calculated from computed displacement is given in Figure 6.8. For a more quantitative comparison, we computed the relative error in \mathbb{U} -norm (equation (6.23)), that was $5.8030e - 13$. We also assess p -convergence behavior and h -convergence behavior, using an discretized space of polynomial of degree 1, of the computed displacement field with the analytical displacement field in Figure 6.9.



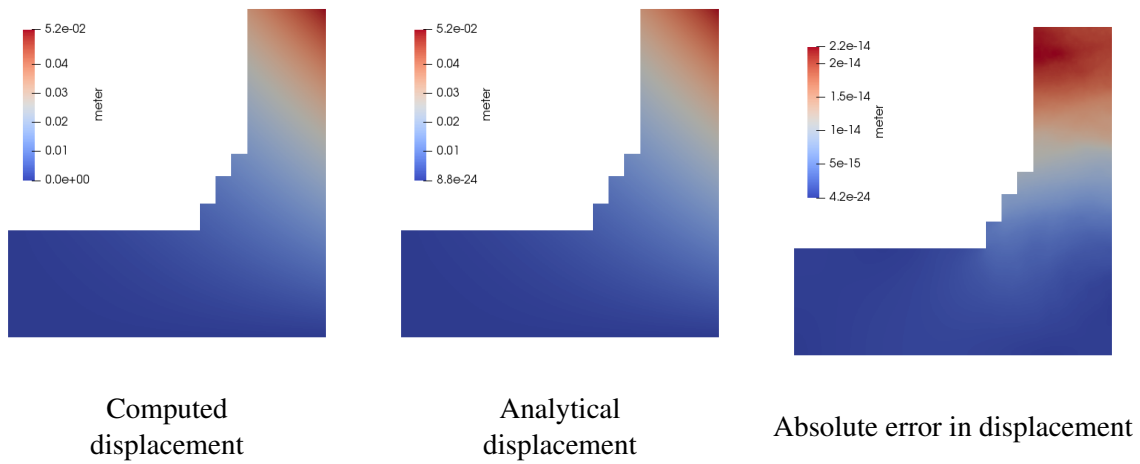


Figure 6.7: Benchmark tests for mechanical model : Displacement

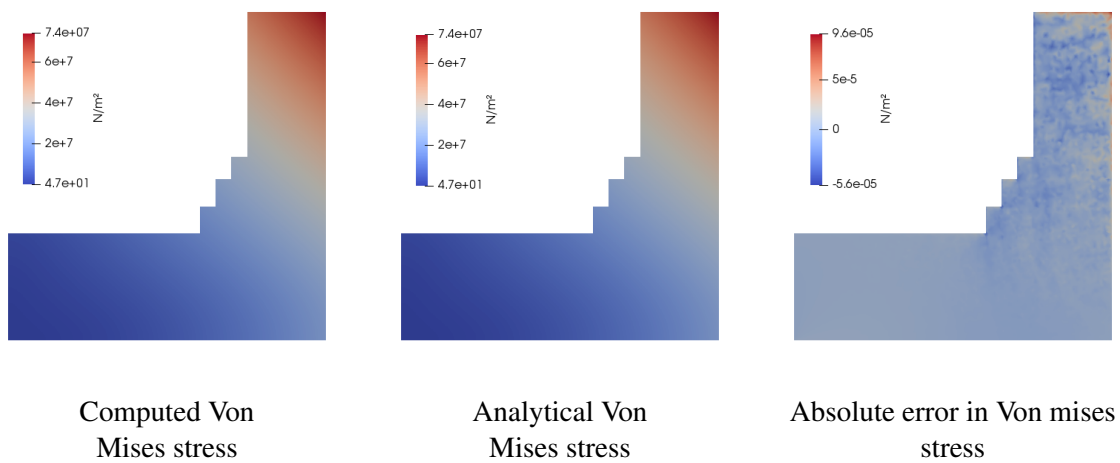


Figure 6.8: Benchmark tests for mechanical model : Von Mises stress

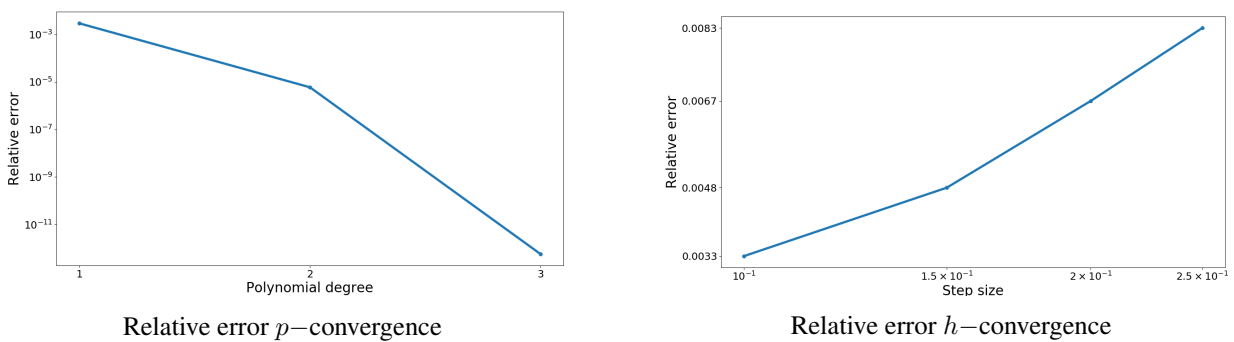


Figure 6.9: Convergence behavior in displacements for mechanical model

6.5.1.3. Coupled model

We first compute the hydrostatic thermomechanical stress (equation (6.35)) when the domain is subjected to combined mechanical and thermal effect. We assume the displacement function,

$$\vec{u}_a = C(r^2y, r^2y), \quad C = 1e - 4/m^2, \quad (6.45)$$

and the temperature function,

$$T_a = C'r^2y, \quad C' = 1K/m^3.$$

\vec{u}_a is the displacement solution of mechanical model (M1) for the data given by expressions (6.43a)-(6.44e) and $T = T_0$ as indicated in Section 6.5.1.2. T_a is the solution of thermal model for the data given by expressions (6.39)-(6.41c) as indicated in Section 6.5.1.1. From mechanical model (M1) following data can be derived.

The source term $\vec{f}_0 = [f_{0,r} \ f_{0,y}]$ is given by,

$$f_{0,r} = - \left(\frac{\partial \sigma_{rr}}{\partial r} + \frac{\partial \sigma_{ry}}{\partial y} + \frac{\sigma_{rr} - \sigma_{\theta\theta}}{r} \right) = - \left(\frac{2E\nu Cr}{(1-2\nu)(1+\nu)} + \frac{2ECr}{(1+\nu)} - \frac{2C'ryE\alpha}{(1-2\nu)} \right), \quad (6.46a)$$

$$f_{0,y} = - \left(\frac{\partial \sigma_{ry}}{\partial r} + \frac{\partial \sigma_{yy}}{\partial y} + \frac{\sigma_{ry}}{r} \right) = - \left(\frac{4ECy}{(1+\nu)} + \frac{4E\nu Cy}{(1-2\nu)(1+\nu)} - \frac{C'r^2E\alpha}{(1-2\nu)} \right). \quad (6.46b)$$

The boundary tractions are given by,

$$\text{on } \gamma_+ : g_{+,r} = \frac{2ECry}{(1+\nu)}, \quad g_{+,y} = \frac{E}{(1-2\nu)(1+\nu)} (2\nu Cy^2 + (1-\nu)Cr^2) - \frac{E\alpha}{(1-2\nu)} (C'r^2y - T_0), \quad (6.47a)$$

$$\text{on } \gamma_- : u_y = 0, \quad g_{-,r} = -\frac{2ECry}{(1+\nu)}, \quad (6.47b)$$

$$\text{on } \gamma_{sf} : g_{sf,r} = \left(\frac{E}{(1-2\nu)(1+\nu)} (Cy^2 + \nu Cr^2) - \frac{E\alpha}{(1-2\nu)} (C'r^2y - T_0) \right) n_r + \frac{2ECry}{(1+\nu)} n_y, \quad (6.47c)$$

$$g_{sf,y} = \frac{2ECry}{(1+\nu)} n_r + \left(\frac{E}{(1-2\nu)(1+\nu)} (2\nu Cy^2 + (1-\nu)Cr^2) - \frac{E\alpha}{(1-2\nu)} (C'r^2y - T_0) \right) n_y, \quad (6.47d)$$

$$\text{on } \gamma_{out} : g_{out,r} = \frac{E}{(1-2\nu)(1+\nu)} (Cy^2 + \nu Cr^2) - \frac{E\alpha}{(1-2\nu)} (C'r^2y - T_0), \quad g_{out,y} = \frac{2ECry}{(1+\nu)}. \quad (6.47e)$$

Firstly, we analyse the analytical solution of the thermomechanical model, in displacements and stresses, with the values computed (see Figures 6.10, 6.11). Finally, notice that the difference between the hydrostatic stress computed with the thermomechanical model and the one computed with the mechanical model should be equal to the thermal part of the thermomechanical stress; this comparison is shown in Figure 6.12. For a more quantitative comparison, we computed the relative error in \mathbb{U} -norm (equation (6.23)), that was $7.3134e - 13$. We also assess p -convergence behavior and h -convergence behavior of the computed displacement field, using an discretized space of polynomial of degree 1, with the analytical one in Figure 6.13.



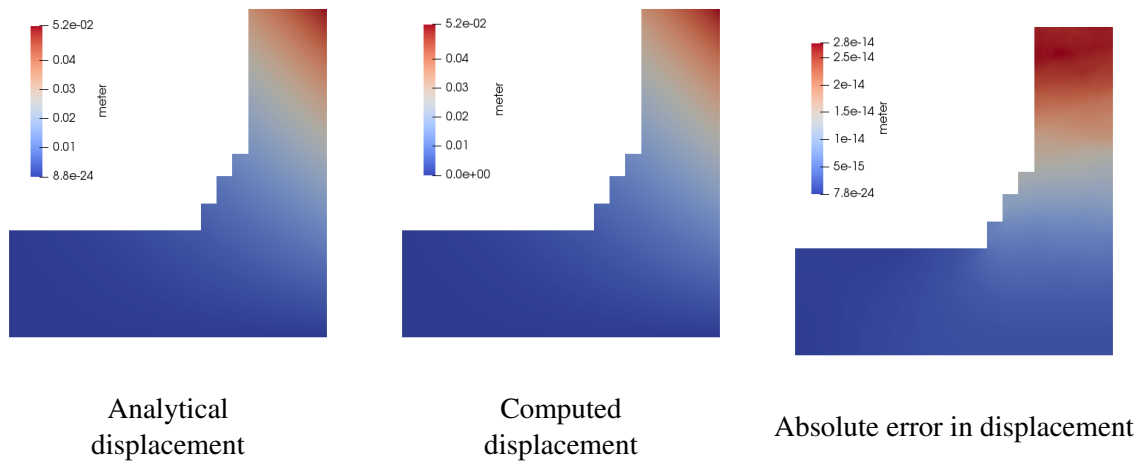


Figure 6.10: Benchmark tests for coupling : Displacement

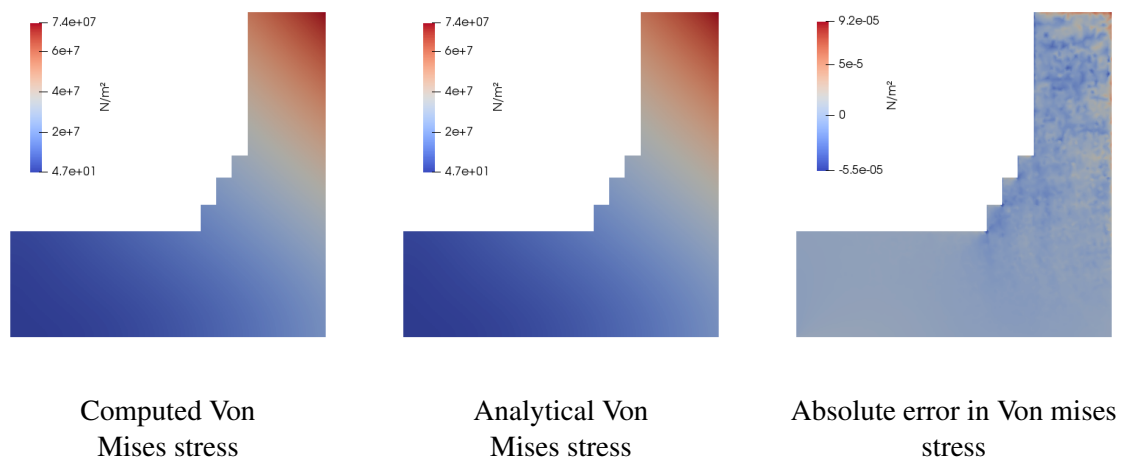


Figure 6.11: Benchmark tests for coupling : Von Mises stress

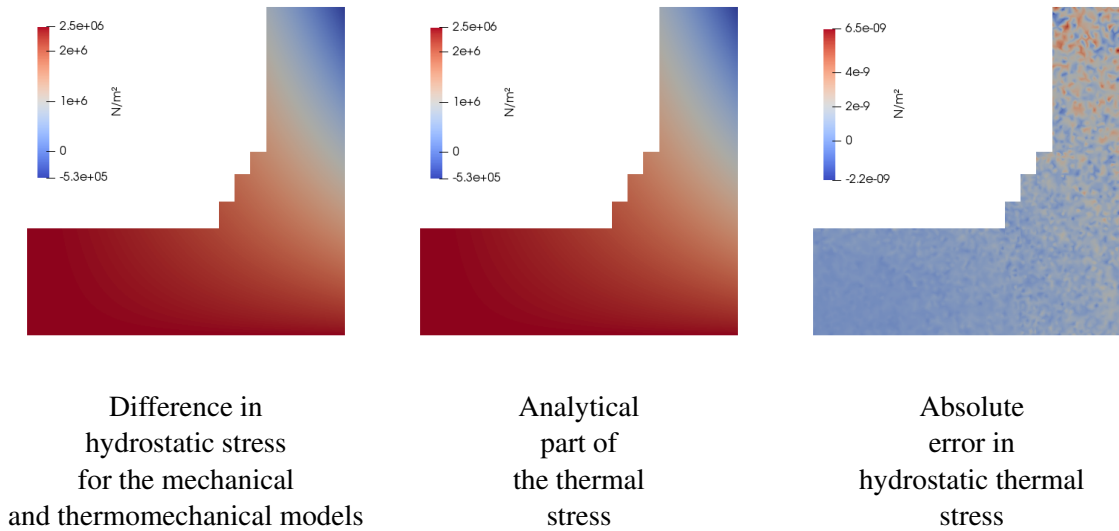


Figure 6.12: Benchmark tests for coupling : Hydrostatic stress

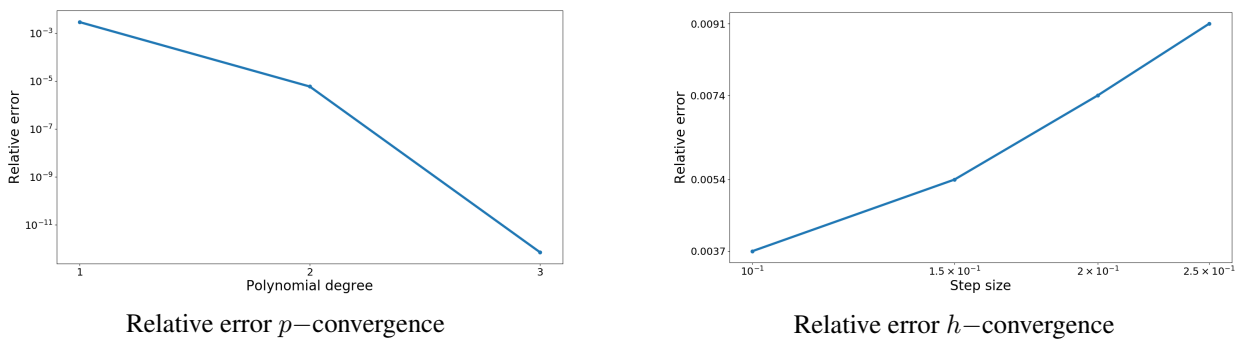


Figure 6.13: Convergence behavior in displacements for coupled model

6.5.1.4. Simulation for actual problem

We now present the computed temperature and the computed displacement profiles for the problem of blast furnace hearth (see Figure 6.14). The geometry and the physical parameters are those introduced in Section 6.5.1. Besides, the following data were considered:

$$T_0 = 298K, Q = 0, q_+ = 0, T_f = 1773K, T_{out} = 313K, T_- = 313K.$$

$$g_{+,r} = g_{+,y} = g_{-,r} = g_{out,r} = g_{out,y} = 0, f_{0,r} = 0, f_{0,y} = 0,$$

$$\text{on } \gamma_{sf} : \vec{g}_{sf} = -\rho_m * g * (y_{max} - y) = -7460 * 10 * (7.265 - y) \vec{n},$$

$$\text{on } \gamma_s \cup \gamma_- : \vec{u} \cdot \vec{n} = 0.$$

In particular, \vec{g}_{sf} is the hydrostatic pressure exerted by the molten metal on the surface γ_{sf} .





Figure 6.14: Temperature and displacement profiles

6.5.2. Test problem for reduced basis method

We now perform the model order reduction for the actual problem presented in the section 6.5.1.4 using the methodology described in the section 6.2.6. We introduce two important quantities in order to investigate the efficiency of the reduced order model developed :

- The Speedup, that is the ratio of the time taken to solve the finite element system of equations to the time taken to solve the reduced basis system of equations.
- The relative error, defined in the following way

$$\epsilon_{rel, X_h} = \frac{\|X_h - X_h^{rb}\|}{\|X_h\|}$$

where X_h and X_h^{rb} are finite element solution and the corresponding reduced basis solution respectively. $\|\cdot\|$ is the relevant norm ($\|\cdot\|_{H_r^1(\omega)}$ or $\|\cdot\|_{\mathbb{U}}$).

Analysis of the speedup and the relative error between the finite element solution and the reduced basis solution in respective norms are given in figures 6.15, 6.16 and 6.17. The number of parameter tuples considered for training, error analysis and speedup are given in Table 6.1. The range of parameters for training and testing, and the values of the parameters related to the parametric domain are given in Table 6.2. The parameters were generated randomly. The minimum admissible eigenvalue for reduced basis space was kept at $1e - 4$.

System	Training parameters	Error analysis	Speedup
Thermal system	500	10	10
Mechanical system	500	10	10
Coupling system	500	10	10

Table 6.1: Number of parameters for training, testing and speedup analysis



Parameter	Minimum value	Maximum value	Parametric domain
t_0	2.3	2.4	2.365
t_1	0.5	0.7	0.6
t_2	0.5	0.7	0.6
t_3	0.4	0.6	0.45
t_4	3.05	3.35	3.2
D_0	13.5	14.5	14.10
D_1	8.3	8.7	8.30
D_2	8.8	9.2	9.2
D_3	9.8	10.2	9.9
D_4	10.4	10.8	10.6
k	9.8	10.2	10
μ	1.9e9	2.5e9	2.08e9
λ	1.2e9	1.8e9	1.39e9
α	0.8e-6	1.2e-6	1e-6

Table 6.2: Ranges of parameters used for training, testing and speedup analysis, and values of the parameters related to the parametric domain

The finite element solution and the reduced basis solution are compared by assessing the spatial distribution of absolute error between the finite element solution and the reduced basis solution for the parametric domain (see Table 6.2 and figures 6.18, 6.19, 6.20).

6.5.2.1. Thermal system

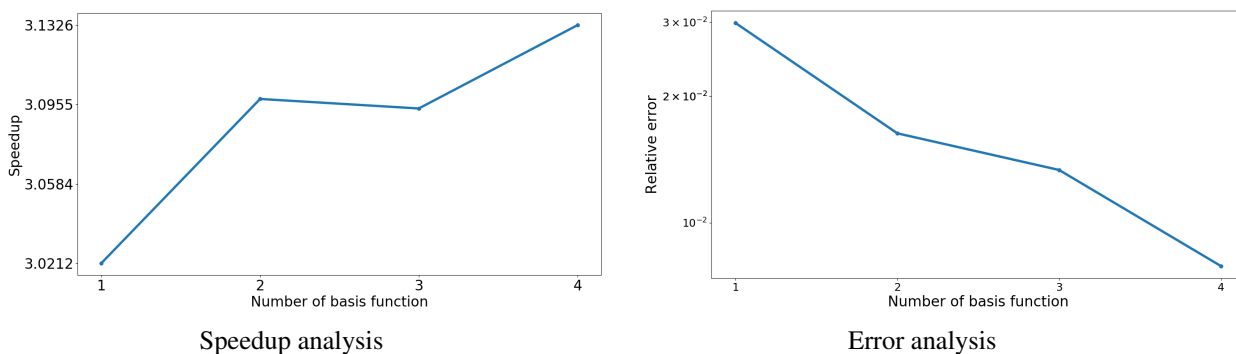


Figure 6.15: Thermal system reduced basis analysis : temperature field

6.5.2.2. Mechanical system

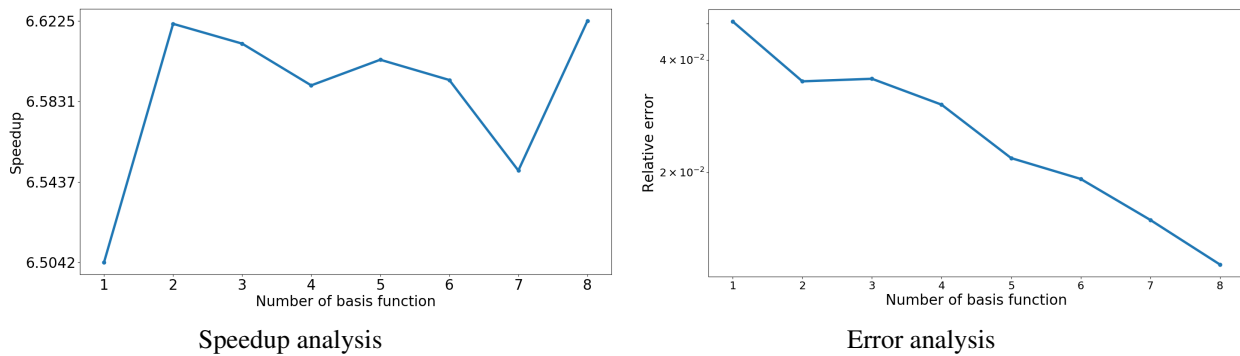


Figure 6.16: Mechanical system reduced basis analysis : displacement field

6.5.2.3. Coupling system

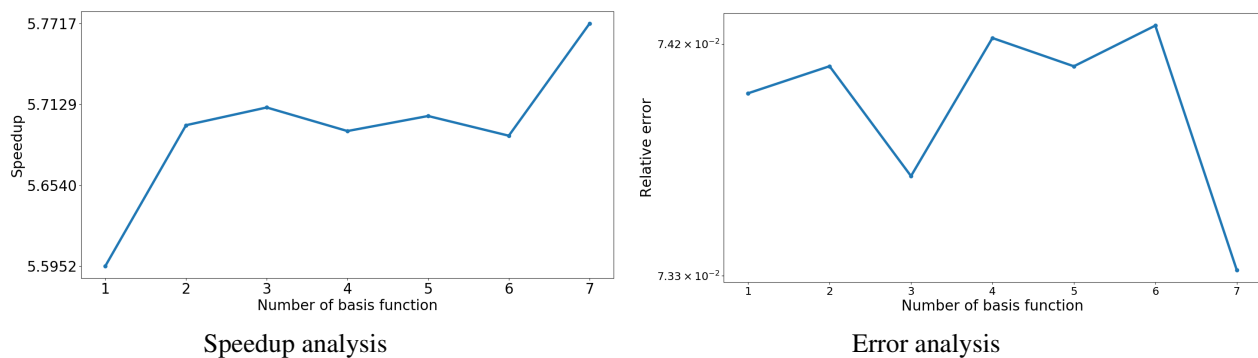


Figure 6.17: Coupling system reduced basis analysis : displacement field

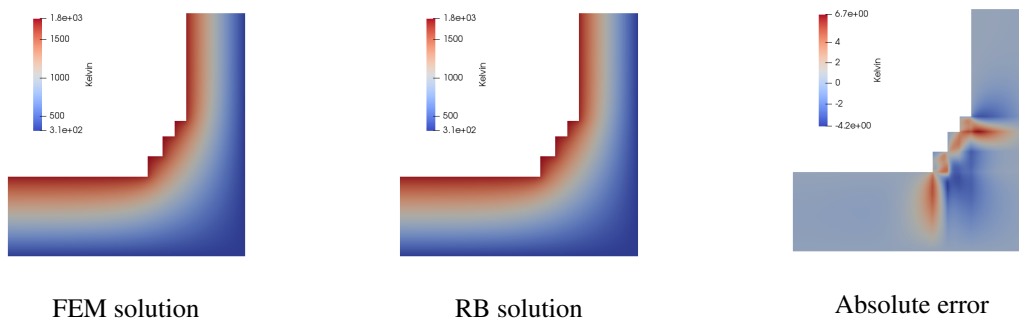


Figure 6.18: FEM and RB solution for thermal system : temperature field

6.3. Hierarchy of thermal model

6.4. Hierarchy of mechanical model

Bibliography

- [1] M. Gurtin, *An Introduction to Continuum Mechanics*. Academic Press, 1981.

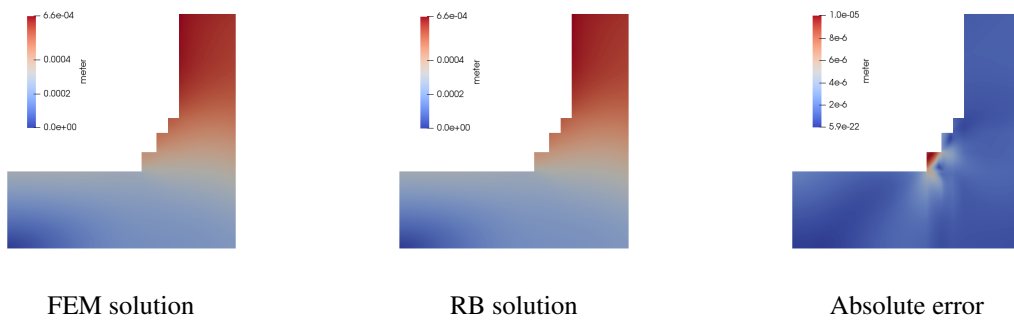


Figure 6.19: FEM and RB solution for mechanical system : displacement field

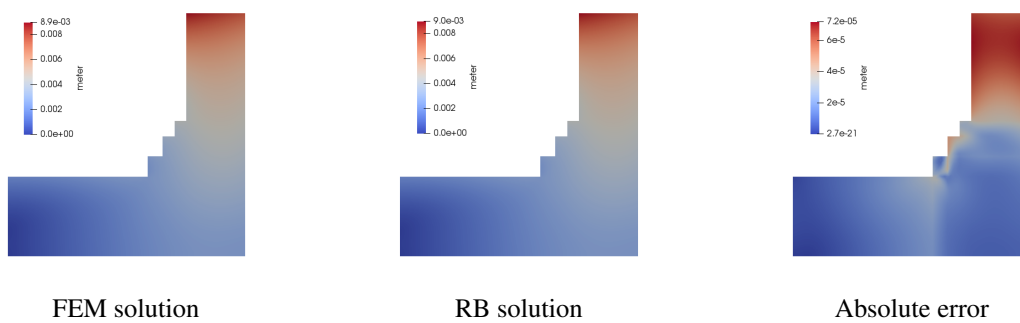


Figure 6.20: FEM and RB solution for coupling system : displacement field

Energy conservation on the deformed configuration

Energy conservation on the reference configuration

Steady state heat transfer

Fourier's law for anisotropic materials

Isotropic materials

Axisymmetric hypothesis

Axisymmetric heat conduction equation

[2] A. Bermúdez de Castro, "Continuum thermomechanics. progress in mathematical physics," *Meccanica*, vol. 41, pp. 697–698, 12 2006.

[3] H. Li, "Finite element analysis for the axisymmetric laplace operator on polygonal domains," *J. Computational Applied Mathematics*, vol. 235, pp. 5155–5176, 07 2011.



Momentum conservation on the deformed configuration

Momentum conservation on the reference configuration

Steady state momentum balance

Model associated with elastic materials

Linearized model

Axisymmetric hypothesis

Axisymmetric momentum conservation equation

- [4] I. Hlaváček, “Shape optimization of elastic axisymmetric bodies,” *Aplikace matematiky*, vol. 34, no. 3, pp. 225–245, 1989. [Online]. Available: <http://eudml.org/doc/15578>
- [5] I. Hlaváček, “Korn’s inequality uniform with respect to a class of axisymmetric bodies,” *Aplikace Matematiky*, vol. 34, 01 1989.
- [6] O. C. Zienkiewicz et. al., *The Finite Element Method: Its Basis and Fundamentals, Sixth Edition*, 6th ed. Butterworth-Heinemann, May 2005.
- [7] J. S. Hesthaven et. al., *Certified Reduced Basis Methods for Parametrized Partial Differential Equations*, ser. SpringerBriefs in Mathematics. Springer International Publishing, 2015.
- [8] D. B. P. Huynh et. al., “Reduced basis approximation and a posteriori error estimation for stress intensity factors,” *International Journal for Numerical Methods in Engineering*, vol. 72, no. 10, pp. 1219–1259, 2007. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.2090>
- [9] G. Rozza et. al., “Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations,” *Archives of Computational Methods in Engineering*, vol. 15, pp. 1–47, 09 2007.
- [10] K. C. Hoang et. al., “Fast and accurate two-field reduced basis approximation for parametrized thermoelasticity problems,” *Finite Elements in Analysis and Design*, vol. 141, pp. 96 – 118, 2018. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0168874X17304870>
- [11] J. S. Hesthaven et. al., “Non-intrusive reduced order modeling of nonlinear problems using neural networks,” *Journal of Computational Physics*, vol. 363, 02 2018.
- [12] W. H. A. Schilders et. al., *A Novel Approach to Model Order Reduction for Coupled Multiphysics Problems*. Cham: Springer International Publishing, 2014, pp. 1–49. [Online]. Available: https://doi.org/10.1007/978-3-319-02090-7_1
- [13] S. Zhang et. al., “Reduced order variational multiscale enrichment method for thermo-mechanical problems,” *Computational Mechanics*, 02 2017.
- [14] M. S. Alnæs et.al., “The fenics project version 1.5,” *Archive of Numerical Software*, vol. 3, no. 100, 2015.
- [15] C. Geuzaine et. al., “Gmsh: A 3-d finite element mesh generator with built-in pre- and post-processing



- facilities,” *International Journal for Numerical Methods in Engineering*, vol. 79, no. 11, pp. 1309–1331, 2009. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.2579>
- [16] S. Vázquez-Fernández et. al., “Mathematical modelling and numerical simulation of the heat transfer in a trough of a blast furnace,” *International Journal of Thermal Sciences*, vol. 137, pp. 365 – 374, 2019. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1290072917320434>



Part III.

Optimization methods



7. A benchmark for atmospheric tomography

Bernadett Stadler¹, Ronny Ramlau¹, Andreas Obereder²

¹*Industrial Mathematics Institute, Johannes Kepler Universität Linz*

²*MathConsult*

Abstract. The new generation of ground-based extremely large telescopes requires highly efficient algorithms to achieve an excellent image quality in a large field of view. These systems rely on adaptive optics (AO), where one aims to compensate the rapidly changing optical distortions in the atmosphere in real-time. Many of systems require the reconstruction of the turbulence layers, which is called atmospheric tomography. Mathematically, this problem is ill-posed, due to the small angle of separation. The dimension of the problem depends on the telescope size and has increased in the last years. Altogether, efficient solution methods are of great interest. Within this benchmark case we will use the standard, however, not most efficient method, called Matrix Vector Multiplication, to deal with the problem of atmospheric tomography.

Keywords: Inverse Problems, Adaptive Optics, Atmospheric Tomography, Matrix-Vector-Multiplication, Extremely Large Telescope

7.1. Introduction

The new generation of planned earthbound Extremely Large Telescopes (ELT) require highly efficient algorithms to achieve an excellent image quality in a large field of view. These systems rely on technique called Adaptive Optics (AO) with the task to compensate in real-time the rapidly changing optical distortions caused by atmospheric turbulences. To achieve such a correction, the deformations of optical wavefronts emitted by natural or artificial guide stars are measured via wavefront sensors and, subsequently, corrected using deformable mirrors.

Many of those systems require the reconstruction of the turbulence profile in the atmosphere, which is called atmospheric tomography. Mathematically, this problem is ill-posed, i.e., there is an unstable relation between measurement data and the solution. Hence, regularization techniques must be applied. A common way of dealing with this problem is the Bayesian formulation, where the statistical information regarding the turbulence model and sensor noise can be incorporated. The dimension of the atmospheric tomography problem depends on the number of subapertures of the used wavefront sensors and on the number of degrees of freedom of the correcting mirrors, which are in general higher for bigger telescopes. Moreover, the solution has to be computed in real-time. Altogether, efficient solution methods are of great interest for this kind of problems.

So far, the standard method for atmospheric tomography is the matrix-vector-multiplication (MVM). The computational costs of the MVM scales at $\mathcal{O}(n^2)$, where n is the dimension of the AO system. These dimension is increasing drastically in the next generation of ground-based telescopes, as e.g., the Extremely Large Telescope.

Within this work, the MVM method will serve as the benchmark case. Before the benchmark method is described in detail in Section 7.2.3, we first introduce the mathematical formulations of guide stars, turbulence statistics, deformable mirrors, wavefront sensors and the atmospheric tomography problem in the upcoming section. For more details about AO we refer to [1]. In Section 7.3 the implementation of the benchmark method is described, including programming language and used libraries. Section 7.4 provides information about computer requirements, the compiler and the build process. The last section shows a numerical example with the input physical parameter setting and the output numerical result.



7.2. Mathematical formulation

In this section, we describe the mathematical models used in the context of adaptive optics for, e.g., guide stars, wavefront sensors or deformable mirrors. Further, we show the problem formulation of atmospheric tomography and a possible solution method called MVM.

7.2.1. Adaptive optics

7.2.1.1. Guide stars

Guide stars (GS) are either natural stars up in the sky near the object of interest or generated by a laser beam.

Natural Guide Star

A natural guide star (NGS) is a bright star that serves as a reference point for the WFS to detect atmospheric distortions. The star is modelled as a point source at a height of infinity. Assuming the layered atmospheric model, the wavefront aberrations in the direction θ of a NGS are given by

$$\varphi_{\theta}(x) = (P_{\theta}^{NGS} \phi)(x) := \sum_{\ell=1}^L \phi_{\ell}(x + \theta h_{\ell}), \quad (7.1)$$

where ϕ_{ℓ} is the turbulent layer at altitude h_{ℓ} for $\ell = 1, \dots, L$. We call P_{θ}^{NGS} the geometric propagation operator in the direction of the NGS.

Within our benchmark case we assume that the photon noise from the NGS, that affects the WFS measurements, is modeled by a Gaussian random variable with zero mean and covariance matrix C_{η} . The noise is identically distributed in each subaperture and the x- and y-measurements noise is uncorrelated. Thus, the covariance matrix can be defined by

$$C_{\eta} = \sigma^2 I, \quad (7.2)$$

where σ^2 is the noise variance of a single measurement, which is given by

$$\sigma^2 = \frac{1}{n_{photons}}, \quad (7.3)$$

where $n_{photons}$ is the number of photons per subaperture.

Laser Guide Star

For a laser guide star (LGS) the model is slightly more complicated than for NGS. In particular, two important effects are taken into account in our benchmark case.

In contrast to the infinite height that is assumed for a NGS, the LGS is considered to be a fixed point at a finite height H . Due to the finite altitude, the light detected by the telescope passes through a cone-like volume in the atmosphere (see Figure 7.1). This behaviour is referred to as the **cone effect**.



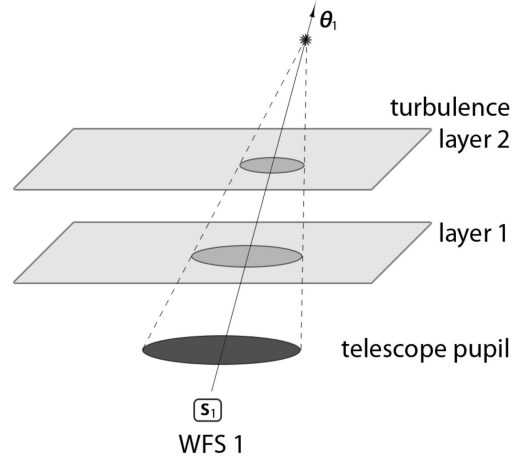


Figure 7.1: Cone effect

Assuming a layered model of the atmosphere, as for the NGS case, the incoming wavefront aberrations in the direction θ of a LGS are given by

$$\varphi_{\theta}(x) = (P_{\theta}^{LGS} \phi)(x) := \sum_{\ell=1}^L \phi_{\ell} \left(\left(1 - \frac{h_{\ell}}{H}\right)x + \theta h_{\ell} \right), \quad (7.4)$$

where P_{θ}^{LGS} is called the geometric propagation operator in the direction of the LGS.

For a LGS the sodium layer thickness has to be taken into account for modelling the photon noise. As the sodium layer has a certain width, the scattering of the laser beam happens in a vertical stripe, instead of in a single point. This stripe is observed as an elongated spot by the charge-coupled device (CCD) detector of the WFS. Thus, this effect is called **spot elongation**.

The vertical density profile of the laser beam scatter is modelled by a Gaussian random variable with mean H and a full width at half maximum (FWHM) parameter, which is defined by

$$FWHM = 2\sqrt{2\ln(2)}\sigma. \quad (7.5)$$

Further, we define the laser launch positions as (x_1^{LL}, x_2^{LL}) and the midpoint of a subaperture Ω_{ij} by

$$\bar{x}_i = \frac{x_i + x_{i+1}}{2}, \quad (7.6)$$

for $0 \leq i < n_s$ where the x_i are given by (7.19).

The elongation vector in a subaperture Ω_{ij} is given by

$$\beta_{ij} = (\beta_{ij,1}, \beta_{ij,2}) = \frac{FWHM}{H^2} ((\bar{x}_i, \bar{x}_j) - (x_1^{LL}, x_2^{LL})). \quad (7.7)$$

The spot elongated noise covariance matrix in a subaperture is given by

$$C_{ij} = \sigma^2 \left(I + \frac{\alpha_{\eta}^2}{f^2} \begin{pmatrix} \beta_{ij,1}^2 & \beta_{ij,1}\beta_{ij,2} \\ \beta_{ij,1}\beta_{ij,2} & \beta_{ij,2}^2 \end{pmatrix} \right), \quad (7.8)$$

where σ is defined as in (7.3), f is the FWHM of the non-elongated spot and α_{η} is a fine-tuning parameter to



cope with other sources of noise (e.g. read out noise).

Summarized, the noise model for the WFS associated to an LGS is given by a Gaussian random variable with zero mean and covariance matrix

$$C_{\eta} = \text{diag}(C_{ij}), \quad (7.9)$$

with $0 \leq i, j < n_s$ for an active subaperture Ω_{ij} .

7.2.1.2. Operating modes

Depending on the number of NGS and LGS the AO systems operates in different modes, which are listed in the following subsections.

Single Conjugate AO

If the object of interest, e.g., a star or a galaxy, is located near a bright NGS, the classical AO system Single Conjugate AO (SCAO) is used. In a SCAO system the wavefront is reconstructed using one WFS, that measures the data, and one DM, where the shape is chosen according to the reconstruction. One issue with SCAO systems is that the further away the object of interest is from the NGS, the worse is the correction of the wavefront.

Laser Tomography AO

If no NGS is available in the vicinity of the object of interest, the usage of an SCAO system is not possible. The idea is to generate LGS to obtain a good correction. This LGS is combined with at least one NGS to correct for the low order modes, which are not available using only LGS. In the general, a combination of several LGS and NGS is possible.

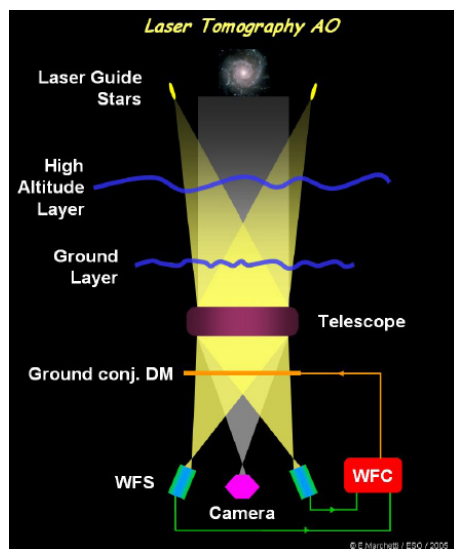


Figure 7.2: Principle of LTAO

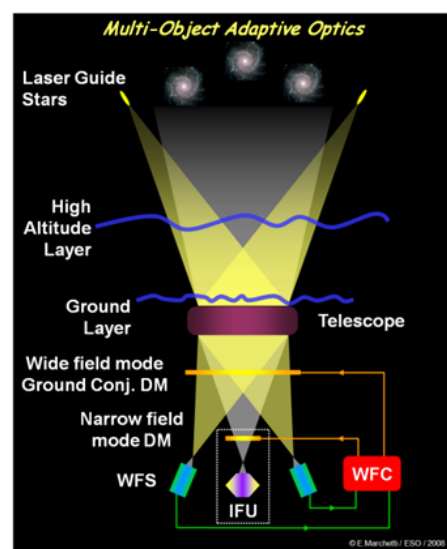


Figure 7.3: Principle of MOAO

Within the framework of a laser tomography AO (LTAO) G_{LGS} and G_{NGS} are used in combination with a single mirror to reconstruct the wavefront. The correction is performed through two steps. The first step is called atmospheric tomography, where the turbulent layers are reconstructed from sensor measurements. In the second step, the shape of the DM is chosen according to the projection of the wavefront through the reconstructed layers in the direction of interest.

Multi Object AO

In contrast to LTAO multi object AO (MOAO) corrects for multiple directions of interest, simultaneously, by using several mirrors. Each mirror corrects for a specific direction. As in the LTAO case a combination of NGS and LGS is used for reconstructing the layers.

Multi Conjugate AO

As in MOAO, a Multi Conjugate AO (MCAO) system corrects for multiple directions, however, with the aim to achieve a uniformly optimal correction over the whole field of view and not into specific directions. For that purpose, several DMs are used conjugated to different heights in the atmosphere.

7.2.1.3. Turbulence statistic in the atmosphere

The main source of distortions of the wavefront are atmospheric turbulences, which emerge from irregular mixing of cold and hot air affected by the sun and wind. Due to these irregularities the refractive index of air is inhomogeneous. This leads to a distorted wavefront arriving at the telescope pupil. Since these turbulence effects are not predictable, we model the turbulent layers as a Gaussian random variable with zero mean and covariance matrix C_ϕ .

Each layer $\ell = 1, \dots, L$ is statistically independent, thus, the layers' covariance matrix $C_\phi = \text{diag}(C_1, \dots, C_L)$. Based on the Karman turbulence model [2] these sub-matrices are given by

$$C_\ell = \mathcal{F}^{-1} \phi_\ell \mathcal{F}, \text{ for } \ell = 1, \dots, L. \quad (7.10)$$

The operator \mathcal{F} is the Fourier transform and ϕ_ℓ is the spectral density of the turbulent layer given by

$$\phi_\ell(\kappa) := \frac{0.023 r_0^{-5/3} C_n^2(h_\ell)}{4\pi(|\kappa|^2 + |\kappa_0|^2)^{11/6}}, \kappa_0 < |\kappa| < 2\pi l_0^{-1}, \kappa_0 = 2\pi L_0^{-1}. \quad (7.11)$$

7.2.1.4. Deformable mirror

A deformable mirror (DM) typically consists of a thin surface to reflect light and a set of actuators that drive the mirror. Within this benchmark case we assume the simple model of a bilinear DM. The shape of a bilinear DM is described using a piecewise continuous bilinear function a .

We define the domain on which the DM operates, also called **actuator grid**, by

$$\Omega := [-D/2, D/2]^2, \quad (7.12)$$

where D is the telescope diameter. Further, we denote by n_a^2 the number of actuators or nodal points of the piecewise bilinear function and assume that they are arranged in a rectangular grid with spacing $d := D/(n_a - 1)$. Due to the circular shape of the telescope, not all of these actuators need to be active.

The **actuator positions** are given by (x_i, x_j) for $0 \leq i, j \leq n_a$, where

$$x_i := -D/2 + i \cdot d. \quad (7.13)$$

In relation to this, we define the square sub-domains of Ω by

$$\Omega_{ij} := [x_i, x_{i+1}] \times [x_j, x_{j+1}]. \quad (7.14)$$



To each subdomain we associate a bilinear function defined on $[0, 1]^2$

$$b_{ij}(x, y) = a_{ij}(1 - x - y + xy) + a_{i,j+1}(x - xy) + a_{i+1,j}(y - xy) + a_{i+1,j+1}xy, \quad (7.15)$$

where the values a_{ij} are called **actuator commands**.

In the fitting step, mirror shapes are fit to the reconstructed atmosphere. This is different for each AO system.

For an SCAO system, the reconstructed layer is located at the altitude of the DM, hence, the grid points of the reconstructed layer are aligned with the mirror nodal values and nothing has to be done.

For a LTAO system, the mirror is optimized towards a certain direction of interest θ_1 . Thus, the fitting step is defined by projecting through the reconstructed layers towards θ_1

$$a_1 = [P_{\theta_1,1}^{NGS} \cdots P_{\theta_1,L}^{NGS}] \begin{pmatrix} \phi_1 \\ \vdots \\ \phi_L \end{pmatrix}, \quad (7.16)$$

where $P_{\theta_1,\ell}^{NGS}$ is a bilinear interpolation on layer $\ell = 1, \dots, L$ towards the direction θ_1 .

The difference to MOAO is that we are optimizing towards M directions of interest $\theta_1, \dots, \theta_M$, instead of only one, leading to

$$\begin{pmatrix} a_1 \\ \vdots \\ a_M \end{pmatrix} = \begin{pmatrix} P_{\theta_1,1}^{NGS} & \cdots & P_{\theta_1,L}^{NGS} \\ \vdots & & \vdots \\ P_{\theta_M,1}^{NGS} & \cdots & P_{\theta_M,L}^{NGS} \end{pmatrix} \begin{pmatrix} \phi_1 \\ \vdots \\ \phi_L \end{pmatrix}. \quad (7.17)$$

For a MCAO system, the fitting operator is more complex, since M mirrors are aligned at various altitudes to obtain a good correction over a wide field of view. For the sake of simplicity we omit this case here.

7.2.1.5. Wavefront sensor

A wavefront sensor (WFS) measures the wavefront aberrations indirectly. The most common WFS is called Shack-Hartmann WFS [3], which utilizes an array of little lenses, each focused on a CCD detector plane. The vertical and horizontal shifts of the focal points determine the average slope of the wavefront over the area of the lens, known as subaperture. Similar to the actuator grid in (7.12) we define the subaperture grid for n_s^2 subapertures by

$$\Omega := [-D/2, D/2]^2, \quad (7.18)$$

and the points with equidistant spacing inside the grid by

$$(x_i, x_j) : 0 \leq i, j \leq n_s, \text{ where } x_i := -D/2 + i \cdot d. \quad (7.19)$$

A subaperture is then defined as an open square sub-domain of Ω

$$\Omega_{ij} := (x_i, x_{i+1}) \times (x_j, x_{j+1}). \quad (7.20)$$



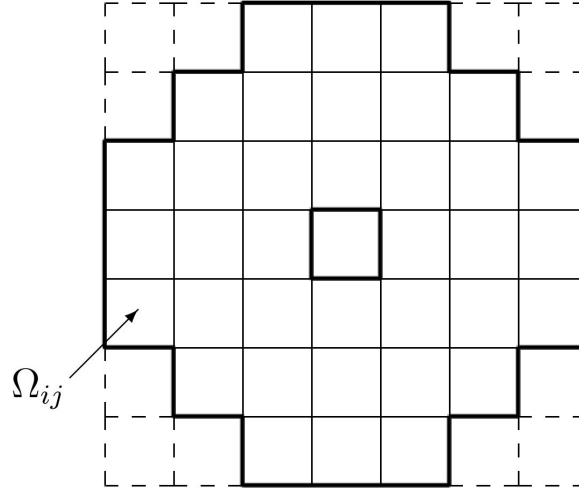


Figure 7.4: Active subapertures

The Shack-Hartmann measurement vector is defined by $s := (s^x, s^y)$. The vectors s^x and s^y are a concatenation of values s_{ij}^x and s_{ij}^y for (i, j) a set of indices that belongs to an active subaperture Ω_{ij} . The subapertures where no measurements are available are excluded from s . To the above defined relation between measurements s and wavefront aberrations φ we associate a Shack-Hartmann WFS operator which we denote by $\Gamma = (\Gamma_x, \Gamma_y)$, where Γ_x and Γ_y determine the slopes in x- and y-direction, respectively

$$s = \begin{pmatrix} s^x \\ s^y \end{pmatrix} = \begin{pmatrix} \Gamma_x \varphi \\ \Gamma_y \varphi \end{pmatrix} = \Gamma \varphi. \quad (7.21)$$

The incoming wavefront aberration is approximated by a continuous piecewise bilinear function φ with nodal values φ_{ij} at points defined by Equation (7.19)

$$s_{ij}^x \simeq \frac{(\varphi_{i,j+1} - \varphi_{i,j}) + (\varphi_{i+1,j+1} - \varphi_{i+1,j})}{2}, \quad (7.22)$$

$$s_{ij}^y \simeq \frac{(\varphi_{i+1,j} - \varphi_{i,j}) + (\varphi_{i+1,j+1} - \varphi_{i,j+1})}{2}. \quad (7.23)$$

7.2.2. Problem formulation - atmospheric tomography

Atmospheric Tomography is the fundamental problem in many AO systems used in the new generation of extremely large telescopes. Assuming a layered model of the atmosphere, the goal of the atmospheric tomography problem is to reconstruct the turbulent layers from the wavefront sensor measurements.



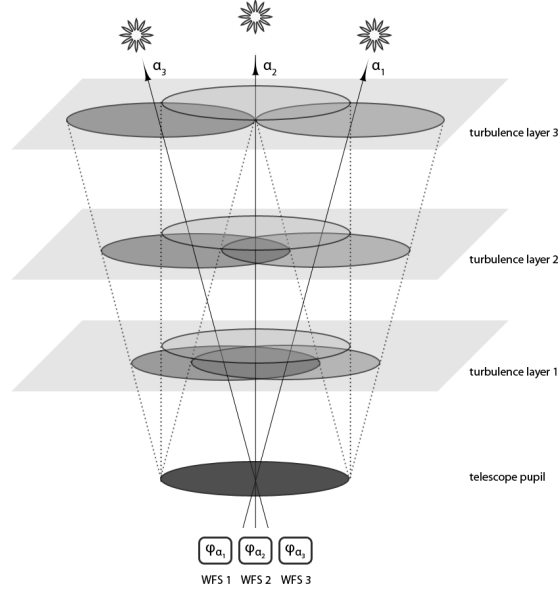


Figure 7.5: Atmospheric Tomography

The atmospheric tomography problem is defined by

$$s = (s_g)_{g=1}^G = A\phi, \quad (7.24)$$

where $\phi = (\phi_1, \dots, \phi_L)$ denote the L turbulent layers, s the sensor measurements and A is the tomographic operator. This operator is a concatenation of a Shack-Hartmann operator Γ , as described in Equations (7.21) and a geometric propagation operator P , defined by (7.1) and (7.4), in the direction of the guide star. This leads to the following equivalent formulation of Equation (7.24)

$$s_g = \Gamma_g P_g \phi \text{ for } g = 1, \dots, G. \quad (7.25)$$

A common way of dealing with the problem of atmospheric tomography is the Bayesian framework [4]. The advantage here is that it allows to incorporate the statistics of turbulence and noise. Within this framework we consider \mathbf{S} and ϕ to be random variables corresponding to the vectors of measurements and turbulence layers, respectively. Further, we assume the presence of noise and model that via a noise random variable η . Altogether, leading to a re-formulation of Equation (7.24)

$$\mathbf{S} = A\phi + \eta. \quad (7.26)$$

The optimal solution of Equation (7.26) is given by the maximum a-posteriori estimate (MAP), which is obtained by solving the linear system of equations

$$(A^T C_\eta^{-1} A + C_\phi^{-1})\phi = A^T C_\eta^{-1} s, \quad (7.27)$$

where C_ϕ^{-1} and C_η^{-1} are the inverse covariance matrices of layers ϕ and noise η .

This problem is ill-posed, due to the small angle of separation. The size of the matrix A depends on the number of subapertures, which is in general higher for bigger telescopes and has increased in the last years. Moreover, the solution has to be computed in real-time. Altogether, efficient solution methods are of great interest for such problems. The standard way of solving this equation, however, not the most efficient one, is called Matrix Vector Multiplication (MVM). This method will serve as benchmark method throughout this document and is described in the following subsection.



7.2.3. Solution method - Matrix Vector Multiplication

The standard approach to solve Equation (7.27) is called Matrix Vector Multiplication (MVM) [5], where the inverse of the discretized left-hand side matrix is computed explicitly by

$$R := (A^T C_\eta^{-1} A + C_\phi^{-1})^{-1} A^T C_\eta^{-1}. \quad (7.28)$$

and then multiplied with the sensor measurements. Typically, a mirror fitting operator F (as defined in Section 7.2.1.4) is combined with the atmospheric reconstruction, mapping sensor measurements onto mirror shapes

$$a = (FR)s. \quad (7.29)$$

The calculation of FR is often referred to as soft real-time, since the re-computation has to be done whenever the noise level, which changes the entries of C_η , or the turbulence parameters, that effect C_ϕ , change. In contrast, the multiplication with the vector of sensor measurements s , which is done at approximately 500 - 1000 Hz, is called hard real-time.

The algorithm described above can be summarized as follows:

1. Compute the tomographic operator A as a concatenation of
 - The Shack-Hartmann operator Γ which is given by equations (7.21), (7.22) and (7.23).
 - The geometric propagation operator P which is defined by (7.4) for LGS and (7.1) for NGS.
2. Set up the inverse covariance matrix of noise C_η^{-1} by using Equation (7.2) and Equation (7.9) for NGS and LGS, respectively.
3. Set up the inverse covariance matrix of layers C_ϕ^{-1} by Equation (7.10) and Equation (7.11).
4. Calculate the reconstruction operator

$$R := (A^T C_\eta^{-1} A + C_\phi^{-1})^{-1} A^T C_\eta^{-1}. \quad (7.30)$$

Use Cholesky Decomposition for inverting the matrix $(A^T C_\eta^{-1} A + C_\phi^{-1})$.

5. Set up the fitting operator F depending on the operating mode of the telescope (see Section 7.2.1.4).
6. Multiply R by the fitting operator F to obtain the control matrix (FR) .
7. Multiply the vector of sensor measurement s by the control matrix to obtain the mirror commands

$$a = (FR)s. \quad (7.31)$$

7.3. Implementation

We implemented the benchmark algorithm in C++ using common libraries for matrix and vector operations, Cholesky decomposition and Fourier transformation. If you want to set-up an environment for C++, you just need to have a text editor to write your program and a C++ compiler to compile your source code into the final executable program. However, we highly recommend to use an integrated development environment (IDE) for C++, as Visual Studio, Eclipse or CLion, instead to profit from an easier way to debug and re-factor your code.

We simply followed the algorithm described in Section 7.2.3 step by step to implement the MVM in C++. First, we set up the required matrices A , C_η^{-1} , C_ϕ^{-1} and F as described in the introduction. Afterwards, we computed the FR matrix and, in a last step, we multiplied these matrix by the vector of sensor measurements s .

The Basic Linear Algebra Subprograms (BLAS) library provides routines for performing basic vector and matrix operations. The Linear Algebra Package (LAPACK) is a C++ library that provides routines for solving systems of linear equations, least-squares solutions of linear systems of equations, eigenvalue problems



and singular value problems. Moreover, matrix factorizations, such as LU or Cholesky decomposition, are provided. BLAS and LAPACK are free libraries that can be downloaded on the following website: <http://www.netlib.org/lapack/> and <http://www.netlib.org/lapack/>. For the benchmark case we use BLAS for matrix- and vector operations and LAPACK to perform Cholesky decomposition for inverting the matrix in Step 4 of the MVM.

Fastest Fourier Transform in the West (FFTW) is a C library for computing the discrete Fourier transform in one or more dimensions. It is a free software and can be downloaded on the FFTW website (<http://www.fftw.org/download.html>). Within the benchmark case we use this library to perform the Fourier transform and inverse Fourier transform when computing the layers covariance matrix C_ϕ with Equation (7.10).

7.4. Computer requirements

One of the most common C++ compiler is called GNU Compiler Collection (GCC) and can be simply downloaded and installed from the GCC website (<https://gcc.gnu.org/>). GCC is just our recommendation, you can use any C++ compiler you prefer.

Beside GCC, we use CMake, which manages the build process in an operating system in a compiler-independent manner. A simple configuration file called *CMakeLists.txt*, placed inside the source directory, is used to generate standard build files (e.g. Makefiles). The most basic *CMakeLists.txt* without using any libraries in the code and any further subdirectories looks as follows

```
cmake_minimum_required (VERSION 2.6)
project (MVM)
SET (CMAKE_C_COMPILER gcc)
SET (CMAKE_CXX_COMPILER g++)
add_executable (MVM mvm.cpp)
```

Including the libraries required for the benchmark case the *CMakeLists.txt* changes to

```
cmake_minimum_required (VERSION 2.6)
project (MVM)
find_package (BLAS)
find_package (LAPACK)
if (LAPACK_FOUND AND BLAS_FOUND)
set (lapackblas_libraries ${BLAS_LIBRARIES} ${LAPACK_LIBRARIES})
endif ()
add_executable (MVM mvm.cpp)
target_link_libraries (MVM ${BLAS_LIBRARIES} ${LAPACK_LIBRARIES} fftw3)
```

7.5. Numerical example

The numerical example we consider within our benchmark case uses LTAO (see 7.2.1.2 for details) for performing atmospheric tomography. Utilizing the input parameters, which are listed in the following subsection, we can use the algorithm described in Section 7.2.3 to deal with the problem of atmospheric tomography and, finally, obtain as output the actuator commands to control the deformable mirror.

7.5.1. Input parameters

To obtain the actuator and subaperture mask I_{act} and I_{sub} , respectively, we can use the provided data files *I_act.txt* and *I_sub.txt*. These two files contain 0 at positions where the actuator or subaperture is inactive and 1 for active actuators or subapertures. Both matrices are stored as an 1D-array inside the data files. The relation



between an index k in the 1d-array and entries (i, j) of the corresponding $n \times n$ matrix is given by

$$k = i \cdot n + j.$$

Operating mode	LTAO
Telescope diameter D	42 m
Type of WFS	Shack-Hartmann
Number of WFS	9
Number of layers L	9
Layer heights h_ℓ	[0, 140, 281, 562, 1125, 2250, 4500, 9000, 18000]
Layer strength c_n^2	[0.5224, 0.0260, 0.0444, 0.1160, 0.0989, 0.0295, 0.0598, 0.0430, 0.0600]
Discretization spacing on layer δ_ℓ	[0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 1, 1]
Number of subapertures n_s	84 x 84 = 7056
Number of actuators n_a	85 x 85 = 7225
Number of photons $n_{photons}$	100
Number of LGS G_{LGS}	6
LGS positions	(3.75, 0), (3.75/2, 3.75·√3/2), (-3.75/2, 3.75 ·√3/2), (-3.75, 0), (-3.75/2, -3.75·√3/2), (3.75/2, -3.75 ·√3/2)
LGS wavelength λ_{LGS}	589 nm
LGS FWHM	11.4 km
LGS height H	90 km
Laser launch positions (x_i^{LL}, x_j^{LL})	(16.26, -16.26), (16.26, 16.26), (-16.26, 16.26), (-16.26, 16.26), (-16.26, -16.26), (16.26, -16.26)
Number of NGS G_{NGS}	3
NGS positions	(-5, 0), (5/2, 5·√3/2), (5/2, -5 ·√3/2)
NGS wavelength λ_{NGS}	500 nm
FWHM of non-elongated spot f	1.1
Outer scale L_0	25 m
Fine-tuning parameter α_η	0.4
Fried parameter r_0	0.129
Number of measurements n_{meas}	84 · 84 · 2 · 9 = 127008
Sensor measurements s	[1, ..., 1] ∈ ℝ ¹²⁷⁰⁰⁸

If for a parameter in the table above no unit is specified, the SI-Unit is meant.

Based on these input parameters we can start with the algorithm described in Section 7.2.3. First, we set up the required matrices A , C_ϕ and C_η . Then we use the libraries described in Section 7.3 to perform matrix and vector operations and Cholesky Decomposition.

7.5.2. Output - DM commands

The output of the MVM algorithm are the mirror commands, with whom the deformable mirror can be adjusted such that atmospheric distortions are corrected. For our specific benchmark case the resulting DM commands are stored in an array of size 85 × 85, thus, we omitted to put the output inside this document and provided a data file *output.txt* where all DM commands are listed.

Bibliography

- [1] F. Roddier, “Adaptive optics in astronomy,” Cambridge, U.K. ; New York: Cambridge University Press, 1999.



- [2] T. von Karman, “Mechanische Ähnlichkeit und Turbulenz,” International Congress of Applied Mechanics, 1930.
- [3] J. Primot, “Theoretical description of Shack–Hartmann wave-front sensor,” *Optics Communications* 222(1):81–92, 2003.
- [4] T. Fusco, J.-M. Conan, G. Rousset, L. Mugnier, and V. Michau, “Optimal wavefront reconstruction strategies for multi conjugate adaptive optics,” *J. Opt. Soc. Am. A* 18(10):2527–2538, 2001.
- [5] I. Foppiano, E. Diolaiti, and P. Ciliegi, “Maory adaptive optics real-time computer user requirements,” *Tech. Rep.*, 2018.



8. Acceleration of Sinkhorn Algorithm using ϵ scaling with applications to the Reflector Problem

Jean-David Benamou¹, Guillaume Chazareix¹, Wilbert Ijzerman², Giorgi Rukhaia¹

¹*Institut National de Recherche en Informatique et en Automatique*

²*Signify*

Abstract. FreeForm Optics is the branch of Optics concerned with the design of non-conventional asymmetric refractive and reflective optical elements or systems of such elements. This research is important to improve the energy efficiency of lighting devices and reduce light pollution (for example of street lighting). A classic application of FreeForm Optics (amongst many) is the irradiance tailoring problem: design an optical system transferring a given light source emittance (e.g a car headlight bulb) to a prescribed irregular target irradiance (e.g. the angular far-field distribution of projected light). At the industrial level, FreeForm Optics design has remained so far largely heuristic.

On the academic side, two classes (collimated or point source illuminance) of idealized tailoring irradiance problems can be exactly modeled and solved using Optimal Transport theory. Optimal Transport defines a unique map or a coupling between prescribed distributions representing given illuminance and irradiance. This map can then be used to construct the optical element shape. Recent advances in Optimal Transport numerical solvers allow tackling systems described by millions of degrees of freedom. This offers a sound mathematical and numerical background to FreeForm Optics.

There are several different approaches for finding numerical solutions of Optimal Transport problems, varying in efficiency, accuracy, and complexity. This work concentrates on the Sinkhorn algorithm. The main advantages of the Sinkhorn algorithm are its simple structure of implementation, involving only simple basic linear algebra operations, and its fastness both from the mathematical foundation and from a wide selection of fast linear algebra libraries. Also, this algorithm can be drastically speeded up using model hierarchy techniques such as discretization and regularization parameter scaling.

Keywords: Reflector Problem, FreeForm Optics, Optimal Transport, Entropic Regularization, Sinkhorn Algorithm.

8.1. Introduction

A light source, also called “illuminance”, is sufficiently small compared to the reflecting surface so that it can be regarded as a point in space. It can therefore be modelled as a probability distribution on the sphere, it will be denoted μ in this paper. The light hits a perfect mirror and we are also given a desired target light distribution, the “illumination” in the far field. From the far field the reflecting surface can be regarded as a point and the illumination again modelled as a probability distribution, denoted ν , on the sphere. Total light conservation is assumed. The reflector problem is to determine the shape of the mirror which produces the specular reflection from the source to the target distribution. This can be interpreted as the inverse problem of generating some illumination given an illuminance and a reflector (see figure 8.1).

8.1.1. Optimal Transport model

This problem has an elegant mathematical modelization and solution based on the optimal transportation (OT) theory due to [1] and [2]. We briefly recall the main result as presented in [2]. In its Kantorovich primal and dual form (see [3]) :

Theorem 4 (Kantorovich duality). *Given two compact manifold X and Y endowed with a continuous, bounded*



htdp

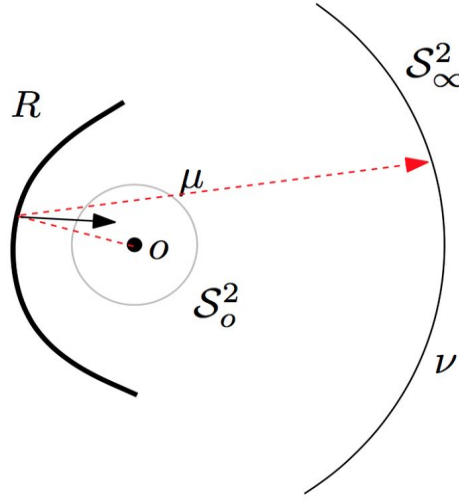


Figure 8.1: Reflector problem from Point source O to Far Field.

from below cost function $c : X \times X \rightarrow \mathbf{R}$ and two borel probability measures $(\mu, \nu) \in \mathcal{P}(X) \times \mathcal{P}(Y)$. Then, Kantorovich problem in primal and dual forms (8.1) has solutions.

$$OT(\mu, \nu) := \min_{\gamma \in \Pi(\mu, \nu)} \langle c, \gamma \rangle_{X \times Y} = \max_{f, g \in C} \langle f, \mu \rangle_X + \langle g, \nu \rangle_Y \quad (8.1)$$

with respectively primal :

$$\Pi(\mu, \nu) := \{\gamma \in \mathcal{P}(X \times Y), \langle 1_X, \gamma \rangle_Y = \nu \langle 1_Y, \gamma \rangle_X = \mu\},$$

and dual :

$$C = \{(f, g) \in \mathcal{C}(X) \times \mathcal{C}(Y), f \oplus g \leq c\},$$

constraints sets

The notation $\langle f, \alpha \rangle_\Omega$ stands for the duality product $\int_\Omega f d\alpha$ between bounded continuous functions $f \in \mathcal{C}(\Omega)$ and probability measures $\alpha \in \mathcal{P}(\Omega)$, $\{f \oplus g\}(x, y) = f(x) + g(y)$ is the direct sum and $\mu \otimes \nu \in \mathcal{P}(X \times Y)$ the tensor product. Finally 1_Ω is the characteristic function, i.e. a constant 1 on Ω .

Under suitable hypothesis on c (as they are technical and satisfied for the costs in this paper, we skip this part), the OT problem is well posed and the the optimal transference plan γ is concentrated on a graph of the OT map $y = T(x)$ implicitly defined by the saturation of the dual constraint :

$$f(x) + g(T(x)) = c(x, T(x)), \quad \mu \text{ a.e.} \quad (8.2)$$

The pair (f, g) are called the Kantorovich potentials and is unique up to an additive constant .

By construction T is a measure preserving map characterizing the transport. The measure preserving property is usually denoted $\nu = T\#\mu$ (T pushes forward μ to ν). The pushforward of μ is the measure defined as

$$\nu(A) = T\#\mu(A) = \mu(T^{-1}(A)) \text{ for all } \nu \text{ measurable subset } A \quad (8.3)$$

Remark 1 (L^p Wasserstein metric). For complete separable metric space X and L^p costs $c := 1/p d^p(x, y)$,



this OT problems defines a separable metric on the set of probability measures with finite second moments: the “Wasserstein” distance, which is given by $W_p^p(\mu, \nu) := OT(\mu, \nu)$.

This metric metrizes weak convergence of measures is a fundamental tool in image processing (see [4]).

In [2], Wang shows that the point source reflector model can be translated to an OT problem. More precisely, he proved the following theorem :

Theorem 5. *Let $S_0 \in \mathbb{S}^{d-1}$ and $S_\infty \in \mathbb{S}^{d-1}$ be connected domains in northern and southern hemispheres respectively, μ and ν which represent the given illuminance and illumination probability distributions. Then theorem 4 applies to the cost function*

$$c(x, y) = -\log(1 - x \cdot y). \quad (8.4)$$

A transport map T satisfying (8.2) exists and the solution of the corresponding OT problem can be used to build the desired reflector.

The construction of the reflector can be summarized as follows : Taking the exponential of the dual constraints and the saturation property (8.2) we get

$$\frac{e^{-g(T(x))}}{1 - x \cdot T(x)} = e^{f(x)} \leq \frac{e^{-g(y)}}{1 - x \cdot y}, \quad \mu \otimes \nu \text{ a.e.} \quad (8.5)$$

We now define in \mathbb{R}^d a family of parabolic reflectors with axis $y \in S_\infty : x \in S_0 \rightarrow P_y(x) := \frac{e^{-g(y)}}{1 - x \cdot y}$. And directly infer that the reflector shape parameterized over the directions in S_0 and given as :

$$R = \{xe^{f(x)} | x \in S_0\}. \quad (8.6)$$

Under this choice the map $x \rightarrow T(x)$ can be interpreted as the specular reflection of an optical ray at $R(x)$ onto a parabola of axis $T(x)$ while the illumination and illuminance constraints are enforced by (8.3).

8.1.2. Entropic Regularization of Optimal Transport

Entropic regularization has been introduced for OT computations in [5] (see [4] for a comprehensive review). The entropic regularization of the Kantorovich problem (4) is based on the following KullBack-Leibler divergence or “relative entropy” (KL) penalization :

$$OT_\epsilon(\mu, \nu) := \min_{\gamma_\epsilon \in \Pi(\mu, \nu)} \langle c, \gamma_\epsilon \rangle_{X \times Y} + \epsilon \text{KL}(\gamma_\epsilon | \mu \otimes \nu) = \max_{f_\epsilon, g_\epsilon} \langle f_\epsilon, \mu \rangle_X + \langle g_\epsilon, \nu \rangle_Y - \epsilon \langle \exp(\frac{1}{\epsilon}(f_\epsilon \oplus g_\epsilon - c)) - 1, \mu \otimes \nu \rangle_{X \times Y} \quad (8.7)$$

where $\epsilon > 0$ is a small “temperature” parameter (see [6] for a Statistical Physics interpretation of this problem due to Schroedinger) and

$$\text{KL}(\gamma | \mu \otimes \nu) := \int_{X \times Y} \log\left(\frac{d\gamma}{d\mu \otimes d\nu}\right) d\gamma \text{ if } \gamma \text{ is absolutely continuous w.r.t to } \mu \otimes \nu \text{ and } +\infty \text{ else.}$$

The primal-dual optimality condition is given by

$$\gamma_\epsilon = \exp\left(\frac{1}{\epsilon}(f_\epsilon \oplus g_\epsilon - c)\right) \mu \otimes \nu. \quad (8.8)$$

The optimal entropic plan is therefore the scaling by the Kantorovich potentials of a fixed Kernel $\exp(-\frac{1}{\epsilon}c)$.



Remark 2. Of course, altering the desired target functional (4) results in altered solution and thereof γ_ϵ is not the exact transport plan that we are looking for. It is diffuse, i.e. not concentrated on a map, and ϵ can be interpreted as a bandwidth under which the transport is blurred.

Although, this entropic plan γ_ϵ converges to γ , the minimizer of (4), when ϵ goes to 0.(see [4])

8.1.3. Sinkhorn Algorithm for Regularized Optimal Transport

Numerical solutions are produced using the discretization of this problem, i.e. replacing (X, Y, c, μ, ν) by $(X_N, Y_N, c_N, \mu_N, \nu_N)$ in the following way:

$$\mu_N = \sum_{i=1}^N p_i \delta_{x_i}, \quad \nu_N = \sum_{j=1}^N q_j \delta_{y_j}, \quad \text{where} \quad \sum_{i=1}^N p_i = \sum_{j=1}^N q_j = 1. \quad (8.9)$$

Of course the number of discrete points for μ and ν may differ, we keep N for both to simplify the presentation. This discretisation provides a natural discretization of the OT problem (4). Setting $X_N = \{x_i\}_{i=1..N}$, $Y_N = \{y_j\}_{j=1..N}$, $c_N = \{c(x_i, y_j)\}_{i,j=1..N}$, $q = \{q_j\}_{j=1..N}$ and $p = \{p_i\}_{i=1..N}$. we can again use the $\langle \cdot, \cdot \rangle$ notation :

$$OT_N(p, q) := \min_{\gamma_N \in \Pi(p, q)} \langle c_N, \gamma_N \rangle_{X_N \otimes Y_N} \quad (8.10)$$

where

$$\Pi(p, q) := \left\{ \gamma_N \in \mathbb{R}_+^{N \times N} \mid \langle 1_{X_N}, \gamma_N \rangle_{Y_N} = p, \langle 1_{Y_N}, \gamma_N \rangle_{X_N} = q \right\} \quad (8.11)$$

Similarly, discretization of regularized problem (8.7) gives

$$OT_{\epsilon, N} := \max_{f_\epsilon, g_\epsilon} \langle f_\epsilon, \mu_N \rangle_{X_N} + \langle g_\epsilon, \nu_N \rangle_{Y_N} - \epsilon \langle \exp\left(\frac{1}{\epsilon}(f_\epsilon \oplus g_\epsilon - c_N)\right) - 1, \mu_N \otimes \nu_N \rangle_{X_N \times Y_N}. \quad (8.12)$$

where we use the same notation (f_ϵ, g_ϵ) for discrete vectors in \mathbb{R}^N .

We solve (8.12) with Sinkhorn algorithm. It corresponds to a block coordinate $(f_\epsilon$ and $g_\epsilon)$ ascent : Initialize with $g_\epsilon^0 = 0_{Y_N}$ and then iterate (in k) :

$$\begin{aligned} f_\epsilon^{k+1} &= -\epsilon \log(\langle \exp\left(\frac{1}{\epsilon}(g_\epsilon^k - c_N)\right), \nu_N \rangle_{Y_N}) \\ g_\epsilon^{k+1} &= -\epsilon \log(\langle \exp\left(\frac{1}{\epsilon}(f_\epsilon^{k+1} - c_N)\right), \mu_N \rangle_{X_N}) \end{aligned} \quad (8.13)$$

As discussed in [4](Remark 4.13), for sufficiently regular data (for example when exact map T is guaranteed to be smooth) following estimate holds for sufficiently large number of iterations k in (8.13):

$$\sup_{X_N} |f_\epsilon(x) - f_\epsilon^k(x)| = O(1 - \epsilon)^k \quad (8.14)$$

Where f_ϵ is an exact regularized potential of (8.12).

8.1.4. Benchmark Cases

The following benchmark cases are being discussed in this chapter:

- Computing a reflector for the problem where the Source is a uniform distribution with support on a set, which is the inverse stereographic projection of unit square centered at the origin and the desired light distribution is a uniform distribution with support on a set, which is an inverse stereographic projection of circle centered at the origin and Diameter 1.



- Computing a reflector for the problem where the Source is a uniform distribution with support on a set, which is an inverse stereographic projection of unit square centered at the origin and the desired light distribution is a sum of two gauss distributions which are centered respectively at inverse stereographic projection of points $(0.25,-0.25),(0.25,0.25)$.

8.2. Hierarchical approach to Sinkhorn Algorithm

8.2.1. ϵ scaling

As mentioned in remark (2), decreasing ϵ would result in a more accurate solution for (4). On the other hand, estimate (8.14) suggests that smaller ϵ we take, higher number of iterations will be required for Sinkhorn algorithm to converge. Also, taking ϵ too small, would result into numerical overflows due to the exponential terms of order $e^{\frac{1}{\epsilon}}$ in (8.13)

As discussed in [7], problem of numerical stability can be tackled by working with the increments of the potentials rather than full potentials during the iterative steps.

That is, if we look at the updates f_ϵ^{k+1} and g_ϵ^{k+1} in (8.13) as $f_\epsilon^{k+1} = f_\epsilon^k + \hat{f}_\epsilon^{k+1}$ and $g_\epsilon^{k+1} = g_\epsilon^k + \hat{g}_\epsilon^{k+1}$, then by moving previous approximations to the right hand side, we will get the following new iterative scheme for the increments:

$$\begin{aligned}\hat{f}_\epsilon^{k+1} &= -\epsilon \log(\langle \exp(\frac{1}{\epsilon}(g_\epsilon^k + f_\epsilon^k - c_N)), \nu_N \rangle_{Y_N}) \\ f_\epsilon^{k+1} &= f_\epsilon^k + \hat{f}_\epsilon^{k+1} \\ \hat{g}_\epsilon^{k+1} &= -\epsilon \log(\langle \exp(\frac{1}{\epsilon}(f_\epsilon^{k+1} + g_\epsilon^k - c_N)), \mu_N \rangle_{X_N}) \\ g_\epsilon^{k+1} &= g_\epsilon^k + \hat{g}_\epsilon^{k+1}\end{aligned}\tag{8.15}$$

Those iterations will be more stable due to the saturation property of the optimizing potentials (8.2). This property tells us that quantity $f(x_i) + g(y_j) - c(x_i, y_j)$ is zero for exact potentials and optimal pairs (x_i, y_j) while being strictly negative for non-optimal pairs. Thereof, when the iterates f_ϵ^k and g_ϵ^k are close to the true potentials, new updating steps would not cause a numerical overflow.

Although, this approach alone would not help at the first steps of the algorithm, since we have no guarantees that initial approximations would be close to the exact potentials, and for small ϵ we would get an overflow at the first step of the iterations. In order to avoid this, possible approach would be to start with higher values of ϵ and gradually decrease it to the desired final value ϵ_{final} (see [7] [8]).

More formally, one can define a sequence of regularization parameters $\epsilon_k \rightarrow \epsilon_{final}$ and use ϵ_k at k -th iteration in (8.15). A common choice is to start with $\epsilon_0 = 1$. We use a scaling parameter $\lambda \in (0, 1)$ and define $\epsilon_k := \max\{\epsilon_{final}, \lambda^k \epsilon_0\}$.

Remark 3. *It has been empirially established (see [7] and references therein), that above discussed approach of gradually increasing ϵ_k at each iteration, not only provides more numerically stable scheme, but also increases the convergence speed. In other words, a smaller number of iterations is required for achieving a given error threshold with decreasing ϵ_k at each iteration, then while using fixed ϵ_{final} for all iterations.*

8.2.2. Discretization scaling

In [7] (see also [9]), it is discussed that the entropic regularization with ϵ acts as a smoothing filter on the data, which smoothers out any details that are on the finer scale than ϵ . This means that using Sinkhorn iterations with discretizations such that $\min_{i,j} d(x_i, x_j) \ll \epsilon$ does not provide any valuable improvement over working with discretizations that are on the scale of ϵ .

Thereof, it would be more efficient to also use a sequence of discretizations $(X_{N_k}, Y_{N_k}, c_{N_k}, \mu_{N_k}, \nu_{N_k})$ where $N_k = O(\frac{1}{\epsilon_k})^d$ (where d is the dimension of the problem). In order to implement this approach, one would need



to find a way to interpolate approximations $f_\epsilon^k, g_\epsilon^k$ on the discretization $X_{N_{k+1}}, Y_{N_{k+1}}$ respectively, while they are computed on the grids X_{N_k}, Y_{N_k} .

Luckily, Sinkhorn algorithm provides a canonical way of computing such interpolations, even for the full spaces X and Y . If we expand the definition of scalar product in (8.13) and replace $c_N = c_N(x_i, y_j)$ by $c(x, y_j)$ and $c(x_i, y)$ respectively, we obtain following continuous extensions for given approximations $f_{\epsilon_k}^k$ and $g_{\epsilon_k}^k$:

$$\tilde{f}_{\epsilon_k}^k(x) := -\epsilon_k \log\left(\sum_{j=1..N_k} \exp\left(\frac{1}{\epsilon_k}(g_{\epsilon_k}^k(y_j) - c(x, y_j))\right) \nu_{N_k}(y_j)\right), \quad \forall x \in X. \quad (8.16)$$

$$\tilde{g}_{\epsilon_k}^k(y) := -\epsilon_k \log\left(\sum_{i=1..N_k} \exp\left(\frac{1}{\epsilon_k}(f_{\epsilon_k}^k(x_i) - c(x_i, y))\right) \mu_{N_k}(x_i)\right), \quad \forall y \in Y. \quad (8.17)$$

Thereof, at k -th iteration, we can take $k - 1$ -th approximations to be restrictions of $\tilde{f}_\epsilon^{k-1}(x)$ and $\tilde{g}_\epsilon^{k-1}(x)$ on the spaces X_{N_k} and Y_{N_k} respectively.

Putting it all together, we obtain the following iterative procedure in k :

$$\begin{aligned} f_{\epsilon_k}^{k-1} &= \tilde{f}_{\epsilon_{k-1}}^{k-1}|_{X_{N_k}} & g_{\epsilon_k}^{k-1} &= \tilde{g}_{\epsilon_{k-1}}^{k-1}|_{Y_{N_k}} \\ \hat{f}_{\epsilon_k}^k &= -\epsilon_k \log(\langle \exp(\frac{1}{\epsilon_k}(g_{\epsilon_{k-1}}^{k-1} + f_{\epsilon_{k-1}}^{k-1} - c_{N_k})), \nu_{N_k} \rangle_{Y_{N_k}}) \\ f_{\epsilon_k}^k &= f_{\epsilon_k}^{k-1} + \hat{f}_{\epsilon_k}^k \\ \hat{g}_{\epsilon_k}^k &= -\epsilon_k \log(\langle \exp(\frac{1}{\epsilon_k}(f_{\epsilon_k}^k + g_{\epsilon_k}^{k-1} - c_{N_k})), \mu_{N_k} \rangle_{X_{N_k}}) \\ g_{\epsilon_k}^k &= g_{\epsilon_k}^{k-1} + \hat{g}_{\epsilon_k}^k \end{aligned} \quad (8.18)$$

In this setting, taking ϵ_{final} to 0 means also refining the discretization. To the best of our knowledge the joint convergence in N and ϵ has only be studied in [9]:

Theorem 6 (Berman joint convergence - corollary 1.3 [9]). *We assume μ and ν are in $C^{2,\alpha}$ and positive, and that N and ϵ are dependent parameters : $N = (1/\epsilon)^d$ where d is the dimension of the problem. A technical condition on the sequence of discretization $(X_N, Y_N, c_N, \mu_N, \nu_N)$ called “density property” (see remark 5 below) is also necessary. Then there exists a positive constant A_0 such that for any $A > A_0$ the following holds : setting $m_\epsilon = \lceil -A \log(\epsilon)/\epsilon \rceil$ the continuous interpolation provided by $\tilde{f}_\epsilon^{m_\epsilon}$, built using the canonical extension (8.16) from the discrete Sinkhorn iterate at $k = m_\epsilon$, satisfies the estimate*

$$\sup_X |\tilde{f}_\epsilon^{m_\epsilon} - f| \leq -C\epsilon \log(\epsilon) \quad (8.19)$$

for some constant C (depending on A) and f an optimal potential for (4).

Remark 4. *Assumptions of Theorem 6 holds on the sphere for the reflector cost (see section 6.3.3 [9]). However, while estimating the necessary number of iterations m_ϵ , this theorem does not take into account the improved effect on the convergence, coming from the ϵ -scaling.*

Remark 5 (Density property Lemma 3.1 [9]). *For any given open set U intersecting the support X of μ (same for Y and ν)*

$$\liminf_{\epsilon \rightarrow 0} \epsilon \log(\mu_N(U)) = 0$$

For the flat space $X \subset \mathbb{R}^d$, this condition is enough. For curved surfaces, a technical generalization is required. But in both cases, this density property ensures the discretization of X and μ (8.9) is such that, for U the sequence of approximations $\mu_N(U)$ never converges faster to 0 than ϵ (remember that $N = (1/\epsilon)^d$).

For the sphere this can be achieved by either the Quasi Monte-Carlo discretizations that are sampled uniformly with respect to the surface element of the sphere (see [10]), or by adjusting the weights of the discretization points according to the deviation from the surface element (e.g. for the orthogonal grids projected from a plane to the sphere).



8.3. Entropic Bias

8.3.1. Entropic Bias and Sinkhorn Divergences

The rate of convergence, both in estimate (8.14) and in theorem 6, have infinite slope at $\epsilon = 0$. Because of this, it is a known issue, that even with above-discussed modifications, using computationally feasible values of ϵ will leave certain "entropic bias" in the approximate potentials.

This problem is discussed in depth in [11] where it is proposed, that in order to correct the bias, to add "diagonal terms" to correct the entropic cost :

$$S_\epsilon(\mu, \nu) = OT_\epsilon(\mu, \nu) - \frac{1}{2}(OT_\epsilon(\mu, \mu) + OT_\epsilon(\nu, \nu)). \quad (8.20)$$

Quite remarkably, the authors show that this quantity, called Sinkhorn divergence, remains positive and is convex. It also obviously vanishes for $\mu = \nu$ which is not the case for OT_ϵ . Thanks to the symmetry, there is only one dual potential for each of diagonal problems. We denote them $f_{OT_\epsilon}^\mu$ and $f_{OT_\epsilon}^\nu$. They can be computed using the independent Sinkhorn iterations :

$$\begin{aligned} f_{OT_\epsilon}^{\mu, k+1} &= -\epsilon \log(\langle \exp(\frac{1}{\epsilon}(f_{OT_\epsilon}^{\mu, k} - c_N)), \mu \rangle_X) \\ f_{OT_\epsilon}^{\nu, k+1} &= -\epsilon \log(\langle \exp(\frac{1}{\epsilon}(f_{OT_\epsilon}^{\nu, k} - c_N)), \nu \rangle_Y) \end{aligned} \quad (8.21)$$

The μ gradient of S_ϵ , denoted f_{S_ϵ} may be formed by a simple subtraction.

$$f_{S_\epsilon} = f_\epsilon - f_{OT_\epsilon}^\mu \quad (8.22)$$

Numerical simulations of gradient flows in [11] indicate that f_{S_ϵ} is a better approximation of exact potential f . For more comprehensive review of entropic bias and its effect on the reflector problem see [12].

8.4. Implementation

Although theoretically it is more efficient to conduct the iterations on the appropriate discretization at every iteration as in (8.18), in practice, altering memory and memory containers at every iteration is not feasible due to hardware properties.

It is a common knowledge in software engineering, that arranging computations in a way that memory is accessed in a continuous way, so that processor doesn't have to wait for the delivery of necessary memory components, produces better practical computational time even when theoretical count of operations is far larger.

With this in mind, depending on the desired ϵ_{final} and $N_{\epsilon_{final}}$, we define two discretization levels : $N_{small} = O(N_{final})^{\frac{1}{2d}}$ and $N_{large} = O(N_{final})$. Similarly, 2 different intermediate values of ϵ are chosen: $\epsilon_{small} = \frac{1}{2}\epsilon_{final}$, $\epsilon_{large} = \epsilon_{final}$.

The sequence ϵ_k is initialized by $\epsilon_0 = 1$ and for $k > 0$ $\epsilon_k = \left(\epsilon_{k-1}^{-1} + \epsilon_{current}^{-\frac{1}{3}} \right)^{-1}$ where $\epsilon_{current}$ is either ϵ_{small} or ϵ_{large} .



With this choice, outline of hierarchical sinkhorn algorithm for reflector problem would be following:

Data: Source and target distributions μ, ν

Result: Approximations of Kantorovich potentials $f_{\epsilon_{final}}, g_{\epsilon_{final}}$

Initialization: $k = 0, N = N_{small}, \epsilon_{current} = \epsilon_{small}, f^0 \equiv 0, g^0 \equiv 0$;

```

while  $\epsilon_k < \epsilon_{final}$  do
     $\hat{f}_\epsilon^{k+1} = -\epsilon_k \log(\langle \exp(\frac{1}{\epsilon_k}(g_\epsilon^k + f_\epsilon^k - c_N)), \nu_N \rangle_{Y_N})$  ;
     $f_\epsilon^{k+1} = f_\epsilon^k + \hat{f}_\epsilon^{k+1}$  ;
     $\hat{g}_\epsilon^{k+1} = -\epsilon_k \log(\langle \exp(\frac{1}{\epsilon_k}(f_\epsilon^{k+1} + g_\epsilon^k - c_N)), \mu_N \rangle_{X_N})$  ;
     $g_\epsilon^{k+1} = g_\epsilon^k + \hat{g}_\epsilon^{k+1}$  ;
     $k=k+1$ 
    if  $\epsilon_k > \epsilon_{current}$  then
        if  $\epsilon_{current} = \epsilon_{final}$  then
            Stop;
        else
             $N = N_{final}, \epsilon_{current} = \epsilon_{final}$  ;
             $f_\epsilon^k = \tilde{f}_{\epsilon_k}^k |_{X_N}, g_\epsilon^k = \tilde{g}_{\epsilon_k}^k |_{Y_N}$  ;
        end
    end

```

end
 $f_{\epsilon_{final}} = f_{\epsilon_k}^k - f_{OT_\epsilon}^\mu$;
 $g_{\epsilon_{final}} = g_{\epsilon_k}^k - f_{OT_\epsilon}^\nu$;

Here de-biasing terms $f_{OT_\epsilon}^\mu$ and $f_{OT_\epsilon}^\nu$ coming from (8.21) are computed using the diagonalized versions of above algorithm:

Data: Source or target distribution μ and corresponding space X

Result: Approximation of Kantorovich potential $f_{\epsilon_{final}}^\mu$

Initialization: $k = 0, N = N_{small}, \epsilon_{current} = \epsilon_{small}, f^0 \equiv 0$;

```

while  $\epsilon_k < \epsilon_{final}$  do
     $\hat{f}_\epsilon^{k+1,\mu} = -\epsilon_k \log(\langle \exp(\frac{1}{\epsilon_k}(f_\epsilon^{k+1,\mu} + f_\epsilon^{k+1,\mu} - c_N)), \mu_N \rangle_{X_N})$  ;
     $f_\epsilon^{k+1,\mu} = f_\epsilon^{k,\mu} + \frac{1}{2} \hat{f}_\epsilon^{k+1,\mu}$  ;
     $k=k+1$  ;
    if  $\epsilon_k > \epsilon_{current}$  then
        if  $\epsilon_{current} = \epsilon_{final}$  then
            Stop;
        else
             $N = N_{final}, \epsilon_{current} = \epsilon_{final}$  ;
             $f_\epsilon^{k,\mu} = \tilde{f}_{\epsilon_k}^{k,\mu} |_{X_N}$  ;
        end
    end

```

end
 $f_{\epsilon_{final}}^\mu = f_{\epsilon_k}^{k,\mu}$;

8.5. Computer Requirements

The software is written in C++, but most of the computational data structures are C-style fixed-size Arrays and most of the computational work is done either by basic operations available in C as well, or by Intel's Math Kernel Library. No manually written classes are used and C++ standard library functions are used for secondary tasks, such as data filling, data sorting or time counting.



Intel's Math Kernel Library (MKL) is a library of optimized math routines. It includes BLAS, LAPACK, ScaLAPACK, sparse solvers, fast Fourier transforms, and vector math. The routines in MKL are hand-optimized specifically for Intel processors. In this software, only BLAS and vector math functions are used.

The library is available free of charge under the terms of Intel Simplified Software License which allows redistribution. Commercial support is available when purchased as standalone software or as part of Intel Parallel Studio XE or Intel System Studio. It can be downloaded from Intel's official web-page <https://software.intel.com/mkl> where installation instructions are also provided.

Due to high importance of good memory management, and software's primary purpose for now being the development of the method, code doesn't follow standard suggestions for C++ code development.

C++ compiler with version 11 or higher is required, as code uses timing functions and arithmetic of "inf" and "nan" values introduced in this version.

The code is OS independent as long as appropriate compiler and ability to link with MKL library are available, except, for convenience of output handling, system command is called to create and move folders. Right now code uses Linux commands "mkdir" and "mv". For other operating systems one could just change those commands in the main function.

Due to the use of Intel's MKL library, the software will be much more efficient when running on the Intel processor, compared to other processors of the same power. Other than the capability of installing Intel's MKL Library, there is no other definitive hardware requirement, but for computational stability, it is desirable for double-precision floating-point variable to hold numbers up to 16 digit precision, so it is recommended that hardware is capable of handling such precision.

8.6. Numerical Demonstration

Here we demonstrate that, for the benchmark cases, using ϵ -scaling speeds up the computation by decreasing number of iterations required for achieving a given precision in Sinkhorn algorithm. Also, using discretization scaling does not worsen total number of iterations, while achieving speedup by using cheaper iterations on the first stage.

As stopping criteria for the native sinkhorn algorithm (8.15), we use the absolute value of the change \tilde{f}_ϵ^{k+1} in the first potential. In order to achieve a fair comparison with the ϵ -scaling algorithms, we also force ϵ -scaling algorithms to continue with native iterations after reaching final value of regularization parameter, until they achieve same threshold.

Our input are the discretization of the analytical descriptions of the illumination/illuminance μ and ν described below. All benchmark cases presented in this paper will have the same source and target domains X and Y . The source domain $X \subset \mathbb{S}^2$ will be the inverse stereographic projection in the northern hemisphere of the square domain centered at the origin $\{(x_1, x_2) \in \mathbb{R}^2 \mid -0.6 \leq x_1 \leq 0.6, -0.6 \leq x_2 \leq 0.6\}$. Similarly, $Y \subset \mathbb{S}^2$ will be the inverse stereographic projection in the southern hemisphere of same domain.

As discussed in remark 5 (see also [9]), we discretize those domains using Quasi Monte-Carlo discretizations from [10]. We take $N = 16488 \approx 128 * 128$ points in each discretization. For discretization scaling, we use $N_{small} = 381$. We compute each benchmark case with two different values for final ϵ , $\frac{1}{4*128}$ and $\frac{1}{16*128}$. For the second value, native sinkhorn algorithm is not applicable as we get an overflow on the very first iteration.

As according to convergence result from Theorem 6, for given $\epsilon = O(N)^{-\frac{1}{d}}$ we can expect only approximations of order $\epsilon \log(\epsilon)$, we take $\tilde{f}_\epsilon^k < 1.e - 5$ as a stopping criteria, since when changes become smaller, each new iteration adds less improvement, and approximation error becomes dominant.

Benchmark Case 1: Square To Circle. The source distribution μ will be the uniform distribution over the set with a square stereographic projection $StP(supp(\mu)) = \{(x_1, x_2) \in \mathbb{R}^2 \mid -0.5 \leq x_1 \leq 0.5, -0.5 \leq x_2 \leq 0.5\}$. The target distribution ν will be an uniform distribution over the set with a disk (circle) stereographic projection $StP(supp(\mu)) = \{(x_1, x_2) \in \mathbb{R}^2 \mid x_1^2 + x_2^2 \leq 0.5^2\}$. Even though the densities are constant, mapping from a non smooth support geometry of the square to the smooth geometry of a circle is not a trivial task.



ϵ_{final}	Type of Algorithm	Number of iterations
$\frac{1}{4*128}$	Native	146
	ϵ -scaling	116
	Discretization scaling	121
$\frac{1}{16*128}$	Native	NA(Numeric Overflow)
	ϵ -scaling	171
	Discretization scaling	182

Benchmark Case 2: Square To Two Gaussians. The source distribution μ is the same as in the previous. The target distribution ν is a gaussian distribution with density on the projected domain $\rho(x_1, x_2) = e^{-16*((x_1-0.25)^2+(x_2-0.25)^2)} + e^{-16*((x_1-0.25)^2+(x_2+0.25)^2)}$ over whole target domain Y

ϵ_{final}	Type of Algorithm	Number of iterations
$\frac{1}{4*128}$	Native	NA (Slow convergence)
	ϵ -scaling	202
	Discretization scaling	175
$\frac{1}{16*128}$	Native	NA(Numeric overflow)
	ϵ -scaling	502
	Discretization scaling	184

An interesting phenomena occurs for this case. Since the target has very high steepness, convergence speed for native algorithm, even for moderate value of ϵ , is extremely slow. Even after 1000 iterations, absolute value of the incrementing term was of order $1.e - 3$. On the other hand, scaling algorithm managed to converge with comparable number of iterations as in previous case.

Bibliography

- [1] T. Glimm and V. Olikar, "Optical design of single reflector systems and the monge-kantorovich mass transfer problem," *Journal of Mathematical Sciences*, vol. 117, pp. 4096–4108, 09 2003.
- [2] X.-J. Wang, "On the design of a reflector antenna ii," *Calculus of Variations and Partial Differential Equations*, vol. 20, no. 3, pp. 329–341, Jul 2004. [Online]. Available: <https://doi.org/10.1007/s00526-003-0239-4>
- [3] C. Villani, *Optimal Transport: Old and New*, ser. Grundlehren der mathematischen Wissenschaften. Springer Berlin Heidelberg, 2008. [Online]. Available: <https://books.google.fr/books?id=NZXiNAEACAAJ>
- [4] G. Peyré and M. Cuturi, "Computational Optimal Transport," *ArXiv e-prints*, Mar. 2018.
- [5] M. Cuturi, "Sinkhorn distances: Lightspeed computation of optimal transport," in *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2013, pp. 2292–2300. [Online]. Available: <http://papers.nips.cc/paper/4927-sinkhorn-distances-lightspeed-computation-of-optimal-transport.pdf>
- [6] C. Léonard, "A survey of the schrödinger problem and some of its connections with optimal transport," 2013.
- [7] B. Schmitzer, "Stabilized Sparse Scaling Algorithms for Entropy Regularized Transport Problems," *arXiv e-prints*, p. arXiv:1610.06519, Oct 2016.
- [8] A. M. Oberman and Y. Ruan, "An efficient linear programming method for optimal transportation," 2015.
- [9] R. J. Berman, "The Sinkhorn algorithm, parabolic optimal transport and geometric Monge-Ampere equations," *ArXiv e-prints*, Dec. 2017.
- [10] R. S. Womersley, "Efficient Spherical Designs with Good Geometric Properties," *ArXiv e-prints*, Sep. 2017.



- [11] J. Feydy, T. Séjourné, F.-X. Vialard, S.-i. Amari, A. Trouvé, and G. Peyré, “Interpolating between Optimal Transport and MMD using Sinkhorn Divergences,” Oct. 2018, working paper or preprint. [Online]. Available: <https://hal.archives-ouvertes.fr/hal-01898858>
- [12] J.-D. Benamou, W. L. Ijzerman, and G. Rukhaia, “An Entropic Optimal Transport Numerical Approach to the Reflector Problem,” Apr. 2020, working paper or preprint. [Online]. Available: <https://hal.archives-ouvertes.fr/hal-02539799>



9. Data driven model adaptations of coil sensitivities in magnetic particle imaging

Sören Dittmer¹, José Carlos Gutiérrez Pérez¹, Lena Hauberg-Lotte¹, Tobias Kluth¹, Peter Maass¹, Daniel Otero Bager¹

¹University of Bremen

Abstract. In Section 9.1 first basic ideas of magnetic particle imaging (MPI) are introduced, model and data based cases are discussed and different problem scenarios are formulated. In the following the basic ideas behind scenarios of applying Deep Learning to Inverse Problems are presented and five broad categorizations are introduced: 1. Learned Penalty Terms, 2. Plug-and-Play Prior Methods, 3. Gradient-Descent-by-Gradient-Descent type Methods, 4. Regularization by Architecture (Deep Prior), 5. Image Post-Processing via Deep Learning. The application of Deep Learning to MPI is described with two approaches, surrogate data sets and regularization via architecture. In Section 9.2 the implementation as well as the underlying network is explained in detail. The computer requirements are stated (Section 9.3) and numerical examples are given by using the publically available data sets generated by the Bruker preclinical MPI system at the Medical Center Hamburg-Eppendorf (Section 9.4).

Keywords: Deep learning, neural networks, inverse problems, deep image priors.

9.1. Introduction and literature

The content of this section is a slightly modified version of the corresponding sections in the description of 'Benchmark Case 4' described in the [first report](#).

Magnetic particle imaging (MPI) is a relatively new non-invasive tomographic imaging technique that directly detects superparamagnetic iron oxide nanoparticles (SPIO). It combines high tracer sensitivity with submillimetre resolution and imaging is performed in milliseconds to seconds.

MPI is suitable for several medical applications: The nonlinear magnetization behaviour of nanoparticles in an applied magnetic field is employed to reconstruct a spatial distribution of the concentration of nanoparticles in the cardiovascular system. A high temporal resolution and a potentially high spatial resolution make MPI suitable for several *in-vivo* applications without the need for harmful radiation. The potential for imaging blood flow was demonstrated first in *in-vivo* experiments using a healthy mouse [1]. The usability of a circulating tracer for long-term monitoring was recently investigated [2]. The high temporal resolution of MPI is advantageous for potential flow estimation [3] and for tracking medical instruments [4]. Recently, MPI was also shown to be suitable for tracking and guiding instruments for angioplasty [5]. Further promising applications of MPI include cancer detection [6] and cancer treatment by hyperthermia [7].

The main goal of MPI is to reconstruct the spatially dependent concentration of particles and for that computationally efficient reconstruction methods are required to allow real time observations. Therefore mathematical models are advantageous for the development of such new methods.

Inverse Problems have been a key tool in many areas of science, technology in general, and more particularly in the field of medical imaging for many years now. The key idea is to calculate causes from effects, opposed to so-called direct problems trying to predict effects from causes.

Deep Learning (DL) on the other hand is a relatively new field which studies big machine learning models.



It is not clear what all the strength of DL are, but a major one is the ability to predict labels from data via supervised learning, e.g., a label of a computer tomographic (CT) image could be cancer or no cancer. A major problem with this is the fact that one usually needs massive amounts of labeled data to train a learning model.

The success of neural networks (NN) in many computer vision tasks in the past has motivated attempts at using Deep Learning to achieve better performance in solving Inverse Problems [8]. It was proposed to apply data-driven approaches to inverse problems, using neural networks as a regularization functional. The network learns to discriminate between the distribution of ground truth images and the distribution of unregularized reconstructions and the approach was used for computer tomography reconstruction [9]. Furthermore, deep learning was used in medical applications such as tumour classification with MALDI Imaging [10], magnetic resonance imaging (MRI) [11] [12], low-dose X-ray CT [13] as well as positron emission tomography (PET) [14].

The need for massive amounts of data is also a major challenge in bringing Deep Learning and Inverse Problems together, since it creates a chicken-and-egg-problem. The problem lies therein that one can think about the "cause" in Inverse Problems as (or at least connected to) the label in Deep Learning. This means that one usually doesn't have labels for Inverse Problems which are required to train a deep model to solve the Inverse Problem. Most approaches that try to bring Deep Learning to Inverse Problems ignore this fact and only focus on problems where there is enough ground truth data for the cause already – due to some special circumstances. In fact they can simplify the practical application of already solved Inverse Problems massively, but they can usually not be applied to novel unsolved Inverse Problems. In summary: there exists some kind of "information gap" that creates a boot strap problem. We are planning on exploring ways to solve this problem, in particular for MPI.

9.1.1. Magnetic Particle Imaging

To determine the distribution of nanoparticles, which is the quantity $c(x)$, the nonlinear magnetization behavior of ferromagnetic nanoparticles is exploited as follows, see also [15]: A static magnetic field (selection field), which is given by a gradient field, generates a field free point (FFP) (or alternatively a field free line (FFL) [16]). The larger the distance between nanoparticles and FFP, the more is the magnetization caused by the nanoparticles in saturation. The superposition with a spatially homogeneous but time-dependent field (drive field) moves the field free region along a predefined trajectory defining the field-of-view (FOV). An interplay between gradient strength and drive field amplitude determines the FOV size but when guaranteeing a certain resolution the FOV is strictly limited due to safety reasons. The rapid change of the applied field $\mathbf{H}(x, t)$ causes a measurable change of the magnetization $\mathbf{M}(x, t)$ of the nanoparticles.

In the first approximation the change of the magnetization can be characterized by using the Langevin function. Neglecting the interactions between multiple particles and doing the transition from microscopic to macroscopic scale (see [17]) allows the approximation of the magnetization \mathbf{M} by multiplying the particles' mean magnetic moment vector $\bar{\mathbf{m}}(x, t)$ and the particle concentration $c(x)$.

The temporal change of the particles' magnetization induces a voltage $u^P(t)$ in the receive coil units. Using a quasi-static approximation in the induction principle and the law of reciprocity (see [18]) allows for the description via a linear integral operator with respect to the particle concentration:

$$u^P(t) = -\mu_0 \int_{\Omega} \mathbf{p}^R(x) \cdot \frac{\partial}{\partial t} \mathbf{M}(x, t) dx = \int_{\Omega} c(x) \underbrace{(-\mu_0 \mathbf{p}^R(x) \cdot \frac{\partial}{\partial t} \bar{\mathbf{m}}(x, t))}_{=s(x,t)} dx. \quad (9.1)$$

Here, \mathbf{p}^R is the receive coil sensitivity, which is the magnetic field which is generated by the receive coil unit when applying a unit current. Analogously, the applied field $\mathbf{H}(x, t)$ also induces a voltage $u^E(t)$ which is



known as direct feedthrough. Since this value is several orders of magnitude larger than that of the particle signal, it must be removed prior to digitization. This is done by applying an analog filter which is described by a temporal convolution with a kernel function $a(t)$ (\tilde{a} denotes its periodic continuation). The measured signal $v(t)$ is then given by $v = (u^P + u^E) * a$. One common choice for the analog filter is a band-stop filter such that some frequency bands of the particle signal are also removed. The resulting integral kernel $\tilde{s}(x, t) = (s(x, \cdot) * a)(t)$ is primarily determined by the analog filter (receive-unit-dependent), the receive coil sensitivity (receive-unit-dependent), the behavior of the nanoparticles, the particle parameters, and the applied magnetic fields. Due to missing accurate models for the particle magnetization, the whole function \tilde{s} is commonly determined in a time-consuming calibration process limiting the FOV size as well as the resolution. As the calibration data strictly relies on the particle properties and the measurement sequence, changing tracer material or measurement sequences requires a complete recalibration. Model-based approaches including less simplified behavior of the particles' magnetic moments in the applied fields are highly desirable to reduce the calibration costs and to enable more sophisticated measurement sequences.

9.1.1.1. Scenarios in MPI

The main problem in MPI is given by the forward operator

$$F : X(\Omega) \rightarrow Y(0, T)^L$$

$$c \mapsto (A_k c + a_k * u_k^E)_{k=1, \dots, L},$$

for $L \in \mathbb{N}$ receive coil units with suitable function spaces $X(\Omega)$ and $Y(0, T)$ (assumed to be Hilbert spaces in the following). $A_k : X(\Omega) \rightarrow Y(0, T)$, $c \mapsto \int_{\Omega} \tilde{s}_k(x, t) c(x) dx$, $k = 1, \dots, L$, is the forward operator mapping to the analog filtered particle signal for individual receive coil units. The operator F describes the actual measurement process. In MPI we are mainly aiming for solving problems given by the linear operator

$$A : X(\Omega) \rightarrow Y(0, T)^L$$

$$c \mapsto (A_k c)_{k=1, \dots, L}.$$

We thus need to get rid of the direct feedthrough u_k^E . In an ideal situation it holds $A = F$ as one assumes $a_k * u_k^E = 0$ but in general this not the case. We can now distinguish two cases in MPI regarding a formal description of the forward operator, the *data-based* case where a full calibration of the linear forward operator is performed and the *model-based* case where a suitable model for the mean magnetic moment $\bar{\mathbf{m}}$ is formulated. Due to the fact that finding a suitable model for the particles' magnetization is still an unsolved problem and the full system matrix calibration is still state of the art in MPI, we distinguish these two cases in the following. As it is possible to specify physical models which might not include relevant aspects, hybrid approaches combining best of both are also desirable. In MPI possible problem setups are as follows (to improve reading convenience, we formulate the scenarios for one coil unit only, i.e., $L = 1$):

- *Data-based* case: Let $\Gamma \subset \mathbb{R}^3$ be a reference volume placed at the origin. The data-based approach uses single measurements of a small sample at predefined positions $\{x^{(i)}\}_{i=1, \dots, N} \in \Omega^N$. The concentration phantoms are given by $c^{(i)} = c_0 \chi_{x^{(i)} + \Gamma}$ for some reference concentration $c_0 > 0$. Typical choices for Γ are small cubes ($\sim 1 \text{ mm} \times 1 \text{ mm} \times 1 \text{ mm}$). The measurements $v^{(i)} = \frac{1}{c_0} F c^{(i)}$, $i = 1, \dots, N$, are then used to characterize the data-based forward operator.

Background subtraction case (D1): Let $v^{(0)} = F \mathbf{0}$. Assuming the calibration positions are chosen such that $x^{(i)} + \Gamma$ are pairwise disjoint and $\Omega = \cup_{i=1}^N x^{(i)} + \Gamma$, the specific discretised problem of the model-based approach corresponds to the data-based approach (solving $Ac = v - v^{(0)}$) with $S\tilde{c} = v - v^{(0)}$ with $S = [v^{(1)} - v^{(0)} | \dots | v^{(N)} - v^{(0)}]$ and where $c = \sum_{i=1}^N \tilde{c}_i \chi_{x^{(i)} + \Gamma}$. The full discrete setup is obtained by a finite dimensional approximation in $Y_M \subset Y(0, T)$, i.e., assume $v = \sum_{i=1}^M \langle \phi_i, v \rangle \phi_i$ where $\{\phi_j\}_{j \in \mathbb{N}}$ is an ONB of $Y(0, T)$. It then reads: Find $\tilde{c} \in \mathbb{R}^N$ for given measurement $\tilde{v} \in \mathbb{K}^M$, $\mathbb{K} \in \{\mathbb{R}, \mathbb{C}\}$,



($v \in Y(0, T)$ obtained from an evaluation of F) such that

$$\tilde{S}_{D1}\tilde{c} = \tilde{v} - \tilde{v}^{(0)}, \quad \tilde{S}_{D1} = [\tilde{v}^{(1)} - \tilde{v}^{(0)} | \dots | \tilde{v}^{(N)} - \tilde{v}^{(0)}] \quad (9.2)$$

where $\tilde{v} = (\langle \phi_i, v \rangle)_{i=1, \dots, M}$ ($\tilde{v}^{(k)}$ obtained analogously).

Partial temporal information case (D2): Alternative strategies to remove the influence of the background signal $v^{(0)}$ (due to the structure of $v^{(0)}$ in a certain basis) is to restrict the problem to partial data. The discretisation of $Y(0, T)$ is again obtained via the ONB $\{\phi_i\}_{i=1, \dots, M}$. Let $\{\psi_j\}_{j \in \mathbb{N}}$ be another ONB of $Y(0, T)$ (can also be equal to $\{\phi_i\}_{i=1, \dots, M}$). We further assume $v^{(0)}$ is sparse in $\{\psi_j\}_{j \in \mathbb{N}}$, i.e. $|\langle v^{(0)}, \psi_j \rangle| \neq 0, j \in J, |J| < \infty$. We thus formulate the reduced data problem within the data-based case: It then reads finding $\tilde{c} \in \mathbb{R}^N$ for given measurement $\tilde{v} \in \mathbb{K}^M, \mathbb{K} \in \{\mathbb{R}, \mathbb{C}\}, (v \in Y(0, T)$ obtained from an evaluation of F) such that

$$\langle S\tilde{c}, \psi_j \rangle = \langle v - v^{(0)}, \psi_j \rangle, \quad j \in \mathbb{N} \setminus J, \quad S = [v^{(1)} - v^{(0)} | \dots | v^{(N)} - v^{(0)}] \quad (9.3)$$

Exploiting the sparsity assumption on $v^{(0)}$ and using the finite-dimensional approximation in Y_M yields the final problem

$$\begin{aligned} \langle \tilde{S}_{D2}\tilde{c}, (\langle \phi_i, \psi_j \rangle)_{i=1, \dots, M} \rangle &= \langle \tilde{v}, (\langle \phi_i, \psi_j \rangle)_{i=1, \dots, M} \rangle, \quad j \in \mathbb{N} \setminus J, \\ \tilde{S}_{D2} &= [\tilde{v}^{(1)} | \dots | \tilde{v}^{(N)}] \end{aligned} \quad (9.4)$$

The crucial part is to obtain the correct index set J . In the literature this is commonly done by an SNR-threshold technique with respect to $\{\psi_j\}_{j \in \mathbb{N}}$ being the Fourier basis for T -periodic signals. If the index set is determined incorrectly, one inverts an affine linear system assuming it is linear causing additional artifacts in the reconstruction (i.e., in a noise-free case the reconstruction c^* of a true c^\dagger is obtained via $c^* = A^{-1}Fc^\dagger = c^\dagger + A^{-1}v^{(0)}$).

- *Model-based case:* The challenging part in the model-based case is formulating the correct model for the mean magnetic moment $\bar{\mathbf{m}}$.

Equilibrium model (monodisperse / polydisperse) (M1): One of the most extensively studied models in MPI is based on the Langevin function. This model is motivated by the assumptions that the applied magnetic field is static and the particles are in equilibrium. Under these assumptions, we assume that the mean magnetic moment vector of the nanoparticles immediately follows the magnetic field, i.e.:

$$\bar{\mathbf{m}}(x, t) = m_0 \mathcal{L}_\beta(|\mathbf{H}(x, t)|) \frac{\mathbf{H}(x, t)}{|\mathbf{H}(x, t)|} \quad (9.5)$$

where $\mathcal{L}_\beta : \mathbb{R} \rightarrow \mathbb{R}$ is given in terms of the Langevin function by the following:

$$\mathcal{L}_\beta(z) = \left(\coth(\beta z) - \frac{1}{\beta z} \right) \quad (9.6)$$

for $m_0, \beta > 0$. The final problem with respect to the Langevin function is to obtain the concentration c from the following system of equations:

$$\begin{cases} v(t) = - \int_0^T \int_\Omega c(x) \tilde{a}_k(t-t') s_k(x, t') dx dt' \\ s = \mu_0 m_0 (\mathbf{p}^R)^T \frac{\partial}{\partial t} \left(\mathcal{L}_\beta(|\mathbf{H}|) \frac{\mathbf{H}}{|\mathbf{H}|} \right) \end{cases} \quad (9.7)$$

$I_3 \in \mathbb{R}^{3 \times 3}$ being the identity matrix. The *equilibrium model* in (9.7) can be extended to polydisperse tracers by adapting the function defining the length of the mean magnetic moment vector in (9.6). The



tracer material is then modeled by a distribution of particles with different diameters $D > 0$. Assuming that the particle's diameter distribution is given by the density function $\rho : \mathbb{R}^+ \rightarrow \mathbb{R}^+ \cup \{0\}$ with $\|\rho\|_{L^1(\mathbb{R}^+)} = 1$, we obtain the extended problems by the following:

$$\begin{cases} v(t) = - \int_0^T \int_{\Omega} c(x) \tilde{a}(t-t') s(x, t') dx dt' \\ s = \mu_0 (\mathbf{p}^R)^T \frac{\partial}{\partial t} \left(L_{\rho}(|\mathbf{H}|) \frac{\mathbf{H}}{|\mathbf{H}|} \right) \end{cases} \quad (9.8)$$

where $L_{\rho} : \mathbb{R} \rightarrow \mathbb{R}$ is given in terms of the Langevin function by

$$L_{\rho}(z) = \int_{\mathbb{R}^+} \rho(D) m_0(D) \mathcal{L}_{\beta(D)}(z) dD \quad (9.9)$$

with $m_0, \beta : \mathbb{R}^+ \rightarrow \mathbb{R}^+$ describing the influence of the particle diameter on the volume of the core, respectively the magnetic moment.

Imperfect models suitable for lower quality image reconstruction (M2): This category includes a rather large number of possible model approaches. Generally spoken, this comprises models which approximate the behavior of the system matrix (commonly fitting some model parameters to real data) but cannot reach the reconstruction quality of the data-based case. For example, one can treat the analog filter a as one unknown parameter and fit the model in (M1) to real data.

Suitable model for magnetization dynamics (M3): The operator is properly described by a mathematical model including a sufficient model for the magnetization behavior of the tracer. This requires considering (or approximating) Brownian and Néel rotation mechanisms in the magnetic moment rotation of the nanoparticles. The important difference to (M2) is that here we assume that this model is qualitatively an alternative to the data-based case. This is still an unsolved problem but it is added to the list of possible cases to emphasize the opportunities in the context of Deep Learning approaches.

Using the previously formulated standard setups in MPI we can formulate different problem scenarios (S) which are discussed in the context of Deep Learning in the remainder of this article:

- **Scenario (S1):**

Given information (D1): measured and background-corrected system matrix (\tilde{S}_{D1}), background measurements $\tilde{v}^{(0)}$, a phantom measurement \tilde{v} .

Possible targets: (i) reduce number of calibration scans (larger delta sample and/or missing regions), (ii) obtain fast and improved concentration reconstructions, (iii) obtain memory-efficient representation, (iv) obtain cleaned particle signal

- **Scenario (S2):**

Given information (D2): measured system matrix (\tilde{S}_{D2}), background measurements $\tilde{v}^{(0)}$ (optional), a phantom measurement \tilde{v} .

Possible targets: (i) reduce number of calibration scans (larger delta sample and/or missing regions), (ii) obtain fast and improved concentration reconstructions, (iii) obtain memory-efficient representation, (iv) obtain correct index set J , (iv) obtain completed particle signal

- **Scenario (S3):**

Given information (D1 or D2, M1): measured system matrix (\tilde{S}_{D1} or \tilde{S}_{D2}), background measurements $\tilde{v}^{(0)}$ (in (D2) optional), a phantom measurement \tilde{v} , similar but oversimplified model for $\bar{\mathbf{m}}$ (see M1).

Possible targets: (i) reduce number of calibration scans (larger delta sample and/or missing regions), (ii) obtain fast and improved concentration reconstructions, (iii) obtain memory-efficient representation, (iv) obtain correct index set J (in (D2)), (iv) obtain completed particle signal, (v) measured signal correction (background), completion, and mapping to range of oversimplified model (reconstruction is than obtained via oversimplified model).



- **Scenario (S4):**

Given information (M1): background measurements $\tilde{v}^{(0)}$, a phantom measurement \tilde{v} , similar but oversimplified model for $\bar{\mathbf{m}}$ (see M1).

Possible targets: (i) obtain fast and improved concentration reconstructions, (ii) obtain improved operator, (iii) obtain cleaned particle signal from measured phantom data.

- **Scenario (S5):**

Given information (M2): background measurements $\tilde{v}^{(0)}$, a phantom measurement \tilde{v} , imperfect model for $\bar{\mathbf{m}}$ which allows reasonable reconstruction (see M2).

Possible targets: (i) obtain fast and improved concentration reconstructions, (ii) obtain improved operator, (iii) obtain cleaned particle signal from measured phantom data.

- **Scenario (S6):**

Given information (D1 or D2, M2): measured system matrix (\tilde{S}_{D1} or \tilde{S}_{D2}), background measurements $\tilde{v}^{(0)}$ (in (D2) optional), a phantom measurement \tilde{v} , imperfect model for $\bar{\mathbf{m}}$ which allows reasonable reconstruction (see M2).

Possible targets: (i) reduce number of calibration scans (larger delta sample and/or missing regions), (ii) obtain fast and improved concentration reconstructions (e.g., post-processed reconstruction of lower quality), (iii) obtain memory-efficient representation, (iv) obtain correct index set J (in (D2)), (v) obtain completed particle signal, (vi) measured signal correction (background), completion, and optional mapping to range of imperfect model (reconstruction is than obtained via oversimplified model).

- **Scenario (S7) (hypothetical):**

Given information (M3): background measurements $\tilde{v}^{(0)}$, a phantom measurement \tilde{v} , suitable model for $\bar{\mathbf{m}}$ (see M3).

Possible targets: (i) obtain fast and improved concentration reconstructions, (ii) obtain cleaned particle signal from measured phantom data.

9.1.2. Deep Learning and Inverse Problems

In the following subsections we will present the basic ideas behind scenarios of applying Deep Learning to Inverse Problems and briefly introduce explicit methods to deploy them.

We use the following naming conventions:

- The forward operator:

$$A : X \rightarrow Y,$$

where X (a Banach space) the reconstruction space and Y (a Banach space) the signal space.

- $p_X : X \rightarrow \mathbb{R}_{g_\epsilon 0}$ the probability density function of the elements/images/concentrations that we want to reconstruct.
- $p_Y : Y \rightarrow \mathbb{R}_{g_\epsilon 0}$ the probability density function of the elements/signals/voltages that we are measuring.
- $p_{XY} : X \times Y \rightarrow \mathbb{R}_{g_\epsilon 0}$ the joint probability density function.

9.1.2.1. A: Learned Penalty Terms

In variational approaches to Inverse Problems one usually incorporates a so-called penalty function

$$\phi : X \rightarrow \mathbb{R}_{g_\epsilon 0}$$

to favor good reconstructions, via minimizing a Tikhonov-type functional

$$\min_x \|Ax - y\|_2^2 + \phi(x)$$



(or similar). This penalty function is usually handcrafted and at best a very rough approximation of what one would want to have as a penalty term. Given enough data to be representative of p_X , one can use neural networks to learn the “ideal” penalty function. This approach has been explored in [19] [20].

Like for the Plug-and-Play Prior Methods, (B), these methods only require knowledge about p_X . This can be provided via handcrafted parts, via known ground truth data or via surrogate, which represents p_X good enough.

- **What do we learn:**

The parametrization θ of a penalty function $\phi_\theta : X \rightarrow \mathbb{R}_{g \in 0}$ (for example represented by a neural network).

- **Intrinsically required knowledge:**

Ground truth data about p_X e. g. in the form of many samples.

9.1.2.2. B: Plug-and-Play Prior Methods

This approach is in some sense (w. r. t. proximal operators) dual to approach A.

Plug-and-Play Prior where introduced in [21], outside the Deep Learning context. The authors pointed out that the alternating direction method of multipliers (ADMM) algorithm used for reconstruction in Inverse Problems could easily be adapted to incorporate any type of denoising method (or more general, any type of algorithm) as a prior during reconstruction. The prior is incorporated by replacing the proximal-operator, $\text{prox}_\phi : X \rightarrow X$, of some penalty function, ϕ (within the reconstruction algorithm e.g. ADMM) with some other operator. It is worth noting, that this operator does not have to be a proximal operator of some penalty function anymore. With the rise of Deep Learning methods people started to learn these proximal-operator which are replaced by neural networks.

It was also possible to extend the basic idea to other algorithms like proximal descent or primal dual hybrid gradient, not just ADMM, see for example [22, 23].

- **What do we learn:**

The parametrization θ of an operator $p_\theta : X \rightarrow X$ that – incorporated in some reconstruction algorithm, in which it replaces the proximal operator of some penalty function – improves the reconstruction.

- **Intrinsically required knowledge:**

Ground truth data about p_X , e. g., in the form of many samples.

9.1.2.3. C: Gradient-Descent-by-Gradient-Descent type Methods

A major field of research in Inverse Problems and Deep Learning is to learn an iterative method from data. This field makes implicit use of the Plug-and-Play prior idea but goes way further. Two very prominent papers in this field are the “Learning Fast Approximations of Sparse Coding” (LISTA) paper [24] and the “Learning to learn by gradient descent by gradient descent” paper [25]. Despite the fact that the paper [25] is not explicitly solving an Inverse Problem their method can be seen as the most data driven form of this type of method. Other authors applied the ideas from [25] directly to solve Inverse Problems [26].

The basic idea of this method has be extended in multiple ways to incorporate prior knowledge e.g. by unrolling existing iteration methods up until a fixed number of iterations and replacing different parts of them by neural networks [27, 28, 29, 30, 31, 32].

One tries to learn parameters θ of a function $f_\theta : Y \rightarrow X$, such that it produces “nice” reconstructions via

$$\min_{\theta} \mathbb{E}_{p_{XY}} [d(f_\theta(y), x)],$$

where d is some measure of distance.

- **What do we learn:**



The parametrization θ of an operator $f_\theta : Y \rightarrow X$ that directly produces “nice” reconstructions.

- **Intrinsically required knowledge:**

Ground truth data about p_{XY} e. g. in the form of many samples.

9.1.2.4. D: Regularization by Architecture

From an abstract point most (if not all) of machine learning methods – and therefore all methods above – can be seen as parameter identification problems. Despite this being the case for all of the above mentioned types this one is the closest to the original notion of parameter identification problems, since no training data is required; one solely fits the parameters θ of a function

$$F_\theta : G \rightarrow \mathbb{R}_{g \in 0}$$

for (usually) a single point of data, $g \in G$ via

$$\min_{\theta} F_\theta(g).$$

These methods are very new, there is only some internal work by us and one very recent paper by a group from the University of Texas [33]. Both works are inspired by a recent paper [34] that noticed, that the inherent structure of so-called convolutional neural networks (CNN) is a good prior for images (even without training). This inspired the authors in [34] to solve Inverse Problems with the prior that the reconstructed image has to lie in the range of a CNN-architecture (non-specific to a given parameterization of the network).

- Deep Image Prior:

$$F_\theta(u) = \|u - Ac(\theta)\|_2^2,$$

where u is the measured signal, A the MPI operator and c (the output of an untrained neural network parameterized by θ whose input is a constant) the concentration.

- Our “Deep Operator Priors” are all of the form:

$$F_\theta(A) = \sum_{i=1}^N w_i \|A_i - f_\theta(c_i)\|_2^2 + \phi(f_\theta, c_i),$$

where f_θ is a neural network representing and enforcing the form of the forward operator in its architecture, A_i usually the columns of a measured operator corresponding to sample concentrations c_i and ϕ a penalty function enforcing structures on weights in the internal representations of f_θ . The w_i are weights to incorporate SNR and similar knowledge. Possible architectures are mixtures of CNN for the spatial structure and RNN (recurrent neural network) for the temporal (frequency) structure.

It is also a still open question whether the $\|\cdot\|_2$ -loss is really the best way to enforce the regression, we are also thinking about using the Wasserstein loss [29].

We see huge potential in these methods, since they allow one to incorporate abstract structural knowledge about the object one ones to regularize. The main difference to “classic” parameter identification is, that one uses deep models as structures (instead of e. g. differential equations). This allows one to incorporate more abstract and less precise knowledge about some underlying structure of a given point of data These methods relate to traditional parameter identification problems, like Deep Learning relates to machine learning.

We not only want to use these type of methods via the one described in [33], but also via own approaches. We see massive potential in applying these types of ideas to the forward operator via casting abstract knowledge about it into into the architectural design of a neural network that in-turn is fitted to a measured (noisy and error prone, maybe incomplete) version of the forward operator.

These methods are especially interesting for Magnetic Particle Imaging, since they do not rely on ground truth



data. This is crucial, since for MPI – as well as for nearly any other novel Inverse Problem – one can not expect to have sufficient ground truth data, see chicken-and-egg-problem.

- **What do we learn:**
A regularized version of some data point (e. g. an image or even an operator itself).
- **Intrinsically required knowledge:**
Any kind of abstract knowledge about the data point could potentially be incorporated.

9.1.2.5. E: Image Post-Processing via Deep Learning

Of course, all kinds of image post-processing techniques can be applied to improve MPI reconstructions: inpainting, denoising, deblurring, etc. The leading methods in this field are mostly Deep Learning based nowadays. The amount of literature is expanding rapidly, see for examples [35, 36, 37, 38, 39, 40, 41].

9.1.3. Applying Deep Learning to Magnetic Particle Imaging

In MPI we have to deal with the full extend of the chicken-and-egg problem described earlier. Therefore many of the methods described above that rely on ground truth data (and which in general do not solve new Inverse Problems) are not applicable. This makes MPI an example par excellence for bringing Inverse Problems and Deep Learning together in solving previously unsolved Inverse Problems.

Possible approaches to tackle this union:

- Use surrogate data sets.
- Regularization via Architecture.

Using surrogate data sets means to use data that is presumably similar to MPI data, like MRI data, to boot strap a Deep Learning approach. Approaches that lean to that are especially approaches of the kind, where one learns a penalty term, since these methods are intrinsically from the Inverse Problem itself.

Using architecture as a regularization is a very new field. Obviously Deep Image Prior should be implemented for MPI, but there are a myriad of other opportunities to evaluate. For example one could use the structure provided by a recurrent neural network, like a long-short-term-memory network (LSTM) combined with a convolutional neural network to do inpainting/deblurring on the measured operator to reconstruct measurements with bad signal-to-noise ratios. This could be done via one big optimization in which one optimizes the structure of an extremely deep LSTM that reaches over all frequency-(or time) measurements of an operator at the same time. The goal of the optimization is to fit the output of the network to the measured operator (learning its structure) weighted by the signal-to-noise ratio of the measurements.

9.2. Implementation

We will now describe the **network** we are using to deploy our deep image prior / regularization by architecture approach to magnetic particle imaging as described above. Since it is not clear what a good prior for MPI is or how one would encode one would cast it into a regularizing architecture. Here, we use the deep image prior introduced by [42], specifically their U-net architecture. Our implementation is based on Tensorflow [43] and Keras [44] and has the following specifications: Between the encoder and decoder part of the U-net our skip connections have 4 channels. The convolutional encoder goes from the input to 32, 32, 64 and 128 channels each with strides of 2×2 and filters of size 3×3 . Then the convolutional decoder has the mirrored architecture plus first a resize-nearest-neighbor layer to reach the desired output shape and second an additional ReLU convolutional layer with filters of size 1. The number of channels of this last layer is 3 for DS1 to accommodate for the 3 slices and 1 for DS2. The input of the network is given by a fixed Gaussian random input of size $1 \times 32 \times 32$ or $3 \times 32 \times 32$. For further details on this architecture we refer to [42].

Since Tensorflow does not support auto gradients for complex numbers, we split up our **loss function** into the



form

$$\|A\phi_W(z) - y^\delta\|^2 = \|\operatorname{real}(A)\phi_W(z) - \operatorname{real}(y^\delta)\|^2 \quad (9.10)$$

$$+ \|\operatorname{imag}(A)\phi_W(z) - \operatorname{imag}(y^\delta)\|^2, \quad (9.11)$$

where `real` and `imag` denote the real and imaginary parts respectively. If nothing else is said we used Adam [45] for our optimizations. Sometimes our optimization apparently got stuck in some undesirable local minimum early on, such that it quickly became apparent that the result would not be anything close desirable. In those cases we simply restarted the optimization (with a new random initialization of the network). The implementation can be found at Google Drive https://drive.google.com/open?id=1F_Pp5VYhrnLDXh3FEhaEnJ-HQNJYCrF5.

The implementation was made in [Python](#), an interpreted, high-level, general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

For presentation we used [Jupyter](#), which is a non-profit, open-source project, born out of the IPython Project in 2014 as it evolved to support interactive data science and scientific computing across all programming languages. Jupyter will always be 100% open-source software, free for all to use and released under the liberal terms of the modified BSD license.

For plotting and visualisation was used [Matplotlib](#), a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc., with just a few lines of code.

Also [SciPy](#), a free and open-source Python library used for scientific computing and technical computing was used. SciPy contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering. SciPy builds on the [NumPy](#) array object and is part of the NumPy stack which includes tools like Matplotlib, pandas and SymPy, and an expanding set of scientific computing libraries. This NumPy stack has similar users to other applications such as MATLAB, GNU Octave, and Scilab. The NumPy stack is also sometimes referred to as the SciPy stack.

For the deep learning part we mainly used two libraries. The first one [Tensorflow](#), is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications. TensorFlow allow you to build and train state-of-the-art models without sacrificing speed or performance. TensorFlow gives you the flexibility and control with features like the Keras Functional API and Model Subclassing API for creation of complex topologies. For easy prototyping and fast debugging, use eager execution.

The other deep learning tool used was [Keras](#), a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research. we use Keras because it allows for easy and fast prototyping (through user friendliness, modularity, and extensibility), supports both convolutional networks and recurrent networks, as well as combinations of the two, and runs seamlessly on CPU and GPU.

Those libraries need to be installed for run the codes. To simplify package management and deployment was used [Anaconda](#), a free and open-source distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.).



The Anaconda distribution is used by over 15 million users and includes more than 1500 popular data-science packages suitable for Windows, Linux, and MacOS. The main reason to use Anaconda was that it has installed almost all libraries that we use, except Tensorflow and Keras, which were installed separately.

9.3. Computer requirements

The codes provided were runned in a [SuperServer 4028GR-TR](#) with the following environment:

- Operating system: Ubuntu 16.04.5 LTS
- Processor: Intel Xeon E5-2630 v4 (10 cores, 20 threads)
- System RAM: 64 GiB
- Graphic cards: 4x Nvidia GeForce GTX 1080 Ti (3584 cores, 11Gbps memory speed, 11GB GDDR5X Standard Memory Config)
- Storage: 2x 240GB Intel SSD DC S4500 240GB SATA, 8x 2TB SATA3-HD Seagate Exos E7E2000

Since the implementation is in Python, there is no need to compile or build the code, the notebooks can be runned once all the required libraries are installed. So no Makefiles or other computer setting environment files are needed. It is recommended to have at least one GPU to run the code.

9.4. Numerical examples

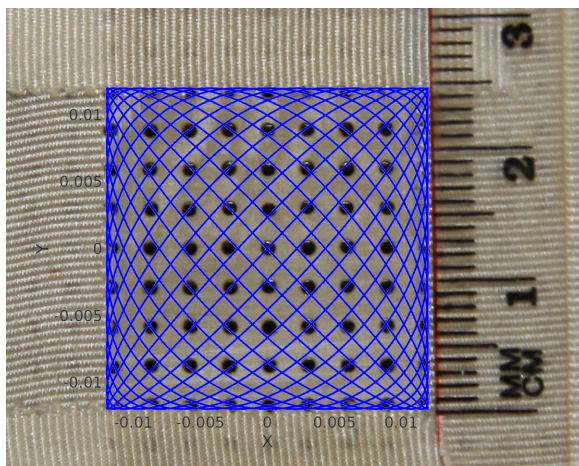


Figure 9.1: Experimental platform used to obtain dataset DS1 with FFP Lissajous trajectory in blue. Photo taken at University Medical Center Hamburg-Eppendorf by T. Kluth.

We test the capability of the Deep Imaging Prior approach to improve image reconstruction. This is done by using two datasets generated by using the Bruker preclinical MPI system at the University Medical Center Hamburg-Eppendorf. A 2D excitation in the x/y -direction is used with excitation frequencies of 2.5/102 MHz (≈ 24.51 kHz) and 2.5/96 MHz (≈ 26.04 kHz) resulting in a 2D Lissajous trajectory with a period of approximately 0.6528 ms. The drive field amplitude in x - and y -direction is $12 \text{ mT}/\mu_0$ respectively. The gradient strength of the selection field is $2 \text{ T/m}/\mu_0$ in z -direction and $-1 \text{ T/m}/\mu_0$ in x - and y -direction. The time-dependent voltage signal is sampled with a rate of 2.5 MHz from $L = 3$ receive coil units. The discretization in time and the real-valued signal results in 817 available Fourier coefficients (for $\psi_j, j \in \{0, \dots, 816\}$) for each receive coil. Thus each system matrix S has at most $3 \cdot 817 = 2451$ rows. The two datasets are as follows:

DS1 2D phantom dataset: The system matrix is obtained by using a cubic sample with edge length of 1 mm. The calibration is done with Resovist[®] tracer with a concentration of 0.25 mol/l. The field-of-view has a size of $29 \text{ mm} \times 29 \text{ mm} \times 3 \text{ mm}$ and the sample positions have a distance of 1 mm in each direction resulting in a size of $29 \times 29 \times 3$ voxels, i.e., 2523 columns in the system matrix. System matrix entries

are averaged over 200 repetitions and empty scanner measurements are performed every 29 calibration scans. It is ensured that the used phantoms are positioned within the calibrated FOV by moving an experimental platform in the desired region, see Figure 9.1. The phantom measurements are averaged over 10000 repetitions of the excitation sequence. We use the three following phantoms:

- “4mm”: Two cylindrical glass capillary with an inner diameter of 0.7 mm filled with Resovist[®] with a concentration of 0.25 mol/l are placed in the x/y-plane oriented in y-direction. The height of the tracer in the capillaries are 10 mm (left capillary) and 21 mm (right capillary). The distance between the capillaries in x-direction is 4 mm. See also Table 9.1 for an illustration.
- “2mm”: Like the “4mm” phantom with 2 mm distance in x-direction between the glass capillary. See also Table 9.1 for an illustration.
- “one”: The same capillaries from the “4mm” phantom are used and arranged as the digit one.

DS2 *Open MPI dataset*: The dataset is publicly available at [46]. From the dataset the 2D calibration system matrix for the x/y-plane located at $z=0$ mm is used for the reconstruction. Here, the system matrix is obtained by using a cuboid sample with an edge length of 2 mm \times 2 mm \times 1 mm. The calibration is done with Perimag[®] tracer with a concentration of 0.1 mol/l. The considered field-of-view has a size of 38 mm \times 38 mm \times 1 mm and the sample positions have a distance of 2 mm in x- and y-direction resulting in a size of 19 \times 19 \times 1 voxels, i.e., 361 columns in the system matrix. System matrix entries are averaged over 1000 repetitions and empty scanner measurements are performed every 19 calibration scans. In contrast to the previous dataset the used phantoms are not limited to the covered field of view of the system matrix. The phantom measurements are averaged over 1000 repetitions of the excitation sequence. According to the description on [46] we have the following three phantoms:

- “concentration”: This phantom consists of 8 cubes of 2mm edge length resulting in 8 μ l volume each. The distance of the cubes is 12 mm between centers (10 mm between edges) within the x/y-plane and 6 mm between centers (4 mm between edges) in z-direction. The sample chambers are numbered from 1 to 8 starting with the top layer on the top left position (positive x- and y-direction), counting clockwise. Then starting with the lower layer with number 5 on the top left (positive X and Y direction), counting clockwise. The concentrations in the 8 sample chambers are diluted with a factor of 1.5 in each step and the values are 100.0, 66.6, 44.4, 29.6, 19.7, 13.1, 8.77, and 5.85 mmol/l. See also Table 9.1 for an illustration.
- “shape”: The phantom is a cone defined by a 1 mm radius tip, an apex angle of 10 deg, and a height of 22 mm. The total volume is 683.9 μ L. Perimag[®] tracer with a concentration of 0.05 mol/L is used. See also Table 9.1 for an illustration.
- “resolution”: The resolution phantom consists of 5 tubes filled with Perimag[®] with a concentration of 0.05 mol/l. The 5 tubes have a common origin on one side of the phantom. From there they extend in different angles from this origin within the x/y- and the y/z-plane. In z-direction the angles in the y/z-plane are chosen smaller (10 deg and 15 deg) than in x/y-plane (20 deg and 30 deg). See also Table 9.1 for an illustration.

All data is provided in the Magnetic Particle Imaging Data Format Files (MDF) encoded according to [47, 48].

In MPI there exist two standard approaches which are commonly combined to determine the index sets J_ℓ , $\ell = 1, 2, 3$, for the purpose of **preprocessing**: a band pass approach and an SNR-type thresholding. Let $I_{BP} = \{j \in \mathbb{Z} \mid b_1 \leq |j|/T \leq b_2\}$ be the band pass indices for frequency band limits $0 \leq b_1 < b_2 \leq \infty$. For the SNR-type thresholding one standard quality measure is determined by computing a ratio of mean absolute values from individual measurements $v_\ell^{(i)}$ (as previously described) and a set of empty scanner measurements $\{v_{\ell,0}^{(k)}\}_{k=1}^K$ [49]:

$$d_{\ell,j} = \frac{\frac{1}{N} \sum_{i=1}^N |\langle v_\ell^{(i)} - \mu_\ell^{(i)}, \psi_j \rangle|}{\frac{1}{K} \sum_{k=1}^K |\langle v_{\ell,0}^{(k)} - \mu_\ell, \psi_j \rangle|} \quad (9.12)$$



where $\mu_\ell = \frac{1}{K} \sum_{k=1}^K v_0^{(k)}$ and $\mu_\ell^{(i)} = \kappa_i v_{\ell,0}^{(k_i)} + (1 - \kappa_i) v_{\ell,0}^{(k_i+1)}$ is a convex combination of the previous (k_i -th) and following ($k_i + 1$ -th) empty scanner measurement with respect to the i -th calibration scan; $\kappa_i \in [0, 1]$ chosen equidistant for all calibration scans between two subsequent empty scanner measurements. For a given threshold $\tau \geq 0$ we thus obtain

$$J_\ell = \{j \in I_{BP} | d_{\ell,j} \geq \tau\} \quad (9.13)$$

for $\ell = 1, 2, 3$.

We will now describe the general setup we use to apply the deep inversion prior approach to the reconstruction of 2 dimensional magnetic particle imaging data. We do the **processing** of the data in the following manner:

1. We build the system matrix S and the measurement v which are associated with the index sets J_ℓ , $\ell = 1, 2, 3$, based on an SNR-type thresholding with $\tau = 2 ((d_{\ell,j})_{j,\ell}$ also provided by the MDF file) and the bandpass index set with the passband boundaries $b_1 = 80$ kHz and $b_2 = 625$ kHz.
2. We subtract the signal of an empty scanner measurement from the phantom data to correct for the background signal.
3. The resulting linear equation system $Sc = v$ is multiplied with a diagonal matrix W with the reciprocal of the 2-norm of the respective row of the system matrix on the diagonal.

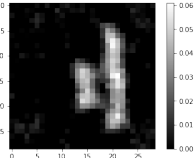
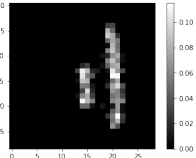
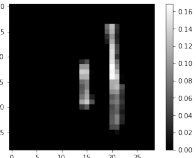
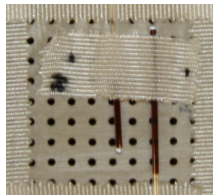
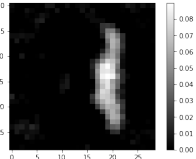
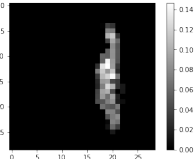
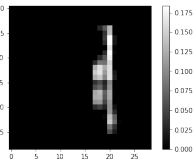
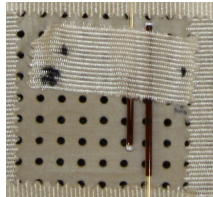
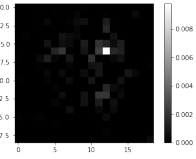
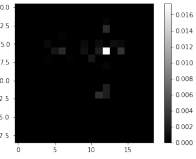
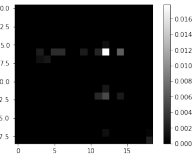
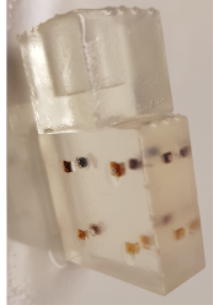
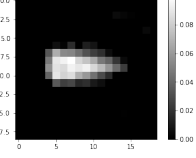
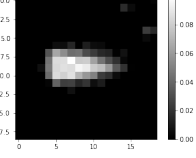
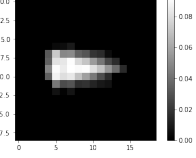
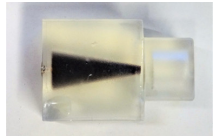
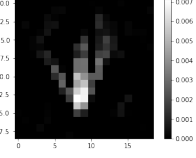
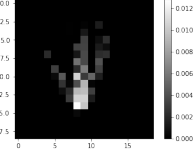
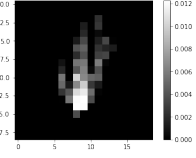

This leaves us with the a processed system matrix, to which we will from now on refer to as $A = WS \in \mathbb{C}^{M \times N}$, and signals to which we will from now on refer to as $y^\delta = W(v - v_0) \in \mathbb{C}^M$ $M = \sum_{\ell=1}^L |J_\ell|$. For DS1 we end up with $M = 211$ and $N = 29^2 \cdot 3 = 2523$ and for DS2 with $M = 842$ and $N = 19^2 = 361$.

Bibliography

- [1] J. Weizenecker, B. Gleich, J. Rahmer, H. Dahnke, and J. Borgert, "Three-dimensional real-time in vivo magnetic particle imaging," *Physics in Medicine and Biology*, vol. 54, no. 5, p. L1, 2009.
- [2] A. Khandhar, P. Keselman, S. Kemp, R. Ferguson, P. Goodwill, S. Conolly, and K. Krishnan, "Evaluation of peg-coated iron oxide nanoparticles as blood pool tracers for preclinical magnetic particle imaging," *Nanoscale*, vol. 9, no. 3, pp. 1299–1306, 2017.
- [3] J. Franke, R. Lacroix, H. Lehr, M. Heidenreich, U. Heinen, and V. Schulz, "Mpi flow analysis toolbox exploiting pulsed tracer information – an aneurysm phantom proof," *International Journal on Magnetic Particle Imaging*, vol. 3, no. 1, 2017. [Online]. Available: <https://journal.iwmpi.org/index.php/iwmpi/article/view/36>
- [4] J. Haegele, J. Rahmer, B. Gleich, J. Borgert, H. Wojtczyk, N. Panagiotopoulos, T. Buzug, J. Barkhausen, and F. Vogt, "Magnetic particle imaging: visualization of instruments for cardiovascular intervention," *Radiology*, vol. 265, no. 3, pp. 933–938, 2012.
- [5] J. Salamon, M. Hofmann, C. Jung, M. G. Kaul, F. Werner, K. Them, R. Reimer, P. Nielsen, A. vom Scheidt, G. Adam, T. Knopp, and H. Ittrich, "Magnetic particle/magnetic resonance imaging: In-vitro MPI-guided real time catheter tracking and 4D angioplasty using a road map and blood pool tracer approach," *PLoS ONE*, vol. 11, no. 6, pp. e0156899–14, 2016.
- [6] E. Y. Yu, M. Bishop, B. Zheng, R. M. Ferguson, A. P. Khandhar, S. J. Kemp, K. M. Krishnan, P. W. Goodwill, and S. M. Conolly, "Magnetic particle imaging: A novel in vivo imaging platform for cancer detection," *Nano Letters*, vol. 17, no. 3, pp. 1648–1654, 2017. [Online]. Available: <http://dx.doi.org/10.1021/acs.nanolett.6b04865>
- [7] K. Murase, M. Aoki, N. Banura, K. Nishimoto, A. Mimura, T. Kuboyabu, and I. Yabata, "Usefulness of magnetic particle imaging for predicting the therapeutic effect of magnetic hyperthermia," *Open Journal of Medical Imaging*, vol. 5, no. 02, p. 85, 2015.
- [8] K. H. Jin, M. T. McCann, E. Froustey, and M. Unser, "Deep convolutional neural network for inverse problems in imaging," *IEEE Transactions on Image Processing*, vol. 26, no. 9, pp. 4509–4522, 2017.



Table 9.1: Different Reconstructions. Photos for phantoms “4mm” and “2mm” taken at University Medical Center Hamburg-Eppendorf by T. Kluth. Photos for phantoms “concentration”, “shape”, and “resolution” as provided by [46].

Phantom	Kaczmarz with ℓ_2	ℓ_1	DIP	Photo
“4mm” (DS1)				
	$\tilde{\lambda} = 5e - 4$	$\tilde{\lambda} = 5e - 3$	$\eta = 5e - 5$	
“2mm” (DS1)				
	$\tilde{\lambda} = 5e - 4$	$\tilde{\lambda} = 5e - 3$	$\eta = 5e - 5$	
“concentration” (DS2)				
	$\tilde{\lambda} = 5e - 3$	$\tilde{\lambda} = 1e - 2$	$\eta = 5e - 5$	
“shape” (DS2)				
	$\tilde{\lambda} = 5e - 3$	$\tilde{\lambda} = 1e - 2$	$\eta = 5e - 5$	
“resolution” (DS2)				
	$\tilde{\lambda} = 5e - 3$	$\tilde{\lambda} = 5e - 3$	$\eta = 5e - 5$	

- [9] S. Lunz, O. Öktem, and C.-B. Schönlieb, “Adversarial regularizers in inverse problems,” *arXiv preprint arXiv:1805.11572*, 2018.
- [10] J. Behrmann, C. Etmann, T. Boskamp, R. Casadonte, J. Kriegsmann, and P. Maaß, “Deep learning for tumor classification in imaging mass spectrometry,” *Bioinformatics*, vol. 34, no. 7, pp. 1215–1223, 2017.
- [11] S. Wang, Z. Su, L. Ying, X. Peng, S. Zhu, F. Liang, D. Feng, and D. Liang, “Accelerating magnetic resonance imaging via deep learning,” in *Biomedical Imaging (ISBI), 2016 IEEE 13th International Symposium on*. IEEE, 2016, pp. 514–517.
- [12] J. Schlemper, J. Caballero, J. V. Hajnal, A. N. Price, and D. Rueckert, “A deep cascade of convolutional

- neural networks for dynamic mr image reconstruction,” *IEEE transactions on Medical Imaging*, vol. 37, no. 2, pp. 491–503, 2018.
- [13] E. Kang, J. Min, and J. C. Ye, “A deep convolutional neural network using directional wavelets for low-dose x-ray ct reconstruction,” *Medical physics*, vol. 44, no. 10, 2017.
- [14] K. Gong, J. Guan, K. Kim, X. Zhang, G. E. Fakhri, J. Qi, and Q. Li, “Iterative pet image reconstruction using convolutional neural network representation,” *arXiv preprint arXiv:1710.03344*, 2017.
- [15] B. Gleich and J. Weizenecker, “Tomographic imaging using the nonlinear response of magnetic particles,” *Nature*, vol. 435, no. 7046, pp. 1214–1217, June 2005.
- [16] J. Weizenecker, B. Gleich, and J. Borgert, “Magnetic particle imaging using a field free line,” *Journal of Physics D: Applied Physics*, vol. 41, no. 10, p. 105009, 2008. [Online]. Available: <http://stacks.iop.org/0022-3727/41/i=10/a=105009>
- [17] T. Kluth, “Mathematical models for magnetic particle imaging,” *Inverse Problems*, 2018, accepted manuscript online available at <http://iopscience.iop.org/article/10.1088/1361-6420/aac535>. [Online]. Available: <http://iopscience.iop.org/article/10.1088/1361-6420/aac535>
- [18] T. Knopp and T. M. Buzug, *Magnetic Particle Imaging: An Introduction to Imaging Principles and Scanner Instrumentation*. Berlin/Heidelberg: Springer, 2012.
- [19] S. Lunz, O. Öktem, and C.-B. Schönlieb. Adversarial regularizers in inverse problems. [Online]. Available: <https://arxiv.org/abs/1805.11572>
- [20] H. Li, J. Schwab, S. Antholzer, and M. Haltmeier, “Nett: Solving inverse problems with deep neural networks,” *arXiv preprint arXiv:1803.00092*, 2018.
- [21] S. V. Venkatakrishnan, C. A. Bouman, and B. Wohlberg, “Plug-and-play priors for model based reconstruction,” in *Global Conference on Signal and Information Processing (GlobalSIP), 2013 IEEE*. IEEE, 2013, pp. 945–948.
- [22] J. R. Chang, C. Li, B. Póczos, B. V. K. V. Kumar, and A. C. Sankaranarayanan, “One network to solve them all - solving linear inverse problems using deep projection models,” *CoRR*, vol. abs/1703.09912, 2017. [Online]. Available: <http://arxiv.org/abs/1703.09912>
- [23] T. Meinhardt, M. Möller, C. Hazirbas, and D. Cremers, “Learning proximal operators: Using denoising networks for regularizing inverse imaging problems,” *CoRR*, vol. abs/1704.03488, 2017. [Online]. Available: <http://arxiv.org/abs/1704.03488>
- [24] K. Gregor and Y. LeCun, “Learning fast approximations of sparse coding,” in *Proceedings of the 27th International Conference on International Conference on Machine Learning*. Omnipress, 2010, pp. 399–406.
- [25] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. De Freitas, “Learning to learn by gradient descent by gradient descent,” in *Advances in Neural Information Processing Systems*, 2016, pp. 3981–3989.
- [26] P. Putzky and M. Welling, “Recurrent inference machines for solving inverse problems,” *arXiv preprint arXiv:1706.04008*, 2017.
- [27] J. Adler and O. Öktem. (2017) Solving ill-posed inverse problems using iterative deep neural networks. [Online]. Available: <https://arxiv.org/abs/1704.04058>
- [28] B. Kelly, T. P. Matthews, and M. A. Anastasio, “Deep learning-guided image reconstruction from incomplete data,” *arXiv preprint arXiv:1709.00584*, 2017.
- [29] J. Adler, A. Ringh, O. Öktem, and J. Karlsson, “Learning to solve inverse problems using wasserstein loss,” *CoRR*, vol. abs/1710.10898, 2017. [Online]. Available: <http://arxiv.org/abs/1710.10898>
- [30] J. Adler and O. Öktem. (2017) Learned primal-dual reconstruction. [Online]. Available: <https://arxiv.org/abs/1707.06474>



- [31] A. Hauptmann, F. Lucka, M. M. Betcke, N. Huynh, B. T. Cox, P. C. Beard, S. Ourselin, and S. R. Arridge, “Model based learning for accelerated, limited-view 3d photoacoustic tomography,” *CoRR*, vol. abs/1708.09832, 2017. [Online]. Available: <http://arxiv.org/abs/1708.09832>
- [32] K. Hammernik, T. Klatzer, E. Kobler, M. P. Recht, D. K. Sodickson, T. Pock, and F. Knoll, “Learning a variational network for reconstruction of accelerated mri data,” *Magnetic resonance in medicine*, vol. 79, no. 6, pp. 3055–3071, 2018.
- [33] D. Van Veen, A. Jalal, E. Price, S. Vishwanath, and A. G. Dimakis, “Compressed sensing with deep image prior and learned regularization,” *arXiv preprint arXiv:1806.06438*, 2018.
- [34] D. Ulyanov, A. Vedaldi, and V. Lempitsky, “Deep image prior,” *arXiv preprint arXiv:1711.10925*, 2017.
- [35] C. Dong, C. C. Loy, K. He, and X. Tang, “Learning a deep convolutional network for image super-resolution,” in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 184–199.
- [36] L. Xu, J. S. Ren, C. Liu, and J. Jia, “Deep convolutional neural network for image deconvolution,” in *Advances in Neural Information Processing Systems*, 2014, pp. 1790–1798.
- [37] C. Dong, C. C. Loy, K. He, and X. Tang, “Image super-resolution using deep convolutional networks,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 2, pp. 295–307, 2016.
- [38] H. C. Burger, C. J. Schuler, and S. Harmeling, “Image denoising: Can plain neural networks compete with bm3d?” in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 2392–2399.
- [39] J. Xie, L. Xu, and E. Chen, “Image denoising and inpainting with deep neural networks,” in *Advances in neural information processing systems*, 2012, pp. 341–349.
- [40] X. Tao, H. Gao, X. Shen, J. Wang, and J. Jia, “Scale-recurrent network for deep image deblurring,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8174–8182.
- [41] C. Chen, Q. Chen, J. Xu, and V. Koltun, “Learning to see in the dark,” *arXiv preprint arXiv:1805.01934*, 2018.
- [42] D. Ulyanov, A. Vedaldi, and V. S. Lempitsky, “Deep image prior,” *CoRR*, vol. abs/1711.10925, 2017. [Online]. Available: <http://arxiv.org/abs/1711.10925>
- [43] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, “TensorFlow: Large-scale machine learning on heterogeneous systems,” 2015, software available from tensorflow.org. [Online]. Available: <https://www.tensorflow.org/>
- [44] F. Chollet *et al.*, “Keras,” <https://keras.io>, 2015.
- [45] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
- [46] “Open mpi data,” <https://magneticparticleimaging.github.io/OpenMPIData.jl/latest/index.html>, accessed: 2018-11-16.
- [47] T. Knopp, T. Viereck, G. Bringout, M. Ahlborg, A. von Gladiss, C. Kaethner, A. Neumann, P. Vogel, J. Rahmer, and M. Möddel, “Mdf: Magnetic particle imaging data format,” *ArXiv e-prints*, vol. 1602.06072v6, pp. 1–15, jan 2018, article, MDF. [Online]. Available: <http://arxiv.org/abs/1602.06072v6>
- [48] “Github mdm,” <https://github.com/MagneticParticleImaging/MDF>, accessed: 2018-11-16.
- [49] J. Franke, U. Heinen, H. Lehr, A. Weber, F. Jaspard, W. Ruhm, M. Heidenreich, and V. Schulz, “System characterization of a highly integrated preclinical hybrid mpi-mri scanner,” *IEEE Transactions on Medical Imaging*, vol. 35, no. 9, pp. 1993–2004, Sept 2016.



10. A mixed-integer programming (MIP) model for a joint assignment of drivers and locomotives to trains at a rail freight company

Jonasz Staszek¹, Andreas Bärmann¹, Alexander Martin¹

¹*Friedrich-Alexander-Universität Erlangen-Nürnberg*

Abstract. In this work we present a mixed-integer programming model (MIP) for a joint assignment of drivers and locomotives to trains (i.e. transportation tasks) in rail freight traffic. The significance of the challenge at hand lies in both the complexity of the planning problem and its prospective economic and environmental impact. We use set packing approach with compatibility, conflict and multiple choice constraints in our modelling approach. At this stage, our objective is to maximize the number of trains performed, i.e. to maximize the number of trains for which both a locomotive and a driver were found. We take into consideration the compatibilities between the drivers and locomotives, the complex labour code regulations regarding working time as well as the locomotives' maintenance requirements. Our approach can efficiently solve small instances. Our further efforts will be focusing on studying the mathematical properties of the problem further to allow reduction of the running time, in order to enable instances connected to longer planning period to solve efficiently. We would also like to develop an approach to schedule the locomotive empty runs.

Keywords: discrete optimization, joint vehicle and staff assignment, integer model, joint vehicle and crew scheduling.

10.1. Introduction

In this work, we present our approach to optimization of the joint assignment of locomotives and drivers to transportation tasks (also called trains), which takes into consideration the compatibilities between the drivers and locomotives, the complex labour code regulations regarding working time as well as the locomotives' maintenance requirements.

The significance of the challenge at hand lies in both the complexity of the planning problem and its prospective economic and environmental impact. The complexity of the discussed problem can be viewed from both theoretical and applied perspective. From the theoretical perspective, set packing problems, long known to be NP-complete [1], are frequently used in the resource assignment problems (see for example [2]). The novelty of our approach lies in considering the compatibilities between distinct resource groups as well as the Polish labor code requirements. In the reality of rail freight planning, the process of assigning locomotives and drivers to trains is carried out sequentially, using many complexity-reduction techniques, in order to make it solvable to human planners. Such an approach can lead to suboptimal solutions. The economic benefit of the studied problem is the increased efficiency in the assignment of locomotives and drivers, and identification of excess capacities that could then be used to transport even more cargo via rail.

Our work focuses on presentation of the planning problem and the model considered. We also present the promising initial results on the real-life instances supplied by DB Cargo Polska S.A., with some simple instances are solved efficiently.

10.2. Methods

We present a mixed-integer programming model (MIP) for a joint assignment of drivers and locomotives to trains (i.e. transportation tasks). We use a set packing approach with compatibility, conflict and multiple choice constraints in our modelling approach. At this stage, our objective is to maximize the number of trains performed, i.e. to maximize the number of trains for which both a locomotive and a driver were found. The inputs to the model are: a set of trains to be performed T (together with departure and arrival times, origin and



destination stations and locomotive requirements), a set of locomotives L (together with information about their type and fuel; their maintenance requirements are stored in set $M^l \forall l \in L$) and a set of drivers D (incl. their geographical assignment, licensing to locomotive types and licensing to train routes), as well as information about the compatibilities between them. We assume that the locomotive type is pre-determined for each train. We could relax this assumption by choosing to search for any suitable locomotive given the mass of the train and the respective power requirements, as well as the availability of the overhead electric line.

In the model, we need to make sure that each train is staffed by one suitable driver and one suitable locomotive. These decisions are modelled with binary variables δ_d^t (for drivers) and λ_l^t (for locomotives). Next, we need to make sure that no locomotive or driver is planned in such a way that they would be required to be present in two places at the same time. Although we allow limited mobility of drivers between stations (i.e. a driver can pick up another train from the station where his previous train has arrived, or in a station in the neighbourhood defined as a maximum of 120 minute car ride from the arrival station of the previous train), we do not allow it for locomotives (i.e. locomotive needs to pick up another train exactly from the station where it has arrived with the previous one).

Furthermore, the Polish labor code stipulates that we need to make sure that the drivers work for a maximum of 12 hours per working day, and then are given a break of at least 11 hours. In our calculations, we assume that the break shall be at least 12 hours long. We also need to ensure that at least one 35-hour break is planned every week for each driver. Additionally, on at least one of every four Sundays the driver has to have a 24-hour break. To comply with these requirements, we need to distinguish between the first job in a shift, denoted by a binary variable y_d^t , last job in a shift before a short (12h) break, denoted by a binary variable v_d^t and the last job in a shift before a long (35h) break, denoted by a binary variable z_d^t . We also need to know whether the driver has worked on a given Sunday. This is denoted by a binary variable $h_{w,d}^{work}$.

Based on the provided maintenance schedule, we also consider maintenance requirements of the locomotives. The model enforces a selection of a train which will bring the locomotives due for maintenance to a depot station. It also prohibits it from use in the period between the train which brought it to maintenance station and the beginning of the maintenance period. Hence, we need to know which train is the last one of a locomotive before undergoing maintenance. We model that using the binary variable $u_{l,m}^t$.

Finally, for modelling purposes we also need to know which trains t are the first and the last job for locos and drivers in the planning period. We do that with the help of binary variables α_l^t , α_d^t and ω_l^t , ω_d^t , which denote that the train t is the first (α) / the last (ω) one in the planning period for a locomotive $l \in L$ or a driver $d \in D$. All the variables are summarized in Table 1.

Table 10.1: Summary of variables

Name	Description	Type
λ_l^t	Train t is served by a locomotive l	binary
δ_d^t	Train t is served by a driver d	binary
y_d^t	Train t is the first job of a driver d in their shift	binary
v_d^t	Train t is the last job of a driver d before a 12h break	binary
z_d^t	Train t is the last job of a driver d before a 35h break	binary
$u_{l,m}^t$	Train t is the last job of a locomotive l before a maintenance period	binary
α_l^t	Train t shall be the first train of locomotive l in the planning period	binary
ω_l^t	Train t shall be the last train of locomotive l in the planning period	binary
α_d^t	Train t shall be the first train of driver d in the planning period	binary
ω_d^t	Train t shall be the last train of driver d in the planning period	binary
$h_{w,d}^{work}$	Driver d has worked on the Sunday of the week w	binary



In order to facilitate the construction of constraints, we construct a number of auxiliary sets. A brief overview of these is included in the appendix.

Our model reads as follows:

$$\max \sum_{t \in T} \sum_{d \in D^t} \delta_d^t$$

s.t.

$$\sum_{d \in D^t} \delta_d^t \leq 1 \quad \forall t \in T \quad (10.1)$$

$$\sum_{l \in L^t} \lambda_l^t \leq 1 \quad \forall t \in T \quad (10.2)$$

$$\delta_d^t + \delta_d^{t_1} \leq 1 \quad \forall d \in D, \quad \forall t \in T^d, \quad \forall t_1 \in T_{t,d}^{time} \quad (10.3)$$

$$\lambda_l^t + \lambda_l^{t_1} \leq 1 \quad \forall l \in L, \quad \forall t \in T^l, \quad \forall t_1 \in T_{t,l}^{time} \quad (10.4)$$

$$\lambda_l^{t_1} \leq \sum_{t_2 \in T_{t_1,l}^{next.l.pruned}} \lambda_l^{t_2} + \omega_l^{t_1} \quad \forall l \in L, \quad \forall t_1 \in T^l \quad (10.5)$$

$$\lambda_l^{t_1} \leq \sum_{t_2 \in T_{t_1,l}^{prev.l}} \lambda_l^{t_2} + \alpha_l^{t_1} \quad \forall l \in L, \quad \forall t_1 \in T^l \quad (10.6)$$

$$\sum_{t_1 \in T_{t,l}^{next.l.pruned}} \lambda_l^{t_1} \leq 1 \quad \forall l \in L, \quad \forall t \in T^l \quad (10.7)$$

$$\sum_{t \in T^l} \alpha_l^t \leq 1 \quad \forall l \in L \quad (10.8)$$

$$\sum_{t \in T^l} \omega_l^t \leq 1 \quad \forall l \in L \quad (10.9)$$

$$\delta_d^t \leq \sum_{t_1 \in T_{t,d}^{next.d}} \delta_d^{t_1} + v_d^t \quad \forall d \in D, \quad \forall t \in T^d \quad (10.10)$$

$$\delta_d^t \leq y_d^t + \sum_{t_1 \in T_{t,d}^{prev.d}} \delta_d^{t_1} \quad \forall d \in D, \quad \forall t \in T^d \quad (10.11)$$

$$v_d^t \leq \sum_{t_1 \in T_{t,d}^{potential.starts}} y_d^{t_1} + \omega_d^t \quad \forall d \in D, \quad \forall t \in T^d \quad (10.12)$$

$$y_d^t \leq \sum_{t_1 \in T_{t,d}^{potential.ends}} v_d^{t_1} + \alpha_d^t \quad \forall d \in D, \quad \forall t \in T^d \quad (10.13)$$

$$\sum_{t \in T^d} \alpha_d^t \leq 1 \quad \forall d \in D, \quad \forall t \in T^d \quad (10.14)$$



$$\sum_{t \in T^d} \omega_d^t \leq 1 \quad \forall d \in D, \quad \forall t \in T^d \quad (10.15)$$

$$y_d^t \leq \delta_d^t \quad \forall d \in D, \quad \forall t \in T^d \quad (10.16)$$

$$\sum_{t_2 \in T_{t,d}^{\text{common.blocking.beginnings}}} y_d^{t_2} + \delta_d^{t_1} \leq 1 \quad \forall d \in D, \quad \forall t \in T^d, \quad \forall t_1 \in T_{t,d}^{B-} \quad (10.17)$$

$$\sum_{t_2 \in T_{t,d}^{\text{common.beginnings}}} \delta_d^{t_2} \leq \sum_{t_1 \in T_{t,d}^{\text{shift.beginning}} \cup \{t\}} y_d^{t_1} \quad \forall d \in D, \quad \forall t \in T^d \quad (10.18)$$

$$\sum_{t_2 \in T_{t,d}^{\text{common.forward.blockings}}} v_d^{t_2} + \delta_d^{t_1} \leq 1 \quad \forall d \in D, \quad \forall t \in T^d, \quad \forall t_1 \in T_{t,d}^{B+} \quad (10.19)$$

$$\sum_{t_2 \in T_{t,d}^{\text{common.ends}}} v_d^{t_2} \leq \sum_{t_1 \in T_{t,d}^{\text{shift.beginning}} \cup \{t\}} y_d^{t_1} \quad \forall d \in D, \quad \forall t \in T^d \quad (10.20)$$

$$\sum_{t_2 \in T_{t,d}^{\text{common.ends}}} \delta_d^{t_2} \leq \sum_{t_1 \in T_{t,d}^{\text{shift.end}} \cup \{t\}} v_d^{t_1} \quad \forall d \in D, \quad \forall t \in T^d \quad (10.21)$$

$$z_d^t \leq v_d^t \quad \forall d \in D, \quad \forall t \in T^d \quad (10.22)$$

$$\sum_{t_2 \in T_{t,d}^{\text{common.long.breaks}}} z_d^{t_2} + \delta_d^{t_1} \leq 1 \quad \forall d \in D, \quad \forall t \in T^d, \quad \forall t_1 \in T_{t,d}^{35h} \quad (10.23)$$

$$\delta_d^t \leq \sum_{t_1 \in T_{w,d}^{\text{week}}} z_d^{t_1} \quad \forall d \in D, \quad \forall w \in T, \quad \forall t \in T^d \cap T_{w,d}^{\text{week}} \quad (10.24)$$

$$\delta_d^t \leq h_{w,d}^{\text{work}} \quad \forall d \in D, \quad \forall w \in T, \quad \forall t \in T_{w,d}^{\text{Sunday}} \quad (10.25)$$

$$h_{w_n,d}^{\text{work}} + h_{w_{n+1},d}^{\text{work}} + h_{w_{n+2},d}^{\text{work}} + h_{w_{n+3},d}^{\text{work}} \leq 3 \quad \forall d \in D, \quad \forall w_n \in T \quad (10.26)$$

$$u_{l,m}^t \leq \lambda_l^t \quad \forall l \in L, \quad \forall m \in M^l, \quad \forall t \in T_{l,m}^{\text{depo}} \quad (10.27)$$

$$\sum_{t_2 \in T_{t,l,m}^{\text{common.premaint.blocking}}} u_{l,m}^{t_2} + \lambda_l^{t_1} \leq 1 \quad \forall l \in L, \quad \forall m \in M^l, \quad \forall t \in T_{l,m}^{\text{depo}}, \quad \forall t_1 \in T_{t,l,m}^{\text{block}} \quad (10.28)$$

$$\sum_{t \in T_{l,m}^{\text{depo}}} u_{l,m}^t = 1 \quad \forall l \in L, \quad \forall m \in M^l \quad (10.29)$$

$$\delta_d^t \leq \sum_{l \in L^d \cap L^t} \lambda_l^t \quad \forall t \in T \quad (10.30)$$



$$\lambda_l^t \leq \sum_{d \in D^l \cap D^t} \delta_d^t \quad \forall t \in T \quad (10.31)$$

$$\delta_d^t \in \{0, 1\} \quad \forall t \in T, \quad \forall d \in D \quad (10.32)$$

$$\lambda_l^t \in \{0, 1\} \quad \forall t \in T, \quad \forall l \in L \quad (10.33)$$

$$y_d^t \in \{0, 1\} \quad \forall t \in T, \quad \forall d \in D \quad (10.34)$$

$$v_d^t \in \{0, 1\} \quad \forall t \in T, \quad \forall d \in D \quad (10.35)$$

$$z_d^t \in \{0, 1\} \quad \forall t \in T, \quad \forall d \in D \quad (10.36)$$

$$h_{w_n, d}^{work} \in \{0, 1\} \quad \forall d \in D, \quad \forall w_n \in T \quad (10.37)$$

$$u_t^{l, m} \in \{0, 1\} \quad \forall l \in L, \quad \forall m \in M^l, \quad \forall t \in T_{l, m}^{pre-maint} \quad (10.38)$$

$$\alpha_d^t \in \{0, 1\} \quad \forall t \in T, \quad \forall d \in D^t \quad (10.39)$$

$$\omega_d^t \in \{0, 1\} \quad \forall t \in T, \quad \forall d \in D^t \quad (10.40)$$

$$\alpha_l^t \in \{0, 1\} \quad \forall t \in T, \quad \forall l \in L^t \quad (10.41)$$

$$\omega_l^t \in \{0, 1\} \quad \forall t \in T, \quad \forall l \in L^t \quad (10.42)$$

In summary, our model enforces the following aspects of resource allocation at a rail freight company (in the brackets one can find the corresponding constraints):

1. Each train requires both a locomotive and a driver to be performed (1, 2)
2. No trains in time conflict (e.g. running in parallel timewise) can be served by the same locomotive / driver (3, 4)
3. Locomotives can only serve trains which begin in the station in which they are currently located (5 - 9)
4. Drivers can serve trains which begin in the station in which their previous train has ended or in stations in its neighbourhood - up to 120 minute car ride (10 - 15)
5. Each driver requires an 12-hour uninterrupted break after each at most 12 hours long shift (16 - 21)
6. Each driver needs at least one uninterrupted 35-hour break once a week (22 - 24)
7. Each driver needs to have at least one in four Sundays off (25, 26)
8. Locomotives with a maintenance period scheduled in the planning period need to serve a train which will bring them to the station, where the maintenance shall take place (27 - 29)
9. Locomotive and driver must be compatible with one another (30, 31)
10. Finally, all the variables need to be binary (32 - 42)

10.3. Implementation

The model discussed above was implemented in Python 3.7. We used Anaconda as our distribution of choice. It comes with a New BSD License.

In order to process data supplied by the industrial partner, we have used many libraries which are a part of Python Standard Library, available with all Python distributions.

To manipulate the source data, we have also used `pandas` library, available under Three-Clause BSD License. To implement some of the auxiliary sets, we use `networkx` library, which supplies a toolset for creation, manipulation, and study of the structure of complex networks. It comes under the BSD license.

The most important software package that we use is `Gurobi` and its Python module called `gurobipy`. It is a state-of-the-art commercial solver, handling Linear, Quadratic and Mixed Integer Programs. We are using its



free-of-charge academic license.

All of the above-mentioned software is easily available online, with a handy installation manual included.

10.4. Results

To test our approach, we have developed a number of test instances based on the data supplied by our industrial partner, DB Cargo Polska S.A. To solve the model, we have used a node at University of Erlangen's High Performance Computing Center with with Gurobi 9.0, using 2x AMD Opteron 6134 ("Magny Cours") at 2,3 GHz and 120 GB RAM, with a time limit of 48 hours.

We have developed two families of instances. *Full* instances capture the whole complexity of the problem, whereas the *Frequent* instances consider only trains which are planned to go at least ten times a week. Each name is accompanied by the indication of the planning horizon (e.g. 1M - one month, 1W - one week etc.). The results are summarized in the Table 2 below.



Table 10.2: Results

Instance	# of trains	# of drivers	# of locos	Before presolve		After presolve		Optimum	Time
				# rows	# columns	# rows	# columns		
Frequent_1D	56	217	112+16	78 141	17 664	65 012	9 041	35	3.34 s
Frequent_3D	148	217	112+29	206 143	49 871	57 441	36 532	118	57.27 s
Frequent_1W	363	217	112+82	876 726	135 472	168 488	95 765	270 (4.07%)	172 800s
Frequent_1M	1512	217	112+315	1 941 780	317 143	394 114	204 171	-	-
Full_1D	80	217	112+10	103 042	25 186	23 925	14 989	66	9.00 s
Full_3D	240	217	112+31	411 566	78 066	99 575	62 911	209	5491.26 s
Full_1W	629	217	112+88	1 925 357	225 077	313 587	173 529	-	-

The results table presents the name of the instance, then the numbers of trains, locomotives and drivers which were found feasible to be used. Then, it presents the number of rows and columns, both before and after Gurobi's preprocessing routine. Finally, we can see the number of trains performed in the optimum solution and its share in the total trains in the planning period (this does not have to be equal to the number of feasible trains in the table).

Our approach can efficiently solve small instances - the two smallest *Frequent* instances optimize in less than a minute. Three-day (3D) *Full* instances can be solved to optimality in a longer period - approximately 1.5 hours. Within the time limit, we could also close the optimality gap to less than 5% for *Frequent* one-week instance.

In our approach, we do not plan empty runs of the locomotives in this model (e.g. locomotives travelling between stations without cargo, just to pick up another train), and such movements of locomotives are required to enable all trains to run. We also assume that drivers need to perform both their first and last job in the planning horizon within a certain set of stations. We also noticed that some of our secondary set implementations are too strict. Hence we see the small number of trains performed in the optimal solution when compared to the total number of trains in an instance. Extending the planning perspective above three days is still challenging.

10.5. Conclusions and Contributions

The current status of our work shows that the optimization of the assignment of locomotives and drivers, including all the peculiarities of driver-locomotive and driver-route licensing can be efficiently applied at least to short planning horizons.

Our further efforts will be focusing on studying the mathematical properties of the problem further to allow reduction of the running time, in order to enable also weekly, and potentially also monthly instances, to solve efficiently. In particular, as we deal with a set packing problem, we will employ some of the results related to that family of problems to strengthen the existing constraints as well as to decrease their number. We will also design a suitable relaxation approach to further speed up the optimization process. Finally, we would also like to develop an approach to schedule the locomotive empty runs.



Table 10.3: Summary of secondary sets

Name	Description
D^t	Drivers allowed to drive on the route of train t
L^t	Locomotives allowed to drive on the route of train t
D^l	Drivers allowed to drive the locomotive l
T^l	Trains which can be served by a locomotive l
L^d	Locomotives which can be driven by a driver d
T^d	Trains which driver d is licensed to
$T_{t,d}^{B+}$	Trains which cannot be assigned to a driver d if t is his last job in a working day
$T_{t,d}^{35h}$	Trains which cannot be assigned to a driver d if t is his last job before a weekly 35h break
$T_{t,d}^{B-}$	Trains which cannot be assigned to a driver d if t is his first job in a working day
$T_{t,d}^{-12h}$	Trains which could timewise be assigned to a driver d as first jobs in a shift if t is to be performed by them
$T_{w,d}^{week}$	Trains which belong to a calculation week w
$T_{w,d}^{Sunday}$	Trains which belong to a Sunday h_w
$T_{l,m}^{pre-maint}$	Trains which are scheduled to run shortly before a locomotive l is due for maintenance period m
$T_{l,m}^{depo}$	Trains $t \in T_{l,m}^{pre-maint}$ which are heading to the depot station
$T_{t,l,m}^{block}$	Trains $t \in T_{l,m}^{pre-maint}$ which cannot be served by locomotive l after it has arrived at its depot station
$T_{t,d}^{prev-d}$	Past trains which could have been assigned to driver d if it is assigned to a job t
$T_{t,d}^{next-d}$	Future trains which could be assigned to driver d if it is assigned to a job t
$T_{t,d}^{next-d-pruned}$	Future trains which could be assigned to driver d if it is assigned to a job t , excl. trains which have predecessors in $T_{t,d}^{next-d}$
$T_{t,l}^{next-l}$	Future trains which could be assigned to driver l if it is assigned to a job t
$T_{t,l}^{next-l-pruned}$	Future trains which could be assigned to locomotive l if it is assigned to a job t , excl. trains which have predecessors in $T_{t,l}^{next-l}$
$T_{t,d}^{shift-beginning}$	Past trains which could have been assigned - both timewise and locationwise - to driver d if he is assigned to a job t .
$T_{t,d}^{shift-end}$	Future trains which can be assigned - both timewise and locationwise - to driver d if he is assigned to a job t .
$T_{t,d}^{time \& T_{t,l}}$	Trains which are feasible for driver d / locomotive l and in time conflict with the train t
$T_{t,d}^{potential-starts}$	Trains which could be the first jobs of the next shift after the 12h break following train t
$T_{t,d}^{potential-ends}$	Trains which could have been the last job of the previous shift before the 12h break preceding train t
$T_{t,d}^{common-blocking-beginnings}$	Trains which block a train t as violating the past 12h break
$T_{t,d}^{common-beginnings}$	Trains which share an identical $T_{t,d}^{shift-beginning}$
$T_{t,d}^{common-forward-blockings}$	Trains which block a train t as violating the future 12h break
$T_{t,d}^{common-ends}$	Trains which share an identical $T_{t,d}^{shift-end}$
$T_{t,d}^{common-long-breaks}$	Trains which block a train t as violating the future 35h break
$T_{t,l,m}^{common-remaint-blocking}$	Trains which block a train t as departing from the depo station

Bibliography

- [1] R. M. Karp, “Reducibility among combinatorial problems,” in *Complexity of Computer Computations*, R. Miller, J. Thatcher, and J. Bohlinger, Eds. Springer, 1972, pp. 85–103. [Online]. Available: http://doi.org/10.1007/978-1-4684-2001-2_9
- [2] R. Velásquez and M. T. Melo, “A set packing approach for scheduling elective surgical procedures,” in *Operations Research Proceedings 2005*, H.-D. Haasis, H. Kopfer, and J. Schönberger, Eds. Springer, 2006, pp. 425–430. [Online]. Available: http://doi.org/10.1007/3-540-32539-5_67





The ROMSOC project

June 9, 2020

ROMSOC-D5.2-2.0

Horizon 2020