

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 0.2



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:	127
Keywords:	Benchmarks, Model hierarchies
Authors:	<p>Marcus Bannenberg, BUW Patricia Barral, ITMATI-USC Jean-David Benamou, INRIA Federico Bianco, Danieli Andres Binder, MathConsult Sören Dittmer, U-HB Daniel Fernández Comesaña, 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, Philips Onkar Jadhav, MathConsult Tobias Kluth, U-HB Karl Knall, MathTec Alejandro Lengomin, AMIII Peter Maass, U-HB Gianfranco Marconi, Danieli Marco Martinolli, MOX, PoliMi Volker Mehrmann, TU Berlin Pier Paolo Monticone, CorWave SA Umberto Morelli, ITMATI Ashwin Nayak, ITMATI Andreas Obereder, MathConsult Daniel Otero Bager, U-HB Luc Polverelli, CorWave SA 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 Giovanni Stabile, SISSA Bernadett Stadler, Industrial Mathematics Institute JKU Christian Vergara, MOX, PoliMi</p>
Peer review:	<p>Andrés Prieto, ITMATI-UDC Peregrina Quintela, ITMATI-USC</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.

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	A benchmark for atmospheric tomography	1
1.1	Introduction	1
1.1.1	Guide Stars	1
1.1.2	Operating Modes	3
1.1.3	Turbulence Statistic in the Atmosphere	4
1.1.4	Deformable Mirror	5
1.1.5	Wavefront Sensor	6
1.2	Mathematical Problem Formulation - Atmospheric Tomography	7
1.2.1	Matrix Vector Multiplication	8
1.3	Implementation and Computer Requirements	9
1.3.1	Building Code	9
1.3.2	Libraries	10
1.4	Numerical Example	10
1.4.1	Input Parameters	10
1.4.2	Ouput - DM commands	11
II	Implementing acoustic scattering simulations for external geometries within a porous enclosure	12
2.1	Introduction	12
2.2	Implementation	15
2.2.1	Geometry and Meshing	16
2.2.2	Solver	17
2.2.3	Post-processing and Visualization	18
2.3	Case study : Acoustic transmission of a vibrating sphere in a porous enclosure	18
2.3.1	Description of test case sphere	18
2.3.2	Exact solution	19
2.3.3	Geometry and Mesh	20
2.3.4	Solver	22
2.3.5	Visualization	25
2.4	Conclusion	25
III	Software implementation description for Point Source Far field Reflector computation using Sinkhorn algorithm	28
3.1	Introduction	28
3.2	Implementation	28
3.3	Computer requirements	29

IV	Data driven model adaptations of coil sensitivities in magnetic particle imaging	30
4.1	Introduction and literature	30
4.1.1	Magnetic Particle Imaging	31
4.1.2	Deep Learning and Inverse Problems	35
4.1.3	Applying Deep Learning to Magnetic Particle Imaging	38
4.2	Implementation	38
4.3	Computer requirements	40
4.4	Numerical examples	40
V	Multirate time integration and model order reduction for coupled thermal electrical systems	44
5.1	Introduction	44
5.1.1	Thermal-Electric Benchmark Circuit	44
5.1.2	Multirate time integration	45
5.1.3	Model order reduction	46
5.2	Implementation	46
5.2.1	Parameters and functions	47
5.2.2	Setting up the IRK iteration	47
5.2.3	The IRK iteration	47
5.2.4	The POD algorithm	48
5.3	Requirements	49
5.4	Numerical examples	49
5.4.1	Results	50
5.5	Conclusion	52
VI	Model order reduction for parametric high dimensional interest rate models in the analysis of financial risk	53
6.1	Introduction	53
6.2	Mathematical Description	54
6.2.1	Bank Account and Short-Rate	54
6.2.2	Yield Curve	55
6.2.3	Zero-Coupon Bonds	56
6.2.4	Forward Rate	56
6.2.5	Options	57
6.2.6	Interest Rate Cap and Floor	57
6.2.7	No-Arbitrage Pricing	58
6.2.8	Short-Rate Models	58
6.2.9	Yield Curve Simulation	62

6.2.10	Parameter Calibration	65
6.3	Numerical Methods	68
6.3.1	Finite Difference Method	68
6.3.2	Parametric Model Order reduction	72
6.3.3	Greedy Sampling Method	73
6.3.4	Adaptive Greedy Sampling Method	75
6.4	Numerical Example	76
6.4.1	Technical and Software Details	76
6.4.2	Model Parameters	76
6.4.3	Finite Difference Method	78
6.4.4	Model Order Reduction	79
6.5	Calibration of Hull-White Model	81
VII	Software-based representation of an inverse heat conduction problem	83
7.1	Introduction	83
7.2	Implementation and Computer Requirements	86
7.3	Numerical Example	86
VIII	Software-based representation and XFEM-based implementation of the benchmark entitled '<i>Experimental-based Validation of FSI Simulations in Blood Pumps</i>'	90
8.1	Introduction	90
8.2	Experimental Data	92
8.3	Benchmark Plan	93
8.4	Implementation and Computer Requirements	94
8.5	Appendix	97
8.5.A	Licences of Use	97
8.5.B	Configuration files	97
IX	Benchmark for numerical simulation of thermo-mechanical phenomena arising in blast furnaces	100
9.1	Conceptual model	100
9.2	Mathematical model	102
9.2.1	Problem statement in strong formulation	102
9.2.2	Boundary conditions	104
9.2.3	Mathematical model in cylindrical coordinates	105
9.2.4	Axisymmetric model	107

9.3 Weak formulation of governing equations	109
9.3.1 Function spaces	109
9.3.2 Weak formulation of the energy equation	110
9.3.3 Weak formulation of the momentum equation	111
9.4 Strong and weak formulation : general form	112
9.4.1 Energy equation	112
9.4.2 Momentum equation	113
9.5 Numerical experiments and results	114
9.5.1 Essential characteristic conditions	114
9.5.2 Energy equation	115
9.5.3 Momentum equation	115
9.5.4 Coupling	116
9.5.5 Software comparison for actual problem	117
9.6 Software installation and licensing	118

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
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	Hydraulic power-Flow
NS	Navier-Stokes
PQ	Pressure-Flow
XFEM	Extended Finite Element Method
3D	three dimensions
SISSA	Scuola Internazionale Superiore di Studi Avanzati

Part I.

A benchmark for atmospheric tomography

Bernadett Stadler, Ronny Ramlau, Andreas Obereder

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: Adaptive optics, atmospheric tomography, benchmark, MVM.

1.1. Introduction

The new generation of planned earthbound Extremely Large Telescopes (ELT) aims at excellent image quality in a large field of view. Such systems rely on Adaptive Optics (AO) with the task to correct 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.

Before we define the mathematical problem of atmospheric tomography, we first introduce some basics about adaptive optics (AO) such as the concept of guide stars, operating systems, turbulence statistics, deformable mirrors and wavefront sensors. For more details about AO we refer to [1].

1.1.1. Guide Stars

Guide stars (GS) are either natural stars up in the sky near the object of interest or generated by a laser beam.

1.1.1.1. 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

$$\varphi_{\theta}(x) = (P_{\theta}^{NGS}\phi)(x) := \sum_{\ell=1}^L \phi_{\ell}(x + \theta h_{\ell}), \quad (1.1)$$

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

$$C_\eta = \sigma^2 I, \quad (1.2)$$

where σ^2 is the noise variance of a single measurement, which is given by

$$\sigma^2 = \frac{1}{n_{photons}}, \quad (1.3)$$

where $n_{photons}$ is the number of photons per subaperture.

1.1.1.2. 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.

Cone effect

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 1.1). This behaviour is referred to as the **cone effect**.

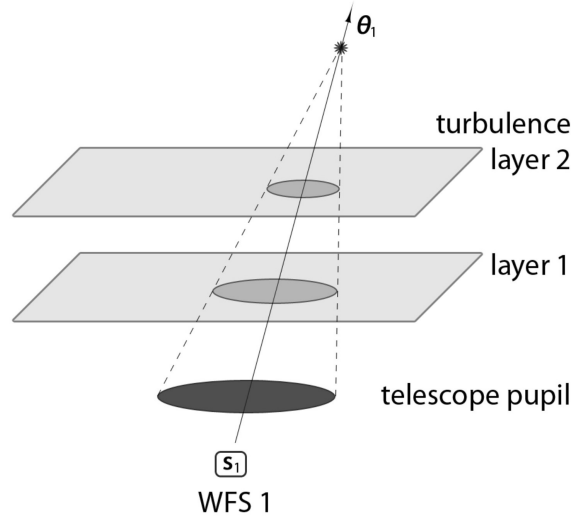


Figure 1.1: Active Subapertures

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

$$\varphi_\theta(x) = (P_\theta^{LGS} \phi)(x) := \sum_{\ell=1}^L \phi_\ell \left(\left(1 - \frac{h_\ell}{H}\right)x + \theta h_\ell \right), \quad (1.4)$$

where P_θ^{LGS} is called the geometric propagation operator in the direction of the LGS.

Spot elongation

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

$$FWHM = 2\sqrt{2\ln(2)}\sigma. \quad (1.5)$$

Further, we define the laser launch positions as (x_1^{LL}, x_2^{LL}) and the midpoint of a subaperture Ω_{ij} by

$$\bar{x}_i = \frac{x_i + x_{i+1}}{2}, \quad (1.6)$$

for $0 \leq i < n_s$ where the x_i are given by (1.19).

The elongation vector in a subaperture Ω_{ij} is given by

$$\beta_{ij} = (\beta_{ij,1}, \beta_{ij,2}) = \frac{FWHM}{H^2}((\bar{x}_i, \bar{x}_j) - (x_1^{LL}, x_2^{LL})). \quad (1.7)$$

The spot elongated noise covariance matrix in a subaperture is given by

$$C_{ij} = \sigma^2 \left(I + \frac{\alpha_\eta^2}{f^2} \right) \begin{pmatrix} \beta_{ij,1}^2 & \beta_{ij,1}\beta_{ij,2} \\ \beta_{ij,1}\beta_{ij,2} & \beta_{ij,2}^2 \end{pmatrix}, \quad (1.8)$$

where σ is defined as in (1.3), 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

$$C_\eta = \text{diag}(C_{ij}), \quad (1.9)$$

with $0 \leq i, j < n_s$ for an active subaperture Ω_{ij} .

1.1.2. Operating Modes

Depending on the number of NGS and LGS the AO systems operates in different modes, which are listed in the following subsections.

1.1.2.1. 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.

1.1.2.2. 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.

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



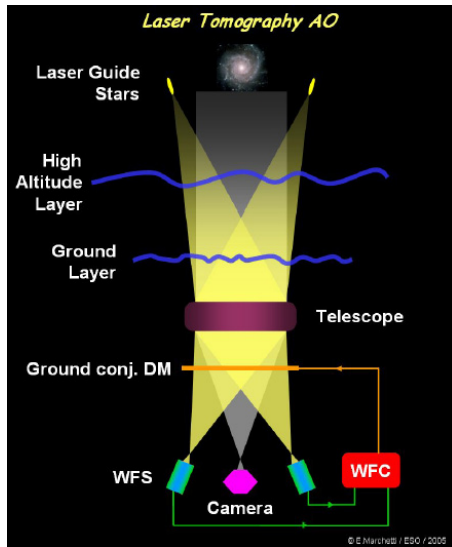


Figure 1.2: Principle of LTAO

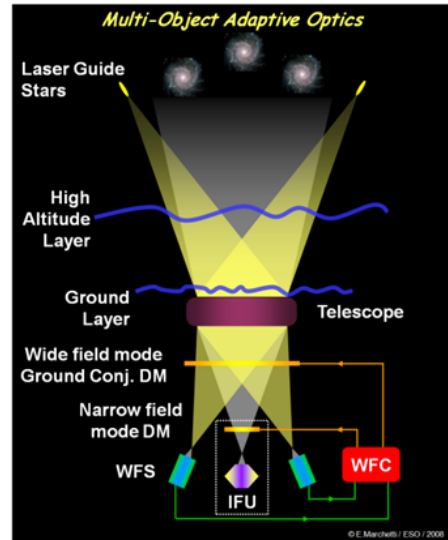


Figure 1.3: Principle of MOAO

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.

1.1.2.3. 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.

1.1.2.4. 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.

1.1.3. 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. Within atmospheric tomography, we assume a layered model of the atmosphere with the goal to reconstruct the turbulent layers. 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

$$C_\ell = \mathcal{F}^{-1} \phi_\ell \mathcal{F}, \text{ for } \ell = 1, \dots, L. \quad (1.10)$$

The operator \mathcal{F} is the Fourier transform and ϕ_ℓ is the spectral density of the turbulent layer given by

$$\phi_\ell(\kappa) := \frac{0.023r_0^{-5/3}C_n^2(h_\ell)}{4\pi(|\kappa|^2 + |\kappa_0|^2)^{11/6}}, \quad (1.11)$$

for a $\kappa_0 < |\kappa| < 2\pi l_0^{-1}$ with $\kappa_0 = 2\pi L_0^{-1}$.

1.1.4. 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

$$\Omega := [-D/2, D/2]^2, \quad (1.12)$$

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

$$x_i := -D/2 + i \cdot d. \quad (1.13)$$

In relation to this, we define the square sub-domains of Ω by

$$\Omega_{ij} := [x_i, x_{i+1}] \times [x_j, x_{j+1}]. \quad (1.14)$$

To each subdomain we associate a bilinear function defined on $[0, 1]^2$

$$b_{ij}(x, y) = a_{ij}(1 - x - y + xy) + a_{i,j+1}(x - xy) + a_{i+1,j}(y - xy) + a_{i+1,j+1}xy, \quad (1.15)$$

where the values a_{ij} are called **actuator commands**.

1.1.4.1. Mirror Fitting

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

$$a_1 = [P_{\theta_1,1}^{NGS} \dots P_{\theta_1,L}^{NGS}] \begin{pmatrix} \phi_1 \\ \vdots \\ \phi_L \end{pmatrix}, \quad (1.16)$$

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

$$\begin{pmatrix} a_1 \\ \vdots \\ a_M \end{pmatrix} = \begin{pmatrix} P_{\theta_{1,1}}^{NGS} & \dots & P_{\theta_{1,L}}^{NGS} \\ \vdots & & \vdots \\ P_{\theta_{M,1}}^{NGS} & \dots & P_{\theta_{M,L}}^{NGS} \end{pmatrix} \begin{pmatrix} \phi_1 \\ \vdots \\ \phi_L \end{pmatrix}. \quad (1.17)$$

For a MCAO system, the fitting operator is more complex, since fitting here requires aligning M mirrors at various altitudes to obtain a good correction over a wide field of view. For the sake of simplicity we omit this case here for the benchmark.

1.1.5. 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 (1.12) we define the subaperture grid for n_s^2 subapertures by

$$\Omega := [-D/2, D/2]^2, \quad (1.18)$$

and the points with equidistant spacing inside the grid by

$$(x_i, x_j) : 0 \leq i, j \leq n_s, \text{ where } x_i := -D/2 + i \cdot d. \quad (1.19)$$

A subaperture is then defined as an open square sub-domain of Ω

$$\Omega_{ij} := (x_i, x_{i+1}) \times (x_j, x_{j+1}). \quad (1.20)$$

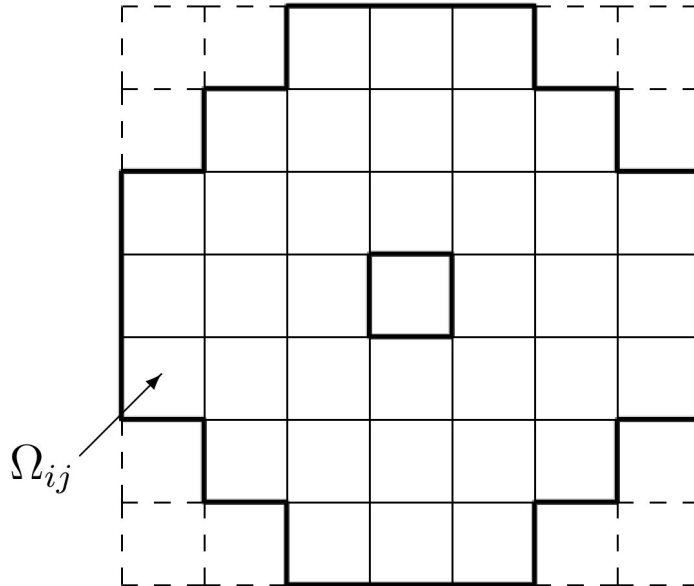


Figure 1.4: Active subapertures

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

$$s = \begin{pmatrix} s^x \\ s^y \end{pmatrix} = \begin{pmatrix} \Gamma_x \varphi \\ \Gamma_y \varphi \end{pmatrix} = \Gamma \varphi. \quad (1.21)$$

The incoming wavefront aberration is approximated by a continuous piecewise bilinear function φ with nodal values φ_{ij} at points defined by Equation (1.19)

$$s_{ij}^x \simeq \frac{(\varphi_{i,j+1} - \varphi_{i,j}) + (\varphi_{i+1,j+1} - \varphi_{i+1,j})}{2}, \quad (1.22)$$

$$s_{ij}^y \simeq \frac{(\varphi_{i+1,j} - \varphi_{i,j}) + (\varphi_{i+1,j+1} - \varphi_{i,j+1})}{2}. \quad (1.23)$$

1.2. Mathematical 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.

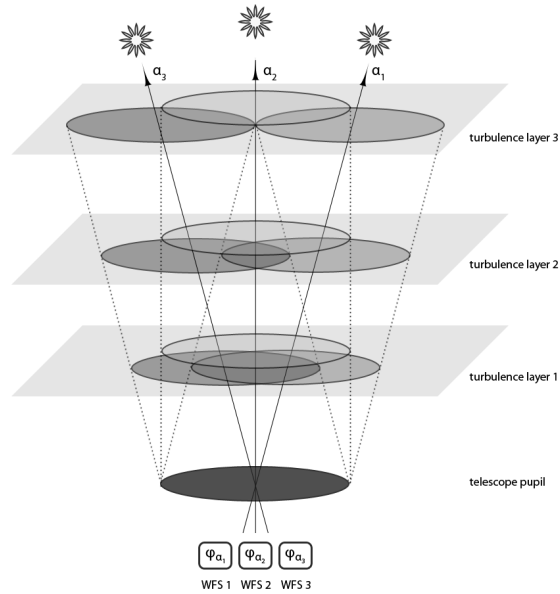


Figure 1.5: Atmospheric Tomography

The atmospheric tomography problem is defined by

$$s = (s_g)_{g=1}^G = A\phi, \quad (1.24)$$

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 Equation (1.21), and a geometric propagation operator P , defined by (1.1) and (1.4), in the direction of the guide star. This leads to the following equivalent formulation of Equation (1.24)

$$s_g = \Gamma_g P_g \phi \text{ for } g = 1, \dots, G. \quad (1.25)$$

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 (1.24)

$$\mathbf{S} = A\phi + \eta. \quad (1.26)$$

The optimal solution of Equation (1.26) is given by the maximum a-posteriori estimate (MAP), which is obtained by solving the linear system of equations

$$(A^T C_\eta^{-1} A + C_\phi^{-1})\phi = A^T C_\eta^{-1} s, \quad (1.27)$$

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.

1.2.1. Matrix Vector Multiplication

The standard approach to solve Equation (1.27) is called Matrix Vector Multiplication (MVM) [5], where the inverse of the discretized left-hand side matrix is computed explicitly by

$$R := (A^T C_\eta^{-1} A + C_\phi^{-1})^{-1} A^T C_\eta^{-1}. \quad (1.28)$$

and then multiplied with the sensor measurements. Typically, a mirror fitting operator F (as defined in Section 1.1.4) is combined with the atmospheric reconstruction, mapping sensor measurements onto mirror shapes

$$a = (FR)s. \quad (1.29)$$

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.

1.2.1.1. Algorithm

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 (1.21), (1.22) and (1.23).
 - The geometric propagation operator P which is defined by (1.4) for LGS and (1.1) for NGS.



2. Set up the inverse covariance matrix of noise C_η^{-1} by using Equation (1.2) and Equation (1.9) for NGS and LGS, respectively.
3. Set up the inverse covariance matrix of layers C_ϕ^{-1} by Equation (1.10) and Equation (1.11).
4. Calculate the reconstruction operator

$$R := (A^T C_\eta^{-1} A + C_\phi^{-1})^{-1} A^T C_\eta^{-1}. \quad (1.30)$$

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 1.1.4).
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

$$a = (FR)s. \quad (1.31)$$

1.3. Implementation and Computer Requirements

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 1.2.1.1 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 .

1.3.1. Building Code

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)
```

1.3.2. Libraries

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 (see Section 1.2.1.1).

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 (1.10).

1.4. Numerical Example

The numerical example we consider within our benchmark case uses LTAO (see 1.1.2.3 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 1.2.1.1 to deal with the problem of atmospheric tomography and, finally, obtain as output the actuator commands to control the deformable mirror.

1.4.1. 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 1.2.1.1. First, we set up the required matrices A , C_ϕ and C_η . Then we use the libraries described in Section 1.3 to perform matrix and vector operations and Cholesky Decomposition.

1.4.2. 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.



Part II.

Implementing acoustic scattering simulations for external geometries within a porous enclosure

Ashwin Nayak, Andrés Prieto, Daniel Fernández

Abstract

The details in implementing an acoustic scattering simulation of a rigid exterior domain enclosed in a porous layer are outlined. Details of the mathematical model used are highlighted alongside numerical procedures implemented in obtaining an approximate solution. An elaborate end-to-end strategy using open-source software tools to compute the solution is also provided. The developed tool is validated for a test case spherical geometry and the methodology to reproduce the same is thoroughly indicated.

Keywords: Scattering, aeroacoustics, porous materials.

2.1. Introduction

Sound sensors are generally housed in a casing and use porous enclosures to filter *noise* - a critical component in acoustic measurements. Understanding the transmission of sound through different media is key in designing and improving accuracy of an acoustic sensor. Of relevance in the project is the Microflow sensor, distinguished by their ability to measure both the intensity and direction of sound. The sensor is commercially available in a variety of housings and porous enclosures, suiting different acoustic environments. A computational model is sought to accurately predict the sound transmission in windy conditions in presence of porous enclosures. While the physics dictates a different sound transmission model in porous media and flow conditions, the model needs to couple these behaviors effectively to capture the combined influence on the incident signal. An earlier report[6] 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.

The implementation considered is 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.2.1a. 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.

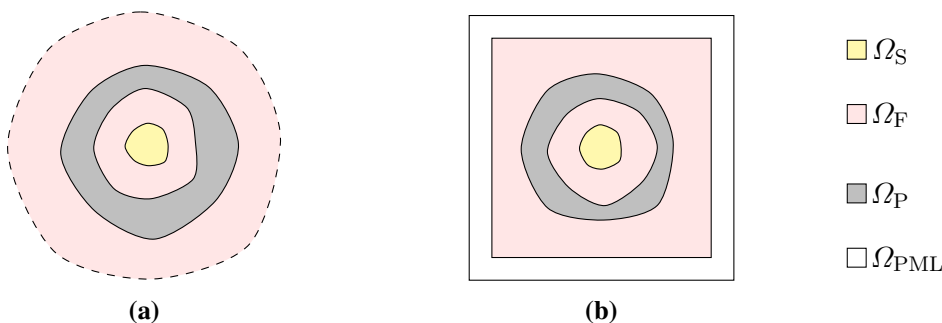


Figure 2.1: Schematic of the original problem configuration on unbounded domain (a), and the model configuration with perfectly matched layers (b).

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 [7]. In this particular implementation, the acoustic oscillations are chosen to be represented by the displacement vector field.

A series of assumptions are considered to arrive at a feasible mathematical model 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 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. Pragmatically, this is mimicked by a model configuration with a finite truncation of the domain and an artificial boundary enclosing it with absorbing properties, known in literature as the perfectly matched layers (PML) technique [8, 9]. It is represented by the Cartesian box Ω_{PML} in Fig.2.1b.

The mathematical formulation for the coupled problem may be surmised as the following system of equations: for a particular frequency, ω ,

$$-\nabla(\rho_{\text{F}}c_{\text{F}}^2 \operatorname{div} \mathbf{u}_{\text{F}}) - \rho_{\text{F}}\omega^2 \mathbf{u}_{\text{F}} = \mathbf{f}_{\text{F}} \quad \text{in } \Omega_{\text{F}}, \quad (2.1)$$

$$-\nabla(K_{\text{P}}(\omega) \operatorname{div} \mathbf{u}_{\text{P}}) - \rho_{\text{P}}(\omega)\omega^2 \mathbf{u}_{\text{P}} = \mathbf{f}_{\text{P}} \quad \text{in } \Omega_{\text{P}}, \quad (2.2)$$

$$-\operatorname{div}(\rho_{\text{F}}c_{\text{F}}^2 \tilde{\mathbf{C}}(\nabla \mathbf{u}_{\text{PML}})) - \rho_{\text{F}}\omega^2 \tilde{\mathbf{M}} \mathbf{u}_{\text{PML}} = \mathbf{f}_{\text{PML}} \quad \text{in } \Omega_{\text{PML}}, \quad (2.3)$$

$$\mathbf{u}_{\text{F}} \cdot \mathbf{n} = g \quad \text{on } \Gamma_{\text{S}}, \quad (2.4)$$

$$\mathbf{u}_{\text{F}} \cdot \mathbf{n} - \mathbf{u}_{\text{P}} \cdot \mathbf{n} = 0 \quad \text{on } \Gamma_{\text{C}}, \quad (2.5)$$

$$\rho_{\text{F}}c_{\text{F}}^2 \operatorname{div} \mathbf{u}_{\text{F}} - K_{\text{P}}(\omega) \operatorname{div} \mathbf{u}_{\text{P}} = 0 \quad \text{on } \Gamma_{\text{C}}, \quad (2.6)$$

$$\mathbf{u}_{\text{F}} \cdot \mathbf{n} - \mathbf{u}_{\text{PML}} \cdot \mathbf{n} = 0 \quad \text{on } \Gamma_{\text{PML}}, \quad (2.7)$$

$$\operatorname{div} \mathbf{u}_{\text{F}} - \operatorname{div} \mathbf{u}_{\text{P}} = 0 \quad \text{on } \Gamma_{\text{PML}}. \quad (2.8)$$

Here, \mathbf{u}_{F} , \mathbf{u}_{P} and \mathbf{u}_{PML} are the displacement vector fields in the fluid, porous and PML domains respectively. Γ_{S} , Γ_{C} and Γ_{PML} represent the boundaries making up the interfaces between structure-fluid, fluid-porous and fluid-PML domains with outward facing normals, \mathbf{n} . The model includes material properties like fluid mass density ρ_{F} , sound speed in the fluid c_{F} , the dynamic porous mass density ρ_{P} and the dynamic porous bulk modulus K_{P} . Equations (2.1)-(2.3) represent the Helmholtz-like equations in each of the domains. Equation (2.4) is a boundary condition at the object boundary and (2.5)-(2.8) represent the pressure and displacement continuity conditions on the interfaces. The source-terms \mathbf{f}_{F} , \mathbf{f}_{P} , \mathbf{f}_{PML} and function g appear according to initial sources of disturbances and are explained later in this document.

The porous material properties 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 [9, 10]. 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,

$$\rho_{\text{P}}(\omega) = \frac{\rho_{\text{F}}}{\phi} \alpha_{\infty} \left(1 - i \frac{\sigma \phi}{\omega \rho_{\text{F}} \alpha_{\infty}} \sqrt{1 + i \frac{4\alpha_{\infty}^2 \eta \rho_{\text{F}} \omega}{\sigma^2 \Lambda^2 \phi^2}} \right), \quad (2.9)$$

$$K_{\text{P}}(\omega) = \frac{\gamma P_{\text{F}} / \phi}{\gamma - (\gamma - 1) \left(1 - i \frac{\eta \phi}{\rho_{\text{F}} k_0' \omega \text{Pr}} \sqrt{1 + i \frac{4k_0'^2 \rho_{\text{F}} \omega \text{Pr}}{\eta \Lambda'^2 \phi^2}} \right)^{-1}}. \quad (2.10)$$

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 model also requires the fluid state properties like density ρ_F , specific heat ratio γ , Prandtl Number Pr , and equilibrium fluid pressure P_F .

The Helmholtz-like PML governing Equation (2.3) ensures absorption of outgoing waves. This is achieved by a complex stretching of spatial variables[11] by the fourth-order tensor $\tilde{\mathbf{C}}$ and the second-order tensor $\tilde{\mathbf{M}}$ given by,

$$\tilde{\mathbf{C}}(\nabla \mathbf{w}) = \left(\sum_{j=1}^3 \frac{1}{\gamma_j} \frac{\partial w_j}{\partial x_j} \right) \mathbf{I} \quad (2.11)$$

$$\text{and } \tilde{\mathbf{M}} = \sum_{j=1}^3 \gamma_j \mathbf{e}_j \otimes \mathbf{e}_j, \quad (2.12)$$

where, \mathbf{I} is the fourth-order identity tensor and \mathbf{e}_j 's are the unit vectors along the spatial directions. The optimally-tuned functions provided by Bermudez et al.[12] are chosen among the various choices for the complex stretching functions γ_j 's, giving,

$$\gamma_j(x_j) = \begin{cases} 1 & |x_j| \leq L_j, \\ 1 + i \frac{c_F}{\omega(L_j^\infty - |x_j|)} & L_j \leq |x_j| \leq L_j^\infty. \end{cases} \quad (2.13)$$

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 definition of γ_j as a piece-wise function ensures the absorption of waves only along the outward direction of propagation. Consequently, the tensors $\tilde{\mathbf{C}}$ and $\tilde{\mathbf{M}}$ are piece-wise and needs to be considered with care during the implementation.

The model described in Equations (2.1)-(2.8) explain the propagation of a generic acoustic field and needs adaptation for our initial problem of computing the acoustic scattering of an incident wave. The total normal displacement at the object boundary are zero ($g = 0$) since the structure is assumed rigid. The principle of superposition may then be utilized to split the total field into the incident field and scattered field components. The equations are then rewritten in terms of the scattered part of the field to obtain right-hand-sides, some of which are non-null.

Considering that the displacement vector fields, \mathbf{u}_F , \mathbf{u}_P and \mathbf{u}_{PML} , are defined in exclusive domains albeit with different smoothing requirements, it may be unified to be a member of a functional space \mathbf{V} introduced as,

$$\mathbf{V} = \left\{ \mathbf{v} \in [\mathbf{L}^2(\Omega)]^3 : \mathbf{v}|_{\Omega_F} \in \mathbf{H}(\text{div}, \Omega_F), \mathbf{v}|_{\Omega_P} \in \mathbf{H}(\text{div}, \Omega_P), \tilde{\mathbf{M}}\mathbf{v}|_{\Omega_{PML}} \in [\mathbf{L}^2(\Omega_{PML})]^3, \right. \\ \left. \sum_{j=1}^3 \frac{1}{\gamma_j} \frac{\partial v_j}{\partial x_j} \Big|_{\mathbf{v} \in \Omega_{PML}} \in \mathbf{L}^2(\Omega_{PML}), \mathbf{v} \cdot \mathbf{n} = 0 \text{ on } \Gamma_\infty \right\}, \quad (2.14)$$

which also ensures the necessary continuity and differentiable properties at the interfaces. The variational form can then be deduced from Equations (2.1)-(2.3) by multiplying a test function $\mathbf{v} \in \mathbf{V}$ and utilizing the Green's



theorem : Find $\mathbf{u} \in \mathbf{V}$ such that,

$$\begin{aligned}
& \int_{\Omega_F} \rho_F c^2 (\operatorname{div} \mathbf{u})(\operatorname{div} \mathbf{v}) \, dV - \int_{\Omega_F} \rho_F \omega^2 \mathbf{u} \cdot \mathbf{v} \, dV \\
& + \int_{\Omega_P} K_P(\omega) (\operatorname{div} \mathbf{u})(\operatorname{div} \mathbf{v}) \, dV - \int_{\Omega_P} \rho_P(\omega) \omega^2 \mathbf{u} \cdot \mathbf{v} \, dV \\
& + \int_{\Omega_{\text{PML}}} \rho_F c^2 \tilde{\mathcal{C}}(\nabla \mathbf{u}) : \nabla \mathbf{v} \, dV - \int_{\Omega_{\text{PML}}} \rho_F \omega^2 \tilde{\mathcal{M}} \mathbf{u} \cdot \mathbf{v} \, dV = \int_{\Omega_F} \mathbf{f}_F \cdot \mathbf{v} \, dV + \int_{\Omega_P} \mathbf{f}_P \cdot \mathbf{v} \, dV \\
& \qquad \qquad \qquad + \int_{\Omega_{\text{PML}}} \mathbf{f}_{\text{PML}} \cdot \mathbf{v} \, dV \qquad (2.15)
\end{aligned}$$

holds for all $\mathbf{v} \in \mathbf{V}$ and also, $\mathbf{v} = 0$ at Γ_S . The Equation (2.15) 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,

$$\mathcal{A}(\mathbf{u}, \mathbf{v}) = \mathcal{L}(\mathbf{v}). \qquad (2.16)$$

A practical implementation of this model would require the approximation of an 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 (2.16) as,

$$\sum_{r=1}^n \mathcal{A}(\psi_r, \psi_s) \mu_r = \mathcal{L}(\psi_s) \quad \text{for } s = 1, 2, \dots, n; \qquad (2.17)$$

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

2.2. Implementation

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.2.2. 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. 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 would ease the workflow and be capable of handling problems of larger order.



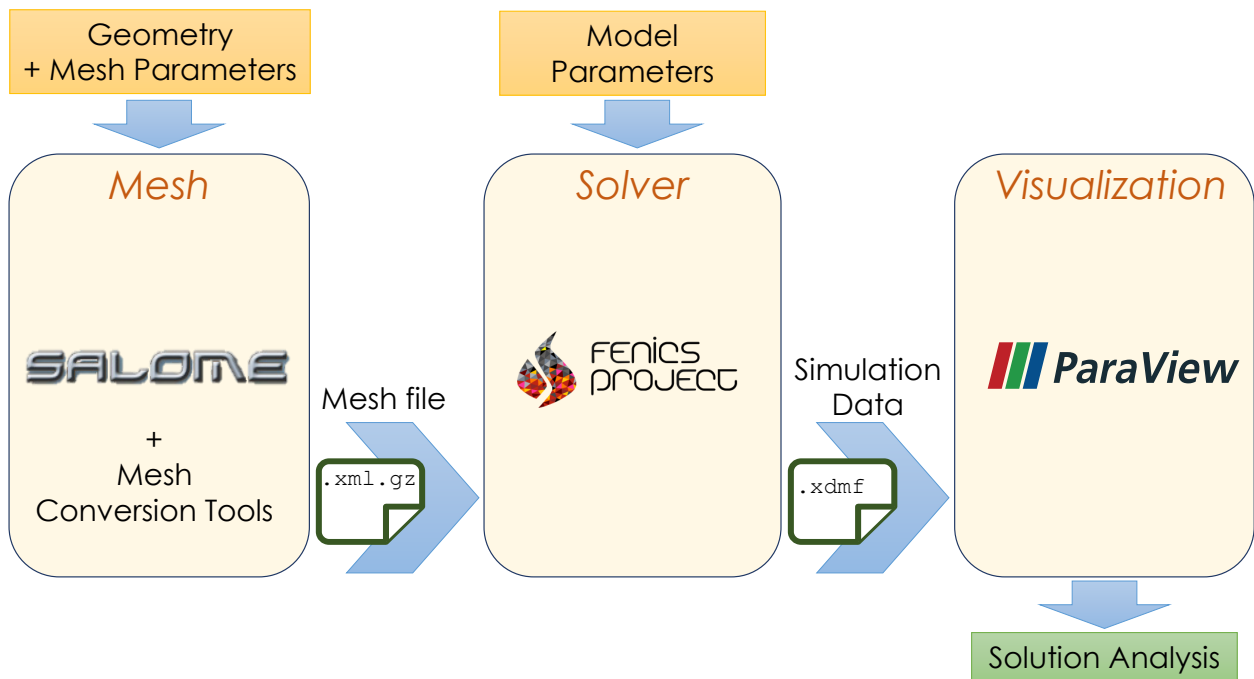


Figure 2.2: Workflow representing implementation stages and their interfaces.

2.2.1. 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 explain 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 entirely by using the GUI which allows for exporting the equivalent state in a TUI script.

The TUI provides `geomBuilder`, a Python module for the creating and editing geometry. An instance of the `geomBuilder` class 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 lists some useful TUI commands available as function attributes, whereas a detailed description along with other functions are available in the documentation [13].

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

Table 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
```

2.2.2. 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 optimized and parallel by design and it is easy to deploy and scale the code into high-performance computing clusters. With its Python and C++ interfaces, FEniCS offers powerful capabilities to integrate into workflows.

The FEniCS library offers a number of component modules and the interfacing is done mainly through its DOLFIN and UFL modules. DOLFIN is the highly optimized computational back-end written in C++ responsible for finite-element machinery. 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. 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 UFL 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.[14] and at the official FEniCS documentation webpage[15]. It is to be noted that the library 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.

2.2.3. 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[16]. 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[17] and many tutorials[18] highlighting the usage and relevance of each of the stages along with the available functionalities to fully exploit its potential.

2.3. 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 in illustrating the use of the described tools. The solution for the case can also be computed analytically.

2.3.1. 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.



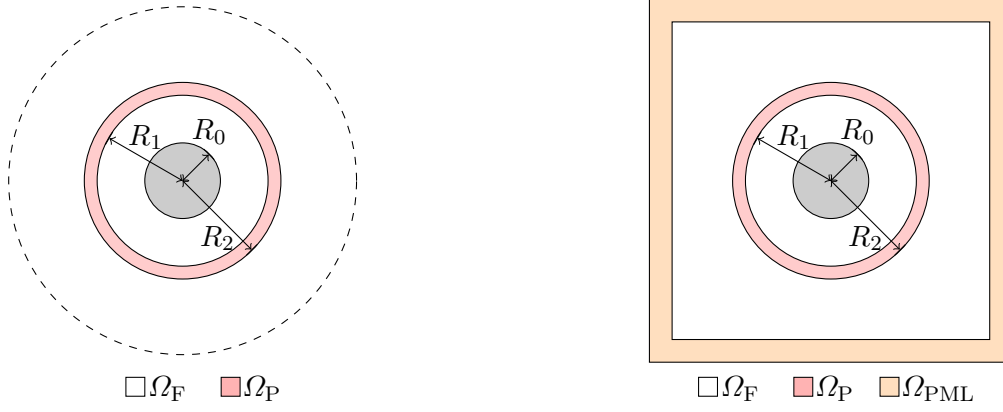


Figure 2.3: Schematic of the spherical test case posed in an unbounded domain (left) and modeled using perfectly matched layers (right).

2.3.2. Exact solution

The exact solution may be obtained by enforcing a constant Neumann boundary condition on the surface of the sphere. The Helmholtz equation then only has 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,

$$p(r) = \begin{cases} A_1 \frac{e^{-ik_F r}}{r} + B_1 \frac{e^{ik_F r}}{r}, & r \in [R_0, R_1], \\ A_2 \frac{e^{-ik_P r}}{r} + B_2 \frac{e^{ik_P r}}{r}, & r \in [R_1, R_2], \\ B_3 \frac{e^{ik_F r}}{r}, & r \in [R_2, \infty). \end{cases} \quad (2.18)$$

where 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,

$$\frac{\partial p}{\partial r}(R_0) = \rho_F \omega^2 g_0, \quad g_0 = \text{constant}, \quad (2.19)$$

$$p(R_1^-) = p(R_1^+), \quad (2.20)$$

$$\rho_P \frac{\partial p}{\partial r}(R_1^-) = \rho_F \frac{\partial p}{\partial r}(R_1^+), \quad (2.21)$$

$$p(R_2^-) = p(R_2^+), \quad (2.22)$$

$$\rho_F \frac{\partial p}{\partial r}(R_2^-) = \rho_P \frac{\partial p}{\partial r}(R_2^+). \quad (2.23)$$

Substituting the general solution (2.18) in (2.19)-(2.23) and rewriting in terms of the coefficients, A_i 's and B_i 's, a 5×5 system of equations is obtained and is solved to compute the exact solution.

2.3.3. Geometry and Mesh

SALOME's batch scripting 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 which performs 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 (2.15) 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 2.4a shows a cross-section of the generated geometry.

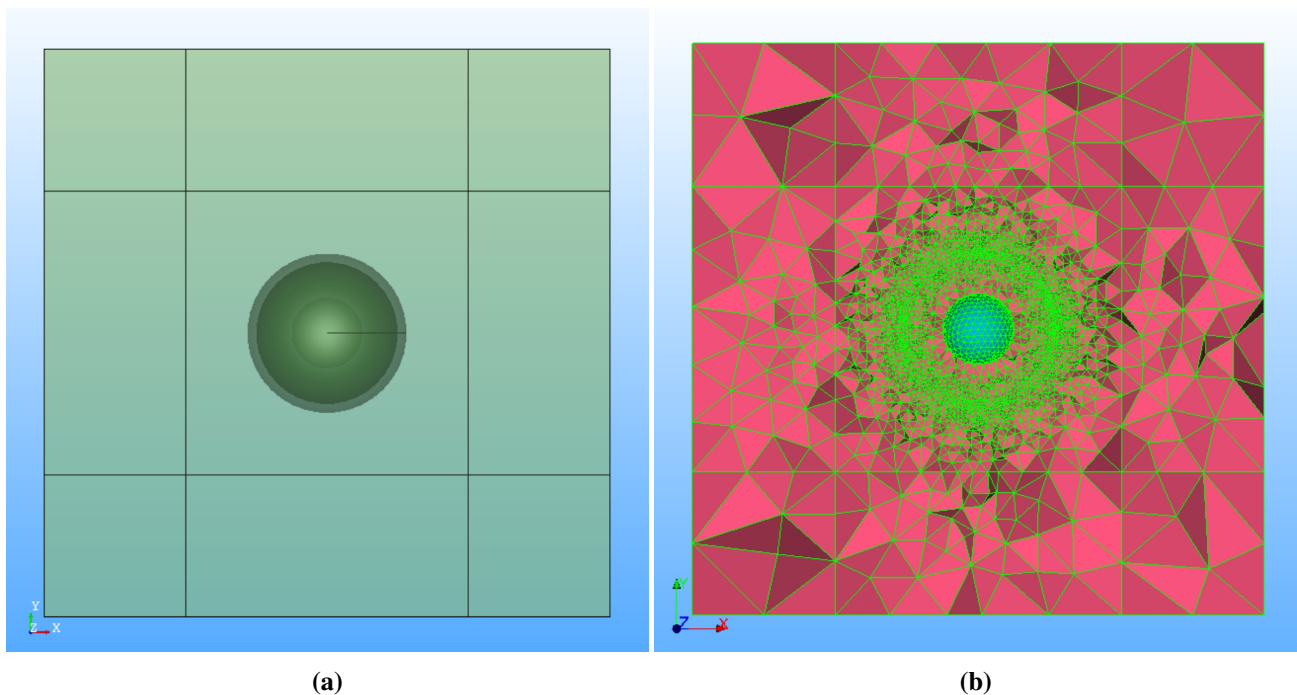


Figure 2.4: View of the geometry(left) and the cross-section of the Mesh(right).

The meshing operation is then carried out using the `smesh` module of SALOME as was similarly suggested in 2.2.1 . Fig.2.4b illustrates 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` [19] and `dolfin-convert` (provided by FEniCS) utilities as,

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

2.3.4. 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 (2.15) in real and imaginary portions. This may be achieved easily by splitting Equations (2.1)-(2.8) into real and imaginary parts and following in the



same process to achieve a similar variational form like (2.15). The wave equations being linear help separate each of the term into two,

$$\begin{aligned}
 & \int_{\Omega_F} \rho_F c^2 (\operatorname{div} \operatorname{Re}(\mathbf{u})) (\operatorname{div} \operatorname{Re}(\mathbf{v})) \, dV - \int_{\Omega_F} \rho_F \omega^2 \operatorname{Re}(\mathbf{u}) \cdot \operatorname{Re}(\mathbf{v}) \, dV \\
 & \int_{\Omega_F} \rho_F c^2 (\operatorname{div} \operatorname{Im}(\mathbf{u})) (\operatorname{div} \operatorname{Im}(\mathbf{v})) \, dV - \int_{\Omega_F} \rho_F \omega^2 \operatorname{Im}(\mathbf{u}) \cdot \operatorname{Im}(\mathbf{v}) \, dV \\
 + & \int_{\Omega_P} [\operatorname{K}_P(\omega) (\operatorname{div} \operatorname{Re}(\mathbf{u}))] (\operatorname{div} \operatorname{Re}(\mathbf{v})) \, dV - \int_{\Omega_P} [\rho_P(\omega) \omega^2 \operatorname{Re}(\mathbf{u})] \cdot \operatorname{Re}(\mathbf{v}) \, dV \\
 & + \int_{\Omega_P} \operatorname{Im}(\operatorname{K}_P(\omega) (\operatorname{div} \mathbf{u})) (\operatorname{div} \operatorname{Im}(\mathbf{v})) \, dV - \int_{\Omega_P} \operatorname{Im}(\rho_P(\omega) \omega^2 \mathbf{u}) \cdot \operatorname{Im}(\mathbf{v}) \, dV \\
 + & \int_{\Omega_{\text{PML}}} \rho_F c^2 \operatorname{Re}(\tilde{\mathbf{C}}(\nabla \mathbf{u})) : \nabla \operatorname{Re}(\mathbf{v}) \, dV - \int_{\Omega_{\text{PML}}} \rho_F \omega^2 \operatorname{Re}(\tilde{\mathbf{M}} \mathbf{u}) \cdot \operatorname{Re}(\mathbf{v}) \, dV \\
 + & \int_{\Omega_{\text{PML}}} \rho_F c^2 \operatorname{Im}(\tilde{\mathbf{C}}(\nabla \mathbf{u})) : \nabla \operatorname{Im}(\mathbf{v}) \, dV - \int_{\Omega_{\text{PML}}} \rho_F \omega^2 \operatorname{Im}(\tilde{\mathbf{M}} \mathbf{u}) \cdot \operatorname{Im}(\mathbf{v}) \, dV = \mathbf{0}. \quad (2.24)
 \end{aligned}$$

Note that the right-hand term is null since there are no source terms in the configuration. The Equation (2.24) 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 [20], 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
s = lambda j : conditional(gt(abs(x[j]), L[j] + tol),
                          c / abs(abs(x[j]) - (L[j] + Lpml)) / omega,
                          Constant("0."))

gamma = lambda j : Complex(Constant("1.0"), s(j))
du_dx = lambda ure, uim, i: 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(ure, uim, i))
             for i in range(3))
Div_im = sum(prod_im(Complex(1., 0.) / gamma(i), du_dx(ure, uim, i))
             for i in range(3))

# # Scaled PML displacement vector to be used in the mass matrix
coef = lambda ure, uim, i: Complex(ure[i], uim[i])
u_scaled_re = as_vector([prod_re(gamma(i), coef(ure, uim, i))
                        for i in range(3)])
u_scaled_im = as_vector([prod_im(gamma(i), coef(ure, uim, 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 (2.9) and (2.10) 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.

2.3.5. Visualization

The saved XDMF file are directly compatible and can be easily visualized using ParaView. The user interface is very intuitive and once the file is opened it allows the user to select the solution fields to import from the dataset 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. 2.5, 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. Fig. 2.6 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.

2.4. 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.



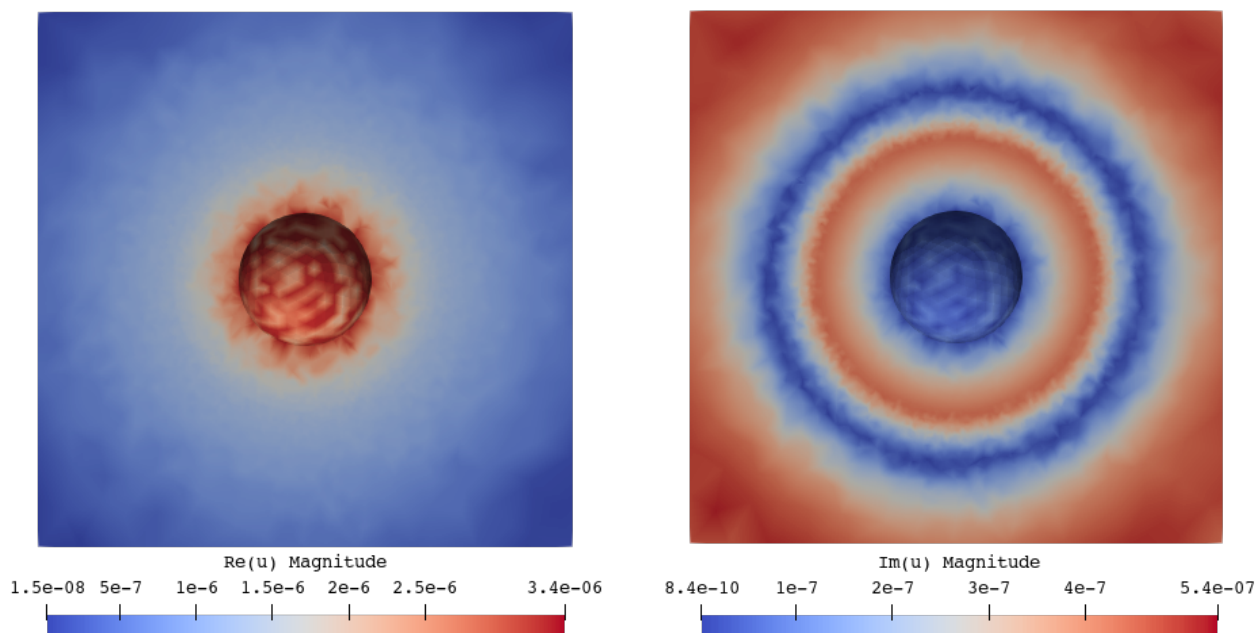


Figure 2.5: Cross-sectional contour views of the magnitude of the real part (left) and the magnitude of the imaginary part (right) of the computed displacement field.

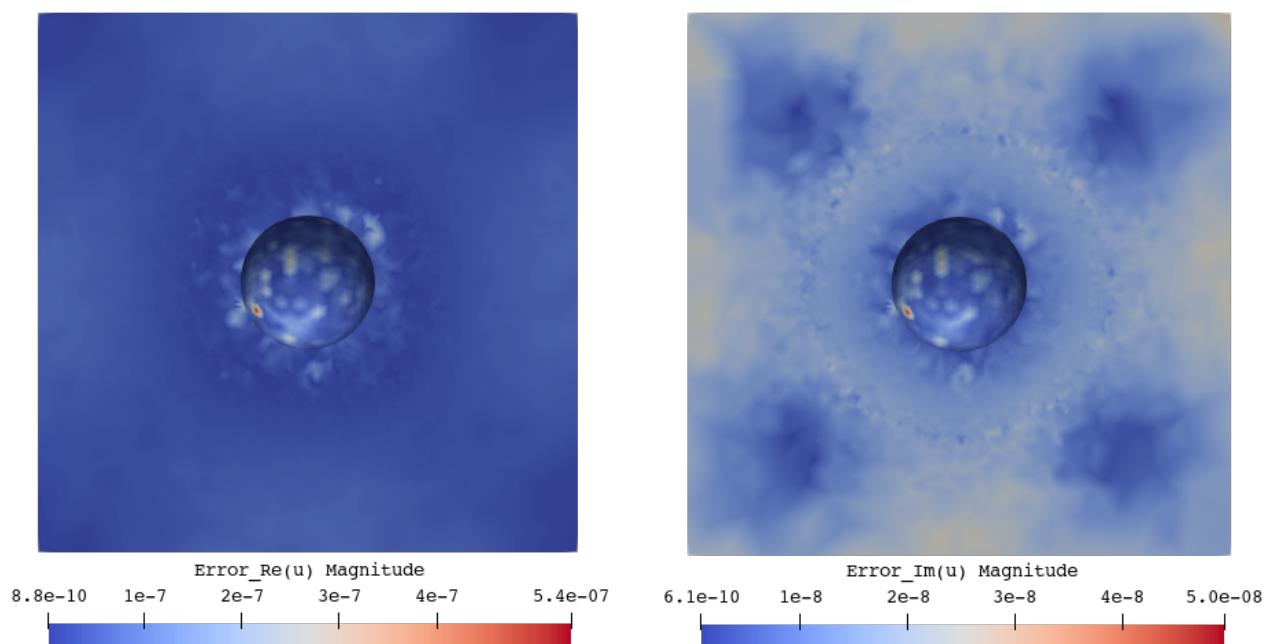


Figure 2.6: Cross-sectional contour views of the errors in the magnitude of the real part (left) and magnitude of the imaginary part (right) of the displacement field.

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 solver stage with the intricate finite element machinery is handled by the FEniCS library offering ease-of-use to the user while focusing their attention towards the development and prototyping of the model. The visualization tool, ParaVIEW is feature-rich providing access to many post-processing algorithms along with an intuitive inter-

face. 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 of running a simulation and obtaining and visualizing results, and hopes to enable the reader to reproduce the results in its entirety.



Part III.

Software implementation description for Point Source Far field Reflector computation using Sinkhorn algorithm

Jean-David Benamou, Wilbert IJzerman, Giorgi Rukhaia

Abstract

This work describes framework where software for Point Source Far field Reflector computation will be implemented using Sinkhorn algorithm and its modifications. It works in C++ environment, using C based memory containers and Intel's MKL libraries.

Keywords: FreeForm Optics, Software framework, C++ implementation, Sinkhorn algorithm, MKL libraries.

3.1. Introduction

This software finds solution for far field reflector problem with a point source. The model consists of a point source of light at the origin \mathcal{O} of \mathbf{R}^3 , a reflecting surface Γ which is a radial graph over $\Omega \subset \mathbf{S}^2$ in the northern hemisphere,

$$\Gamma = \{x\rho(x); x \in \Omega\} \quad \rho > 0 \quad (3.1)$$

and a target area which is far enough from reflector that reflected rays can be identified with their direction, i.e the domain $\Omega^* \subset \mathbf{S}^2$ represents reflected rays in terms of their reflection direction.

Software uses Sinkhorn's algorithm (described in corresponding Benchmark) with several modifications to receive solution surface.

By changing test case file described below, this software solves all 3 corresponding benchmark cases.

3.2. Implementation

Software is written in C++, but most of computational data structures are C-style fixed size Arrays and most of 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 mostly 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 allow redistribution. Commercial support is available when purchased as a 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 development of method, code doesn't follow standard suggestions for C++ code development.

It consists of two files:

One is test case file, where data structures for discretization and density cloud are declared, together with corresponding functions filling them. Also where functions describing source and target density are declared and can be changed manually to receive different test cases.

Second file is main file, where, together with main function, data structures and solving functions, which are independent of test case except dimension and total number of points in discretization, are declared and



implemented.

3.3. Computer requirements

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.

Code is completely 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 main function.

Due to use of Intel's MKL library, software will be much more efficient when run on Intel processor, compared to other processors of same power. Other than 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 be capable of handling such precision.

To compile software, one should link both code files and intel's MKL library file according to instructions from Intel.



Part IV.

Data driven model adaptations of coil sensitivities in magnetic particle imaging

Soren Dittmer, José Carlos Gutiérrez Pérez, Lena Hauberg-Lotte, Tobias Kluth, Peter Maass, Daniel Otero Baguer

Abstract

In Section 4.1 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 4.2 the implementation as well as the underlying network is explained in detail. The computer requirements are stated (Section 4.3) 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 4.4).

Keywords: Deep Learning, neural networks, inverse problems, deep image priors.

4.1. 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 [21]. The usability of a circulating tracer for long-term monitoring was recently investigated [22]. The high temporal resolution of MPI is advantageous for potential flow estimation [23] and for tracking medical instruments [24]. Recently, MPI was also shown to be suitable for tracking and guiding instruments for angioplasty [25]. Further promising applications of MPI include cancer detection [26] and cancer treatment by hyperthermia [27].

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 [28]. 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 [29]. Furthermore, deep learning was used in medical applications such as tumour classification with MALDI Imaging [30], magnetic resonance imaging (MRI) [31] [32], low-dose X-ray CT [33] as well as positron emission tomography (PET) [34].

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 forMPI.

4.1.1. 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 [35]: 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) [36]). 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 [37]) 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 [38]) allows for the description via a linear integral operator with respect to the particle concentration:

$$u^P(t) = -\mu_0 \int_{\Omega} \mathbf{p}^R(x) \cdot \frac{\partial}{\partial t} \mathbf{M}(x, t) dx = \int_{\Omega} c(x) \underbrace{(-\mu_0 \mathbf{p}^R(x) \cdot \frac{\partial}{\partial t} \bar{\mathbf{m}}(x, t))}_{=s(x,t)} dx. \quad (4.1)$$

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.

4.1.1.1. 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

$$\tilde{S}_{D1}\tilde{c} = \tilde{v} - \tilde{v}^{(0)}, \quad \tilde{S}_{D1} = [\tilde{v}^{(1)} - \tilde{v}^{(0)} | \dots | \tilde{v}^{(N)} - \tilde{v}^{(0)}] \quad (4.2)$$

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

$$\langle S\tilde{c}, \psi_j \rangle = \langle v - v^{(0)}, \psi_j \rangle, \quad j \in \mathbb{N} \setminus J, \quad S = [v^{(1)} - v^{(0)} | \dots | v^{(N)} - v^{(0)}] \quad (4.3)$$

Exploiting the sparsity assumption on $v^{(0)}$ and using the finite-dimensional approximation in Y_M yields the final problem

$$\begin{aligned} \langle \tilde{S}_{D2}\tilde{c}, (\langle \phi_i, \psi_j \rangle)_{i=1, \dots, M} \rangle &= \langle \tilde{v}, (\langle \phi_i, \psi_j \rangle)_{i=1, \dots, M} \rangle, \quad j \in \mathbb{N} \setminus J, \\ \tilde{S}_{D2} &= [\tilde{v}^{(1)} | \dots | \tilde{v}^{(N)}] \end{aligned} \quad (4.4)$$

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.:

$$\bar{\mathbf{m}}(x, t) = m_0 \mathcal{L}_\beta(|\mathbf{H}(x, t)|) \frac{\mathbf{H}(x, t)}{|\mathbf{H}(x, t)|} \quad (4.5)$$

where $\mathcal{L}_\beta : \mathbb{R} \rightarrow \mathbb{R}$ is given in terms of the Langevin function by the following:

$$\mathcal{L}_\beta(z) = \left(\coth(\beta z) - \frac{1}{\beta z} \right) \quad (4.6)$$

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:

$$\begin{cases} v(t) = - \int_0^T \int_\Omega c(x) \tilde{a}_k(t - t') s_k(x, t') dx dt' \\ s = \mu_0 m_0 (\mathbf{p}^R)^T \frac{\partial}{\partial t} \left(\mathcal{L}_\beta(|\mathbf{H}|) \frac{\mathbf{H}}{|\mathbf{H}|} \right) \end{cases} \quad (4.7)$$

$I_3 \in \mathbb{R}^{3 \times 3}$ being the identity matrix. The *equilibrium model* in (4.7) can be extended to polydisperse tracers by adapting the function defining the length of the mean magnetic moment vector in (4.6). 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:

$$\begin{cases} v(t) = - \int_0^T \int_{\Omega} c(x) \tilde{a}(t-t') s(x, t') dx dt' \\ s = \mu_0 (\mathbf{p}^R)^T \frac{\partial}{\partial t} \left(L_{\rho}(|\mathbf{H}|) \frac{\mathbf{H}}{|\mathbf{H}|} \right) \end{cases} \quad (4.8)$$

where $L_{\rho} : \mathbb{R} \rightarrow \mathbb{R}$ is given in terms of the Langevin function by

$$L_{\rho}(z) = \int_{\mathbb{R}^+} \rho(D) m_0(D) \mathcal{L}_{\beta(D)}(z) dD \quad (4.9)$$

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 over-simplified 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.

4.1.2. 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.

- $p_X : X \rightarrow \mathbb{R}_{\geq 0}$ the probability density function of the elements/images/concentrations that we want to reconstruct.
- $p_Y : Y \rightarrow \mathbb{R}_{\geq 0}$ the probability density function of the elements/signals/voltages that we are measuring.
- $p_{XY} : X \times Y \rightarrow \mathbb{R}_{\geq 0}$ the joint probability density function.

4.1.2.1. A: Learned Penalty Terms

In variational approaches to Inverse Problems one usually incorporates a so-called penalty function

$$\phi : X \rightarrow \mathbb{R}_{\geq 0}$$

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 [39] [40].

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 $\phi_\theta : X \rightarrow \mathbb{R}_{\geq 0}$ (for example represented by a neural network).

- **Intrinsically required knowledge:**

Ground truth data about p_X e. g. in the form of many samples.

4.1.2.2. 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 [41], 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 [42, 43].

- **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.

4.1.2.3. 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 [44] and the “Learning to learn by gradient descent by gradient descent” paper [45]. Despite the fact that the paper [45] 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 [45] directly to solve Inverse Problems [46].

The basic idea of this method has been 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 [47, 48, 49, 50, 51, 52].

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.

4.1.2.4. 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

$$F_\theta : G \rightarrow \mathbb{R}_{\geq 0}$$

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 [53]. Both works are inspired by a recent paper [54] 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 [54] 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 [49].

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 [53], 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.

4.1.2.5. 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 [55, 56, 57, 58, 59, 60, 61].

4.1.3. 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.

4.2. 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 [62], specifically their U-net architecture. Our implementation is based on Tensorflow [63] and Keras [64] and has the following specifications: Between the encoder and decoder part of the U-net our skip connection 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 layers 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 [62].

Since Tensorflow does not support auto gradients for complex numbers, we split up our **loss function** into the



form

$$\|A\phi_W(z) - y^\delta\|^2 = \|\operatorname{real}(A)\phi_W(z) - \operatorname{real}(y^\delta)\|^2 \quad (4.10)$$

$$+ \|\operatorname{imag}(A)\phi_W(z) - \operatorname{imag}(y^\delta)\|^2, \quad (4.11)$$

where `real` and `imag` denote the real and imaginary parts respectively. If nothing else is said we used Adam [65] 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.

4.3. 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.

4.4. Numerical examples

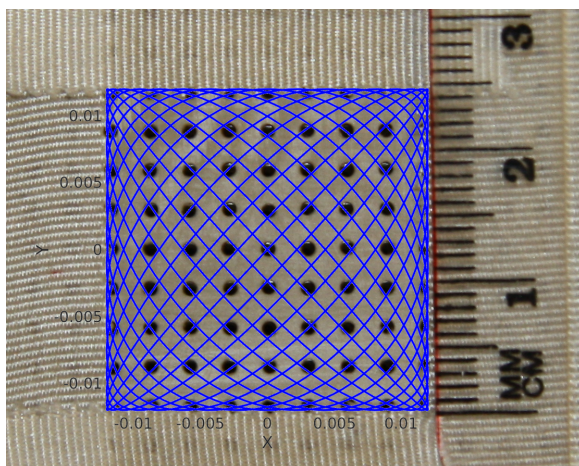


Figure 4.1: Experimental platform used to obtain dataset DS1 with FFP Lissajous trajectory in blue. Photo taken at University Medical Center Hamburg-Eppendorf by T. Kluth.

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 4.1. 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 2 for an illustration.
- “2mm”: Like the “4mm” phantom with 2 mm distance in x-direction between the glass capillary. See also Table 2 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 [66]. 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 × 2 mm × 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 × 38 mm × 1 mm and the sample positions have a distance of 2 mm in x- and y-direction resulting in a size of 19 × 19 × 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 [66] we have the following three phantoms:

- “concentration”: This phantom consists of 8 cubes of 2mm edge length resulting in 8μl 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 2 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 μL. Perimag[®] tracer with a concentration of 0.05 mol/L is used. See also Table 2 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 2 for an illustration.

All data is provided in the Magnetic Particle Imaging Data Format Files (MDF) encoded according to [67, 68].

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 $\{v_{\ell,0}^{(k)}\}_{k=1}^K$ [69]:

$$d_{\ell,j} = \frac{\frac{1}{N} \sum_{i=1}^N |\langle v_\ell^{(i)} - \mu_\ell^{(i)}, \psi_j \rangle|}{\frac{1}{K} \sum_{k=1}^K |\langle v_{\ell,0}^{(k)} - \mu_\ell, \psi_j \rangle|} \quad (4.12)$$



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

$$J_\ell = \{j \in I_{\text{BP}} | d_{\ell,j} \geq \tau\} \quad (4.13)$$

for $\ell = 1, 2, 3$.

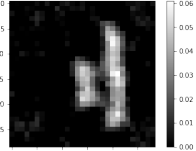
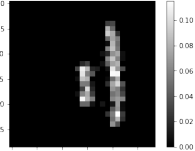
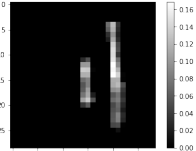
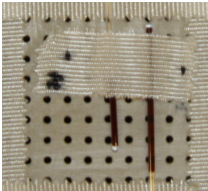
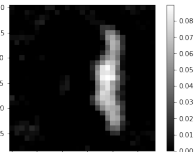
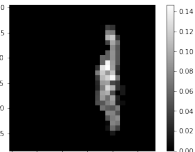
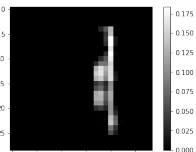
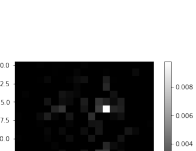


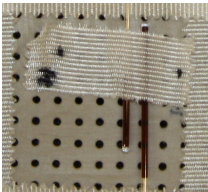
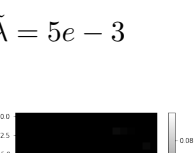
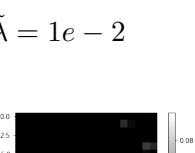
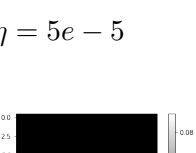
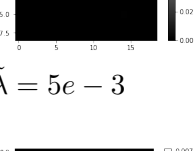
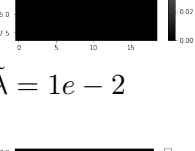
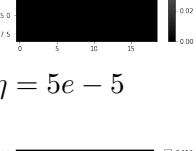
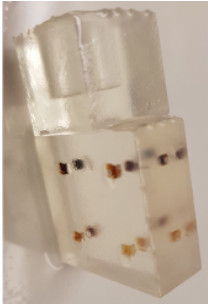
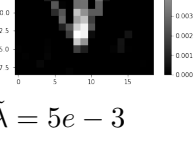
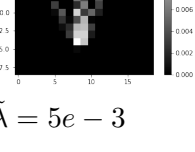
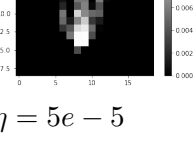



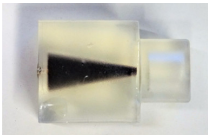



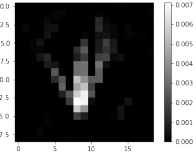
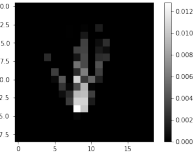
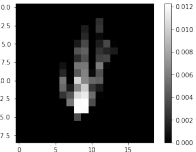

We will now describe the general setup we use to apply the deep inversion prior approach to the reconstruction of 2 dimensional magnetic particle imaging data. We do the **processing** of the data in the following manner:

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 $N = 19^2 = 361$.



Table 2: 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 [66].

Phantom	Kaczmarz with ℓ_2	ℓ_1	DIP	Photo
“4mm” (DS1)	 $\tilde{\lambda} = 5e - 4$	 $\tilde{\lambda} = 5e - 3$	 $\eta = 5e - 5$	
	 $\tilde{\lambda} = 5e - 4$	 $\tilde{\lambda} = 5e - 3$	 $\eta = 5e - 5$	
“2mm” (DS1)	 $\tilde{\lambda} = 5e - 4$	 $\tilde{\lambda} = 5e - 3$	 $\eta = 5e - 5$	
	 $\tilde{\lambda} = 5e - 3$	 $\tilde{\lambda} = 1e - 2$	 $\eta = 5e - 5$	
“concentration” (DS2)	 $\tilde{\lambda} = 5e - 3$	 $\tilde{\lambda} = 1e - 2$	 $\eta = 5e - 5$	
	 $\tilde{\lambda} = 5e - 3$	 $\tilde{\lambda} = 5e - 3$	 $\eta = 5e - 5$	
“shape” (DS2)	 $\tilde{\lambda} = 5e - 3$	 $\tilde{\lambda} = 1e - 2$	 $\eta = 5e - 5$	
	 $\tilde{\lambda} = 5e - 3$	 $\tilde{\lambda} = 5e - 3$	 $\eta = 5e - 5$	
“resolution” (DS2)	 $\tilde{\lambda} = 5e - 3$	 $\tilde{\lambda} = 5e - 3$	 $\eta = 5e - 5$	

Part V.

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

M.W.F.M. Bannenberg, M. Günther

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.

5.1. 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.

5.1.1. 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 [70] is used, Figure 5.1.

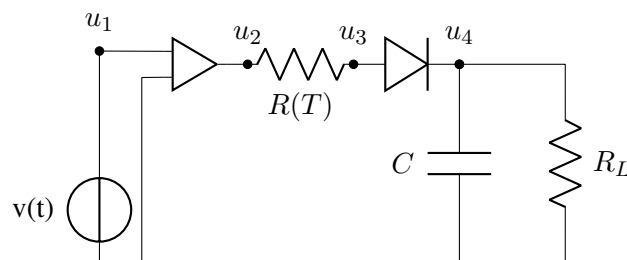


Figure 5.1: Electric description of the benchmark circuit.

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:

$$\begin{aligned} 0 &= (Av(t) - u_3)/R(T) + i_{di}(u_3 - u_4, T_{di}), \\ C\dot{u}_4 &= i_{di}(u_3 - u_4, T_{di}) - u_4/R_L, \end{aligned}$$

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:

$$\begin{aligned} M'_{w,i} h \dot{T}_i &= \Lambda \frac{T_{i+1} - 2T_i + T_{i-1}}{h} + P_w \frac{\tilde{\rho}(X_i, T_i)}{R} h, \\ &\quad - \gamma S'_{w,i} h (T_i - T_{env}), \quad (i = 1, \dots, N-1), \\ (M'_{w,0} \cdot \frac{h}{2} + M_{op}) \dot{T}_0 &= \Lambda \frac{T_1 - T_0}{h} + P_w \frac{\tilde{\rho}(0, T_0)}{R} \frac{h}{2} \\ &\quad - \gamma (S'_{w,0} \frac{h}{2} + S_{op}) \cdot (T_0 - T_{env}) + P_{op}, \\ (M'_{w,N} \cdot \frac{h}{2} + M_{di}) \dot{T}_N &= \Lambda \frac{T_{N-1} - T_N}{h} + P_w \frac{\tilde{\rho}(X_N, T_N)}{R} \frac{h}{2} \\ &\quad - \gamma (S'_{w,N} \frac{h}{2} + S_{di}) \cdot (T_N - T_{env}) \end{aligned}$$

5.1.2. Multirate time integration

To efficiently integrate the thermal-electric system through time we use a multirate scheme, [71]. First consider a more general notation of the thermal-electric system:

$$\begin{aligned} M_e \dot{\mathbf{u}} &= f_e(t, \mathbf{u}, \mathbf{T}), \\ M_t \dot{\mathbf{T}} &= f_t(t, \mathbf{u}, \mathbf{T}). \end{aligned}$$

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 [72]. 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, [73].

5.1.3. 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 [74], 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} :

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

From the scaled singular values matrix $\mathbf{\Sigma} = \text{diag}(\sigma_1, \dots, \sigma_n) / \sum \sigma_i$ 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

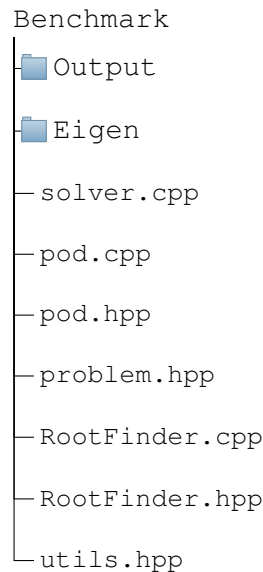
$$\mathbf{M}_e \dot{\mathbf{u}} = f_e(t, \mathbf{u}, \mathbf{V} \tilde{\mathbf{T}}),$$

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

5.2. 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.





5.2.1. 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.

5.2.2. 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.

5.2.3. 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.

5.2.4. 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.



5.3. 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, [75], must be included in the 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.

5.4. 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.



5.4.1. Results

To give an impression of the solution of the circuit a reference solution is shown in Figure 5.2. 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.

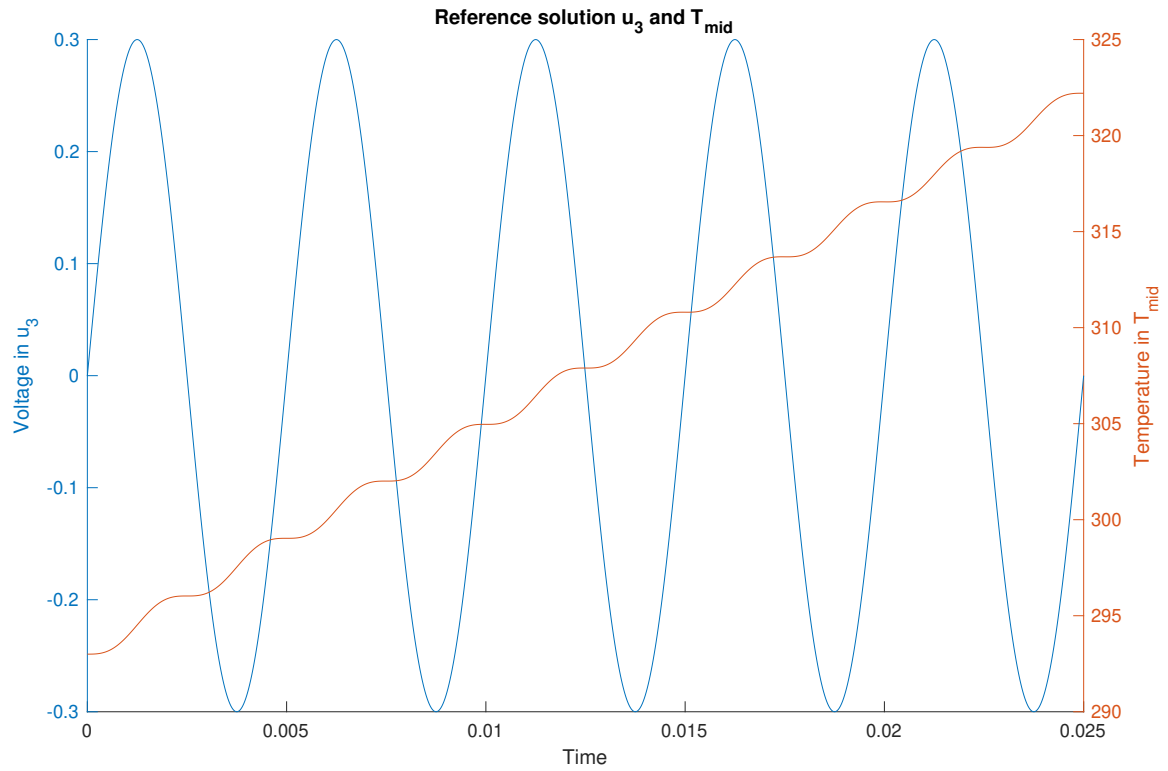


Figure 5.2: Reference solution

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 5.3 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.

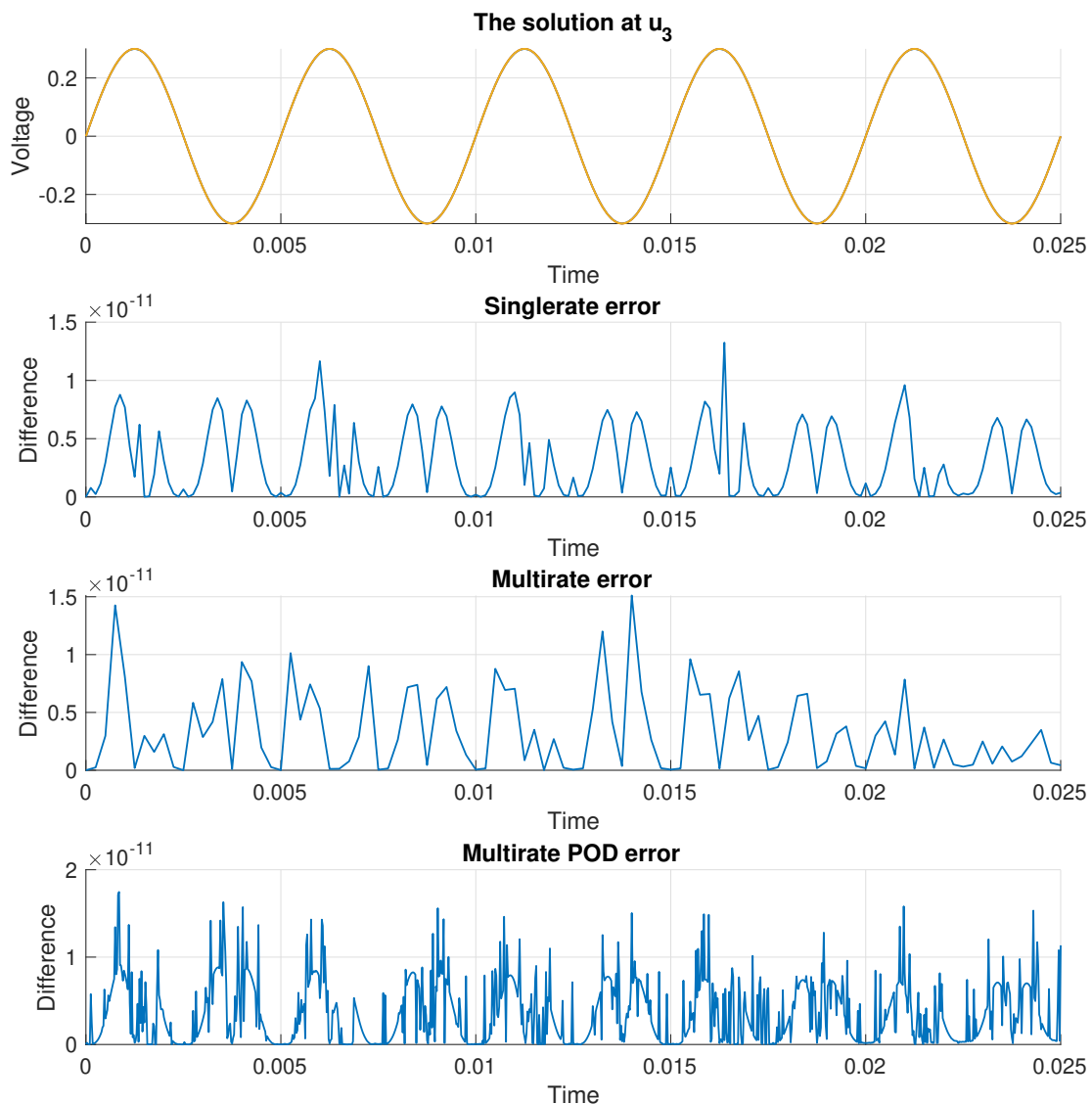


Figure 5.3: The results of the simulations

Lastly, in Table 3 the computation time of the three different approaches is shown. These times have been mea-



sured for 20 runs and then the average is shown. It now shows that the multirate approach roughly doubles the

	SR	MR	MR+MOR
Time	54.8243	28.3296	4.0325

Table 3: Computation time of the solver.

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.

5.5. 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.

Part VI.

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

Andreas Binder, Onkar Jadhav, Volker Mehrmann

Abstract

The European Parliament has introduced regulations (No 1286/2014) on packaged retail investment and insurance products (PRIIPs). According to this regulation, PRIIP manufacturers must provide a key information document (KID) describing the risk and the possible returns of these products. The formation of a KID requires expensive valuations rising the need for efficient computations. To perform such valuations efficiently, we establish a model order reduction approach based on a proper orthogonal decomposition (POD) method. The study involves the computations of high dimensional parametric convection-diffusion reaction partial differential equations. POD requires to solve the high dimensional model at some parameter values to generate a reduced-order basis. We propose a greedy sampling technique 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 cap and floor under the Hull-White model indicate that the MOR approach works well for the short-rate models.

Keywords: PRIIP, model order reduction, POD-greedy approach, interest rate modeling.

6.1. Introduction

Packaged retail investment and insurance 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) [76]. According to this regulation, a PRIIP manufacturer must provide a key information document for an underlying product that is easy to read and understand. The PRIIPs include interest rate derivatives such as the interest rate cap and floor [77], interest rate swaps [78] 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 work evaluates interest rate derivatives based on the dynamics of the short-rate models [79]. For the simulations of short-rate models, techniques based on discretized convection-diffusion reaction partial differential equations (PDEs) are often superior [80]. We implement the finite difference method (FDM) for such simulations [81]. The FDM method has been proven to be efficient for solving the short-rate models [82, 83, 84]. The model parameters are usually calibrated based on market structures like yield curves, cap volatilities, or swaption volatilities [79]. 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 the 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. We need to solve the high dimensional model (HDM) obtained by discretizing the short-rate PDE for such a high dimensional parameter space [85]. These simulations are computationally costly and additionally, have the disadvantage of being affected by the so-called *curse of dimensionality* [86].

To avoid this problem, we establish a parametric model order reduction (MOR) approach based on the proper orthogonal decomposition (POD) method [87, 88]. The method is also known as the Karhunen-Loève decomposition [89] or principal component analysis [90] in statistics. The combination of a Galerkin projection approach and POD creates a powerful method for generating a reduced order model (ROM) from the HDM



that has a high dimensional parametric space [91]. This approach is computationally feasible as it always looks for a linear (or affine) subspaces [92, 93]. 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. Further, the reduced order models are 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 most prominently obtained by the *method of snapshots* introduced in [94]. In this method, the optimal basis is computed based on the set of state solutions. These state solutions are known as snapshots and are calculated by solving the HDM for some parameter values. The quality of the ROM is bounded by the 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 projection subspace. Most of the previous work has implemented either some form of fixed sampling or often only uniform sampling techniques [95]. These approaches are straightforward, but they may neglect the vital regions in the case of large dimensional parameter spaces. In the current work, the greedy sampling algorithm has been implemented to determine the best suitable parameter set. The basic idea is to select the parameters at which the error between the ROM and HDM is maximal. Further, we compute the snapshots using these parameters and thus obtaining the best suitable ROB. The calculation of the relative error between ROM and HDM is expensive, so instead, we use the error estimators like the residual errors.

In this report, we illustrate 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 type of financial instrument. We present the results with a numerical example of a floater with cap and floor under the Hull-White one-factor model. However, we can and will implement the developed model order reduction technique for other models as well. The current research findings indicate that the MOR approach works well for the short-rate models. Also, we will apply the Monte Carlo techniques for the simulation of short rate models.

6.2. Mathematical Description

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. Before introducing the stochastic differential equations (SDEs), we present some basic concepts required to construct the short-rate models.

6.2.1. Bank Account and Short-Rate

Firstly we introduce the definition for a *bank account* or also called as a money-market account. When investing a certain amount in a bank account, we expect it to grow at some rate over time. A money-market account represents a risk-less investment with a continuous profit at a risk-free rate.

Definition 1. Bank account (Money-market account). Let $B(t)$ be the value of a bank account at time $t \geq 0$. We assume that the bank account evolves according to the following differential equation with $B(0) = 1$,

$$dB(t) = B(t)r_t dt, \quad (6.1)$$

where r_t is a short-rate. This gives

$$B(t) = \exp\left(\int_0^t r_s ds\right). \quad (6.2)$$

According to the above definition, investing a unit amount at time $t = 0$ yields the value in (6.2) at time t , and r_t is the short-rate at which the bank account grows.

For simplicity, assume that the interest rate r and the bank account B are deterministic processes. If we deposit C units in the bank account at time $t = 0$, then we will have $C \times B(T)$ units at time T . Therefore, to have



exactly one unit at time T , i.e.,

$$CB(T) = 1, \quad (6.3)$$

we need to invest $1/B(T)$ units in the bank account. So, the value at time t of the C units invested at the initial time is

$$CB(t) = \frac{B(t)}{B(T)}. \quad (6.4)$$

This ratio is known as a *discount factor*.

Definition 2. Discount factor. The discount factor $DF(t, T)$ between time t and T is the amount at time t that is equal to one unit of currency payable at time T , and is given by

$$DF(t, T) = \frac{B(t)}{B(T)} = \exp\left(-\int_0^t r_s ds\right) \quad (6.5)$$

However, when working with interest-rate products, the study about the variability of the interest rates is essential. Therefore, it is necessary to drop the assumption that the process is deterministic and to consider the evaluation of r in time as a stochastic process. Thus, the bank account (6.1) and the discount factors (6.5) will be stochastic processes too. We introduce some stochastic differential equations (SDEs) based on a short-rate r_t later on in this report.

6.2.2. Yield Curve

A fundamental curve that can be obtained from the market data is the zero coupon curve or also known as the *yield curve*. It depends on *maturity* dates and interest rates for an underlying instrument.

Definition 3. Maturity or maturity date. A maturity or maturity date is the final payment date of a financial instrument, at which the principal (along with the remaining interest) is due to be paid.

Definition 4. Yield curve. The yield curve or a zero-coupon curve at time t is a curve showing interest rates plotted against different maturities for a similar financial instrument.

Such a curve is also known as *the term structure of interest rates* at time t . It is a plot of simply-compounded interest rates for all maturities T up to one year and of annually compounded rates for maturities T larger than one year. The maturity time points are also known as *tenor points*. We plot the yields at each tenor point T for $0 \leq T \leq T^*$ where T^* is the last maturity date. An example of such a curve is shown in Fig. 6.1.

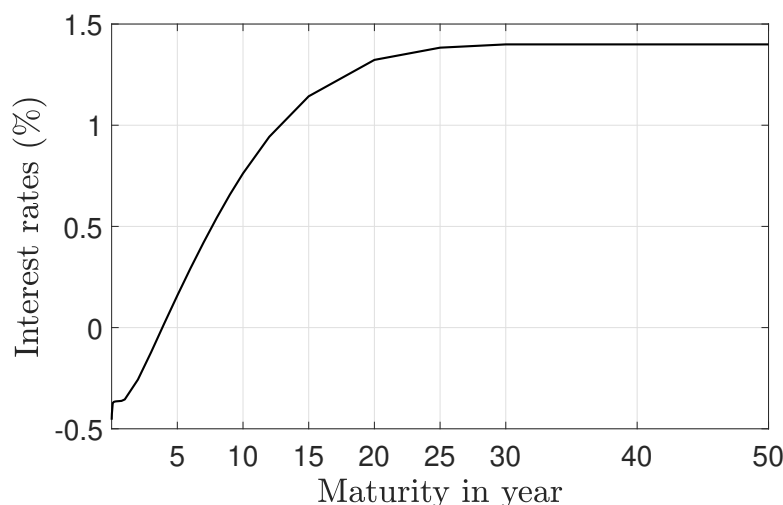


Figure 6.1: Sample yield curve

6.2.3. Zero-Coupon Bonds

Definition 5. Zero-Coupon bond. A T -maturity zero-coupon bond is a contract that ensures its holder the payment of one unit of a currency at time T , with no intermediate payments. The contract value at time $t < T$ is denoted by $B(t, T)$. Evidently, $B(T, T) = 1$.

Based on the stochastic discount factor, we can define a price for a zero-coupon bond paying €1 at time T . If the rate r is a deterministic quantity, then the $DF(t, T)$ is a deterministic quantity as well and hence, $D(t, T) = B(t, T)$. However, if the rate r is a stochastic process, then the $DF(t, T)$ will also be a stochastic process. So, it is necessary to take the weighted average of all the values of the stochastic discount factor, where each possible value is weighted by its respective probability. This weighted average is known as the expected value.

Definition 6. Expected value. Consider a stochastic variable X defined on a probability space Ω , then the expected value of X , denoted by $\mathbb{E}[X]$, is defined by Lebesgue integral [96]

$$\mathbb{E}[X] = \int_{\Omega} X(\omega) dP(\omega), \quad (6.6)$$

where P denotes probability.

The price of a zero-coupon bond based on the stochastic discount factor is given as

$$B(t, T) = \mathbb{E} \left[\exp \left(- \int_t^T r_s ds \right) \right], \quad (6.7)$$

where r is the short rate.

6.2.4. Forward Rate

The *forward rate* is a future yield calculated using the yield curve. Assume that the interest rates are continuously compounded, i.e., (6.1) describes the growth of a bank account continuously compounded by the interest rate r . If the continuous interest rate r depends on t_0 and on $t_1 = t_0 + \Delta t$, then from (6.5) we can write

$$DF(t_0, t_1) = \exp(-r(t_0, t_1)(t_1 - t_0)), \quad (6.8)$$

for the discount factor from time t_1 to t_0 .

Definition 7. Present value. The present value $PV(t_0)$ is the current value of a stream of cash flows $C(t_i)$ for a given bank account with a given specified rate of returns. The future cash flows are discounted at the discount rates $DF(t_0, t_i)$, and the higher the discount rates, the lower the present value.

Based on a definition of a discount factor $DF(t_0, t)$, the present value $PV(t_0)$ is given by

$$P(t_0) = \sum_{i=1}^n C(t_i) DF(t_0, t_i). \quad (6.9)$$

Now suppose we are trying to find the future interest rate $r_{1,2}$ for the time period (t_1, t_2) , where t_1 and t_2 are expressed in years. Assume that we know the interest rate r_1 for the time period (t_0, t_1) and the rate r_2 for the time period (t_0, t_2) from the yield curve. Also, at time t_2 , we are entitled to get a cash flow C and we want to calculate the fair value F of this cash flow at time t_1 .

The present value of the cash flow C will be, $C \cdot DF(t_0, t_2)$. Also, the present value of the cash flow F will



be, $F \cdot DF(t_0, t_1)$. Now, we can write

$$\begin{aligned} F \cdot DF(t_0, t_1) &= C \cdot DF(t_0, t_2), \\ F \cdot \exp(-r(t_0, t_1)(t_1 - t_0)) &= C \cdot \exp(-r(t_0, t_2)(t_2 - t_0)). \end{aligned} \quad (6.10)$$

Therefore

$$\log(F/C) = -\frac{r(t_0, t_2)(t_2 - t_0) - r(t_0, t_1)(t_1 - t_0)}{t_2 - t_1} \cdot (t_2 - t_1), \quad (6.11)$$

is the discount factor used for calculating F from C . Here the rate

$$r_{1,2} = \frac{r(t_0, t_2)(t_2 - t_0) - r(t_0, t_1)(t_1 - t_0)}{t_2 - t_1}, \quad (6.12)$$

is known as the *forward rate*.

6.2.5. Options

An *option* is a right, but not an obligation, to purchase or sell the underlying instrument at some time $t \geq 0$ at a pre-defined price. This price is known as a *strike price*. There are two types of options: *call options* and *put options*.

The call option gives a buyer the right to buy the *asset* at an agreed price, whereas the put option gives the right to sell. The payoff of an option is its value at the time of its exercise. In the case of a call option with a strike price K and an underlying instrument with a value V at the expiry T , the payoff for a call option C_V is given as

$$\begin{aligned} C_V &= \begin{cases} V - K, & \text{if } V > K \\ 0, & \text{if } V \leq K \end{cases} \\ C_V &= \max(V - K, 0) = (V - K)^+. \end{aligned} \quad (6.13)$$

For a call option, in the case of $V \leq K$, the asset can be purchased at a lower price than K in the market, and the buyer will not exercise the call option.

Similarly, for the put option, the payoff will be

$$P_V = \max(K - V, 0) = (K - V)^+. \quad (6.14)$$

For a put option, in case of $V \geq K$, the asset can be sold at a higher price than K in the market.

6.2.6. Interest Rate Cap and Floor

Assume that a borrower pays a floating interest rate (e.g., quarterly payments of Euribor3M¹) on some outstanding loan. The borrower may not be able to pay the interest payments if the Euribor3M rises substantially so, it is agreed that the interest rate shall not exceed 5%. That means the interest rate is capped at 5%.

Typically, the interest rate at time t_i is fixed by the interest rate at time t_{i-1} . For our example, the interest rate can be written as

$$\min(\text{Euribor3M}(t_{i-1}), 5\%).$$

Here Euribor3M is known as the reference interest rate R . The payoff $(R - K)^+$ is called a *caplet* with the strike price K , and is similar to the call option. For the fixed income products, the interest payments are made on scheduled dates (t_1, \dots, t_n) . The collection of caplets for these single payments into one contract gives a

¹The Euro Interbank Offered Rate (Euribor) is a daily reference rate, published by the European Money Markets Institute, based on the averaged interest rates at which Eurozone banks offer to lend unsecured funds to other banks in the euro wholesale money market.



so-called *cap*. A cap has a discounted payoff

$$\sum_{i=1}^n DF(t_i)(t_i - t_{i-1})(R(t_{i-1}) - K)^+. \quad (6.15)$$

Similarly, the interest rate floor is a contract in which the buyer receives payments at the end of each period in the case of the reference interest rate is below the strike price. The payoff $(K - R)^+$ gives a *floorlet* and collection of such floorlets gives a *floor*. The floorlet is similar to the put option.

6.2.7. No-Arbitrage Pricing

Consider a contract which pays cash flow C_k at time $t = k$, where $k = 1, \dots, T$. One has to pay the price p at $t = 0$ to get this contract. The *no-arbitrage condition* bounds this price p for the underlying contract. There are two types of no-arbitrage conditions: a weak no-arbitrage and a strong no-arbitrage.

- Weak no-arbitrage:

The weak no-arbitrage condition states that if the cash flow C_k is non-negative for all time points $k \geq 1$, then the price p for this contract must be greater than or equal to zero, i.e.

$$\text{If } C_k \geq 0 \text{ for all } k \geq 1 \text{ then } p \geq 0.$$

- Strong no-arbitrage:

The strong no-arbitrage condition states that if the cash flow C_k is non-negative for all time points $k \geq 1$, and there exist some time t_l , in the future, such that $C_l > 0$, then the price p must be strictly positive.

$$\text{If } C_k \geq 0 \text{ for all } k \geq 1 \text{ and } C_l > 0 \text{ for some } l \in k \text{ then } p > 0.$$

These no-arbitrage conditions have the following importance. In a market, if there are contracts for which one gets something for nothing, then prices for these contracts will be adjusted based on the arbitrage conditions such that one can not take advantage of this unbalance.

Suppose the price p satisfies $p < 0$, then the buyer has to pay $-p$ price to purchase the contract. In other words, we can say that the buyer will receive an amount p . Also, the seller can keep on increasing the price p as long as $p \leq 0$, and still buyers will be interested in purchasing the contract. The no-arbitrage condition avoids this problem and puts a bound on the price p .

6.2.8. Short-Rate Models

When working with the interest-rate products, the study about the variability of interest rates is essential. Therefore, it is necessary to consider the interest rate as a stochastic process.

Let S be the price of a *stock* at the end of the n th trading day. The *daily return* from day n to day $n + 1$ is given by $(S_{n+1} - S_n)/S_n$. In general, it is common to work with log returns since the log return of k days can be easily computed by adding up the daily log returns:

$$\log(S_k/S_0) = \log(S_1/S_0) + \dots + \log(S_k/S_{k-1}). \quad (6.16)$$

Based on the assumption that the log returns over disjoint time intervals are stochastically independent, and are equally distributed, the central limit theorem [97] of probability theory implies that the log returns are normally distributed [98].

So, it is necessary to define a stochastic model in continuous time in which log returns over arbitrary time intervals are normally distributed. The *Brownian motion* provides these properties [99].



6.2.8.1. Brownian Motion

Definition 8. A Brownian motion. A standard Brownian motion is a stochastic process $W(t)$ where $t \in \mathbb{R}$, i.e., a family of random variables $W(t)$, indexed by non-negative real numbers t with the following properties:

- At $t = 0$, $W_0 = 0$.
- With probability 1, the function $W(t)$ is continuous in t .
- For $t \geq 0$, the increment $W(t + s) - W(s)$ is normally distributed with mean 0 and variance t , i.e.,

$$W(t + s) - W(s) \sim N(0, t). \quad (6.17)$$

- For all n and times $t_0 < t_1 < \dots < t_{n-1} < t_n$, the increments $W(t_j) - W(t_{j-1})$ are stochastically independent.

Another important property of the Brownian motion is that [100]

$$(dW(t))^2 = dt. \quad (6.18)$$

Equation (6.18) says that the $(dW(t))^2$ is a deterministic quantity but not random, and has a magnitude of dt . Based on the definition of the Brownian motion, we can establish an SDE. Consider an ordinary differential equation (ODE)

$$\frac{dx(t)}{dt} = a(t)x(t), \quad (6.19)$$

with an initial condition $x(0) = x_0$. When we consider ODE (6.19) with an assumption that the parameter $a(t)$ is not a deterministic but rather a stochastic parameter, we get an SDE.

In our case, the stochastic parameter $a(t)$ is given as [100]

$$a(t) = f(t) + h(t)\Xi(t), \quad (6.20)$$

where $\Xi(t)$ is a white noise process.

Thus, we get

$$\frac{dX(t)}{dt} = f(t)X(t) + h(t)X(t)\Xi(t). \quad (6.21)$$

This equation is known as Langevin equation [101]. Here $X(t)$ is a stochastic variable having the initial condition $X(0) = X_0$ with a probability one. The Langevin force $\Xi(t) = dW(t)/dt$ is a fluctuating quantity having Gaussian distribution.

Substituting $dW(t) = \Xi(t)dt$ in (6.21), we get

$$dX(t) = f(t)X(t)dt + h(t)X(t)dW(t) \quad (6.22)$$

In the general form an SDE is given by

$$dX(t) = f(t, X(t))dt + g(t, X(t))dW(t), \quad (6.23)$$

where $f(t, X(t)) \in \mathbb{R}$, and $g(t, X(t)) \in \mathbb{R}$ are sufficiently smooth functions.

6.2.8.2. Ito's Lemma

Consider a function $\xi(x, y)$ which depends on variables x and y . According to the chain rule for total derivatives, we can write

$$d\xi = \frac{\partial \xi}{\partial x} dx + \frac{\partial \xi}{\partial y} dy. \quad (6.24)$$



If we have a function ξ which depends not only on a real variable t but also on a stochastic random variable $X(t)$, i.e., $\xi = f(t, X(t))$, then *Ito's lemma* provides an answer to this problem which is the stochastic calculus counterpart of the chain rule.

Theorem 1. Ito's Lemma. [102] *Let $\xi(X(t), t)$ be a sufficiently smooth function and let the stochastic process $X(t)$ be given by (6.23), then with probability one,*

$$d\xi(X(t), t) = \left(\frac{\partial \xi}{\partial X(t)} f(X(t), t) + \frac{\partial \xi}{\partial t} + \frac{1}{2} \frac{\partial^2 \xi}{\partial X^2(t)} g^2(X(t), t) \right) dt + \frac{\partial \xi}{\partial X(t)} g(X(t), t) dW(t). \quad (6.25)$$

Based on the Ito's lemma, we can derive a general partial differential equation for any interest rate derivative depending on the short-rate r_t . An SDE with r_t as a stochastic random variable can be written as

$$dr_t = f(t, r_t)dt + g(t, r_t)dW(t). \quad (6.26)$$

Consider a risk-neutral *portfolio*² Π_t that depends on a short-rate r_t , and consists of (i) a call option on the original interest rate product with maturity T_1 and price V_1 , (ii) Δ units in another product with different maturity T_2 and price V_2 , and (iii) the position of $V_1 - \Delta V_2$ in the risk-less asset. Now consider the value change of the portfolio over an infinitesimal time interval:

- the call option: the change in the product price is described by $dV_1(t) = V_1(t + dt) - V_1(t)$,
- the underlying units: the value change in the second product will be dV_2 , so for Δ units, the change will be ΔdV_2 ,
- the risk-free asset: the interest rate is paid/received so that the change in this position will be $(V_1 - \Delta V_2)r_t dt$.

The total change in the portfolio will be

$$d\Pi_t = \Delta dV_2 + (V_1 - \Delta V_2)r_t dt - dV_1. \quad (6.27)$$

According to Ito's lemma, we can define dV as

$$dV = \left(\frac{\partial V}{\partial r_t} f(r_t, t) + \frac{\partial V}{\partial t} + \frac{1}{2} \frac{\partial^2 V}{\partial r_t^2} g^2(r_t, t) \right) dt + \frac{\partial V}{\partial r_t} g(r_t, t) dW(t). \quad (6.28)$$

Substituting dV_1 and dV_2 in (6.27) we get

$$d\Pi_t = (V_1 - \Delta V_2)r_t dt - \left[\left(\frac{\partial V_1}{\partial r_t} f(r_t, t) + \frac{\partial V_1}{\partial t} + \frac{1}{2} \frac{\partial^2 V_1}{\partial r_t^2} g^2(r_t, t) \right) dt + \frac{\partial V_1}{\partial r_t} g(r_t, t) dW(t) \right] + \Delta \left[\left(\frac{\partial V_2}{\partial r_t} f(r_t, t) + \frac{\partial V_2}{\partial t} + \frac{1}{2} \frac{\partial^2 V_2}{\partial r_t^2} g^2(r_t, t) \right) dt + \frac{\partial V_2}{\partial r_t} g(r_t, t) dW(t) \right]. \quad (6.29)$$

Choosing $\Delta = \frac{\partial V_1}{\partial r_t} / \frac{\partial V_2}{\partial r_t}$ eliminates the stochastic term dW from (6.29). Also, to avoid arbitrage opportunities, the rate of return of this portfolio must be equal to the rate of return of the risk-free asset, r_t , and remains zero

²In finance, a portfolio is a collection of investments held by an investment company, a hedge fund, a financial institution or an individual.



due to the zero net investment requirement, i.e., $dII_t = 0$.

$$\begin{aligned}
0 = & \left[V_1 - \left(\frac{\partial V_1}{\partial r_t} / \frac{\partial V_2}{\partial r_t} \right) V_2 \right] r_t dt \\
& - \left[\left(\frac{\partial V_1}{\partial r_t} f(r_t, t) + \frac{\partial V_1}{\partial t} + \frac{1}{2} \frac{\partial^2 V_1}{\partial r_t^2} g^2(r_t, t) \right) dt + \frac{\partial V_1}{\partial r_t} g(r_t, t) dW(t) \right] \\
& + \left(\frac{\partial V_1}{\partial r_t} / \frac{\partial V_2}{\partial r_t} \right) \left[\left(\frac{\partial V_2}{\partial r_t} f(r_t, t) + \frac{\partial V_2}{\partial t} + \frac{1}{2} \frac{\partial^2 V_2}{\partial r_t^2} g^2(r_t, t) \right) dt + \frac{\partial V_2}{\partial r_t} g(r_t, t) dW(t) \right].
\end{aligned} \tag{6.30}$$

Eliminating the stochastic term

$$\begin{aligned}
& \left[V_1 - \left(\frac{\partial V_1}{\partial r_t} / \frac{\partial V_2}{\partial r_t} \right) V_2 \right] r_t dt \\
& = \left[\frac{\partial V_1}{\partial t} + \frac{1}{2} \frac{\partial^2 V_1}{\partial r_t^2} g^2(r_t, t) - \left(\frac{\partial V_1}{\partial r_t} / \frac{\partial V_2}{\partial r_t} \right) \left(\frac{\partial V_2}{\partial t} + \frac{1}{2} \frac{\partial^2 V_2}{\partial r_t^2} g^2(r_t, t) \right) \right] dt.
\end{aligned} \tag{6.31}$$

Rearranging the terms, and using notation $r = r_t$, we get

$$\frac{\frac{\partial V_1}{\partial t} + \frac{1}{2} \frac{\partial^2 V_1}{\partial r^2} g^2(r, t) - r V_1}{\frac{\partial V_1}{\partial r}} = \frac{\frac{\partial V_2}{\partial t} + \frac{1}{2} \frac{\partial^2 V_2}{\partial r^2} g^2(r, t) - r V_2}{\frac{\partial V_2}{\partial r}}. \tag{6.32}$$

Denoting either of the quotients as $u(r, t)$

$$\frac{\frac{\partial V_1}{\partial t} + \frac{1}{2} \frac{\partial^2 V_1}{\partial r^2} g^2(r, t) - r V_1}{\frac{\partial V_1}{\partial r}} = u(r, t). \tag{6.33}$$

Using notation $V = V_1$, we obtain a PDE for V depending on r

$$\frac{\partial V}{\partial t} + \frac{1}{2} g^2(r, t) \frac{\partial^2 V}{\partial r^2} - u(r, t) \frac{\partial V}{\partial r} - r V = 0, \tag{6.34}$$

where functions g and u depend on market structures like yield curves.

In the next subsection we will introduce some well-known one state variable short-rate models.

6.2.8.3. Vasicek and Cox-Ingersoll-Ross Models

The following models describe the dynamics of the short-rate r as

$$dr_t = b(\theta - r_t)dt + \sigma r_t^\beta dW(t), \tag{6.35}$$

where b , θ , σ , and β are positive constants. The model proposed in [103] considers $\beta = 0$ and is well known as the Vasicek model while the Cox-Ingersoll-Ross model introduced in [104] considers $\beta = 0.5$. One of the drawbacks of the Vasicek model is that the short-rate can be negative. On the other hand, in the case of the Cox-Ingersoll-Ross model, the square root term does not allow negative interest rates. However, the major drawback of these models is that the model parameters are constant, so we can not fit the model to the market structures like yield curves.



6.2.8.4. Hull-White Model

The Hull-White model [105, 106] is an extension of the Vasicek model which can be fitted to the market structures like yield curves. The SDE is given as

$$dr_t = b(t)(\theta(t) - r_t)dt + \sigma(t)dW(t), \quad (6.36)$$

where the model parameters $b(t)$, $\theta(t)$, and $\sigma(t)$ are time dependent. Equation (6.36) can also be represented as

$$dr_t = (a(t) - b(t)r_t)dt + \sigma(t)dW(t), \quad (6.37)$$

where $a(t) = b(t)\theta(t)$ is a deterministic function of time. The term $(a(t) - b(t)r_t)$ is a drift term and $a(t)$ is known as *deterministic drift*.

Definition 9. Drift. *The drift is a rate at which the expected value of the process changes.*

Theorem 2 presents the exact solution for the model (6.37).

Theorem 2. *Consider an SDE for a Hull-White model*

$$dr_t = (a(t) - b(t)r_t)dt + \sigma(t)dW(t),$$

with the initial condition $r(0) = r_0$. The exact solution is given as

$$r_t = e^{-\kappa(t)} \left[r_0 + \int_0^t e^{\kappa(s)} a(s) ds + \int_0^t e^{\kappa(s)} \sigma(s) dW(s) \right], \quad (6.38)$$

where $\kappa(t) = \int_0^t b(s) ds$.

We can define a PDE for any underlying instrument based on the Hull-White model depending on r . We recall (6.34),

$$\frac{\partial V}{\partial t} + \frac{1}{2}g^2(r, t) \frac{\partial^2 V}{\partial r^2} - u(r, t) \frac{\partial V}{\partial r} - rV = 0.$$

In the case of a Hull-White model, $g(r, t) = \sigma(t)$, and $-u(r, t) = (a(t) - b(t)r)$. Substituting $g(r, t)$ and $u(r, t)$, we get,

$$\frac{\partial V}{\partial t} + (a(t) - b(t)r) \frac{\partial V}{\partial r} + \frac{1}{2}\sigma^2(t) \frac{\partial^2 V}{\partial r^2} - rV = 0, \quad (6.39)$$

where the parameters $a(t)$, $b(t)$, and $\sigma(t)$ depend on the yield curves and cap/swap rates. The following sections present the simulation procedure for yield curves and the calibration of model parameters based on these simulated yield curves.

6.2.9. Yield Curve Simulation

The calibration of financial models is based on market structures like yield curves. The problem of determining the model parameters is relatively complex. These time-dependent parameters are derived from yield curves which determine the average direction in which short-rate r moves. The PRIIP's regulation demands to perform yield curve simulations for at least 10,000 times. We explain the detailed yield curve simulation procedure in this section.

- Collect the historical data for the interest rates.

The data set must contain at least 2 years of daily interest rates for an underlying instrument or 4 years of weekly interest rates or 5 years of monthly interest rates. Further, we construct a data matrix $D \in \mathbb{R}^{n \times m}$ of



the interest rates from the collected historical data, where each row of the matrix forms a yield curve, and each column is a tenor point, m . For example, we have collected the daily interest rate data at 20-30 tenor points in time over the past five years. Each year has approximately 260 working days also known as *observation periods*. Thus, there are $n \approx 1306$ observation periods and $m \approx 20$ tenor points in time.

- Calculate the log return over each period.

We take the natural logarithm of the ratio between the interest rate at each observation period and the interest rate at the preceding period. To avoid problems while taking the natural logarithm, we have to ensure that all elements of the data matrix D are positive.

$$\begin{aligned}\bar{D} &= D + \gamma E, \\ \bar{d}_{ij} &= d_{ij} + \gamma \beta_{ij}, \quad \beta_{ij} = 1 \quad \forall i, j\end{aligned}\tag{6.40}$$

where γ is a correction factor ensuring all elements of matrix D are positive and matrix $E \in \mathbb{R}^{n \times m}$ is a binary matrix having all entries as 1. The selection of γ does not affect the final simulated yield curves as we are compensating this shift at the bootstrapping stage by subtracting it from the simulated rates. Now we calculate the log returns over each period and store them into a new matrix $\hat{D} = \hat{d}_{ij} \in \mathbb{R}^{n \times m}$ as

$$\hat{d}_{ij} = \frac{\ln(\bar{d}_{ij})}{\ln(\bar{d}_{(i-1)j})}.\tag{6.41}$$

- Correct the returns observed at each tenor so that the resulting set of returns at each tenor point has a zero mean.

We calculate the arithmetic mean, μ_j of each column of a matrix \hat{D} ,

$$\mu_j = \frac{1}{n} \sum_{i=1}^n \hat{d}_{ij}.\tag{6.42}$$

Further, we subtract this arithmetic mean μ_j from each element of the corresponding j th column of a matrix \hat{D} and store the obtained results in the matrix, $\bar{\bar{D}} \in \mathbb{R}^{n \times m}$,

$$\bar{\bar{d}}_{ij} = \hat{d}_{ij} - \mu_j \beta_{ij}.\tag{6.43}$$

- Compute the singular value decomposition [107] of the matrix $\bar{\bar{D}}$.

The singular value decomposition of the matrix $\bar{\bar{D}}$ is

$$\bar{\bar{D}} = \Phi \Sigma \Psi^T,\tag{6.44}$$

$$\bar{\bar{D}} = \begin{bmatrix} \phi_{11} & \cdots & \phi_{1m} \\ \vdots & \vdots & \vdots \\ \phi_{m1} & \cdots & \phi_{mm} \end{bmatrix}_{m \times m} \cdot \begin{bmatrix} \Sigma_{11} & 0 & \cdots \\ 0 & \ddots & \vdots \\ \vdots & \cdots & \Sigma_{mm} \end{bmatrix}_{m \times m} \cdot \begin{bmatrix} \psi_{11} & \cdots & \psi_{1m} \\ \vdots & \vdots & \vdots \\ \psi_{m1} & \cdots & \psi_{mm} \end{bmatrix}_{m \times m}\tag{6.45}$$

where Σ is a diagonal matrix having singular values arranged in the descending order. The columns of Φ are the normalized eigenvectors $\phi \in \Phi$ which are also known as principal components.

- Select the principal components corresponding to the maximum energy.

The relative importance of i th principal component is determined by the relative energy E_i of that component defined as

$$E_i = \frac{\Sigma_i}{\sum_{i=1}^m \Sigma_i},\tag{6.46}$$



where the total energy is given by $\sum_{i=1}^n E_i = 1$. We select p singular vectors correspond to maximum energies from the matrix $\bar{\Phi}$. We construct a matrix $\bar{\Phi} \in \mathbb{R}^{m \times p}$ composed of these selected singular vectors

$$\bar{\Phi} = \begin{bmatrix} \phi_{11} & \cdots & \phi_{1p} \\ \vdots & \vdots & \vdots \\ \phi_{m1} & \cdots & \phi_{mp} \end{bmatrix}_{m \times p} \quad (6.47)$$

- Calculate the matrix of returns to be used for the simulation of yield curves.

We project the matrix \bar{D} onto the matrix of selected singular vectors $\bar{\Phi}$.

$$\mathfrak{X} = \bar{D} \times \bar{\Phi}, \quad \mathfrak{X} \in \mathbb{R}^{n \times p}. \quad (6.48)$$

Furthermore, we calculate the matrix of returns $M_R \in \mathbb{R}^{n \times m}$ by multiplying the matrix \mathfrak{X} with the transpose of the matrix of singular vectors $\bar{\Phi}$.

$$M_R = \mathfrak{X} \times \bar{\Phi}^T. \quad (6.49)$$

PCA simplifies the statistical data \bar{D} that transforms m correlated tensor points into p uncorrelated principal components. It allows reproducing the same data by simply reducing the total size of the model.

- Bootstrapping

According to the PRIIP regulations, we have to perform the following procedure for the yield curve simulation for at least 10,000 times. The regulations state that a standardized key information document shall include the minimum *recommended holding period* (RHP).

Definition 10. Holding period. *A holding period is a period between the acquisition of an asset and its sale. It is the length of time during which an underlying instrument is 'held' by an investor.*

Remark 1. *The recommended holding period gives an idea to an investor that for how long should an investor hold the product to minimize the risk. Generally, the RHP is given in years.*

The time step in the simulation is one observation period. Let, h be the RHP in days (For example, $h \approx 2600$ days or 10 years). So, there are h observation periods in the RHP. For each observation period in the RHP, we select a row at random from the matrix M_R , i.e., we select h random rows from the matrix M_R . We construct a matrix $S = \mathfrak{s}_{ij} \in \mathbb{R}^{h \times m}$ of these selected random rows. Further, we sum over the selected rows of the columns corresponding to the tenor point j ,

$$\bar{s}_j = \sum_{i=1}^h \mathfrak{s}_{ij}, \quad j = 1, \dots, m. \quad (6.50)$$

Thus, we obtain a row vector $\bar{s} \in \mathbb{R}^{1 \times m}$ such that

$$\bar{s} = [\bar{s}_1 \ \bar{s}_2 \ \cdots \ \bar{s}_m].$$

The final simulated yield rate y_j at tenor point j is the rate at the last observation period \bar{d}_{nj} at the corresponding tenor point j ,

1. multiplied by the exponential of the \bar{s}_j ,
2. adjusted for any shift γ used to ensure positive values for all tenor points.
3. adjusted for the forward rate so that the expected mean matches current expectations.

From the forward rate formula given by (6.12)

$$r_{1,2} = \frac{r(t_0, t_2)(t_2 - t_0) - r(t_0, t_1)(t_1 - t_0)}{t_2 - t_1}.$$



Thus, the final simulated rate r_j is given by

$$y_j = \bar{d}_{nj} \times \exp(\bar{s}_j) - \gamma\beta_{nj} + r_{1,2}, \quad j = 1, \dots, m. \quad (6.51)$$

Finally, we get the simulated yield curve from the calculated simulated returns y_j as

$$y = [y_1 \ y_2 \ \dots \ y_m], \quad j = 1, \dots, m. \quad (6.52)$$

We perform the bootstrapping procedure for at least $s = 10,000$ times and construct a simulated yield curve matrix $Y \in \mathbb{R}^{s \times m}$ as,

$$Y = \begin{bmatrix} y_{11} & \dots & y_{1m} \\ \vdots & \vdots & \vdots \\ y_{s1} & \dots & y_{sm} \end{bmatrix}_{s \times m} \quad (6.53)$$

Section 6.2.10 explains the parameter calibration based on simulated yield curves.

6.2.10. Parameter Calibration

The model parameters $a(t)$, $b(t)$, and $\sigma(t)$ are calibrated based on simulated yield curves. In this work, we present the parameter calibration for a one-factor Hull-White model only. See [108] for the parameter calibration procedures of other financial models.

From (6.7), the price at time $t = 0$ of a zero-coupon bond paying 1 at time T is,

$$B(0, T) = \mathbb{E} \left[\exp \left(- \int_0^T r_t dt \right) \right].$$

The expected value of the exponential function is obtained using a moment-generating function [109].

Definition 11. Moment-generating function. The moment-generating function of a random variable X is

$$M_X(\alpha) = \mathbb{E}[e^{\alpha X}], \quad \alpha \in \mathbb{R}.$$

In the case of a normal distribution, $M_X(\alpha)$ is given as

$$M_X(\alpha) = \exp(\alpha \mathbb{E}[X] + \frac{1}{2} \text{var}(X) \alpha^2). \quad (6.54)$$

Based on this definition

$$\mathbb{E} \left[\exp \left(- \int_0^T r_t dt \right) \right] = \exp \left\{ (-1) \mathbb{E} \left[\int_0^T r_t dt \right] + \frac{1}{2} (-1)^2 \text{var} \left(\int_0^T r_t dt \right) \right\}. \quad (6.55)$$

From equation (6.38), we obtain

$$r_t = e^{-\kappa(t)} \left[r_0 + \int_0^t e^{\kappa(s)} a(s) ds \right] + e^{-\kappa(t)} \Theta_1(t), \quad (6.56)$$

where $\Theta_1(t) = \int_0^t e^{\kappa(s)} \sigma(s) dW(s)$. Furthermore, using notation $\Theta_2(t) = \int_0^t e^{-\kappa(t)} \Theta_1(t) dt$, we get

$$\int_0^T r_t dt = \int_0^T e^{-\kappa(t)} \left[r_0 + \int_0^t e^{\kappa(s)} a(s) ds \right] + \Theta_2(t). \quad (6.57)$$



According to [108], the expected value and the variance of (6.57) are given respectively as

$$\begin{aligned}\mathbb{E} \int_0^T r_t dt &= \int_0^T e^{-\kappa(t)} \left[r_0 + \int_0^t e^{\kappa(s)} a(s) ds \right], \\ \text{Var} \left(\int_0^T r_t dt \right) &= \mathbb{E}[\Theta_2^2(T)], \\ &= \int_0^T e^{2\kappa(v)} \sigma^2(v) \left(\int_v^T e^{-\kappa(z)} dz \right)^2 dv.\end{aligned}\tag{6.58}$$

From (6.55),

$$\begin{aligned}B(0, T) &= \exp \left\{ -r_0 \int_0^T e^{-\kappa(t)} dt - \int_0^T \int_0^t e^{-\kappa(t)+\kappa(s)} a(s) ds dt \right. \\ &\quad \left. + \frac{1}{2} \int_0^T e^{2\kappa(v)} \sigma^2(v) \left(\int_v^T e^{-\kappa(z)} dz \right)^2 dv \right\}, \\ &= \exp \{ -r_0 \Gamma(0, T) - \Lambda(0, T) \},\end{aligned}\tag{6.59}$$

where

$$\begin{aligned}\Gamma(0, T) &= \int_0^T e^{-\kappa(t)} dt, \\ \Lambda(0, T) &= \int_0^T \int_0^t e^{-\kappa(t)+\kappa(s)} a(s) ds dt - \frac{1}{2} \int_0^T e^{2\kappa(v)} \sigma^2(v) \left(\int_v^T e^{-\kappa(z)} dz \right)^2 dv.\end{aligned}\tag{6.60}$$

We consider the parameter $b(t)$ and $\sigma(t)$ as constants. Hull and White questioned whether parameters b and σ should be made functions of time [110]. The problem is the term structure of volatility σ at future times might be different from the volatility structure today. Authors realized that the volatility at future times can collapse and may reach zero, which ultimately could result in implausible option prices. Thus, Hull and White advocate for making b and σ independent of time t . Nonetheless, we can calibrate $b(t)$ and $\sigma(t)$ based on cap data for various strike prices. See Appendix 6.5 for more details.

Based on (6.59), we obtain $B(0, T)$ for all $T \in [0, T^*]$ from the simulated yield returns. We take following input data for the calibration:

1. The zero-coupon bond prices for all maturities T^* , $0 \leq T \leq T^*$.
2. The initial value of $a(t)$ at $t = 0$ as $a(0)$.
3. The value of the volatility $\sigma(t)$ of the short-rate r_t at all maturities $0 \leq T \leq T^*$ is assumed to be constant.
4. The value of the parameter $b(t)$ is known and constant for all maturities $0 \leq T \leq T^*$.

We then determine the value of $\Gamma(0, T)$ as follows:

$$\Gamma(0, T) = \int_0^T e^{-\kappa(t)} dt$$

and then

$$\begin{aligned}\frac{\partial}{\partial T} \Gamma(0, T) &= e^{-\kappa(T)}, \\ \kappa(T) &= -\ln \frac{\partial}{\partial T} \Gamma(0, T), \\ \frac{\partial}{\partial T} \kappa(T) &= \frac{\partial}{\partial T} \int_0^T b(s) ds = b(T).\end{aligned}\tag{6.61}$$

Based on the known value of $b(t)$, we have computed $\Gamma(T)$.



Recall the formula for $\Lambda(0, T)$,

$$\Lambda(0, T) = \int_0^T \int_0^t e^{-\kappa(t)+\kappa(s)} a(s) ds dt - \frac{1}{2} \int_0^T e^{2\kappa(v)} \sigma^2(v) \left(\int_v^T e^{-\kappa(z)} dz \right)^2 dv.$$

We further simplify $\Lambda(0, T)$ by the change of variables s and t such that $z = t$; $v = s$,

$$\Lambda(0, T) = \int_0^T \left[e^{\kappa(v)} a(v) \left(\int_v^T e^{-\kappa(z)} dz \right) - \frac{1}{2} e^{2\kappa(v)} \sigma^2(v) \left(\int_v^T e^{-\kappa(z)} dz \right)^2 \right] dv. \quad (6.62)$$

We can use $\Lambda(0, T)$ to determine $a(t)$, for $0 \leq T \leq T^*$,

$$\begin{aligned} \frac{\partial}{\partial T} \Lambda(0, T) &= \int_0^T \left[e^{\kappa(v)} a(v) e^{-\kappa(T)} \right. \\ &\quad \left. - e^{2\kappa(v)} \sigma^2(v) e^{-\kappa(T)} \left(\int_v^T e^{-\kappa(z)} dz \right) \right] dv, \\ e^{\kappa(T)} \frac{\partial}{\partial T} \Lambda(0, T) &= \int_0^T \left[e^{\kappa(v)} a(v) - e^{2\kappa(v)} \sigma^2(v) \left(\int_v^T e^{-\kappa(z)} dz \right) \right] dv, \\ \frac{\partial}{\partial T} \left[e^{\kappa(T)} \frac{\partial}{\partial T} \Lambda(0, T) \right] &= e^{\kappa(T)} a(T) - \int_0^T e^{2\kappa(v)} \sigma^2(v) e^{-\kappa(T)} dv, \\ e^{\kappa(T)} \left[e^{\kappa(T)} \frac{\partial}{\partial T} \Lambda(0, T) \right] &= e^{2\kappa(T)} a(T) - \int_0^T e^{2\kappa(v)} \sigma^2(v) dv, \\ \frac{\partial}{\partial T} \left[e^{\kappa(T)} \left[e^{\kappa(T)} \frac{\partial}{\partial T} \Lambda(0, T) \right] \right] &= \frac{\partial a(T)}{\partial T} e^{2\kappa(T)} + 2a(T) e^{2\kappa(T)} \frac{\partial}{\partial T} \kappa(T) - e^{2\kappa(T)} \sigma^2(T), \\ \frac{\partial}{\partial T} \left[e^{\kappa(T)} \left[e^{\kappa(T)} \frac{\partial}{\partial T} \Lambda(0, T) \right] \right] &= \frac{\partial a(T)}{\partial T} e^{2\kappa(T)} + 2a(T) e^{2\kappa(T)} b(T) - e^{2\kappa(T)} \sigma^2(T). \end{aligned} \quad (6.63)$$

The yield $y(T)$ at time T is given by

$$y(T) = -\ln B(0, T). \quad (6.64)$$

From (6.59), we can obtain

$$\Lambda(0, T) = [y(T) - r_0 \Gamma]. \quad (6.65)$$

This gives an ordinary differential equation (ODE) for $a(t)$

$$\begin{aligned} \frac{\partial}{\partial T} a(T) e^{2\kappa(T)} + 2a(T) \cdot b(T) \cdot e^{2\kappa(T)} - e^{2\kappa(T)} \sigma^2(T) \\ = \frac{\partial}{\partial T} \left[e^{\kappa(T)} \left[e^{\kappa(T)} \frac{\partial}{\partial T} (y(T) - r_0 \Gamma(0, T)) \right] \right], \end{aligned} \quad (6.66)$$

where $y(T)$ is the simulated yield rate at tenor point T . We can solve (6.66) numerically with the given initial conditions and yield rates for $0 \leq T \leq T^*$. For all $T \in [0, T^*]$, we know $b(T)$, $\sigma(t)$ and $\Gamma(0, T)$ from the given initial conditions and (6.61) respectively. Thus, the right hand side of the (6.66) is the known function in time.



We divide (6.66) by $e^{2\kappa(T)}$

$$\begin{aligned} \frac{\partial}{\partial T} a(T) + 2a(T) \cdot b(T) - \sigma^2(T) \\ = e^{-2\kappa(T)} \frac{\partial}{\partial T} \left[e^{\kappa(T)} \left[e^{\kappa(T)} \frac{\partial}{\partial T} ((T^* - T)y(T) - r_0 \Gamma(0, T)) \right] \right], \end{aligned} \quad (6.67)$$

$$\frac{\partial}{\partial T} a(T) + 2a(T) \cdot b(T) - \sigma^2(T) = e^{-2\kappa(T)} \frac{\partial}{\partial T} \left[e^{2\kappa(T)} \frac{\partial}{\partial T} ((T^* - T)y(T) - r_0 \Gamma(0, T)) \right],$$

If we would assume $a(T)$ to be piecewise constant with values a_i in $((i+1) \cdot \Delta T, i \cdot \Delta T)$, then we could calculate a_i iteratively for $0 \leq T \leq T^*$. This leads to a triangular system of linear equations for the vector with non-zero diagonal elements, and therefore holds a unique solution.

In practice, the authors of [111] found that the right-hand side of the system contains noise. Hence, the use of a naive approach like finite difference scheme may result in some numerical errors. In this work, we use the inbuilt UnRisk PRICING ENGINE functions for the parameter calibrations [112]. The UnRisk engine implements a classical Tikhonov regularization approach to stabilize the ill-posed problems. Based on 10,000 different simulated yield curves, we obtain $s = 10,000$ different parameter vectors $a(t)$. In the matrix form, we write

$$\mathbf{a}(t) = \begin{bmatrix} a_{11} & \cdots & a_{1m} \\ \vdots & \vdots & \vdots \\ a_{s1} & \cdots & a_{sm} \end{bmatrix} \quad (6.68)$$

where m is the number of tenor points. All parameters are assumed to be piecewise constant changing their values only on tenor points, i.e., on model term dates. Thus, if there are m tenor points, then there will be m values for a single parameter vector.

6.3. Numerical Methods

6.3.1. Finite Difference Method

The PDEs obtained for the short-rate models can be interpreted as convection-reaction-diffusion PDEs [113]. Consider a Hull-White PDE given by (6.34)

$$\frac{\partial V}{\partial t} + \underbrace{(a(t) - b(t)r_t) \frac{\partial V}{\partial r}}_{\text{Convection}} + \underbrace{\frac{1}{2} \sigma^2(t) \frac{\partial^2 V}{\partial r^2}}_{\text{diffusion}} - \underbrace{rV}_{\text{reaction}} = 0. \quad (6.69)$$

In this work, we apply the finite difference method to solve the Hull-White PDE. The convection term in the above equation may lead to numerically unstable results. Thus, we implement the so-called upwind scheme [114] to obtain a stable solution. We incorporate the semi-implicit scheme called the Crank-Nicolson method [115] for the time discretization.

6.3.1.1. Spatial Discretization

Consider the following one-dimensional linear advection equation

$$\frac{\partial \zeta}{\partial t} + U \frac{\partial \zeta}{\partial x} = 0, \quad (6.70)$$



describing a wave propagation along the x -axis with a velocity U . We define a discretization of the computational domain in sd spatial dimensions as

$$[\alpha_k, \beta_k]^{sd} \times [0, T] = [\alpha_1, \beta_1] \times \cdots \times [\alpha_d, \beta_d] \times [0, T] = \left(\prod_{k=1}^{sd} [\alpha_k, \beta_k] \right) \times [0, T],$$

where α and β are the cut off limits of the spatial domain. T denotes the final time of the computation. The corresponding indices are, $i_k \in \{1, \dots, M_k\}$ for the spatial discretization and $n \in \{1, \dots, N\}$ for the time discretization. The first order upwind scheme of the order $\mathcal{O}(\Delta x)$ is given by

$$\frac{\zeta_i^{n+1} - \zeta_i^n}{\Delta t} + U \frac{\zeta_i^n - \zeta_{i-1}^n}{\Delta x} = 0 \quad \text{for } U > 0 \quad (6.71a)$$

$$\frac{\zeta_i^{n+1} - \zeta_i^n}{\Delta t} + U \frac{\zeta_{i+1}^n - \zeta_i^n}{\Delta x} = 0 \quad \text{for } U < 0 \quad (6.71b)$$

Let's say,

$$U^+ = \max(U, 0), \quad U^- = \min(U, 0),$$

and

$$\zeta_x^- = \frac{\zeta_i^n - \zeta_{i-1}^n}{\Delta x}, \quad \zeta_x^+ = \frac{\zeta_{i+1}^n - \zeta_i^n}{\Delta x}$$

Combining (6.71a) and (6.71b) in the compact form, we can write

$$\zeta_i^{n+1} = \zeta_i^n - \Delta t [U^+ \zeta_x^- + U^- \zeta_x^+]. \quad (6.72)$$

We implement the above defined upwind scheme for the convection term. The diffusion term is discretized using the second order central differencing scheme of order $\mathcal{O}(\Delta x)^2$.

$$\frac{\partial^2 \zeta}{\partial x^2} = \frac{\zeta_{i+1}^n - 2\zeta_i^n + \zeta_{i-1}^n}{(\Delta x)^2} \quad (6.73)$$

6.3.1.2. Time Discretization

Consider a time-dependent PDE for a quantity ζ

$$\frac{\partial \zeta}{\partial t} + L\zeta = 0, \quad (6.74)$$

where L is the linear differential operator containing all spatial derivatives. Using the Taylor series expansion, we write

$$\zeta(t + \Delta t) = \zeta(t) + \Delta t \frac{\partial \zeta}{\partial t} + \frac{\Delta t^2}{2} \frac{\partial^2 \zeta}{\partial t^2}. \quad (6.75)$$

Neglecting terms of order higher than one

$$\frac{\partial \zeta}{\partial t} = \frac{\zeta(t + \Delta t) - \zeta(t)}{\Delta t} + \mathcal{O}(\Delta t). \quad (6.76)$$

From (6.76),

$$\zeta(t + \Delta t) = \zeta(t) - \Delta t (L(t)\zeta(t)). \quad (6.77)$$

Let us introduce a new parameter Θ such that

$$\frac{\zeta(t + \Delta t) - \zeta(t)}{\Delta t} = (1 - \Theta)(L(t)\zeta(t)) + \Theta(L(t + \Delta t)\zeta(t + \Delta t)) \quad (6.78)$$



We can construct different time discretization schemes for different values of Θ . Setting $\Theta = 0$, we obtain a fully explicit scheme known as the forward difference method, while considering $\Theta = 1$, we get a fully implicit scheme known as the backward difference method [113]. Here we set $\Theta = 1/2$ and obtain a semi-implicit scheme known as the Crank-Nicolson method.

$$\left(1 - \frac{1}{2}\Delta t L(t + \Delta t)\right)\zeta(t + \Delta t) = \left(1 + \frac{1}{2}\Delta t L(t)\right)\zeta(t) \quad (6.79)$$

6.3.1.3. FDM for a Hull-White Model

The computational domain for a spatial dimension r is $[\alpha, \beta]$. According to [80], the cut off values α and β are given as,

$$\alpha = r_{sp} + \sqrt{T}K\sigma_{\max} \text{ and } \beta = r_{sp} - \sqrt{T}K\sigma_{\max}, \quad (6.80)$$

where $K = 7$ and r_{sp} is a yield at the maturity T also known as a spot rate.

$\sigma_{\max} = \max(\sigma(t)), t \in [0, T]$. We divide the spatial domain into M equidistant grid points which generate a set of points $\{r_1, r_2, \dots, r_M\}$. The time interval $[0, T]$ is divided into $N - 1$ time points (N points in time that are measured in days starting from $t = 0$).

Equation (6.74) can be represented as,

$$\frac{\partial V}{\partial t} + L(t)V(t) = 0. \quad (6.81)$$

We specify the spatial discretization operator $L(n)$, where the index n denotes the time-point. From (6.71) and (6.73),

$$\begin{aligned} L(n)V_i^n &= \\ & \frac{1}{2}\sigma^2(n)\frac{V_{i+1}^n - 2V_i^n + V_{i-1}^n}{(\Delta x)^2} + (a(n) - b(n)r_i)\frac{V_i^n - V_{i-1}^n}{\Delta x} - r_i V_i^n \\ & \text{if } (a(n) - b(n)r_i) > 0 \\ L(n)V_i^n &= \\ & \frac{1}{2}\sigma^2(n)\frac{V_{i+1}^n - 2V_i^n + V_{i-1}^n}{(\Delta x)^2} + (a(n) - b(n)r_i)\frac{V_{i+1}^n - V_i^n}{\Delta x} - r_i V_i^n \\ & \text{if } (a(n) - b(n)r_i) < 0 \end{aligned} \quad (6.82)$$

From (6.78), we can write

$$\frac{V(t + \Delta t) - V(t)}{\Delta t} = (1 - \Theta)(L(t)V(t)) + \Theta(L(t + \Delta t)V(t + \Delta t)) \quad (6.83)$$

For $\Theta = 1/2$, we write

$$\underbrace{\left(1 - \frac{1}{2}\Delta t L(t + \Delta t)\right)}_{A(\rho_s(t)) \in \mathbb{R}^{M \times M}} V(t + \Delta t) = \underbrace{\left(1 + \frac{1}{2}\Delta t L(t)\right)}_{B(\rho_s(t)) \in \mathbb{R}^{M \times M}} V(t), \quad (6.84)$$

where matrices $A(\rho_s(t))$, and $B(\rho_s(t))$ depend on parameters $a(t)$, $b(t)$ and $\sigma(t)$. For the simplicity of notations, we denote ρ_s as the s th group of these parameters. Here

$$A(\rho_s(t)) = I - \frac{\sigma^2(n)\Delta t}{2(\Delta x)^2}X - \frac{\Delta t}{2\Delta x}(Z^+Y^- + Z^-Y^+) + R, \quad (6.85)$$

and

$$B(\rho_s(t)) = I + \frac{\sigma^2(n)\Delta t}{2(\Delta x)^2}X + \frac{\Delta t}{2\Delta x}(Z^+Y^- + Z^-Y^+) - R, \quad (6.86)$$



where

$$\begin{aligned}
 X &= \begin{bmatrix} -2 & 1 & 0 & \cdots & 0 \\ 1 & -2 & 1 & \ddots & \vdots \\ 0 & 1 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & 1 \\ 0 & \cdots & 0 & 1 & -2 \end{bmatrix} \\
 Y^- &= \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ -1 & 1 & 0 & \ddots & \vdots \\ 0 & -1 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -1 & 1 \end{bmatrix} & Y^+ &= \begin{bmatrix} 1 & -1 & 0 & \cdots & 0 \\ 0 & 1 & -1 & \ddots & \vdots \\ 0 & 0 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & -1 \\ 0 & \cdots & 0 & 0 & 1 \end{bmatrix} \\
 Z^+ &= \begin{bmatrix} \max(a(n) - b(n)r(1)) & 0 & 0 & \cdots & 0 \\ 0 & \max(a(n) - b(n)r(2)) & 0 & \ddots & \vdots \\ 0 & 0 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 0 & \max(a(n) - b(n)r(M)) \end{bmatrix} \\
 Z^- &= \begin{bmatrix} \min(a(n) - b(n)r(1)) & 0 & 0 & \cdots & 0 \\ 0 & \min(a(n) - b(n)r(2)) & 0 & \ddots & \vdots \\ 0 & 0 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 0 & \min(a(n) - b(n)r(M)) \end{bmatrix} \\
 R &= \begin{bmatrix} r(1) & 0 & 0 & \cdots & 0 \\ 0 & r(2) & 0 & \ddots & \vdots \\ 0 & 0 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 0 & r(M) \end{bmatrix}
 \end{aligned}$$

The above discretization of the PDE generates a parametric HDM of the following form (6.87). 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.

$$A(\rho_s(t))V^{n+1} = B(\rho_s(t))V^n, \quad V(0) = V_0, \quad (6.87)$$

where the matrices $A(\rho) \in \mathbb{R}^{M \times M}$, and $B(\rho) \in \mathbb{R}^{M \times M}$ are parameter dependent matrices. $V \in \mathbb{R}^M$ is a high dimensional state vector. M is the total number of spatial discretization points. t is the time variable. $t = [0, T]$ where T is the final term date. Within the interval $[0, T]$, if we have m tenor points, then we have m values for the parameter $a(t)$ such that $a_s = a_{s1}, \dots, a_{sm}$. We consider the parameters $b(t)$ and $\sigma(t)$ as constants. For example, if we consider an instrument with a contract period of 10 years, then in this case $T = 10$ years. If there are m tenor points such that $m := 0, 1y, 2y, \dots, 10y$, then we will get 11 different values for the parameter vector $a(t)$. That is, considering the time interval of 1 day, for the period of $n = 0$ to 260, $a_n = a_{s1}$ (1 year \approx 260 days). For the simplicity of notations, we denote $\rho_s = [(a_{s1}, \dots, a_{sm}), b, \sigma]$ as the s th group of model parameters where $s = 1, \dots, 10000$. Each parameter group ρ_s has values ranging from ρ_{s1} to ρ_{sm}



where m is the total number of tensor points. We need to solve the system (6.87) for at least 10,000 parameter groups ρ generating a parameter space P of $10000 \times m$.

6.3.2. Parametric Model Order reduction

We employ the projection based MOR technique to solve the HDM (6.87). The idea is to project a high dimensional space onto a low dimensional subspace, Q as

$$\bar{V}^n = QV_d^n, \quad (6.88)$$

where $Q \in \mathbb{R}^{M \times d}$ is a reduced order basis (ROB) with $d \ll M$, V_d is a vector of reduced coordinates, and $\bar{V} \in \mathbb{R}^M$ is the solution obtained using the reduced order model. Substituting (6.88) into the system of equations (6.87) gives the residual of the reduced state as

$$r^n(V_d, \rho_s) = A(\rho_s)QV_d^{n+1} - B(\rho_s)QV_d^n. \quad (6.89)$$

In the case of the Galerkin projection, the residual $r(V_d, \rho_s)$ is orthogonal to the ROB Q

$$Q^T r^n(V_d, \rho_s) = 0. \quad (6.90)$$

Multiplying (6.89) by Q^T , we get

$$\begin{aligned} Q^T A(\rho_s)QV_d^{n+1} &= Q^T B(\rho_s)QV_d^n, \\ A_d(\rho_s)V_d^{n+1} &= B_d(\rho_s)V_d^n, \end{aligned} \quad (6.91)$$

where the matrices $A_d(\rho_s) \in \mathbb{R}^{d \times d}$ and $B_d(\rho_s) \in \mathbb{R}^{d \times d}$ are the parameter dependent reduced matrices. We obtain the parametric reduced order model (6.91) based on a proper orthogonal decomposition method (POD). POD generates an optimal order orthonormal basis Q which serves as the ROB in the least square sense for a given set of computational data. We aim to obtain the subspace Q independent of parameter space P . In this work, we obtain the optimal basis set by the method of snapshots. We compute snapshots by solving the HDM (6.87) for the selected parameter groups (i.e., snapshots taken for some parameter groups $\rho_1 \cdots \rho_l \in [\rho_1, \rho_s]$). Further, we construct a snapshot matrix composed of these snapshots. Finally, we generate an optimally ordered orthonormal basis by performing a singular value decomposition of the snapshot matrix.

A solution V_s of the HDM for a single parameter group ρ_s can be represented as

$$V_s = \begin{bmatrix} V_s(r_1, t_1) & V_s(r_1, t_2) & \cdots & V_s(r_1, t_N) \\ V_s(r_2, t_1) & V_s(r_2, t_2) & \cdots & V_s(r_2, t_N) \\ \vdots & \vdots & \vdots & \vdots \\ V_s(r_M, t_1) & V_s(r_M, t_2) & \cdots & V_s(r_M, t_N) \end{bmatrix} \quad (6.92)$$

We obtain such solutions of the HDM for selected parameter groups and combined them to form a snapshot matrix, \hat{V} such that

$$\hat{V} = \begin{bmatrix} V_1(r_1, t_1) & \cdots & V_1(r_1, t_N) & \cdots & V_l(r_1, t_1) & \cdots & V_l(r_1, t_N) \\ V_1(r_2, t_1) & \cdots & V_1(r_2, t_N) & \cdots & V_l(r_2, t_1) & \cdots & V_l(r_2, t_N) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ V_1(r_M, t_1) & \cdots & V_1(r_M, t_N) & \cdots & V_l(r_M, t_1) & \cdots & V_l(r_M, t_N) \end{bmatrix} \quad (6.93)$$



We perform a truncated SVD of the matrix \hat{V} to obtain the ROB Q

$$\begin{aligned}\hat{V} &= \sum_{i=1}^k \Sigma_i \phi_i \psi_i^T, \\ \hat{V} &= \Phi \Sigma \Psi^T,\end{aligned}\tag{6.94}$$

where ϕ_i and ψ_i are the left and right singular vectors of the matrix \hat{V} respectively. Σ_i are the singular values.

$$\hat{V} = [\phi_1 \ \cdots \ \phi_k]_{M \times k} \begin{bmatrix} \Sigma_1 & 0 & \cdots \\ 0 & \ddots & \vdots \\ \vdots & \cdots & \Sigma_k \end{bmatrix}_{k \times k} [\psi_1 \ \cdots \ \psi_k]_{k \times k}\tag{6.95}$$

The truncated (economy-size) SVD computes only first k columns of matrix Φ . The optimal projection subspace Q consists of d left singular vectors ϕ_i known as POD modes. Here, d makes the dimension of the reduced order model.

The quality of the parametric reduced model mainly depends on the selection of parameters for which the snapshots are computed. Thus, it necessitates defining an efficient sampling technique for the high dimensional parameter space. We could consider standard sampling techniques like uniform or random sampling to generate snapshots. However, the computational cost for the uniform sampling may become too expensive due to the combinatorial explosion of samples used to cover the parameter domain [116]. On the other hand, random sampling might neglect the vital regions in the parameter space. The greedy sampling method introduced in [117] is proven to be an efficient method for sampling a high dimensional parameter space in the framework of MOR.

6.3.3. Greedy Sampling Method

The greedy sampling technique selects the parameter groups at which the error between the ROM and the HDM is maximum. Further, we compute the snapshots using these parameter groups so that we can obtain the best suitable ROB Q . However, the computation of a relative error $\|e(\cdot, \rho)\| = \|V(\cdot, \rho) - \bar{V}(\cdot, \rho)\| / \|V(\cdot, \rho)\|$ between the HDM and the ROM is expensive. Thus, usually, the error is replaced by the error bounds or the relative residual for the approximate solution \bar{V} . We discuss the error estimators $\varepsilon(\rho)$ in Subsubsect. 6.3.3.1. The greedy sampling method is explained in Algorithm 1.

The POD greedy algorithm runs for I_{max} iterations. At each iteration $I = 1, \dots, I_{max}$, we choose the parameter group as the maximizer

$$\rho_I = \operatorname{argmax}_{\rho \in P} \varepsilon(\rho).\tag{6.96}$$

We solve the problem over a pre-defined set of random candidate parameter groups $\bar{P} = \{\rho_1, \rho_2, \dots, \rho_C\}$ from the parameter space P .

6.3.3.1. Error Estimators

To avoid the computational expenses, the error $\|e(\cdot, \rho)\|$ is usually replaced by the error estimators ε like error bounds and the norm of the residual.

- **Error Bonds:** Error bounds $\delta e(\rho)$ tell the maximum possible error in the approximation and can be given as

$$\|e(\cdot, \rho)\| \leq \delta e(\rho) \quad \forall \rho \in P\tag{6.97}$$

However, in some cases, it is not possible to define the error bounds or the error bounds do not exist. In such cases, the relative residual of the approximate solution is a common alternative.



Input: Maximum number of iterations I_{max} , maximum parameter groups C , Parameter space P

Output: Q

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

Solve the HDM for a parameter group ρ_1 and store the results in V_1

Compute an SVD of the matrix V_1 and construct Q_1

Randomly select a set of parameter groups $\bar{P} = \{\rho_1, \rho_2, \dots, \rho_C\} \subset P$

```

for  $i = 2$  to  $I_{max}$  do
  for  $j = 1$  to  $C$  do
    Solve ROM for the parameter group  $\rho_j$  with the projection subspace  $Q_{i-1}$ 
    Compute the error estimator  $\varepsilon(\rho_j)$ 
  end
  Find  $\rho_I = \operatorname{argmax}_{\rho \in P} \varepsilon(\rho)$ 
  if  $\varepsilon(\rho_I) \leq \varepsilon_{tol}$  then
     $Q = Q_{i-1}$ 
    break
  end
  Solve the HDM for the parameter group  $\rho_I$  and store the result in  $V_i$ 
  Construct a snapshot matrix  $\hat{V}$  by concatenating the solutions  $V_s$  for  $s = 1, \dots, i$ 
  Compute an SVD of the matrix  $\hat{V}$  and construct  $Q_i$ 
end

```

Algorithm 1: POD-Greedy Sampling Algorithm

- **Residual:** The relative error $\|e(\cdot, \rho)\|$ is bounded by the residual as

$$\frac{\|r(V_d, \rho_s)\|}{\|A(\rho_s)\|} \leq \|e(\cdot, \rho)\| \leq \|A^{-1}(\rho_s)\| \cdot \|r(V_d, \rho_s)\| \quad (6.98)$$

This error bound holds if and only if $A(\rho)$ is a well-conditioned matrix [116, 118]. The quantity $\frac{\|r(V_d, \rho)\|}{\|A(\rho)\| \cdot \|V\|}$ is known as the relative residual.

The above estimates require the knowledge of the norm of the matrix A and its inverse. According to [119], in some cases, the $\|r\|/\|A\|$ can be a bad estimate of $\|e(\cdot, \rho)\|$. In this work, we use the error estimator proposed in [119] for the solution of linear systems. Here the error estimator $\varepsilon(\rho)$ is an excellent approximation of $\|e(\cdot, \rho)\|$.

$$\varepsilon^2(\rho) = \frac{C_0^2}{C_1}, \quad (6.99)$$

where

$$C_0 = \sum_{i=1}^k (\phi_i, r(V_d, \rho_s))^2,$$

$$C_1 = \sum_{i=1}^k \Sigma_i^2 (\psi_i, r(V_d, \rho_s))^2.$$

where ϕ_i and ψ_i are the left and right singular vectors of the matrix $A(\rho_s)$ respectively. Σ_i are the singular values of $A(\rho_s)$.

6.3.3.2. Drawbacks

The greedy sampling method computes the inexpensive *a posteriori* error estimator for the ROM. However, it is not feasible to calculate the error estimators for all of the parameter space P . An error estimator is based on



the norm of the residual which scales with the dimension of the HDM, M . With an increase in dimension, it is not computationally reasonable to calculate the residual for 10,000 parameter groups. Hence, the POD-greedy technique chooses the pre-defined parameter set \bar{P} randomly as a subset of P . Random sampling is designed to represent the whole parameter space P , but there is no guarantee that \bar{P} is the complete reflective of P . The random selection of a parameter set may neglect the parameter vectors corresponding to the most significant error. These observations motivate to design a new criterion for the selection of the subset \bar{P} . In this work, we implement the adaptive greedy sampling technique.

6.3.4. Adaptive Greedy Sampling Method

To avoid the drawbacks associated with classical POD-greedy sampling technique, we implement an adaptive greedy approach. We propose to select the parameter groups adaptively at each iteration of the greedy procedure, using an optimized exploration approach. We construct a surrogate model $\bar{\varepsilon}$ to approximate the error estimator ε over the entire parameter space. Further, we use the surrogate model to locate the parameter groups \bar{P} , where the probability of having larger values of ε is the highest. The adaptive greedy sampling methods in the framework of MOR are well addressed in [120, 121, 116]. This approach is also implemented to obtain the reduced order models for a parameterized steady Navier Stokes equation [122], for elliptical PDEs [123], and for parabolic PDEs [124].

This report comprises the results obtained from the classical POD-greedy algorithm only. The future work will involve an implementation of the adaptive greedy POD approach and numerical results. Also, we will apply the Monte Carlo method to solve the short rate model. Furthermore, we aim to establish the same MOR approach associated with the newly defined Monte Carlo method.



6.4. Numerical Example

In this section, we compare the results of the POD method introduced in Subsect. 6.3.2 and the HDM results obtained based on the finite difference method (see Subsect. 6.3.1). We use a floater with cap and floor as a test example.

Table 4: Numerical Example of a floater with cap and floor.

Reference interest rate	Euribor3M
Fixing of Euribor3M	in advance
Coupon frequency	quarterly
Cap rate	2.25 % p.a.
Floor rate	0.5 % p.a.
Maturity and Nominal value	10 years with face value of €1000

We calculate the present value of an underlying instrument as a function of a short-rate r and time t . The interest rates are capped at 2.25 % p.a. and floored at 0.5 % p.a. with the reference rate as Euribor3M. The coupon rates can be written as

$$c = \min(2.25\%, \max(0.5\%, \text{Euribor3M})) \quad (6.100)$$

According to [80], if the yield curve $r(t_0, t)$ at time t_0 is given, then the Euribor3M can be computed using forward rate calculations as presented in Subsect. 6.2.4. Note that, the coupon rate $c^{(n)}$ at time t_n is set in advanced by the coupon rate at t_{n-1} . Consider an instrument with the 10 years of maturity and the coupons are paid quarterly, i.e., a total of 40 coupons need to be paid based on the coupon rate given by (6.100). (For example, if an instrument has a face value of €1000, and a coupon rate is 2% at a particular coupon date, then it pays a coupon of €20 at that date).

We can solve this problem using the Hull-White partial differential equation presented in Subsubsect. 6.2.8.4. In the following, we present details of the algorithm used for the valuation.

6.4.1. Technical and Software Details

All computations are carried out on a PC with 4 cores and 8 logical processors at 2.90 GHz (Intel i7 7th generation). We use MATLAB R2018a for the yield curve simulations. The numerical method for the yield curve simulations is tested with real market based historical data. We have collected the daily interest rate data at 26 tenor points in time over the past five years. Each year has 260 working days. Thus, there are 1300 observation periods. We have retrieved this data from the State-of-the-art stock exchange information system, "Thomson Reuters EIKON."

We have used the inbuilt UnRisk tool for the parameter calibration, which is well integrated with Mathematica (version used: Mathematica 11.3). Further, we use calibrated parameters for the construction of a Hull-White model. We have designed the FDM and MOR techniques for the solution of the Hull-White model in MATLAB R2018a.

6.4.2. Model Parameters

We compute the model parameters as explained in Sect. 6.2.10. The yield curve simulation is the first step to compute the model parameters. Based on the procedure described in Sect. 6.2.9, we perform the bootstrapping process for the recommended holding period of 10 years, i.e., for the maturity of the floater. The collected



historical data has 19 tenor points and 1306 observation periods as follows (D: Day, M: Month, Y: Year):

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

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

The ten thousand simulated yield curves are presented in Fig. 6.2. Furthermore, we calculate the model

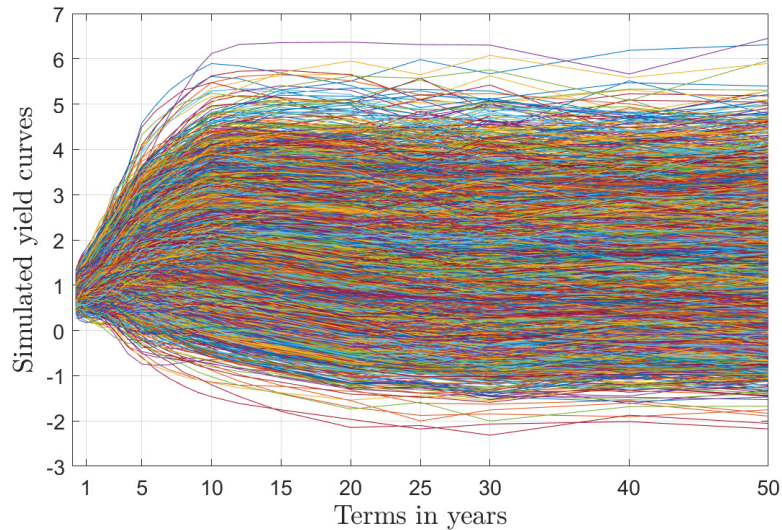


Figure 6.2: 10,000 simulated yield curves obtained by bootstrapping for 10 years in future.

parameters using the procedure explained in Section 6.2.10. For the floater example, we need parameter values only until 10Y tenor point (maturity of the floater). Henceforth, we consider the simulated yield curves with only first 12 tenor points. The calibration generates the real parameter space of dimension $\mathbb{R}^{10000 \times 12}$ for the parameter $a(t)$. We consider the volatility $\sigma(t)$ and the mean reversion $b(t)$ of the short-rate r as constants and equal to 0.005 and 0.1, respectively. The Figure 6.3 presents one of the 10,000 parameter vectors $a(t)$ as a function of time. All parameters are assumed to be piecewise constants between the tenor points (0 – 6M, 6M – 1Y, 1Y – 2Y, 2Y – 3Y, \dots , 9Y – 10Y).

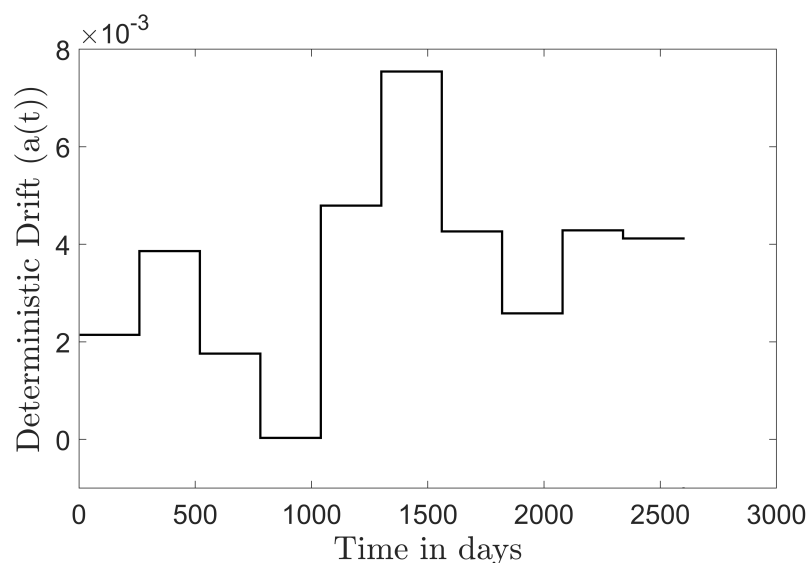


Figure 6.3: One of the 10,000 parameter vectors $a(t)$ as a function of time.



6.4.3. Finite Difference Method

The computational domain for a spatial dimension r is restricted to $r \in [\alpha, \beta]$. Here, $\alpha = 0.1$ and $\beta = -0.1$. Also, we apply Neumann boundary condition of the form

$$\frac{\partial V}{\partial r} \Big|_{r=\alpha} = 0, \quad \frac{\partial V}{\partial r} \Big|_{r=\beta} = 0. \quad (6.101)$$

We divide the spatial domain into M equidistant grid points which generate a set of points $\{r_1, r_2, \dots, r_M\}$. The time interval $[0, T]$ is divided into $N - 1$ time points. N points in time that are measured in days starting from $t = 0$ till the maturity T , i.e., in our case, the number of days until maturity are assumed to be 2600 with an interval $\Delta t = 1$ (10 years \approx 2600 days). Rewriting (6.87),

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

We can apply the first boundary condition by updating the first and the last rows (A_1 and A_M) of the matrix $A(\rho_s)$. Using an FDM, the discretization of (6.101) writes

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

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 (6.87). Note that, if we reach the coupon date, then we update the value of the grid point r_i by adding coupon f^n based on the coupon rate given by (6.100). Figure 6.4 shows the results obtained for the floater with cap and floor as a function of r and time t . The following results are obtained using the parameter vector $a(t)$ as illustrated in the Fig. 6.3, whereas $b = 0.1$, and $\sigma = 0.005$. We discretize the spatial domain into 800 equidistant grid points and the time interval $[0, T]$ is divided into $N - 1$ time points. N points in time that are measured in days starting from $t = 0$ till the maturity T , i.e., in our case, the number of days until maturity are assumed to be 2600 with an interval $\Delta t = 1$ (10 years \approx 2600 days).

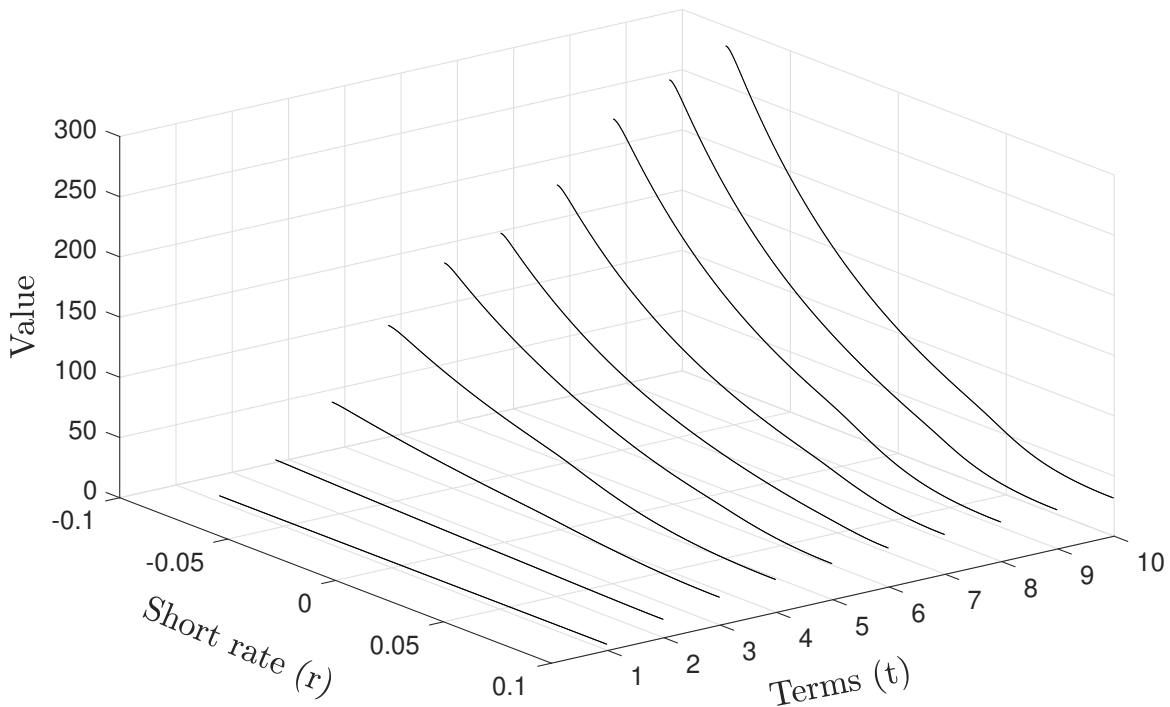


Figure 6.4: The value of the floater with 10 years of maturity and a face value of 1000 as a function of r and t .



6.4.4. Model Order Reduction

We implement the parametric model order reduction approach, as discussed in Subsect. 6.3.2. The quality of a ROM depends on the parameter groups selected for the construction of the ROB Q . In this work, we use the POD-greedy method to obtain Q . At each greedy iteration, the algorithm constructs a ROB using the POD approach as presented in the Algorithm 1. We set the maximum number of pre-defined candidates to construct a set \bar{P} to 40 and the maximum number of iteration I_{max} to 10.

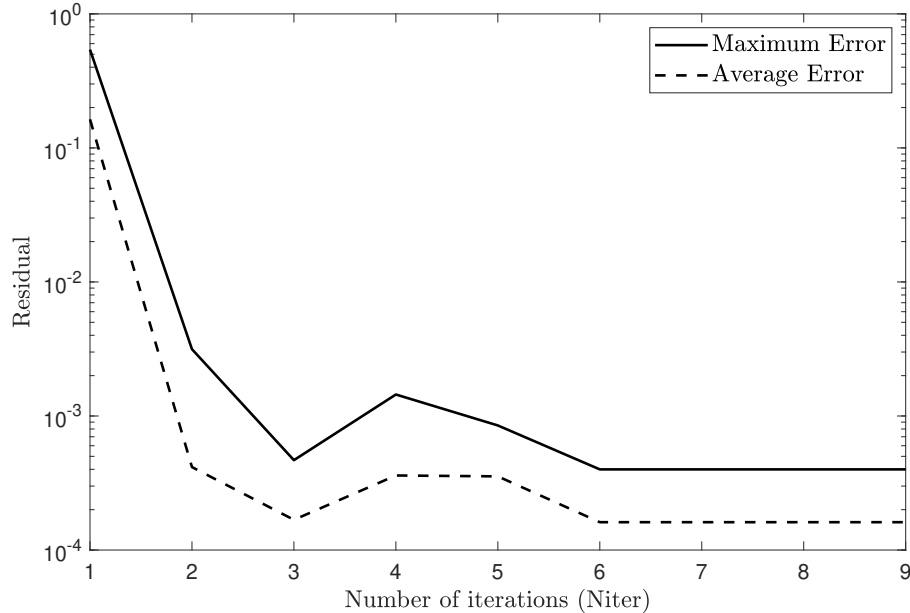


Figure 6.5: Evolution of maximum and average residuals with each iteration of the greedy-POD algorithm.

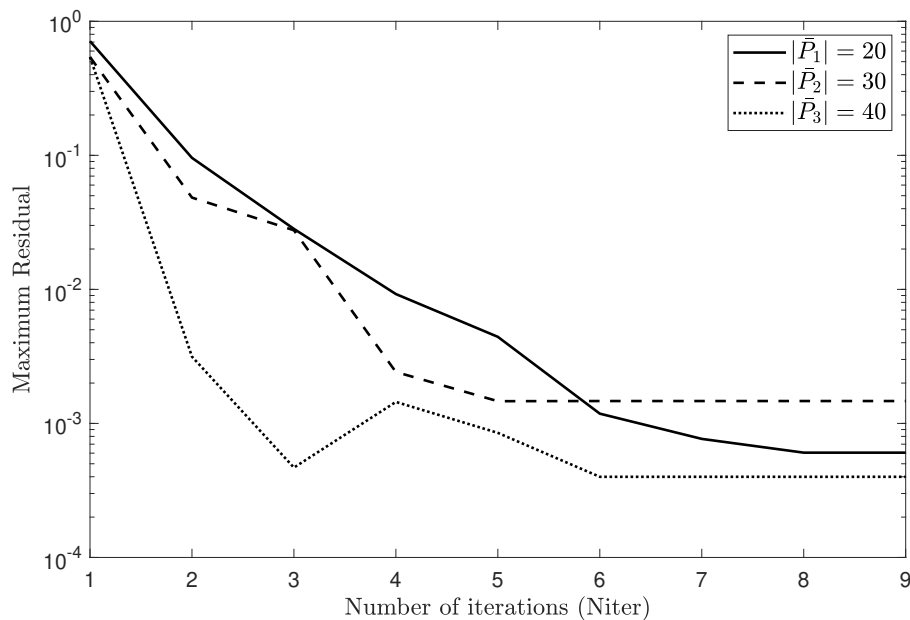


Figure 6.6: Evolution of the maximum residual error for three different cardinalities of set \bar{P} .

The progression of the maximum and average residuals with each iteration of the POD-greedy algorithm is presented in the Fig. 6.5. It is observed that the maximum residual error decreases with each proceeding iteration. We can say that the proposed greedy algorithm efficiently locates the optimal parameter groups and



construct the desired ROB Q . Furthermore, we test the effect of change in the cardinalities of the set \bar{P} . The proposed algorithm is applied with three different cardinalities of \bar{P} : $|\bar{P}_1| = 20$, $|\bar{P}_2| = 30$, $|\bar{P}_3| = 40$. Note that, we construct \bar{P} by randomly selecting the parameter groups from the parameter space P . Figure 6.6 shows the plot of the maximum residual against the number of iterations for three different cardinalities. It is evident that with an increasing number of candidates, the maximum residual error decreases. However, the decrement is not significant enough with an increase in the cardinality of \bar{P} even by 20. Thus, we can say that the twenty randomly selected parameter groups are enough to obtain the parameter independent ROB Q .

Using this projection subspace Q , we construct two different ROMs with two different parameter groups ρ_1 and ρ_2 . The relative errors between the HDMs and ROMs are presented in Fig. 6.7. It is observed that the increase in reduced dimension d improves the quality of the result. Also, we observe that the relative error between the ROM₂ and HDM₂ (dashed line) is larger than that of the ROM₁ and HDM₁ (solid line). This remark reveals

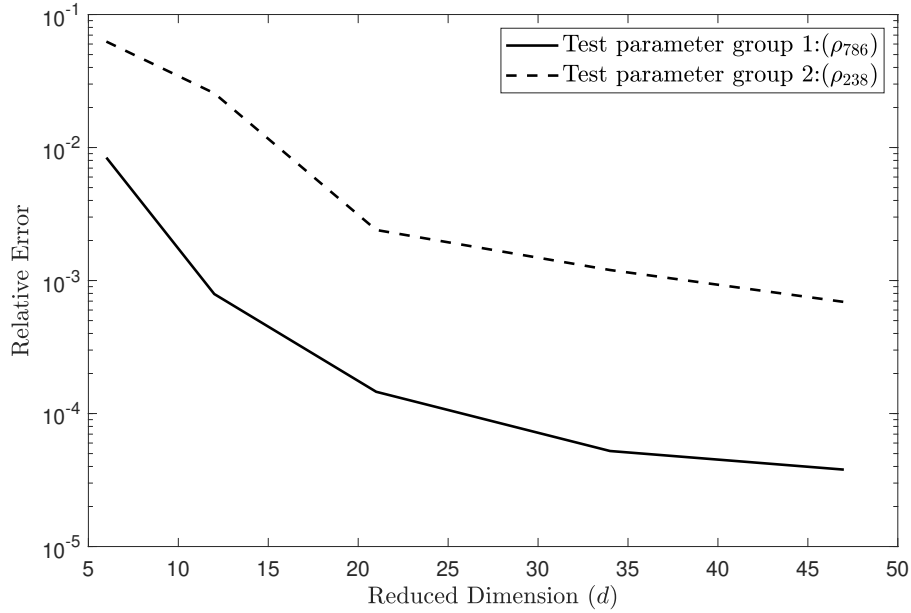


Figure 6.7: The relative error between the HDM and the ROM for two different parameter groups.

that the selection of trial candidates by random sampling may neglect the parameter groups corresponding to the significant error. To overcome these drawbacks of classic greedy-POD approach, we aim to define an adaptive greedy sampling technique. However, the results obtained with the reduced dimension $d = 25$ are still well within the acceptable range.

Table 5 shows the computational times required to generate the ROB Q with different sets of \bar{P} .

Table 5: Computation time/ reduction time (T_Q) to generate projection subspace.

Cardinality $ \bar{P} $	Maximum iterations I_{max}	Computational time
20	10	56.82 s
30	10	82.54 s
40	10	95.04 s

The computational times required to solve the ROMs and the HDMs is presented in Table 6. The evaluation columns give the time needed to solve the linear systems generated for both HDMs and ROMs. The time required to solve the complete system with a parameter space of $10000 \times m$ for both HDMs and ROMs is given in a total time column. We can see that the evaluation time required for the ROM is at least 8-10 times less



Table 6: Evaluation time.

Model	Evaluation time single ρ_s	Total Evaluation time (T_{eva})	Total time $T_Q + T_{eva}$
HDM, $M = 800$	0.1808 s	1895.87 s	1895.87 s
ROM, $ \bar{P} = 20$ $d = 5$	0.0104 s	108.56 s	165.38 s
ROM, $ \bar{P} = 20$ $d = 10$	0.0125 s	128.48 s	185.30 s
ROM, $ \bar{P} = 20$ $d = 15$	0.0155 s	156.31 s	213.13 s
ROM, $ \bar{P} = 20$ $d = 25$	0.0179 s	182.89 s	239.71 s
ROM, $ \bar{P} = 20$ $d = 30$	0.0212 s	214.79 s	271.61 s

than that of the HDM model. However, there is a slight increase in total time due to the addition of reduction time T_Q . Despite of that, the reduced system is at least 7-8 times faster than the high dimensional system. We can also observe that with an increase in the dimension of ROMs, the evaluation time increases as well. Nonetheless, the results obtained with $d = 25$ are satisfactory enough as even in the worst case the relative error is less than 10^{-2} , and it provides a computational speed of order 8.

6.4.4.1. Floater Scenario Values

To design a KID, 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 y_{s1} from the simulated yield curve. The value of a floater at the spot rate r_{sp} is nothing but the value at the 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 several thousand values are further used to calculate three different scenarios: (i) favorable scenario, (ii) moderate scenario, (iii) unfavorable scenario. The favorable, moderate, and unfavorable scenario values are the values at 90th percentile, 50th percentile and 10th percentile of 10,000 values respectively.

Table 7: Results for a floater with cap and floor.

Scenario	5 years	10 years
Favorable	1075.9	1082.9
Moderate	1057.8	1061.5
Unfavorable	1018.3	1025.4

6.5. Calibration of Hull-White Model

In the section, we have considered the parameters $b(t)$ and $\sigma(t)$ of the Hull-White model as constants. However, we can obtain those parameters as piecewise constants on the time interval. To calibrate the mean reversion speed and volatility, we need additional information about the cap data. We collect cap data for various strikes and various maturities. Consider a call option on a zero-coupon bond with strike price K and the expiration time T_1 . The bond maturity is T with $T > T_1$. Then according to [108], the price of the option in terms of



zero-coupon bond is given by

$$\begin{aligned}
 P_{\text{HWCap}} &= \mathbb{E} \left[\exp \left(- \int_0^{T_1} r(s) ds \right) (B(T_1, T) - K)^+ \right] \\
 &= \mathbb{E} \left[\exp \left(- \int_0^{T_1} r(s) ds \right) (\exp\{-r(T_1)\Gamma(T_1, T) - \Lambda(T_1, T)\} - K)^+ \right]
 \end{aligned} \tag{6.102}$$

The idea is to match the analytical cap prices $P_{\text{HWCap}}(K_i, T_i)$ with the market prices $P_{\text{MCap}}(K_i, T_i)$ by optimizing the parameters $b(t)$ and $\sigma(t)$. Let $a(t, b, \sigma)$ be the deterministic drift parameter obtain with the constant b and σ . The typical procedure now is to optimize $b(t)$ and $\sigma(t)$ with the pre-calculated $a(t)$ such that the analytical price approaches the collected market price, i.e., we minimize

$$\min_{b(t), \sigma(t)} \sum (P_{\text{HWCap}}^a(K_i, T_i) - P_{\text{MCap}}^a(K_i, T_i)) \tag{6.103}$$

where the superscript a , $P^a(\cdot)$, denotes that the parameter $a(t)$ has been calculated previously based on simulated yield curves.



Part VII.

Software-based representation of an inverse heat conduction problem

Umberto Morelli, Giovanni Stabile, Gianluigi Rozza, Federico Bianco, Gianfranco Marconi, Peregrina Quintela, Patricia Barral

Abstract

To analyze different techniques for the solution of inverse heat conduction problems, we designed a benchmark. In the first part of this document, we provide a description of the numerical formulation of this benchmark and the method used for the solution of the inverse heat transfer problem. The second part of the document is then devoted to the numerical solution of the inverse problem, its software implementation and all the information needed for the reproducibility of the numerical results.

Keywords: Inverse problem, heat transfer, casting mold

7.1. Introduction

In this report, we discuss the numerical solution of the following inverse problem: the reconstruction of the boundary condition of an heat transfer problem in a solid domain using temperature measurements inside the domain.

We start considering the steady heat conduction problem in an homogeneous isotropic solid in the rectangular parallelepiped domain Ω of Figure 7.1

Problem 1. Given $k \in \mathbb{R}^+$, $g \in L^2(\Gamma_{in})$, $H \in \mathbb{R}^+$ and $T_f \in L^2(\Gamma_{in})$. Find the temperature $T : \Omega \rightarrow \mathbb{R}^+$ such that

$$-k\Delta T(\mathbf{x}) = 0, \quad \forall \mathbf{x} \in \Omega, \quad (7.1)$$

with BC

$$\begin{cases} -k\nabla T(\mathbf{x}) \cdot \mathbf{n} = g(\mathbf{x}) & \forall \mathbf{x} \in \Gamma_{in}, \end{cases} \quad (7.2)$$

$$\begin{cases} -k\nabla T(\mathbf{x}) \cdot \mathbf{n} = 0 & \forall \mathbf{x} \in \Gamma_{ex}, \end{cases} \quad (7.3)$$

$$\begin{cases} -k\nabla T(\mathbf{x}) \cdot \mathbf{n} = H(T(\mathbf{x}) - T_f(\mathbf{x})) & \forall \mathbf{x} \in \Gamma_{sf}. \end{cases} \quad (7.4)$$

We will call this the direct problem.

We state now the correspondent inverse problem. We assume to know the correct measured temperature $\tilde{T}(\mathbf{x}_i)$ for all $x_i, i = 1, 2, \dots, M$. We interpolate the measured temperatures in a plane Σ , which is defined by the thermocouples, obtaining an approximation of the function $\tilde{T}(\mathbf{x})|_{\Sigma}$. Then, we state the inverse problem as

Problem 2. Given the temperature measurements $\tilde{T}(\mathbf{x}) \in \mathbb{R}^+$, for all $x \in \Sigma$, find $g(\mathbf{x}) \in L^2(\Gamma_{in})$ which minimizes the functional

$$J[g] = \frac{1}{2} \int_{\Sigma} (T[g](\mathbf{x}) - \tilde{T}_{int}(\mathbf{x}))^2 d\gamma, \quad (7.5)$$

where $T[g](\mathbf{x})$ is solution of **Problem 1**.

To solve **Problem 2**, we used Alifanov's regularization method [125]. It is an iterative procedure in which the function $g(\mathbf{x})$ is updated at each iteration

$$g^{n+1}(\mathbf{x}) = g^n(\mathbf{x}) - \beta^n P^n(\mathbf{x}), \quad n = 0, 1, 2, \dots \quad (7.6)$$



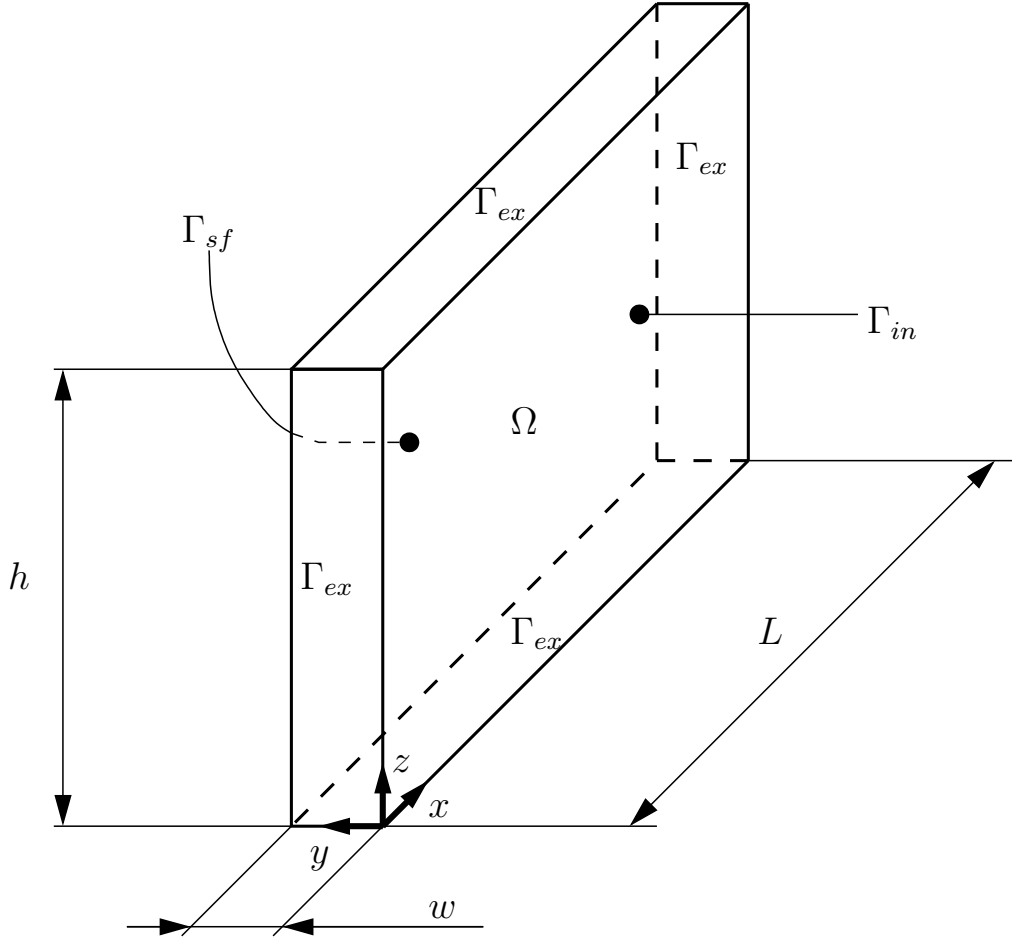


Figure 7.1: Schematic of the solid rectangular parallelepiped domain.

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

$$P^n(\mathbf{x}) = J'_{g^n}(\mathbf{x}) + \gamma^n P^{n-1}(\mathbf{x}), \quad (7.7)$$

where γ^n is the conjugate coefficient with $\gamma^0 = 0$, and $J'_g(\mathbf{x})$ is the Fréchet derivative (gradient) of J . It is the element of $L^2(\Gamma_{sin})$ that represents the Gâteaux derivative (directional derivative) of J , dJ_g , with respect to the inner product $\langle \cdot, \cdot \rangle_{L^2(\Gamma_{sin})}$ (see [126]), i.e. such that

$$dJ_g[b] = \langle J'_g, b \rangle_{L^2(\Gamma_{sin})}, \quad \forall b \in L^2(\Gamma_{sin}). \quad (7.8)$$

Minimizing the functional $J[g^{n+1}] = J[g^n - \beta P^n]$ with respect to β , we find the search stepsize β^n which is used in (7.6). It is the solution of the critical point equation of the functional J , restricted to a line passing through $g^n(\mathbf{x})$ in the direction defined by P^n , i.e. β^n is the critical point of $J[g^n - \beta P^n]$ which then satisfies

$$J[g^n - \beta^n P^n] = \min_{\beta} \left\{ \frac{1}{2} \sum_{i=1}^M \{T[g^n - \beta P^n](\mathbf{x}_i) - \hat{T}(\mathbf{x}_i)\}^2 \right\}. \quad (7.9)$$

In initial and boundary value problems for linear PDE, homogeneous or not, β^n is given by (see [127])

$$\beta^n = \frac{\sum_{i=1}^M \{T[g^n](\mathbf{x}_i) - \hat{T}(\mathbf{x}_i)\} \delta T[P^n](\mathbf{x}_i)}{\sum_{i=1}^M (\delta T[P^n](\mathbf{x}_i))^2}. \quad (7.10)$$



where $\delta T[P](\mathbf{x})$ is the linear operator that solves the sensitivity problem with input P . The sensibility problem is presented later in this section.

To conclude, we compute the conjugate coefficient minimizing $J[g^n - \beta^n P^n]$ with respect of the possible choices of γ^n in the definition of P^n . We then write (see [127])

$$\gamma^n = \frac{\int_{\Gamma_{sin}} [J'_{g^n}(\mathbf{x})]^2 d\mathbf{x}}{\int_{\Gamma_{sin}} [J'_{g^{n-1}}(\mathbf{x})]^2 d\mathbf{x}}, \quad (7.11)$$

which minimizes $J[g^n - \beta^n P^n]$ with respect of the possible choices of γ^n in the definition of P^n .

To use this iterative procedure, we have to compute at each iteration the gradient $J'_g(\mathbf{x})$ and the variation $\delta T[P]$ which are solutions of the adjoint and sensitivity problem respectively.

The sensitivity problem is obtained by perturbing the heat flux $g \rightarrow g + \delta g$, causing a variation of the temperature field, $T \rightarrow T + \delta T$. The sensitivity problem corresponds to the problem satisfied by δT [127].

Problem 3. Find $\delta T : \Omega \rightarrow \mathbb{R}^+$ such that

$$-k\Delta\delta T(\mathbf{x}) = 0, \quad \forall \mathbf{x} \in \Omega, \quad (7.12)$$

with BC

$$\begin{cases} -k\nabla\delta T(\mathbf{x}) \cdot \mathbf{n} = \delta g(\mathbf{x}) & \forall \mathbf{x} \in \Gamma_{in}, \\ -k\nabla\delta T(\mathbf{x}) \cdot \mathbf{n} = 0 & \forall \mathbf{x} \in \Gamma_{ex}, \\ -k\nabla\delta T(\mathbf{x}) \cdot \mathbf{n} = H(\delta T(\mathbf{x})) & \forall \mathbf{x} \in \Gamma_{sf}. \end{cases} \quad (7.13)$$

$$\quad \quad \quad (7.14)$$

$$\quad \quad \quad (7.15)$$

The gradient of the functional $J[g]$ evaluated at g, J'_g , is

$$J'_g(\mathbf{x}) = -\lambda(\mathbf{x}) \quad \forall \mathbf{x} \in \Gamma_{in}. \quad (7.16)$$

Where λ is solution of

Problem 4. (Adjoint) Find $\lambda : \Omega \rightarrow \mathbb{R}$ such that

$$\frac{1}{k}\Delta\lambda(\mathbf{x}) + (T[g](\mathbf{x}) - \tilde{T}(\mathbf{x}))\delta(\mathbf{x} - \mathbf{x}_i) = 0, \quad \forall \mathbf{x} \in \Omega, \mathbf{x}_i \in \Sigma, \quad (7.17)$$

with BC

$$\begin{cases} \frac{1}{k}\nabla\lambda(\mathbf{x}) \cdot \mathbf{n} = 0 & \forall \mathbf{x} \in \Gamma_{in} \cup \Gamma_{ex}, \\ \frac{1}{k}\nabla\lambda(\mathbf{x}) \cdot \mathbf{n} + \frac{1}{k^2}H\lambda(\mathbf{x}) = 0 & \forall \mathbf{x} \in \Gamma_{sf}. \end{cases} \quad (7.18)$$

$$\quad \quad \quad (7.19)$$

Finally, the conjugate gradient algorithm is

Algorithm 1.

1. Let $n = 0$. Choose an estimate, $g^0(\mathbf{x})$.
2. Compute $T[g^n](\mathbf{x}_i)$, $i = 1, \dots, M$, by solving **Problem 1**.
3. Examine the stopping criteria. Stop if it is satisfied.
4. Solve the **Problem 4** to determine the gradient J'_{g^n} .
5. Compute the conjugate coefficient, γ^n , by (7.11).



6. Compute the search direction, $P^n(\mathbf{x})$, by (7.7)
7. Solve the **Problem 3** with $\delta g(\mathbf{x}) = P^n(\mathbf{x})$ and obtain $\delta T[P^n](\mathbf{x})$.
8. Compute the stepsize in the search direction, β^n , by (7.10).
9. Compute the new estimate $g^{n+1}(\mathbf{x})$, with (7.6).
10. Set $n = n + 1$ and return to Step 2.

Regarding the numerical solution of the three problems, we use the finite volume method for the discretization. Given a discretization of the domain \mathcal{T} , we write the discrete unknowns $(T_C)_{C \in \mathcal{T}}$, $(\lambda_C)_{C \in \mathcal{T}}$ and $(\delta T_C)_{C \in \mathcal{T}}$ into the real vectors \mathbf{T} , $\boldsymbol{\lambda}$ and \mathbf{ffiT} , respectively, belonging to \mathbb{R}^{N_h} with $N_h = \text{size}(\mathcal{T})$. Then, we write the finite volume schemes as linear systems for the three problems. In the direct problem, we split the source term in one term due to the BC at Γ_{in} , \mathbf{b}_{T_g} , and one due to the other BC,

$$A_T \mathbf{T} = \mathbf{b}_{T_g} + \mathbf{b}_T. \quad (7.20)$$

The source term of the adjoint problem includes a first term due to the BC, \mathbf{b}_λ , and a second due to the difference between measured and computed temperatures, \mathbf{f}_λ ,

$$A_\lambda \boldsymbol{\lambda} = \mathbf{b}_\lambda + \mathbf{f}_\lambda. \quad (7.21)$$

Finally we treat the sensitivity problem as the direct problem obtaining

$$A_{\delta T} \mathbf{ffiT} = \mathbf{b}_{\delta T_g} + \mathbf{b}_{\delta T}, \quad (7.22)$$

where $A_{T,\lambda,\delta T} \in \mathbb{R}^{N_h \times N_h}$ and $\mathbf{b}_{T,\lambda,\delta T} \in \mathbb{R}^{N_h}$ are the finite volume stiffness matrixes and source terms, respectively. The elements of the stiffness matrices and source terms depend on the finite volume scheme used for the discretization and the mesh used. For further details regarding the finite volume discretization we refer to [128].

7.2. Implementation and Computer Requirements

All numerical computations are performed using ITHACA-FV [129] which is freely available under the GPL 3 License. It is an open-source C++ library based on the finite volume solver OpenFOAM [130]. All the updated informations regarding installation and usage are available at <https://mathlab.sissa.it/ithaca-fv>.

7.3. Numerical Example

We present here a numerical example. Considering the domain in Figure 7.1, the used parameters are summarized in Table 8. Given the simple geometry, we use a structured, orthogonal mesh of $3 \cdot 10^5$ cells. The elements are uniformly distributed in the three directions in the following way: 100 elements along the x- and z-axis while 30 elements were used along the y-axis. We can then solve the direct problem obtaining the temperature field in the domain.

The objective of this example is to reconstruct the heat flux $g(\mathbf{x})$ based on temperature measurements which are taken inside the domain. Here the temperature measurements are given by the solution of the direct problem. We select 15 points in the middle plane of the domain as virtual thermocouples and we use the procedure previously discussed to reconstruct the boundary condition $g(\mathbf{x})$.

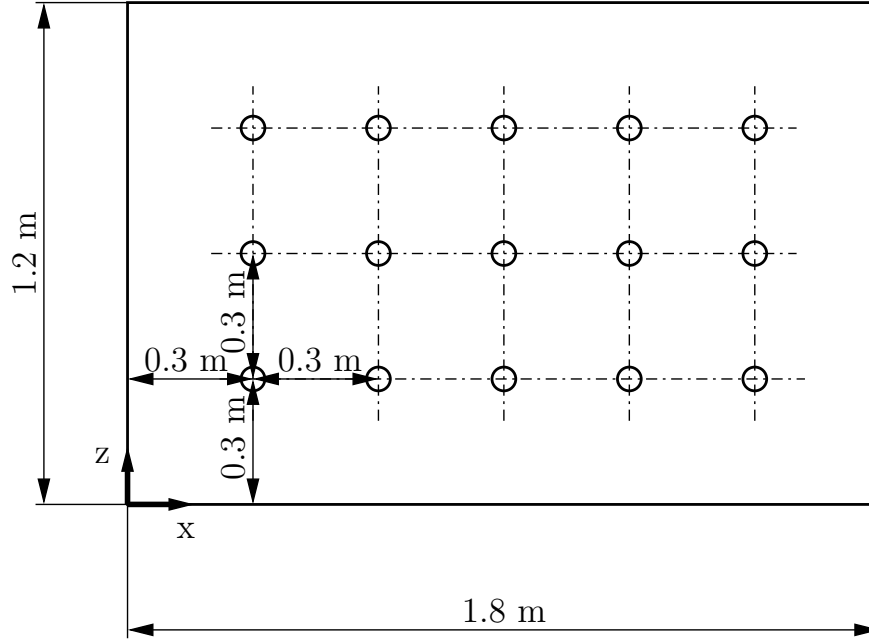
To have a measure of the goodness of the heat flux estimation, we define the relative error

$$\epsilon_{rel}(\mathbf{x}) = \frac{g_{inv}(\mathbf{x}) - g(\mathbf{x})}{g(\mathbf{x})}, \quad (7.23)$$



Table 8: Parameters used in the numerical example.

Parameter	Value
Thermal conductivity, k	300.0 W/(mK)
Heat transfer coefficient, H	6000.0 W/(m ² K)
Heat flux, $g(\mathbf{x})$	$-10^5 x \cdot z$ W/m ²
Convective flow temperature, T_f	300 K
L	1.8 m
w	0.1 m
h	1.2 m

**Figure 7.2:** Thermocouples locations at the plane $y = 0.05$ m. They are uniformly distributed along the z - and x -axis.

where $g_{inv}(\mathbf{x})$ is the reconstructed boundary condition. Moreover, we define the surfaces

$$\Sigma = \{(x) \in \Omega | 0.3 \text{ m} \leq x \leq 1.5 \text{ m}, y = 0.5 \text{ m}, 0.3 \text{ m} \leq z \leq 0.9 \text{ m}\}, \quad (7.24)$$

which is the plane defined by the thermocouples, and

$$\Sigma_{in} = \{(x) \in \Gamma_{in} | 0.3 \text{ m} \leq x \leq 1.5 \text{ m}, 0.3 \text{ m} \leq z \leq 0.9 \text{ m}\}, \quad (7.25)$$

which is the projection of Σ on Γ_{in} . We interpolate the measurements on the plane Σ using radial basis functions with Gaussian kernel functions.

The conjugate gradient algorithm meets the relative tolerance criterium in 81 iterations. Figure 7.3 illustrates the convergence history and the behavior of the relative error norms. The minimum value of the functional J at the 81st iteration is $1.0453 \cdot 10^{-6} \text{ K}^2/\text{m}^2$. The correspondent $L^2(\Sigma_{in})$ - and $L^\infty(\Sigma_{in})$ -norm of the relative error are $1.287 \cdot 10^{-2} \text{ m}^{-1}$ and 1.474, respectively. In Figure 7.4 and Figure 7.5, we see a comparison of the temperature fields at the thermocouples plane and of the heat fluxes.



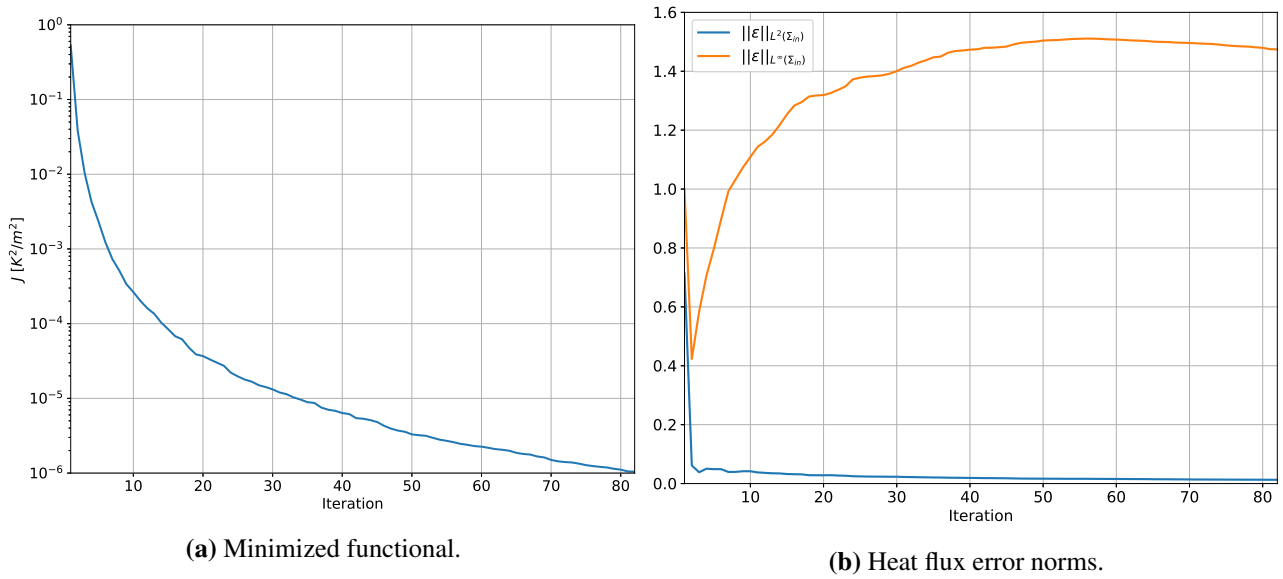


Figure 7.3: Convergence history and error norms of the conjugate gradient algorithm.

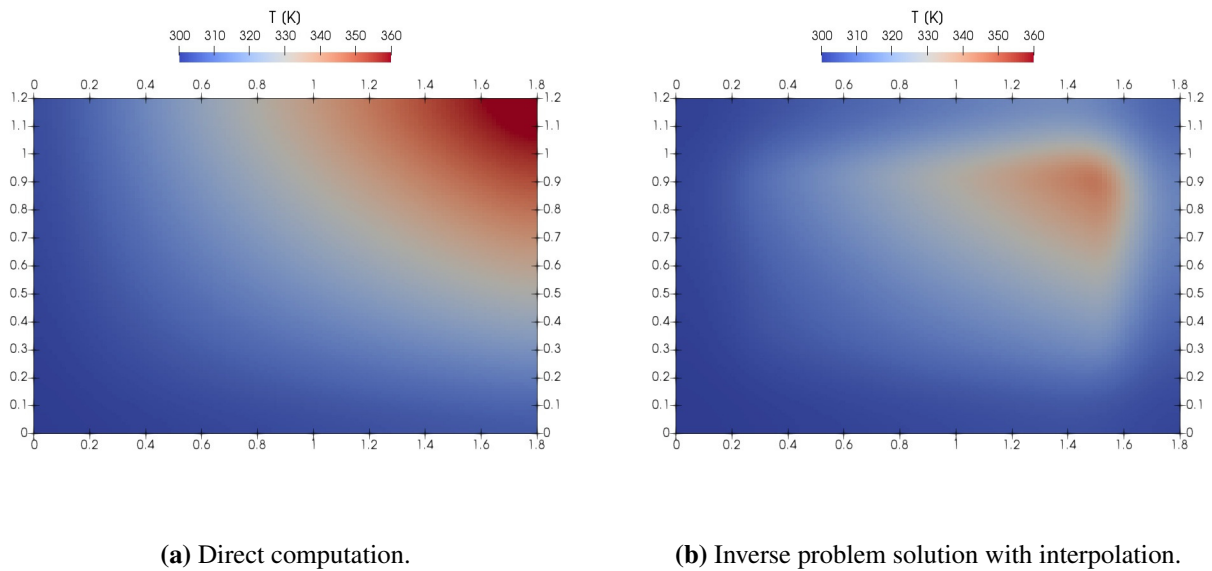
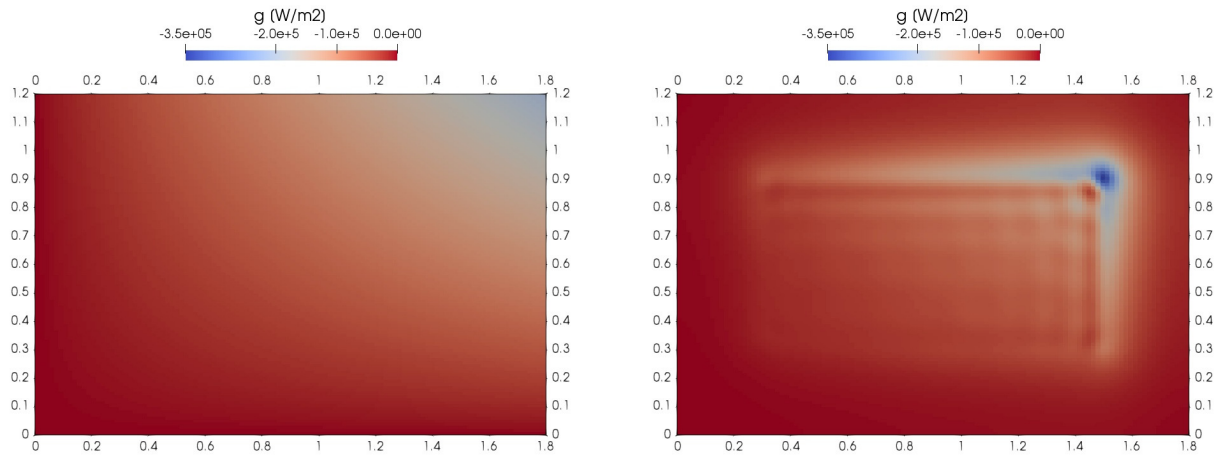
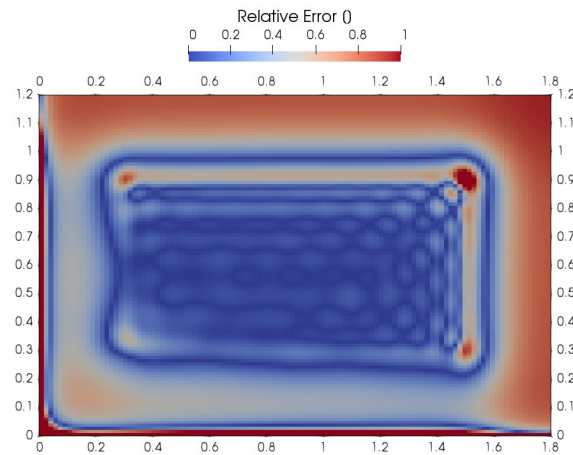


Figure 7.4: Temperature field at $y = 0.05$ m.



(a) Direct computation.

(b) Inverse problem solution.



(c) Relative error.

Figure 7.5: Heat flux at T_{in} .

Part VIII.

Software-based representation and XFEM-based implementation of the benchmark entitled '*Experimental-based Validation of FSI Simulations in Blood Pumps*'

Marco Martinolli, Christian Vergara, Luc Polverelli

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 measure the pressure rise over the pump and the hydraulic power for different inflow velocities and finally compare the results with the experimental PQ and HQ curves. The software for the solution of the benchmark will be implemented in the C++ parallel library of finite elements LIFEV and tackled 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, LIFEV, PQ curves.

The benchmark consists in the validation of a numerical method to solve the Fluid-Structure Interaction (FSI) problem in membrane-based blood pumps against real data provided by CorWave Inc. This report is structured as follows: in Section 8.1, we briefly recall the most important details about the blood pump technology and the fluid-structure model; in Section 8.2, we describe the structure and the format of the experimental data provided by CorWave Inc.; in Section 8.3, we depict the plan for the benchmark for the model validation; and finally, in Section 8.4, we shortly present our numerical approach to solve the benchmark, together with the details of the software environment and the computer requirements to run the benchmark. Licences of use and configuration files can be found in the Appendix 8.5.

8.1. 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 [131]. 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 [132, 133]. You can find a cross sectional view of the pump device in the left panel of Figure 8.1. The possibility to simulate the 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).

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. 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



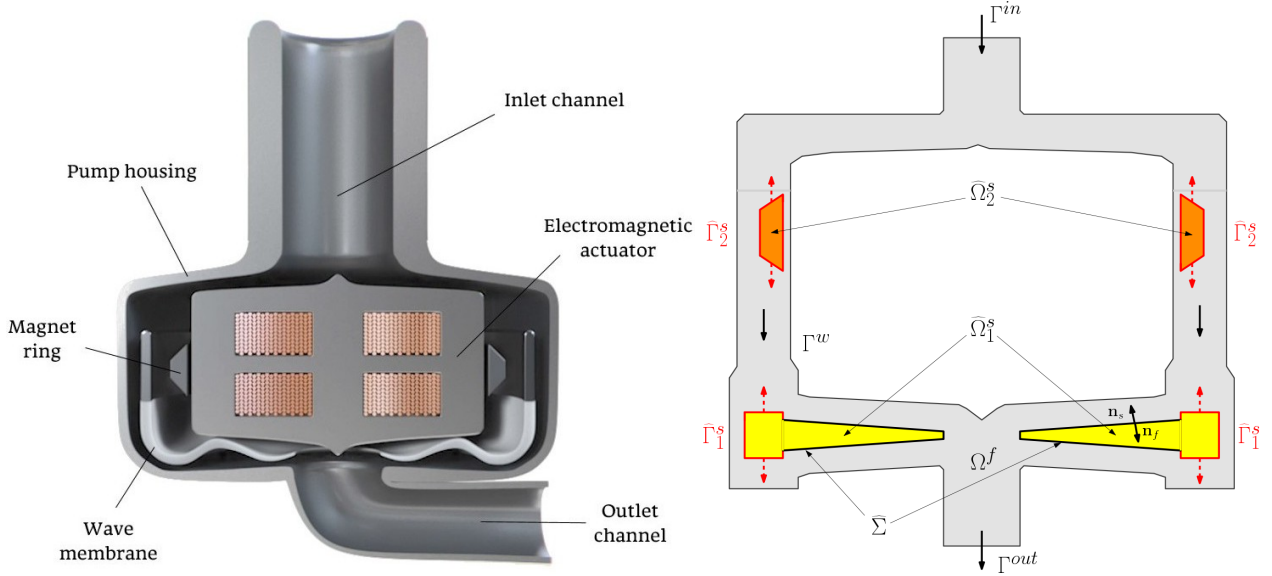


Figure 8.1: Left: Cross section of the implantable progressive wave pump. The blood enters from the superior inlet channel, it flows down along the sides of the central actuator body, it interacts dynamically with the wave silicon membrane and it is ejected into the inferior outlet channel. Right: Sketch of the pump domain.

domain $\widehat{\Omega}^s = \Omega^s(0)$, such that:

$$\rho_f (\partial_t \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u}) - \nabla \cdot \mathbf{T}^f(\mathbf{u}, p) = \mathbf{0} \quad \text{in } \Omega^f(\mathbf{d}^f), \quad (8.1)$$

$$\nabla \cdot \mathbf{u} = 0 \quad \text{in } \Omega^f(\mathbf{d}^f), \quad (8.2)$$

$$\rho_s \partial_{tt} \widehat{\mathbf{d}} - \nabla \cdot \widehat{\mathbf{T}}^s(\widehat{\mathbf{d}}) = \mathbf{0} \quad \text{in } \widehat{\Omega}^s, \quad (8.3)$$

$$\mathbf{d}^f = \mathbf{d} \quad \text{on } \Sigma(\mathbf{d}^f), \quad (8.4)$$

$$\mathbf{u} = \partial_t \mathbf{d} \quad \text{on } \Sigma(\mathbf{d}^f), \quad (8.5)$$

$$\mathbf{T}^f(\mathbf{u}, p) \mathbf{n} = \mathbf{T}^s(\mathbf{d}) \mathbf{n} \quad \text{on } \Sigma(\mathbf{d}^f). \quad (8.6)$$

with proper boundary and initial conditions. Equations (8.1) and (8.2) are the Navier-Stokes (NS) equations modeling the conservation of momentum and mass of an incompressible viscous Newtonian fluid. Equation (8.3), written in the Lagrangian formulation with the quantities defined in the reference configuration ($\widehat{\cdot}$ supercript), is the elastodynamic equation that models the structure deformation. Equation (8.4) provides the adherence condition between the fluid and the solid domains, and Equations (8.5) and (8.6) 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 (8.4). 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 $\mathbf{D}(\mathbf{w}) = \frac{1}{2}(\nabla \mathbf{w} + \nabla \mathbf{w}^T)$
- $\widehat{\mathbf{T}}^s(\widehat{\mathbf{d}})$ is the first Piola-Kirchhoff tensor of the structure, such that $\widehat{\mathbf{T}}^s(\widehat{\mathbf{d}}) = J \mathbf{T}^s(\mathbf{d}) \mathbf{F}^{-T}$ 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.



In our FSI problem, the structures interacting with the fluid are two (as shown in the right panel of Figure 8.1): the silicon membrane and the magnet ring. Therefore, the system includes two structural problems in $\hat{\Omega}_1^s$ and in $\hat{\Omega}_2^s$, to compute the displacement of the membrane $\hat{\mathbf{d}}_1$ and of the magnet ring $\hat{\mathbf{d}}_2$, and the two fluid-structure interfaces Σ_1 and Σ_2 , where the coupling conditions are prescribed. The forcing term of the system is represented by the forced vibrations induced by the electromagnetic actuator on the periphery of the membrane disc $\Gamma_1^s \subsetneq \Sigma_1$ and on the surface of the magnet ring $\Gamma_2^s \equiv \Sigma_2$. The imposed motion is modeled by means of a pair of sinusoidal Dirichlet conditions:

$$\mathbf{d}_1(t) = \Phi \sin(2\pi f t) \mathbf{e}_z \quad \text{on } \Gamma_1^s, \quad (8.7)$$

$$\mathbf{d}_2(t) = \Phi \sin(2\pi f t) \mathbf{e}_z \quad \text{on } \Gamma_2^s, \quad (8.8)$$

where Φ and f are respectively the amplitude and the frequency of the forced oscillation of both the membrane and the magnet ring. The other boundary conditions of the model are: parabolic velocity profile at the inlet section Γ^{in} (non-homogeneous Dirichlet condition), force-free condition at the outlet section Γ^{out} (homogeneous Neumann condition) and non-slip condition at the pump walls Γ^w (homogeneous Dirichlet condition).

8.2. Experimental Data

The experimental data used for the benchmark are Pressure-Flow (PQ) and Hydraulic power-Flow (HQ) curves, which are obtained during the *in vitro* testings of the pump system in a Mock circulation loop, i.e., a system of pipes that mimics the cardiac apparatus in physiologic conditions. Blood is replaced by a mixture of water and glycerin (39% in weight), while a mechanical actuator reproduces the action of the implantable pump. The system is also equipped with pressure sensors placed both at the inlet and at the outlet sections, to measure the pressure gradient arisen over the pump, and an ultrasonic flowmeter clamped at the outlet pipe, to measure the outflow volume rate of the pump. The PQ curves are obtained by varying the volume flow rate by means of a control hand valve and displaying the respective measurements of pressure in the PQ plan. Analogously, the data can be represented in the HQ plan, computing the hydraulic power H as the product between the pressure rise over the pump P and the flow volume rate Q and producing the HQ curves. Figure 8.2 shows the PQ and 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 actuator - passes from 0.53 mm (grey triangles), to 0.63 mm (yellow crosses) and finally to 0.73 mm (blue dots).

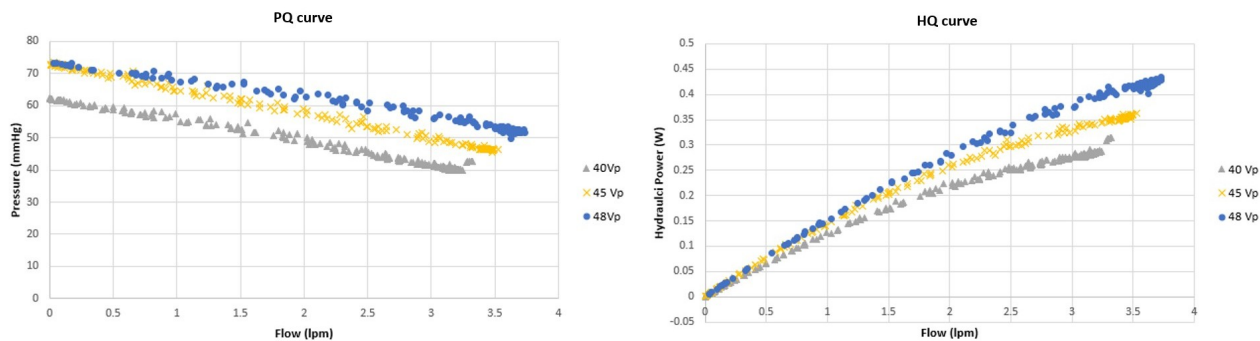


Figure 8.2: PQ and HQ experimental curves for three different values of the voltage of the actuator.

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:

- outflow volume rate, expressed in liters per minute [lpm] ($1 \text{ lpm} = 0.06 \text{ cm}^3/\text{s}$);
- pressure rise over the pump, expressed in millimeters of mercury [mmHg] ($1 \text{ mmHg} = 1333.22 \text{ g/cm}^2 \text{ s}^2$);
- hydrodynamic power, expressed in Watts [W] ($1 \text{ W} = 1e7 \text{ g cm}^2/\text{s}^3$).

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 Volts [V], shown in the Figure above, and the respective amplitude of the induced membrane vibrations Φ , expressed in centimeters [cm].

8.3. Benchmark Plan

The aim of the proposed benchmark is to validate the FSI model via comparison of the experimental PQ and HQ curves with the numerical results. Analogously to the approach for model validation used by Perschal et al. in a similar scenario [134], 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.

```

1: [Q_data, P_data, H_data, M_data] = read_data(PQcurves, HQcurves, settings);
2: [Q_Ndata, P_Ndata, H_Ndata, M_Ndata] = extract_Ndata(N);           // N ≥ 1 data points
3: for n = 1 : N do
4:   input_flow = Q_Ndata[n];
5:   input_frequency = M_Ndata[n].f;
6:   input_amplitude = M_Ndata[n].phi;
7:   define_bc(input_flow, 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:    t = t + dt;
12:  end while
13:  P_output[n] = mean(p|outlet) - mean(p|inlet);                   // Averages in space and time
14:  H_output[n] = P_output[n]*input_flow;
15: end for
16: plot(Q_data, P_data, Q_Ndata, P_output);                           // PQ curves
17: err_PQ = norm2(P_Ndata - P_output);
18: plot(Q_data, H_data, Q_Ndata, H_output);                           // HQ curves
19: err_HQ = norm2(H_Ndata - H_output);

```

Algorithm 2: Benchmark structure: Model validation against N data points.

The plan for the model validation is described in the Algorithm 2. First, the experimental data coming from the PQ and the HQ curves are stored in separated variables: Q_data contains the measurements of the out-flow volume rate, P_data collects the data of the pressure rise between outlet and inlet, H_data contains the measurements of the hydraulic power, and M_data stores the input settings for the parameters of membrane displacement (line 1). Only N data points are extracted by the curves to be reproduced via simulation and used for the model validation (line 2). 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). In this way, the pump system is simulated in the time interval $[0, T_{max}]$ using the XFEM numerical approach (lines 8-12). Then, the pressure rise can be estimated as the difference between the mean values of the pressure over the outlet and the inlet surfaces and averaged in time over the last 2 periods of membrane oscillation (line 13); while the hydraulic power can be computed as the product between the predicted pressure rise and the input inflow rate (line 14). Finally, the PQ curves and the HQ curves can be plotted together with the numerical outputs and an error measure can be computed as the 2-norm of the deviation of the numerical results from the experimental data (lines 15-19).

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.



8.4. Implementation and Computer Requirements

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 Extended Finite Element Method (XFEM) [135, 136]. The XFEM technique is an unfitted method 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. On the other side, since the structure mesh moves in the foreground cutting the underlying fluid mesh, the XFEM approach 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 [137].

The software for the benchmark will be implemented in Library of Finite Elements V (LIFEV) (<https://bitbucket.org/lifev-dev/lifev-env.git>) [138], 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) [139] 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 8.5.A 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 configdelure 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 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 2: defs.sh

3. Clone the source codes of LIFEVEV 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: clone.sh

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

4. Build TetGen and configure LIFEVEV 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 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 8.5.B).

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 5: build.sh

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 on 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.



8.5. Appendix

8.5.A. Licences of Use

8.5.A.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/>.

8.5.A.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.

8.5.A.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.)

8.5.B. 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 6: config.sh



Part IX.

Benchmark for numerical simulation of thermo-mechanical phenomena arising in blast furnaces

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

Abstract

High thermal stress inside the blast furnace hearth walls limits the lifetime of the blast furnace hearth and in turn reduces the blast furnace campaign. We present here linear, time-invariant, weakly-coupled, axisymmetric model for computing thermo-mechanical stress. Proper selection of function spaces along with symmetry conditions ensure that the model is well defined and is well posed. In order to validate the simulation results, benchmark tests are performed.

Keywords: Blast furnace hearth, thermo-mechanical coupled model, weak formulation, benchmark verification.

9.1. Conceptual model

Blast furnace is a metallurgical furnace used to produce iron from “charge”. The “charge” contains iron ore, limestone, coke. The process involves an exothermic reaction, the oxidation of carbon. High temperature is required to increase the oxidation rate of carbon. In the hearth, the hot air, supplied through the tuyeres, acts as source of oxygen and is used for the oxidation of the carbon. The temperature in the hearth is as high as 1500 °C. The general layout of the blast furnace is shown in Figure 9.1.



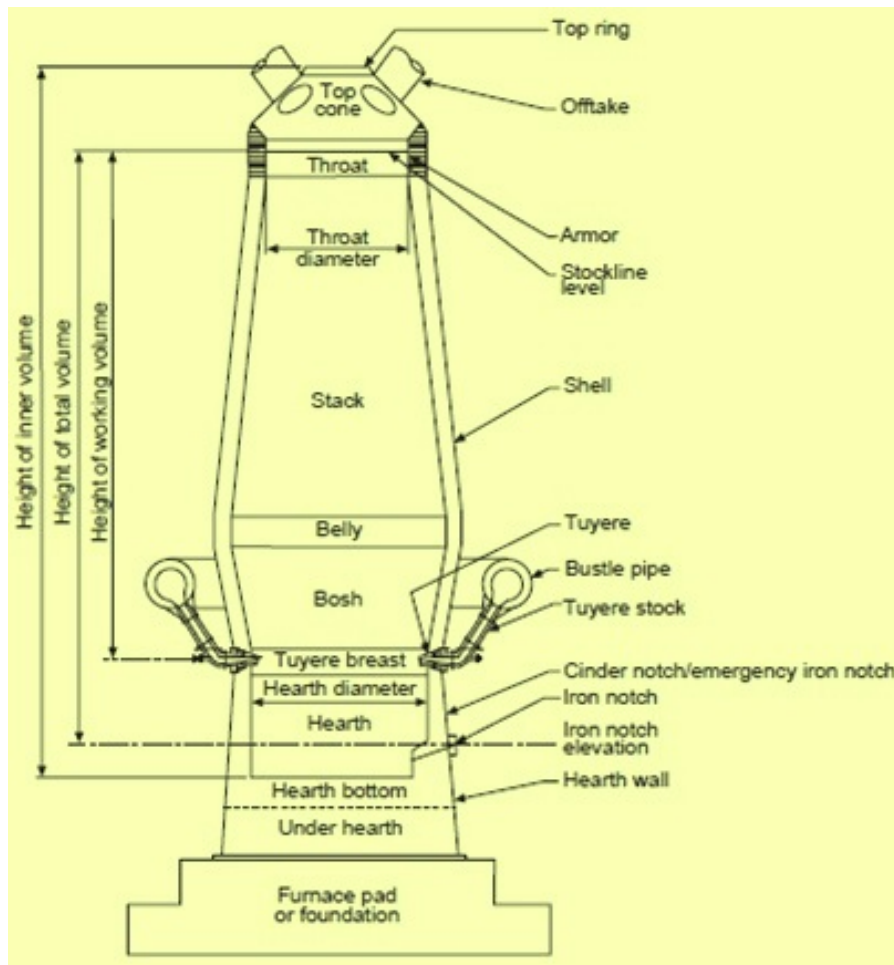


Figure 9.1: Blast furnace divided in zones [140]

As mentioned by Swartling M. et. al. [141], the hearth is made up of several refractory zones (Figure 9.2), with each zone having specific functions and, accordingly, design requirements. The outside of the hearth is covered with a steel shell. The wall and the bottom consist of five types of refractory materials, each with its specific properties depending on type of environment the region is exposed to. The wall is made up of carbon material having relatively high heat conductivity to keep the inner wall at a low temperature. The upper layer of the bottom is a ceramic plate with the composition 23.5 % SiO_2 , 73.5 % Al_2O_3 , and some other additives. It has high resistance to mechanical wear, which helps to keep the bottom layer intact and avoid cracks. That is important to avoid penetration and solidification of liquid iron in the lower bottom layers, since this would have impact on the heat flow. Between the steel shell and the bottom refractory, there is a layer of ramming paste. The taphole region is made up of carbon refractory. It has a very high heat conductivity to transport heat from the taphole during tapping and reduce the thermal stresses in the region.

The proper material selection ensures less heat loss and also less thermal stress. The thermal stresses and hot metal flow cause mechanical design challenge. The shell of the hearth is subjected also to abrasive wear due to oxidation, abrasion/deterioration due to chemical attack and erosion due to hot liquid flow. The heat transfer through the furnace wall occurs by conduction. On the contrary, in the region of fluid flow such as taphole, heat transfer occurs by convection. The high temperature induces thermal stresses in addition to the stresses due to weight of other parts and fixations.

Wear of carbon refractory limits the hearth lifetime, which in turn restricts lifetime of blast furnace hearth. An optimum design enlarges the hearth lifetime and consequently, the blast furnace campaign.



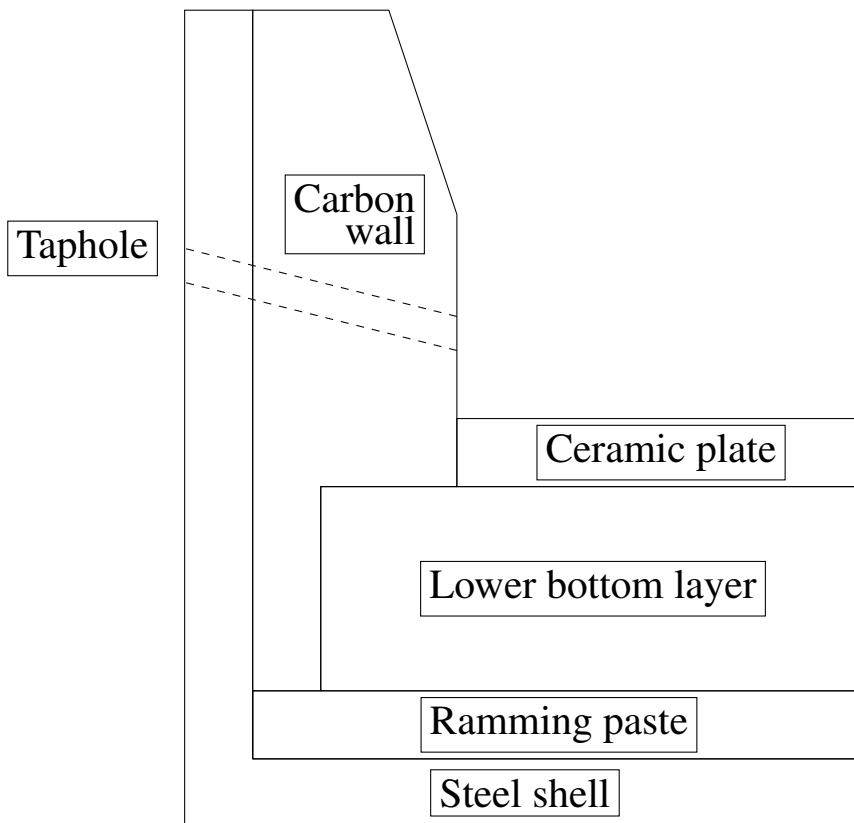


Figure 9.2: Typical cross section of Hearth geometry [141]

The design parameters which affect the hearth design are material properties, operating conditions and geometric dimensions. If carbon blocks are used, frictional contact occurs between block surfaces. Other materials used are ceramic cup bricks or plastic material for ramming mix. The geometric parameters are the thickness of hearth material, number of blocks.

The thermo-mechanical modelling of blast furnace hearth is based on the extensive experience of the global Research and Development center of Arcelor Mittal (AMIII) in Asturias, Spain. The project is carried out under supervision from ITMATI and SISSA.

9.2. Mathematical model

The governing equations for thermo-mechanical problems are linear momentum conservation and energy conservation from continuum mechanics. These equations will be described briefly in the following subsections.

9.2.1. Problem statement in strong formulation

Summarizing, in this document the following simplifications are considered:

- Taphole operation is not part of this study, i.e. only normal operating conditions are considered.
- We only consider steady state operations.
- We assume that the hearth is made up of a single, elastic and isotropic material.
- 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.

In further works, some of the following non linearities will be considered: inhomogeneity, anisotropy, contact



between refractory blocks and dependence of material properties on temperature.

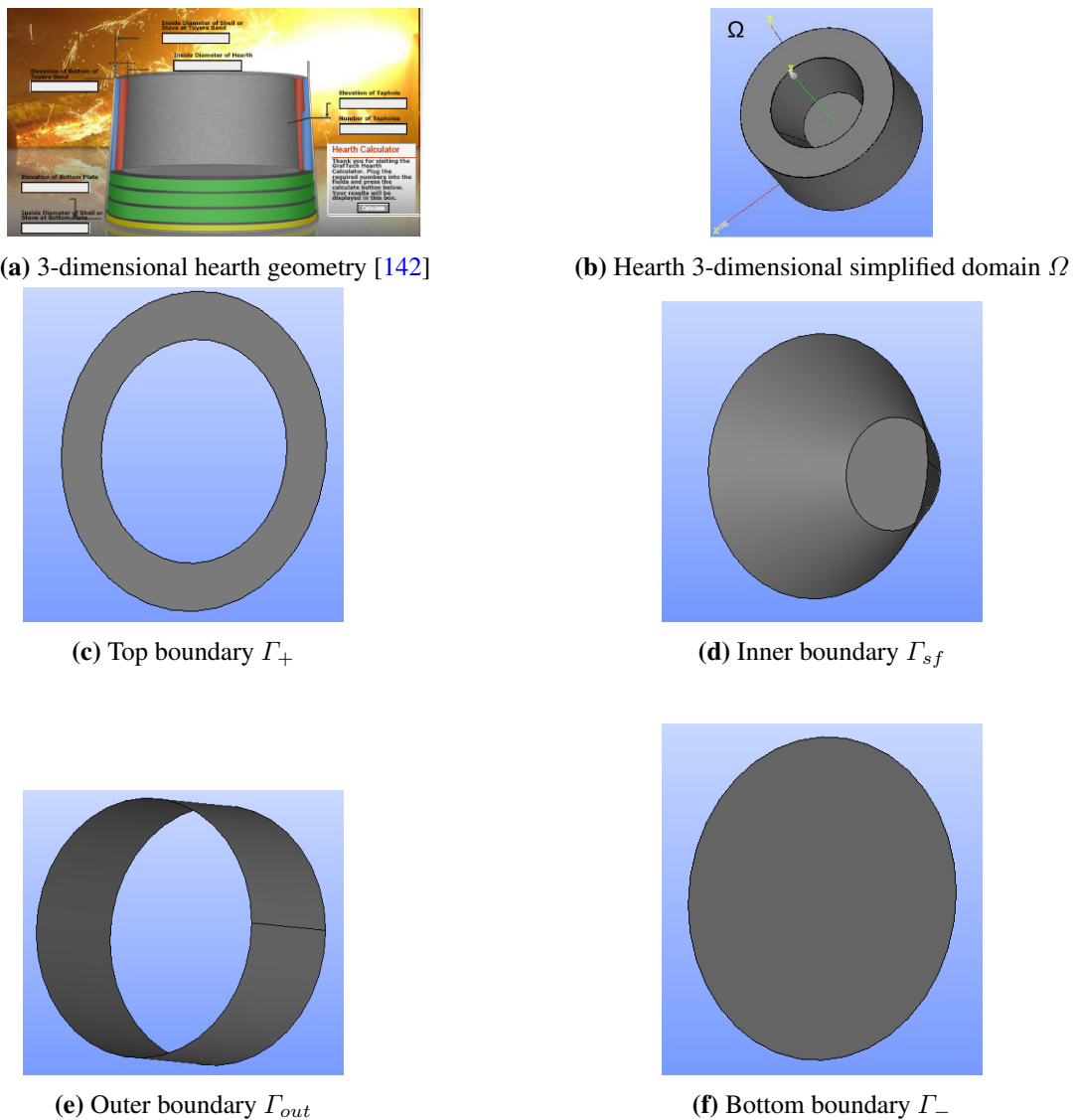


Figure 9.3: Hearth geometry : 3-dimensional computational domain and its boundaries

We consider the three dimensional domain Ω as in Figure 9.3. Ω represents a simplified hearth geometry of the furnace.

Based on the assumptions listed above, in the absence of heat source/sink, and in the absence of external forces, the momentum conservation for small displacements, and the energy conservation can be written as,

$$- \text{Div}(\boldsymbol{\sigma}) = \vec{0} \text{ in } \Omega, \quad (9.1)$$

$$- \text{Div}(\mathbf{K}\nabla T) = 0 \text{ in } \Omega. \quad (9.2)$$

In the case of isotropic material the thermal conductivity \mathbf{K} is expressed as,

$$\mathbf{K} = k\mathbf{I}, \quad k > 0. \quad (9.3)$$



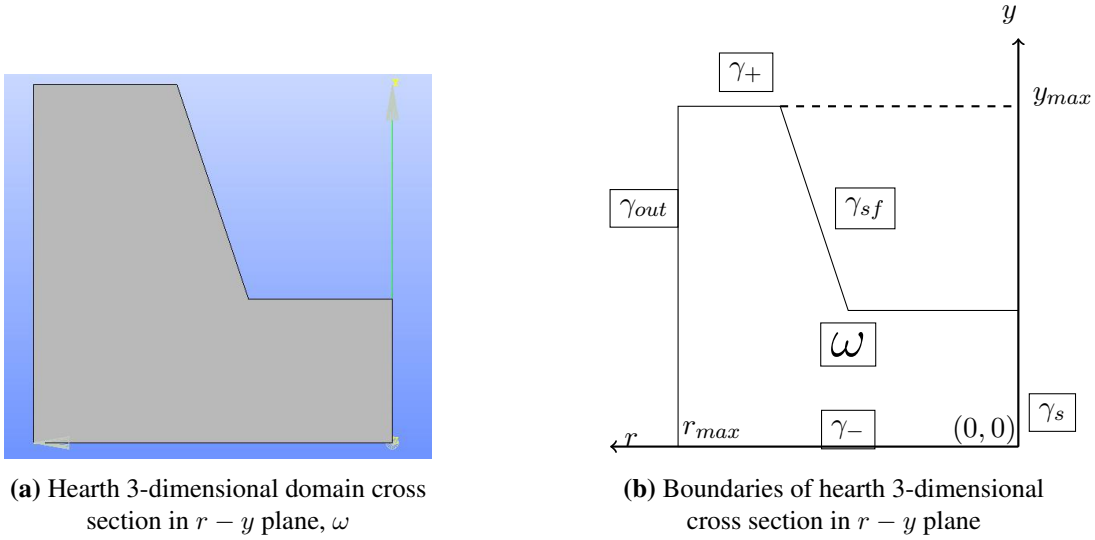


Figure 9.4: Hearth 3-dimensional domain cross section and corresponding boundaries

We represent displacement vector field as \vec{u} and temperature scalar field as T .

The thermo-mechanical stress tensor σ is related to the strain tensor through the Hooke's law:

$$\sigma(\vec{u})[T] = \lambda \text{Tr}(\varepsilon(\vec{u}))\mathbf{I} + 2\mu\varepsilon(\vec{u}) - (2\mu + 3\lambda)\alpha(T - T_0)\mathbf{I}, \quad (9.4)$$

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

$$\varepsilon(\vec{u}) = \frac{1}{2}(\nabla\vec{u} + \nabla\vec{u}^T). \quad (9.5)$$

In addition in (9.4), T_0 is the reference temperature, α is the thermal expansion coefficient, and λ and μ are the Lamé parameters of the material. The Lamé parameters can be expressed in terms of Young modulus, E , and the Poisson ratio, ν , as:

$$\mu = \frac{E}{2(1 + \nu)}, \quad \lambda = \frac{E\nu}{(1 - 2\nu)(1 + \nu)}. \quad (9.6)$$

9.2.2. Boundary conditions

In the following, we introduce the notations for the boundaries of the domain Ω , and its vertical cross section in $r - y$ plane, ω (see Figure 9.3 and 9.4).

$$\begin{aligned} \Gamma_{out} &= \partial\Omega \cap (r \equiv r_{max}) = \gamma_{out} \times [0, 2\pi), \\ \Gamma_+ &= \partial\Omega \cap (y \equiv y_{max}) = \gamma_+ \times [0, 2\pi), \\ \Gamma_- &= \partial\Omega \cap (y \equiv 0) = \gamma_- \times [0, 2\pi), \\ \Gamma_{sf} &= \partial\Omega \setminus (\Gamma_{out} \cup \Gamma_+ \cup \Gamma_-) = \gamma_{sf} \times [0, 2\pi), \\ \gamma_s &= \partial\omega \cap (r \equiv 0), \end{aligned}$$

being $r_{max} \in \mathbb{R}^+$ and $y_{max} \in \mathbb{R}^+$.

- On the upper boundary, Γ_+ , the weight of the upper blast furnace components acts and no conduction heat transfer occurs:



$$\begin{aligned}
 (-\mathbf{K}\nabla T) \cdot \vec{n} &= 0, \\
 \vec{\sigma}_t &= \vec{0}, \\
 \sigma_n &= |\vec{W}|,
 \end{aligned} \tag{9.7}$$

where \vec{W} is the weight per unit surface of the upper furnace components, supported by the upper hearth walls, $\vec{\sigma}_t$ and σ_n are the tangential and normal forces defined by:

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

- On the bottom boundary, Γ_- , the temperature is assumed to be known, T_D , the normal displacement is null and shear forces are assumed to be zero. Therefore, on this boundary,

$$\begin{aligned}
 T &= T_D, \\
 \vec{u} \cdot \vec{n} &= 0, \\
 \vec{\sigma}_t &= \vec{0}.
 \end{aligned} \tag{9.8}$$

- On the inner boundary, Γ_{sf} , a convection heat transfer with the liquid phase occurs and hydrostatic pressure is acting. So, on this boundary the following boundary conditions are considered:

$$\begin{aligned}
 (-\mathbf{K}\nabla T) \cdot \vec{n} &= h_{c,f}(T - T_f), \\
 \boldsymbol{\sigma} \vec{n} &= -p_h \vec{n},
 \end{aligned} \tag{9.9}$$

where T_f is the fluid temperature, assumed to be constant at the steady state, $h_{c,f}$ the convective heat transfer coefficient on Γ_{sf} and p_h is the hydrostatic pressure, which depends on the (r, y) variables.

- On the outer boundary, Γ_{out} , a convective heat flux and a free surface condition are assumed:

$$\begin{aligned}
 (-\mathbf{K}\nabla T) \cdot \vec{n} &= h_{c,out}(T - T_{out}), \\
 \boldsymbol{\sigma} \vec{n} &= \vec{0},
 \end{aligned} \tag{9.10}$$

$h_{c,out}$ being the convective heat transfer coefficient on Γ_{out} , and T_{out} the ambient temperature.

9.2.3. Mathematical model in cylindrical coordinates

We express now the governing equations and boundary conditions in cylindrical coordinate system (r, y, θ) having corresponding unit vectors $(\vec{e}_r, \vec{e}_y, \vec{e}_\theta)$. As we will see in next section, this transformation leads to significant simplification. The operators introduced during the previous sections need to be transformed accordingly. The normal vector will now be represented as $\vec{n} = n_r \vec{e}_r + n_y \vec{e}_y + n_\theta \vec{e}_\theta$.

The displacement vector \vec{u} is expressed as,

$$\vec{u} = u_r(r, y, \theta) \vec{e}_r + u_y(r, y, \theta) \vec{e}_y + u_\theta(r, y, \theta) \vec{e}_\theta, \tag{9.11}$$

the temperature scalar, T , is expressed as,

$$T = T(r, y, \theta), \tag{9.12}$$

and the material point $p = (r, y, \theta)$, with $(r, y) \in \omega$ (see Figure 9.4), and $\theta \in [0, 2\pi)$.



The stress and strain tensors in cylindrical coordinate system (r, y, θ) are given by,

$$\boldsymbol{\varepsilon}(\vec{u}) = \begin{bmatrix} \frac{\partial u_r}{\partial r} & \frac{1}{2} \left(\frac{\partial u_r}{\partial y} + \frac{\partial u_y}{\partial r} \right) & \frac{1}{2} \left(\frac{\partial u_\theta}{\partial r} + \frac{1}{r} \frac{\partial u_r}{\partial \theta} - \frac{u_\theta}{r} \right) \\ \frac{1}{2} \left(\frac{\partial u_r}{\partial y} + \frac{\partial u_y}{\partial r} \right) & \frac{\partial u_y}{\partial y} & \frac{1}{2r} \left(\frac{\partial u_y}{\partial \theta} + r \frac{\partial u_\theta}{\partial y} \right) \\ \frac{1}{2} \left(\frac{\partial u_\theta}{\partial r} + \frac{1}{r} \frac{\partial u_r}{\partial \theta} - \frac{u_\theta}{r} \right) & \frac{1}{2r} \left(\frac{\partial u_y}{\partial \theta} + r \frac{\partial u_\theta}{\partial y} \right) & \frac{1}{r} \frac{\partial u_\theta}{\partial \theta} + \frac{u_r}{r} \end{bmatrix}. \quad (9.13)$$

We represent \mathbf{A} as the fourth order tensor defined as:

$$\mathbf{A} = \frac{E}{(1-2\nu)(1+\nu)} \begin{bmatrix} 1-\nu & \nu & \nu & 0 & 0 & 0 \\ \nu & 1-\nu & \nu & 0 & 0 & 0 \\ \nu & \nu & 1-\nu & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} \end{bmatrix}. \quad (9.14)$$

In Voigt notation, stress-strain relationship can be expressed as,

$$\{\boldsymbol{\sigma}(\vec{u})[T]\} = \mathbf{A}\{\boldsymbol{\varepsilon}(\vec{u})\} - (2\mu + 3\lambda)\alpha(T - T_0)\{\mathbf{I}\}, \quad (9.15)$$

and the following column vectors have been considered:

$$\begin{aligned} \{\boldsymbol{\sigma}\} &= [\sigma_{rr} \quad \sigma_{yy} \quad \sigma_{\theta\theta} \quad \sigma_{y\theta} \quad \sigma_{r\theta} \quad \sigma_{ry}]^T, \\ \{\boldsymbol{\varepsilon}\} &= [\varepsilon_{rr} \quad \varepsilon_{yy} \quad \varepsilon_{\theta\theta} \quad 2\varepsilon_{y\theta} \quad 2\varepsilon_{r\theta} \quad 2\varepsilon_{ry}]^T, \\ \{\mathbf{I}\} &= [1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0]^T. \end{aligned} \quad (9.16)$$

- Stationary thermal model:

For the thermal model, we consider that the tensor of thermal conductivity is isotropic and no dependency on time is considered. Therefore, the energy conservation equation (9.2) can be rewritten as:

$$-\frac{1}{r} \frac{\partial}{\partial r} \left(rk \frac{\partial T}{\partial r} \right) - \frac{\partial}{\partial y} \left(k \frac{\partial T}{\partial y} \right) - \frac{1}{r} \frac{\partial}{\partial \theta} \left(\frac{k}{r} \frac{\partial T}{\partial \theta} \right) = 0, \text{ in } \Omega. \quad (9.17)$$

Boundary conditions:

$$\begin{aligned} \text{on } \Gamma_+ &: -k \frac{\partial T}{\partial r} n_r - k \frac{\partial T}{\partial y} n_y - \frac{k}{r} \frac{\partial T}{\partial \theta} n_\theta = 0, \\ \text{on } \Gamma_- &: T = T_D, \\ \text{on } \Gamma_{sf} &: -k \frac{\partial T}{\partial r} n_r - k \frac{\partial T}{\partial y} n_y - \frac{k}{r} \frac{\partial T}{\partial \theta} n_\theta = h_{c,f}(T - T_f), \\ \text{on } \Gamma_{out} &: -k \frac{\partial T}{\partial r} n_r - k \frac{\partial T}{\partial y} n_y - \frac{k}{r} \frac{\partial T}{\partial \theta} n_\theta = h_{c,out}(T - T_{out}). \end{aligned} \quad (9.18)$$

- Stationary mechanical model:

Vectorial equation (9.1) in cylindrical coordinates, with no time dependency, corresponds to the three



following equations:

$$\begin{aligned}
 \frac{\partial \sigma_{rr}}{\partial r} + \frac{\partial \sigma_{ry}}{\partial y} + \frac{1}{r} \frac{\partial \sigma_{r\theta}}{\partial \theta} + \frac{\sigma_{rr} - \sigma_{\theta\theta}}{r} &= 0, \text{ in } \Omega, \\
 \frac{\partial \sigma_{r\theta}}{\partial r} + \frac{\partial \sigma_{\theta y}}{\partial y} + \frac{1}{r} \frac{\partial \sigma_{\theta\theta}}{\partial \theta} + 2 \frac{\sigma_{r\theta}}{r} &= 0, \text{ in } \Omega, \\
 \frac{\partial \sigma_{ry}}{\partial r} + \frac{1}{r} \frac{\partial \sigma_{\theta y}}{\partial \theta} + \frac{\partial \sigma_{yy}}{\partial y} + \frac{\sigma_{ry}}{r} &= 0 \text{ in } \Omega.
 \end{aligned} \tag{9.19}$$

Boundary conditions:

$$\begin{aligned}
 \text{on } \Gamma_+ : \vec{\sigma}_t &= \vec{0}, \sigma_n = |\vec{W}|, \\
 \text{on } \Gamma_- : \vec{u} \cdot \vec{n} &= 0, \vec{\sigma}_t = \vec{0}, \\
 \text{on } \Gamma_{sf} : \boldsymbol{\sigma} \vec{n} &= -p_h \vec{n}, \\
 \text{on } \Gamma_{out} : \boldsymbol{\sigma} \vec{n} &= \vec{0}.
 \end{aligned} \tag{9.20}$$

9.2.4. Axisymmetric model

It can be observed that the introduced 3-dimensional model (equations (9.17) - (9.20)) is independent of θ , and hence a symmetry hypothesis is applicable. The axisymmetric model leads to significant computational savings as the 3-dimensional model defined in Ω (see Figure 9.3) is replaced by the corresponding 2-dimensional model defined in its vertical section, ω (see Figure 9.4). The normal vector will now be represented as $\vec{n} = n_r \vec{e}_r + n_y \vec{e}_y + 0 \vec{e}_\theta$. In the axisymmetric system, we represent the displacement \vec{u} and temperature T , both independent of θ , as

$$\begin{aligned}
 \vec{u} &= u_r(r, y) \vec{e}_r + u_y(r, y) \vec{e}_y, \\
 T &= T(r, y).
 \end{aligned} \tag{9.21}$$

The associated axisymmetric model is reduced to consider conservation equations (9.17), (9.19) defined in ω and the boundary conditions (9.18), (9.20) replacing the Γ boundaries by γ such that $\Gamma = (\gamma \setminus \gamma_s) \times [0, 2\pi)$ (see figures 9.3c, 9.3d, 9.3e, 9.3f and 9.4b), adding the usual symmetry conditions on γ_s :

$$\begin{aligned}
 -k \frac{\partial T}{\partial r} n_r - k \frac{\partial T}{\partial y} n_y &= 0, \\
 \vec{u} \cdot \vec{n} &= 0, \\
 \vec{\sigma}_t &= \vec{0}.
 \end{aligned} \tag{9.22}$$

Under the assumption of axisymmetry, the strain and stress tensors in cylindrical coordinate system (r, y, θ) given by (9.13) and (9.15), respectively, can be reduced to,

$$\{\boldsymbol{\varepsilon}\} = \begin{bmatrix} \frac{\partial u_r}{\partial r} & \frac{1}{2} \left(\frac{\partial u_r}{\partial y} + \frac{\partial u_y}{\partial r} \right) & 0 \\ \frac{1}{2} \left(\frac{\partial u_r}{\partial y} + \frac{\partial u_y}{\partial r} \right) & \frac{\partial u_y}{\partial y} & 0 \\ 0 & 0 & \frac{u_r}{r} \end{bmatrix} \tag{9.23}$$

$$\{\boldsymbol{\sigma}\} = \begin{bmatrix} \sigma_{rr} & \sigma_{ry} & 0 \\ \sigma_{ry} & \sigma_{yy} & 0 \\ 0 & 0 & \sigma_{\theta\theta} \end{bmatrix}. \tag{9.24}$$

Summarizing, the axisymmetric model considered is :



- Thermal model:

$$-\frac{1}{r} \frac{\partial}{\partial r} \left(rk \frac{\partial T}{\partial r} \right) - \frac{\partial}{\partial y} \left(k \frac{\partial T}{\partial y} \right) = 0, \text{ in } \omega. \quad (9.25)$$

Boundary conditions:

$$\begin{aligned} \text{on } \gamma_+ : & -k \frac{\partial T}{\partial r} n_r - k \frac{\partial T}{\partial y} n_y = 0 \implies \frac{\partial T}{\partial y} = 0, \\ \text{on } \gamma_- : & T = T_D, \\ \text{on } \gamma_{sf} : & -k \frac{\partial T}{\partial r} n_r - k \frac{\partial T}{\partial y} n_y = h_{c,f}(T - T_f), \\ \text{on } \gamma_{out} : & -k \frac{\partial T}{\partial r} n_r - k \frac{\partial T}{\partial y} n_y = -k \frac{\partial T}{\partial r} = h_{c,out}(T - T_{out}), \\ \text{on } \gamma_s : & -k \frac{\partial T}{\partial r} n_r - k \frac{\partial T}{\partial y} n_y = 0 \implies \frac{\partial T}{\partial r} = 0. \end{aligned} \quad (9.26)$$

- Mechanical model :

$$\begin{aligned} \frac{\partial \sigma_{rr}}{\partial r} + \frac{\partial \sigma_{ry}}{\partial y} + \frac{\sigma_{rr} - \sigma_{\theta\theta}}{r} &= 0, \text{ in } \omega, \\ \frac{\partial \sigma_{ry}}{\partial r} + \frac{\partial \sigma_{yy}}{\partial y} + \frac{\sigma_{ry}}{r} &= 0, \text{ in } \omega. \end{aligned} \quad (9.27)$$

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

$$\begin{aligned} \{\boldsymbol{\sigma}(\vec{u})[T]\} &= \mathbf{A}\{\boldsymbol{\varepsilon}(\vec{u})\} - (2\mu + 3\lambda)\alpha(T - T_0)\{\mathbf{I}\}, \\ \mathbf{A} &= \frac{E}{(1 - 2\nu)(1 + \nu)} \begin{bmatrix} 1 - \nu & \nu & \nu & 0 \\ \nu & 1 - \nu & \nu & 0 \\ \nu & \nu & 1 - \nu & 0 \\ 0 & 0 & 0 & \frac{1 - 2\nu}{2} \end{bmatrix}, \\ \{\boldsymbol{\sigma}\} &= [\sigma_{rr} \quad \sigma_{yy} \quad \sigma_{\theta\theta} \quad \sigma_{ry}]^T, \\ \{\boldsymbol{\varepsilon}\} &= [\varepsilon_{rr} \quad \varepsilon_{yy} \quad \varepsilon_{\theta\theta} \quad 2\varepsilon_{ry}]^T, \\ \{\mathbf{I}\} &= [1 \quad 1 \quad 1 \quad 0]^T. \end{aligned} \quad (9.28)$$

Boundary conditions :

$$\begin{aligned} \text{on } \gamma_+ : & \vec{\sigma}_t = \vec{0} \implies \sigma_{ry} = 0, \sigma_n = \sigma_{yy} = -|\vec{W}|, \\ \text{on } \gamma_- : & \vec{u} \cdot \vec{n} = u_y = 0, \vec{\sigma}_t = \vec{0} \implies \sigma_{ry} = 0, \\ \text{on } \gamma_{sf} : & \sigma_{rr} n_r + \sigma_{ry} n_y = -p n_r, \sigma_{ry} n_r + \sigma_{yy} n_y = -p n_y, \\ \text{on } \gamma_{out} : & \sigma_{rr} n_r + \sigma_{ry} n_y = \sigma_{rr} = 0, \sigma_{ry} n_r + \sigma_{yy} n_y = \sigma_{ry} = 0, \\ \text{on } \gamma_s : & \vec{u} \cdot \vec{n} = u_r = 0, \vec{\sigma}_t = \vec{0} \implies \sigma_{ry} = 0. \end{aligned} \quad (9.29)$$

In simplifying boundary conditions in equation (9.26) and (9.29), we have used definition of unit normal vector



on each boundary, as below :

$$\begin{aligned}
 \text{on } \gamma_+ & : \vec{n} = \vec{e}_y, \\
 \text{on } \gamma_- & : \vec{n} = -\vec{e}_y, \\
 \text{on } \gamma_{sf} & : \vec{n} = n_r \vec{e}_r + n_y \vec{e}_y, \\
 \text{on } \gamma_{out} & : \vec{n} = \vec{e}_r, \\
 \text{on } \gamma_s & : \vec{n} = -\vec{e}_r.
 \end{aligned}$$

9.3. Weak formulation of governing equations

We derive the weak formulation related to the conservation equations introduced in last section.

9.3.1. Function spaces

Before discussing the weak formulation, we introduce relevant function spaces.

- Test function space for Temperature is defined as,

$$\mathbb{T} = \{ \psi \in H_r^1(\omega), \psi = 0 \text{ on } \gamma_- \},$$

where, $H_r^1(\omega)$ is weighted Sobolev space equipped with inner product,

$$\langle \psi, T \rangle_{H_r^1(\omega)} = \int_{\omega} \left(\psi T + \frac{\partial T}{\partial r} \frac{\partial \psi}{\partial r} + \frac{\partial T}{\partial y} \frac{\partial \psi}{\partial y} \right) r dr dy,$$

and norm,

$$\|\psi\|_{H_r^1(\omega)} = \left(\int_{\omega} \left(\psi^2 + \left(\frac{\partial \psi}{\partial r} \right)^2 + \left(\frac{\partial \psi}{\partial y} \right)^2 \right) r dr dy \right)^{\frac{1}{2}}.$$

It is to be noted that $\psi = 0$ on γ_- does not refer to absolute zero temperature, but temperature with respect to some specified reference temperature, in this case T_D .

- Test function space for displacement is defined as,

$$\begin{aligned}
 \mathbb{U} = \{ \vec{\phi} \equiv [\phi_r \ \phi_y] \in \mathbb{R}^2, \phi_r \in \left(L_{\frac{1}{r}}^2(\omega) \cap H_r^1(\omega) \right), \phi_y \in H_r^1(\omega), \\
 \phi_y = 0 \text{ on } \gamma_-, \phi_r = 0 \text{ on } \gamma_s \},
 \end{aligned}$$

which is equipped with the inner product,

$$\begin{aligned}
 \langle \vec{u}, \vec{\phi} \rangle_{\mathbb{U}} = \\
 \int_{\omega} \left(\phi_r u_r + \phi_y u_y + \frac{\partial u_r}{\partial r} \frac{\partial \phi_r}{\partial r} + \frac{\partial u_r}{\partial z} \frac{\partial \phi_r}{\partial z} + \frac{u_r}{r} \frac{\phi_r}{r} + \frac{\partial u_y}{\partial r} \frac{\partial \phi_y}{\partial r} + \frac{\partial u_y}{\partial z} \frac{\partial \phi_y}{\partial z} \right) r dr dy,
 \end{aligned}$$

and equipped with the norm,

$$\begin{aligned}
 \|\vec{\phi}\|_{\mathbb{U}} = \\
 \left(\int_{\omega} \left(\phi_r^2 + \phi_y^2 + \left(\frac{\partial \phi_r}{\partial r} \right)^2 + \left(\frac{\partial \phi_r}{\partial z} \right)^2 + \left(\frac{\phi_r}{r} \right)^2 + \left(\frac{\partial \phi_y}{\partial r} \right)^2 + \left(\frac{\partial \phi_y}{\partial z} \right)^2 \right) r dr dy \right)^{\frac{1}{2}}.
 \end{aligned}$$



Here, weighted Sobolev space $L_r^2(\omega)$ is equipped with inner product,

$$\langle \vec{u}, \vec{\phi} \rangle_{L_r^2(\omega)} = \int_{\omega} \vec{u} \cdot \vec{\phi} r dr dy ,$$

and equipped with norm,

$$\|\vec{\phi}\|_{L_r^2(\omega)} = \left(\int_{\omega} \vec{\phi} \cdot \vec{\phi} r dr dy \right)^{\frac{1}{2}} .$$

- The stress tensor σ is symmetric and finite.

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

- The heat flux is also finite.

$$\mathbb{Q} = \{ \vec{q} \in [L^2(\omega)]^2 \} .$$

- The thermal conductivity \mathbf{K} is finite and is positive definite.

$$\{ \mathbf{K} \in [L^\infty(\omega)]^{2 \times 2}, \mathbf{K} \text{ is positive definite} \} .$$

- The material parameters are finite and positive. The poisson's ratio is bounded so that \mathbf{A} is positive definite.

$$\{ E \in L^\infty(\omega), E > 0 \} ,$$

$$\{ \nu \in L^\infty(\omega), 0 < \nu < 0.5 \} .$$

- The convection coefficients $h_{c,f}$ and $h_{c,out}$ are also finite and positive.

$$\{ h_{c,f} \in L^\infty(\gamma_{sf}), h_{c,f} > 0 \} ,$$

$$\{ h_{c,out} \in L^\infty(\gamma_{out}), h_{c,out} > 0 \} .$$

- Pressure p_h and the weight \vec{W} are finite.

$$\{ p_h \in L^\infty(\gamma_{sf}) \} ,$$

$$\{ \vec{W} \in [L^\infty(\gamma_+)]^2 \} .$$

9.3.2. Weak formulation of the energy equation

We multiply the energy equation (9.25) by test function $\psi(r, y) \in \mathbb{T}$ and integrate over axisymmetric domain ω . It is important that the elemental volume in axisymmetric domain is $r dr dy$ corresponds to elemental volume in 3-dimensional domain $r dr d\theta dy$.

$$\int_{\omega} \frac{\psi}{r} \frac{\partial}{\partial r} \left(rk \frac{\partial T}{\partial r} \right) r dr dy + \int_{\omega} \psi \frac{\partial}{\partial y} \left(k \frac{\partial T}{\partial y} \right) r dr dy = 0 , \quad (9.30)$$

$$\int_{\omega} \frac{\partial}{\partial r} \left(rk \psi \frac{\partial T}{\partial r} \right) dr dy - \int_{\omega} rk \frac{\partial T}{\partial r} \frac{\partial \psi}{\partial r} dr dy +$$

$$\int_{\omega} \frac{\partial}{\partial y} \left(rk \psi \frac{\partial T}{\partial y} \right) dr dy - \int_{\omega} rk \frac{\partial T}{\partial y} \frac{\partial \psi}{\partial y} dr dy = 0 . \quad (9.31)$$



By applying Gauss-Divergence theorem and representing boundary normal as $\vec{n} = n_r \vec{e}_r + n_y \vec{e}_y$,

$$\int_{\omega} rk \left(\frac{\partial T}{\partial y} \frac{\partial \psi}{\partial y} + \frac{\partial T}{\partial r} \frac{\partial \psi}{\partial r} \right) = \int_{\partial \omega} \psi k \left(\frac{\partial T}{\partial y} n_y + \frac{\partial T}{\partial r} n_r \right) dS, \quad (9.32)$$

$$\int_{\omega} rk \left(\frac{\partial T}{\partial y} \frac{\partial \psi}{\partial y} + \frac{\partial T}{\partial r} \frac{\partial \psi}{\partial r} \right) = \int_{\partial \omega} \psi k \nabla T \cdot \vec{n} dS. \quad (9.33)$$

Here, dS is the boundary measure. In the case of axisymmetric model this is not the length of edge of 2-dimensional domain ω . We note the invariance with respect to θ . For example the length of Γ_{out} is $\int_0^{2\pi} \int_{\gamma_{out}} r dy d\theta = 2\pi \int_{\gamma_{out}} r dy \implies dS$ for γ_{out} is $r dy$.

By using boundary conditions from equation (9.26),

$$\begin{aligned} \int_{\omega} rk \left(\frac{\partial T}{\partial z} \frac{\partial \psi}{\partial z} + \frac{\partial T}{\partial r} \frac{\partial \psi}{\partial r} \right) dr dy + \int_{\gamma_{sf}} \psi h_{c,f} T r dl + \int_{\gamma_{out}} \psi h_{c,out} T r dy = \\ \int_{\gamma_{sf}} \psi h_{c,f} T_{sf} r dl + \int_{\gamma_{out}} \psi h_{c,out} T_{out} r dy. \end{aligned} \quad (9.34)$$

The left hand side of the equation is bilinear,

$$a_T(T, \psi) = \int_{\omega} rk \left(\frac{\partial T}{\partial z} \frac{\partial \psi}{\partial z} + \frac{\partial T}{\partial r} \frac{\partial \psi}{\partial r} \right) dr dy + \int_{\gamma_{sf}} \psi h T r dl + \int_{\gamma_{out}} \psi h T r dy, \quad (9.35)$$

and the right hand side of the equation is linear,

$$l_T(\psi) = \int_{\gamma_{sf}} \psi h T_{sf} r dl + \int_{\gamma_{out}} \psi h T_{out} r dy. \quad (9.36)$$

The problem now reduces to, find $T \in \mathbb{T}$ such that,

$$a_T(T, \psi) = l_T(\psi), \quad \forall \psi \in \mathbb{T}. \quad (9.37)$$

9.3.3. Weak formulation of the momentum equation

We multiply the equation (9.27) by $\vec{\phi}(r, y) = [\phi_r(r, y) \phi_y(r, y)] \in \mathbb{U}$ and integrate over ω ,

$$\int_{\omega} \left(\phi_r \frac{\partial \sigma_{rr}}{\partial r} + \phi_r \frac{\partial \sigma_{ry}}{\partial y} + \frac{\phi_r}{r} (\sigma_{rr} - \sigma_{\theta\theta}) + \phi_y \frac{\partial \sigma_{ry}}{\partial r} + \phi_y \frac{\partial \sigma_{yy}}{\partial y} + \phi_y \frac{\sigma_{ry}}{r} \right) r dr dy = 0. \quad (9.38)$$

Now,

$$Div(\boldsymbol{\sigma} \vec{\phi}) = \frac{\sigma_{rr} \phi_r}{r} + \frac{\sigma_{ry} \phi_y}{r} + \frac{\partial}{\partial r} (\sigma_{rr} \phi_r) + \frac{\partial}{\partial r} (\sigma_{ry} \phi_y) + \frac{\partial}{\partial y} (\sigma_{ry} \phi_r) + \frac{\partial}{\partial y} (\sigma_{yy} \phi_y),$$

and hence by using Gauss-divergence theorem and stress-strain relation,

$$\int_{\omega} (\mathbf{A} \boldsymbol{\varepsilon}(\vec{u})) : \boldsymbol{\varepsilon}(\vec{\phi}) r dr dy = \int_{\partial \omega} (\boldsymbol{\sigma} \vec{\phi}) \cdot \vec{n} dS + \int_{\omega} (2\mu + 3\lambda) \alpha (T - T_0) \mathbf{I} : \boldsymbol{\varepsilon}(\vec{\phi}) r dr dy. \quad (9.39)$$

It is useful here to recall, as explained in last section, the length of boundary measure dS for axisymmetric model.



By using boundary conditions (Equation (9.29)),

$$\begin{aligned} \int_{\omega} (\mathbf{A}\boldsymbol{\varepsilon}(\vec{u})) : \boldsymbol{\varepsilon}(\vec{\phi}) r dr dy &= \int_{\omega} (2\mu + 3\lambda)\alpha(T - T_0)\mathbf{I} : \boldsymbol{\varepsilon}(\vec{\phi}) r dr dy \\ &- \int_{\gamma_+} |\vec{W}| \vec{n} \cdot \vec{\phi} r dr - \int_{\gamma_{sf}} p_h \vec{n} \cdot \vec{\phi} r dl . \end{aligned} \quad (9.40)$$

The left side of the equation is bilinear,

$$a_M(\vec{u}, \vec{\phi}) = \int_{\omega} (\mathbf{A}\boldsymbol{\varepsilon}(\vec{u})) : \boldsymbol{\varepsilon}(\vec{\phi}) r dr dy , \quad (9.41)$$

and the right hand side of the equation is linear,

$$l_M[T](\vec{\phi}) = \int_{\omega} (2\mu + 3\lambda)\alpha(T - T_0)\mathbf{I} : \boldsymbol{\varepsilon}(\vec{\phi}) r dr dy - \int_{\gamma_+} |\vec{W}| \vec{n} \cdot \vec{\phi} r dr - \int_{\gamma_{sf}} p_h \vec{n} \cdot \vec{\phi} r dl . \quad (9.42)$$

The problem now reduces to find $\vec{u} \in \mathbb{U}$ such that,

$$a_M(\vec{u}, \vec{\phi}) = l_M[T](\vec{\phi}) , \quad \forall \vec{\phi} \in \mathbb{U} . \quad (9.43)$$

9.4. Strong and weak formulation : general form

In this section, we introduce the general form of weak formulation.

9.4.1. Energy equation

The energy equation (9.25) in the presence of axisymmetric heat source $Q(r, y)$ can be written as:

$$-\frac{1}{r} \frac{\partial}{\partial r} \left(r k \frac{\partial T}{\partial r} \right) - \frac{\partial}{\partial y} \left(k \frac{\partial T}{\partial y} \right) = Q , \quad k > 0 , \quad \text{in } \omega , \quad (9.44)$$

subjected to boundary equations (q_0 is known heat flux),

$$\begin{aligned} \text{on } \gamma_+ &: -k \frac{\partial T}{\partial r} n_r - k \frac{\partial T}{\partial y} n_y = q_0 \implies -k \frac{\partial T}{\partial y} n_y = q_0 , \\ \text{on } \gamma_- &: T = T_D , \\ \text{on } \gamma_{sf} &: -k \frac{\partial T}{\partial r} n_r - k \frac{\partial T}{\partial y} n_y = h_{c,f}(T - T_f) , \\ \text{on } \gamma_{out} &: -k \frac{\partial T}{\partial r} n_r - k \frac{\partial T}{\partial y} n_y = -k \frac{\partial T}{\partial r} n_r = h_{c,out}(T - T_{out}) , \\ \text{on } \gamma_s &: -k \frac{\partial T}{\partial r} n_r - k \frac{\partial T}{\partial y} n_y = 0 \implies \frac{\partial T}{\partial r} = 0 . \end{aligned} \quad (9.45)$$

In case of benchmark tests, we apply known temperature $T_{analytical}$ over the domain i.e. $T(r, y) = T_{analytical}(r, y)$ in ω . In such case the source term is determined by equation (9.44) and environmental temperatures T_f and T_{out} are determined by equation (9.45) by substituting $T = T_{analytical}$.

Correspondingly, weak form (9.34) with source term Q and known temperature $T = T_{analytical}$ is given by,

$$\int_{\omega} r k \left(\frac{\partial T}{\partial z} \frac{\partial \psi}{\partial z} + \frac{\partial T}{\partial r} \frac{\partial \psi}{\partial r} \right) dr dy = \int_{\omega} Q r dr dy - \int_{\partial\omega} \psi \vec{q} \cdot \vec{n} dS , \quad \forall \psi \in \mathbb{T} . \quad (9.46)$$



The source term $Q(r, y)$ is given by equation (9.44) and heat flux at the boundary is given by equation (9.45) considering $T = T_{analytical}$.

In case of convection boundary, instead of heat flux, the environmental temperature is required. In case of γ_{sf} this is given by equation (9.45),

$$T_f = \frac{k}{h_{c,f}} \left(\frac{\partial T}{\partial r} n_r + \frac{\partial T}{\partial y} n_y \right) + T_{analytical} . \quad (9.47)$$

Similarly in case of γ_{out} the temperature T_{out} is given by,

$$T_{out} = \frac{k}{h_{c,out}} \left(\frac{\partial T}{\partial r} n_r \right) + T_{analytical} . \quad (9.48)$$

The equation (9.46) with convection boundary, considering $dS = r dr dy$, $\vec{q} \cdot \vec{n} = 0$ on γ_s and by definition of space \mathbb{T} ,

$$\begin{aligned} \int_{\omega} r k \left(\frac{\partial T}{\partial z} \frac{\partial \psi}{\partial z} + \frac{\partial T}{\partial r} \frac{\partial \psi}{\partial r} \right) dr dy + \int_{\gamma_{sf}} h_{c,f} \psi T r dr dy + \int_{\gamma_{out}} h_{c,out} \psi T r dr dy = \\ \int_{\omega} Q r dr dy + \int_{\gamma_{sf}} h_{c,f} \psi T_f r dr dy + \int_{\gamma_{out}} h_{c,out} \psi T_{out} r dy - \\ \int_{\gamma_+} \psi \vec{q} \cdot \vec{n} r dr - \int_{\gamma_s} \psi \vec{q} \cdot \vec{n} r dy , \forall \psi \in \mathbb{T} . \end{aligned} \quad (9.49)$$

The aim of benchmark test now reduces to ensure that the magnitude of error $T - T_{analytical}$ between computed solution T and known analytical $T_{analytical}$ should be negligible point wise.

9.4.2. Momentum equation

The momentum equation (9.27) in the presence of axisymmetric body source $\vec{b}(r, y) = b_r(r, y)\vec{e}_r + b_y(r, y)\vec{e}_y$ can be written as:

$$\begin{aligned} \frac{\partial \sigma_{rr}}{\partial r} + \frac{\partial \sigma_{ry}}{\partial y} + \frac{\sigma_{rr} - \sigma_{\theta\theta}}{r} + b_r = 0 , \text{ in } \omega , \\ \frac{\partial \sigma_{ry}}{\partial r} + \frac{\partial \sigma_{yy}}{\partial y} + \frac{\sigma_{ry}}{r} + b_y = 0 , \text{ in } \omega , \end{aligned} \quad (9.50)$$

subjected to boundary conditions,

$$\begin{aligned} \text{on } \gamma_+ : \boldsymbol{\sigma}(\vec{u}) &= \boldsymbol{\sigma}(\vec{u}_{analytical}) , \\ \text{on } \gamma_- : \vec{u} \cdot \vec{n} &= u_y = 0 , \vec{\sigma}_t(\vec{u}) = \vec{\sigma}_t(\vec{u}_{analytical}) \implies \sigma_{ry}(\vec{u}) = \sigma_{ry}(\vec{u}_{analytical}) , \\ \text{on } \gamma_{sf} : \boldsymbol{\sigma}(\vec{u}) &= \boldsymbol{\sigma}(\vec{u}_{analytical}) , \\ \text{on } \gamma_{out} : \boldsymbol{\sigma}(\vec{u}) &= \boldsymbol{\sigma}(\vec{u}_{analytical}) , \\ \text{on } \gamma_s : \vec{u} \cdot \vec{n} &= u_r = 0 , \vec{\sigma}_t = \vec{0} \implies \sigma_{ry} = 0 . \end{aligned} \quad (9.51)$$

In case of benchmark tests, we apply known displacement $\vec{u}_{analytical}$ over the domain i.e. $\vec{u}(r, y) = \vec{u}_{analytical}(r, y)$ in ω . In such case the body force term is determined by equation (9.50) and boundary displacements are determined by equation (9.51).

Correspondingly, weak form (9.40) with source term $\vec{b}(r, y)$ and known displacement $\vec{u} = \vec{u}_{analytical}$ is



given by,

$$\int_{\omega} \boldsymbol{\sigma}(\vec{u})[T] : \boldsymbol{\varepsilon}(\vec{\phi}) = \int_{\partial\omega} \vec{\phi} \cdot (\boldsymbol{\sigma} \vec{n}) + \vec{b} \cdot \vec{\phi}, \text{ in } \omega, \forall \vec{\phi} \in \mathbb{U}. \quad (9.52)$$

Applying the condition, $\vec{\phi} \cdot \vec{n} = 0$ on $\gamma_s \cup \gamma_-$ and $\vec{\sigma}_t = \vec{0}$ on γ_s ,

$$\begin{aligned} \int_{\omega} \boldsymbol{\sigma}(\vec{u})[T] : \boldsymbol{\varepsilon}(\vec{\phi}) &= \int_{\gamma_{out}} \vec{\phi} \cdot (\boldsymbol{\sigma}(\vec{u}_{analytical}) \vec{n}) + \int_{\gamma_+} \vec{\phi} \cdot (\boldsymbol{\sigma}(\vec{u}_{analytical}) \vec{n}) + \\ &\int_{\gamma_{sf}} \vec{\phi} \cdot (\boldsymbol{\sigma}(\vec{u}_{analytical}) \vec{n}) + \int_{\gamma_-} \vec{\phi} \cdot \vec{\sigma}_t(\vec{u}_{analytical}) \\ &+ \int_{\omega} \vec{b} \cdot \vec{\phi}, \text{ in } \omega, \forall \vec{\phi} \in \mathbb{U}. \end{aligned} \quad (9.53)$$

The aim of benchmark test now reduces to ensure that the magnitude of error $\vec{u} - \vec{u}_{analytical}$ between computed solution \vec{u} and known analytical $\vec{u}_{analytical}$ should be negligible point wise.

9.5. Numerical experiments and results

In this section, we first introduce the conditions, which are required to be fulfilled by benchmark test. Thereafter, we provide results of benchmark tests to verify the numerical implementations. The benchmark tests are based on our proposal in section 7.7 of [143]. The benchmark test results in sections 9.5.2, 9.5.3 and 9.5.4 are based on FEniCS [144] and the benchmark test results in section 9.5.5 are based on FEniCS [144] and code_aster [145].

9.5.1. Essential characteristic conditions

It is pertinent to mention that each benchmark test should satisfy ‘‘essential characteristic conditions’’. These conditions ensure that if a benchmark test is known to compute exact solution, the computed solution lies in the required function space and in turn, ensure that benchmark problem is correct representative of the actual problem.

- The heat flux normal to γ_s should be zero. From equation (9.26),

$$\text{on } \gamma_s : -k \frac{\partial T}{\partial r} n_r - k \frac{\partial T}{\partial y} n_y = 0 \implies \frac{\partial T}{\partial r} = 0.$$

- Zero displacement normal to the boundary on $\gamma_s \cup \gamma_-$. From equation (9.29),

$$\text{on } \gamma_s : u_r = 0,$$

$$\text{on } \gamma_- : u_y = 0.$$

- Zero shear force on γ_s . From equation (9.29) and (9.28),

$$\text{on } \gamma_s : \sigma_{ry} = 0 \implies \varepsilon_{ry} = 0.$$

However, in real-life problem the analytical solution is not known and hence, the fulfillment of all of the essential characteristic condition can be expected only when number of degrees of freedom is very high. This is possible either with high polynomial degree approximation (p -convergence) or with very fine discretization (h -convergence) or with both. However, increase in number of degrees of freedom results in increased computational time.

It is important to note that we verify $\frac{\partial T}{\partial r} = 0$ on γ_s and not $\vec{q} \cdot \vec{n} = 0$ on γ_s . Numerically, the two conditions are not same. Consider that the component of gradient of T in radial direction r , $\frac{\partial T}{\partial r}$ is of the order of $\approx 10^{-11}$



and the order of magnitude of thermal conductivity k is 10^{-6} . The heat flux component $q_r \approx 10^{-17}$, which is less than machine precision. However, the computed temperature T does not satisfy one of the essential characteristic condition but the low order of magnitude of thermal conductivity misleads into believing that the computed temperature satisfies essential characteristic condition.

Similarly on γ_s , we verify that the computed displacement \vec{u} satisfies $\varepsilon_{ry} = 0$ and we do not verify $\sigma_{ry} = 0$. Consider a case in which $\varepsilon_{ry} \approx 10^{-17}$, which is less than machine precision. Consider Young's modulus E and accordingly Lamé parameter λ is of the order of $\approx 10^6$. The shear stress σ_{ry} in this case becomes $\approx 10^{-11}$ i.e. non-zero. Hence, it appears that the solution violates one of the essential characteristic condition although the computed displacement \vec{u} satisfies the essential characteristic condition.

As can be seen, due to round-off error, theoretically same conditions can give different results numerically and it is necessary that the benchmark test is based on correct measurements.

9.5.2. Energy equation

We apply the known temperature,

$$T_{analytical} = r^2 y ,$$

along with corresponding source term and compare the known analytical temperature with computed temperature [Figure 9.5].

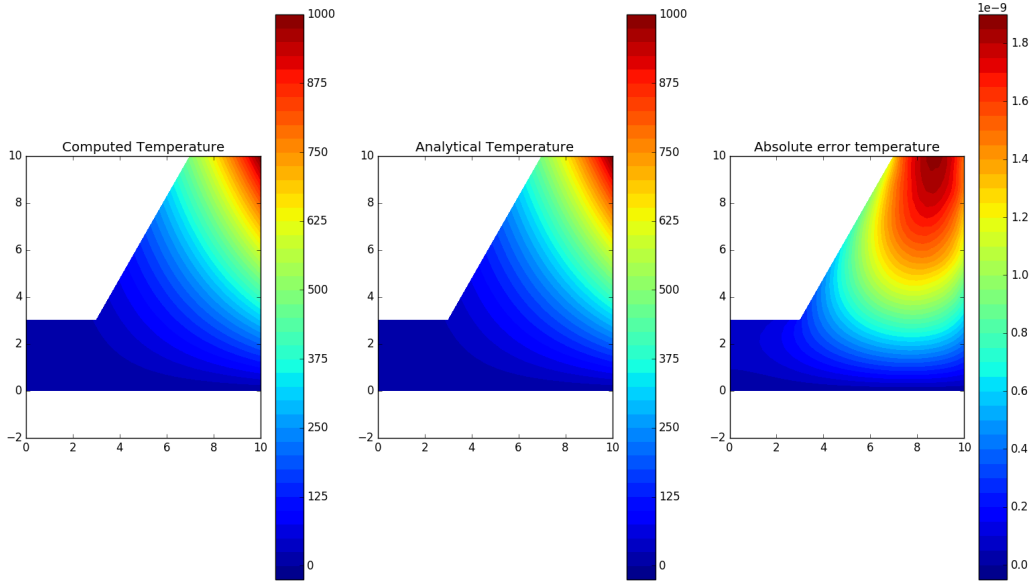


Figure 9.5: Benchmark tests for energy equation

```
file_name :
1. benchmark_thermal
2. readme_benchmark_thermal
```

9.5.3. Momentum equation

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 apply the known displacement function

$$\vec{u}_{analytical} = [ry^2, r^2y] \quad (9.54)$$

to mechanical problem, without considering thermal stresses, along with corresponding body force term and compare the known analytical displacement with computed displacement [Figure 9.6].

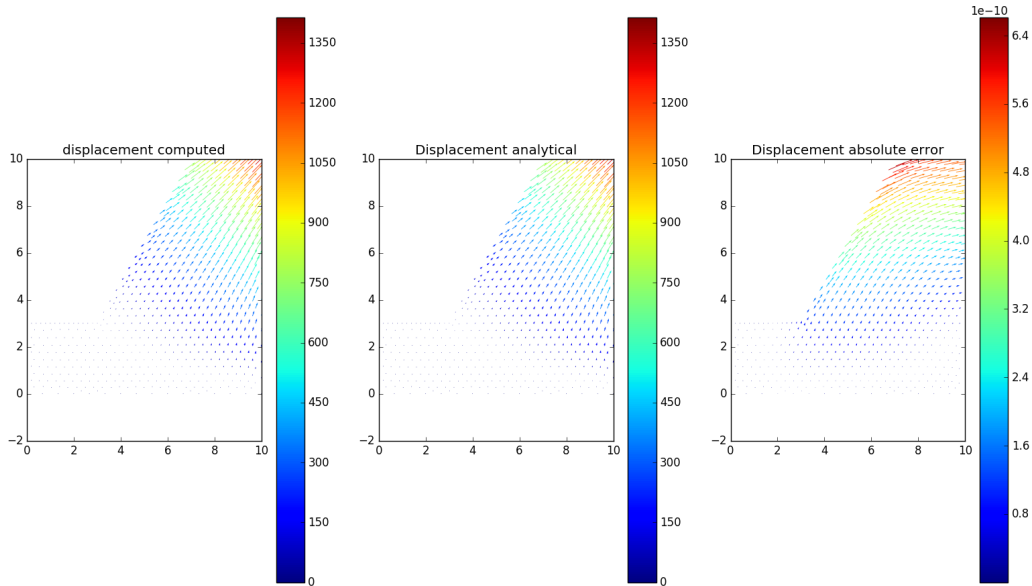


Figure 9.6: Benchmark tests for momentum equation

```
file_name :
1. benchmark_mechanical
2. readme_benchmark_mechanical
```

9.5.4. Coupling

Instead of applying the known spherical stress tensor and thermal stress tensor in separate tests, we apply in our problem thermo-mechanical stress as defined during mechanical model. Also, we apply the known displacement function,

$$\vec{u}_{analytical} = [r^3, y^3]$$

and known temperature function,

$$T_{analytical} = r^2y .$$

We compare the thermal part of the stress tensor with difference between the spherical part of thermo-mechanical stress tensor and the spherical part of mechanical stress tensor [Figure 9.7].



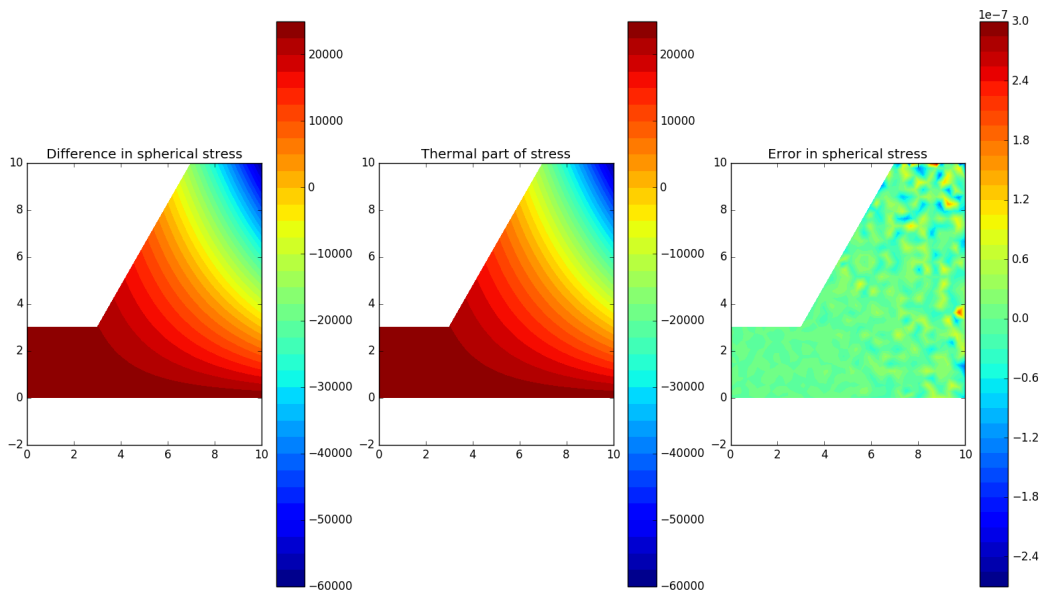


Figure 9.7: Benchmark tests for coupling

```
file_name :
1. coupling_benchmark
2. readme_coupling_benchmark
```

9.5.5. Software comparison for actual problem

We now present the temperature and displacement profiles obtained by FEniCS [144] and Code_aster [145] for the problem of blast furnace hearth described by equations (9.34) and (9.40) [Figures 9.8 and 9.9].

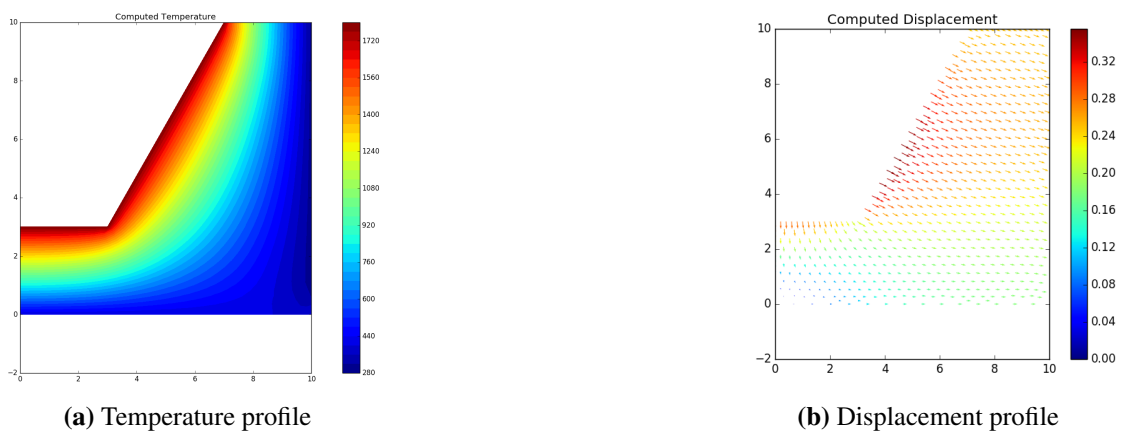


Figure 9.8: Temperature and Displacement profile obtained by FEniCS

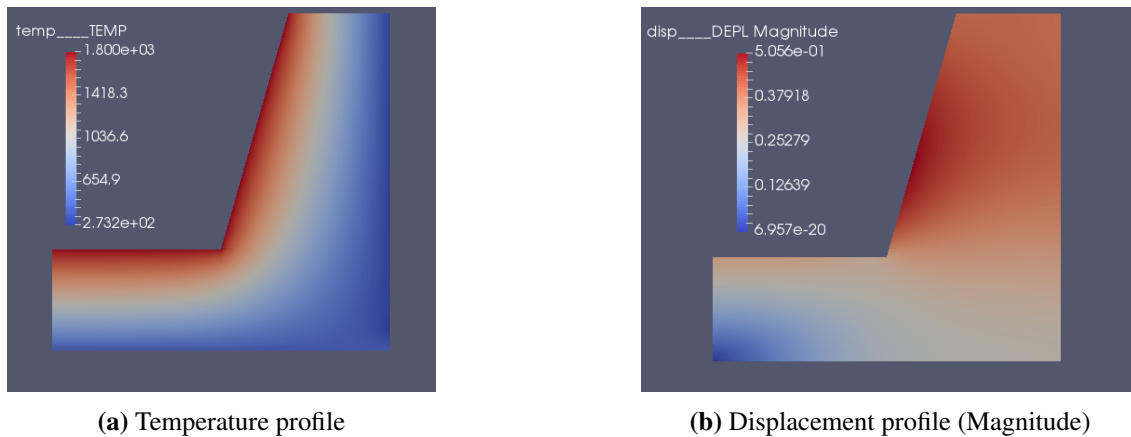


Figure 9.9: Temperature and Displacement profile obtained by Code_aster

For FEniCS [144]:

```
file_name :
1. benchmark_thermal_mechanical
2. readme_benchmark_thermal_mechanical
```

For code_aster [145]:

```
file_name :
1. thermal_mech_benchmark (.comm)
2. thermal_mech_benchmark (.hdf)
```

9.6. Software installation and licensing

The libraries required for all the benchmark tests, its licensing and the software version used for benchmark tests reported here are mentioned below. The operating system used is Ubuntu 16.04.5 LTS.

- 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
- code_aster and Salome_meca
 1. Website : <https://www.code-aster.org/spip.php?rubrique2>
 2. Download and installation : <https://www.code-aster.org/spip.php?rubrique21>
 3. Licensing : code_Aster is distributed under GNU GPL licence (GNU General Public Licence, version 2). Salome-Meca is distributed under GNU LGPL licence (GNU General Public Licence, version 2.1).
 4. Version : V2016
- 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

All benchmark tests performed in FEniCS[144] with python3 can be executed using bash command :

```
$ python3 file_name.py
```

The benchmark test in code_aster is performed by starting the salome_meca, loading the case study file thermal_mech_benchmark.hdf by file → open, activating mesh module and activating aster module. Thereafter, in aster module go to object browser and right click on the case (such as linear-thermic) and run test. To visualize the results, activate paravis module, go to file → Open paraview file → load the *.rmed file. The commands executed in case file *.hdf are defined in *.comm file.

We advise users to read the readme file, if available and comments inside the code for step by step understanding.



References

- [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, Tech. Rep., 1930.
- [3] J. Primot, “Theoretical description of Shack–Hartmann wave-front sensor,” *Optics Communications*, vol. 222, no. 1, pp. 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*, vol. 18, no. 10, p. 2527–2538, 2001.
- [5] I. Foppiano, E. Diolaiti, and P. Ciliegi, “Maory adaptive optics real-time computer user requirements,” European Extremely Large Telescope (E-ELT), Tech. Rep., 2018.
- [6] 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>
- [7] S. Marburg and B. Nolte, *Computational Acoustics of Noise Propagation in Fluids - Finite and Boundary Element Methods*. Springer-Verlag Berlin Heidelberg, 2008.
- [8] 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.
- [9] 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.
- [10] J. Allard and N. Atalla, *Propagation of Sound in Porous Media: Modelling Sound Absorbing Materials 2e*. Wiley, 2009.
- [11] 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.
- [12] 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>
- [13] “Salome documentation.” [Online]. Available: <https://www.salome-platform.org/user-section/documentation/current-release>
- [14] H. P. Langtangen and A. Logg, *Solving PDEs in Python*. Springer, 2017.
- [15] “Fenics documentation.” [Online]. Available: <https://fenicsproject.org/documentation/>
- [16] J. D. Hunter, “Matplotlib: A 2d graphics environment,” *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.
- [17] U. Ayachit, *The ParaView Guide: A Parallel Visualization Application*. USA: Kitware, Inc., 2015.
- [18] “Paraview tutorials.” [Online]. Available: https://www.paraview.org/Wiki/The_ParaView_Tutorial
- [19] “Feconv project page.” [Online]. Available: <http://victorsndvg.github.io/FEconv/>
- [20] “Ufl documentation.” [Online]. Available: <https://fenics.readthedocs.io/projects/ufl/en/latest/>
- [21] 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.
- [22] 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.
- [23] 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>
- [24] 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.
- [25] 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. Itrich, “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.
- [26] 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>
- [27] 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.
- [28] 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.
- [29] S. Lunz, O. Öktem, and C.-B. Schönlieb, “Adversarial regularizers in inverse problems,” *arXiv preprint arXiv:1805.11572*, 2018.
- [30] 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.
- [31] 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.
- [32] 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.
- [33] 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.
- [34] 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.
- [35] B. Gleich and J. Weizenecker, “Tomographic imaging using the nonlinear response of magnetic particles,” *Nature*, vol. 435, no. 7046, pp. 1214–1217, June 2005.
- [36] 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>
- [37] 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>
- [38] T. Knopp and T. M. Buzug, *Magnetic Particle Imaging: An Introduction to Imaging Principles and Scanner Instrumentation*. Berlin/Heidelberg: Springer, 2012.



- [39] S. Lunz, O. Öktem, and C.-B. Schönlieb. Adversarial regularizers in inverse problems. [Online]. Available: <https://arxiv.org/abs/1805.11572>
- [40] H. Li, J. Schwab, S. Antholzer, and M. Haltmeier, “Nett: Solving inverse problems with deep neural networks,” *arXiv preprint arXiv:1803.00092*, 2018.
- [41] 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.
- [42] 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>
- [43] 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>
- [44] 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.
- [45] 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.
- [46] P. Putzky and M. Welling, “Recurrent inference machines for solving inverse problems,” *arXiv preprint arXiv:1706.04008*, 2017.
- [47] J. Adler and O. Öktem. (2017) Solving ill-posed inverse problems using iterative deep neural networks. [Online]. Available: <https://arxiv.org/abs/1704.04058>
- [48] B. Kelly, T. P. Matthews, and M. A. Anastasio, “Deep learning-guided image reconstruction from incomplete data,” *arXiv preprint arXiv:1709.00584*, 2017.
- [49] 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>
- [50] J. Adler and O. Öktem. (2017) Learned primal-dual reconstruction. [Online]. Available: <https://arxiv.org/abs/1707.06474>
- [51] 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>
- [52] 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.
- [53] 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.
- [54] D. Ulyanov, A. Vedaldi, and V. Lempitsky, “Deep image prior,” *arXiv preprint arXiv:1711.10925*, 2017.
- [55] 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.
- [56] 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.
- [57] 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.



- [58] 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.
- [59] 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.
- [60] 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.
- [61] C. Chen, Q. Chen, J. Xu, and V. Koltun, “Learning to see in the dark,” *arXiv preprint arXiv:1805.01934*, 2018.
- [62] 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>
- [63] 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/>
- [64] F. Chollet *et al.*, “Keras,” <https://keras.io>, 2015.
- [65] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
- [66] “Open mpi data,” <https://magneticparticleimaging.github.io/OpenMPIData.jl/latest/index.html>, accessed: 2018-11-16.
- [67] 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>
- [68] “Github mdf,” <https://github.com/MagneticParticleImaging/MDF>, accessed: 2018-11-16.
- [69] 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.
- [70] 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.
- [71] M. Günther, A. Kvaernø, and P. Rentrop, “Multirate partitioned runge-kutta methods,” *BIT Numerical Mathematics*, vol. 41, no. 3, pp. 504–514, 2001.
- [72] G. Wanner and E. Hairer, *Solving ordinary differential equations II*. Springer Berlin Heidelberg, 1996.
- [73] B. Gough, *GNU scientific library reference manual*. Network Theory Ltd., 2009.
- [74] 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.
- [75] G. Guennebaud, B. Jacob *et al.*, “Eigen v3,” <http://eigen.tuxfamily.org>, 2010.
- [76] European Commission, “Commission delegated regulation (EU) 2017/653 OJ L 100,” *Off. J. EU*, vol. 1, pp. 1–52, 2017.
- [77] 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.
- [78] J. Bicksler and A. Chen, “An economic analysis of interest rate swaps,” *J. Finance*, vol. 3, pp. 645–655, 1986.



- [79] D. Brigo and F. Mercurio, *Interest Rate Models - Theory and Practice*, 2nd ed. Berlin: Springer-Verlag, 2006.
- [80] M. Aichinger and A. Binder, *A Workout in Computational Finance*, 1st ed. West Sussex, UK: John Wiley and Sons Inc., 2013.
- [81] 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.
- [82] 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.
- [83] A. Falcó, L. Navarro, and C. Cendón, “Finite difference methods for hull-white pricing of interest rate derivatives with dynamical consistent curves,” *SSRN Elec. J.*, 2014.
- [84] A. Sepp. (2002) Numerical implementation of hull-white interest rate model: Hull-white tree vs finite differences. Available from www.hot.ee/seppar.
- [85] A. Cohen and R. DeVore, “Approximation of high-dimensional parametric pdes,” *Acta Numerica*, vol. 24, pp. 1–159, 2015.
- [86] 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.
- [87] A. Chatterjee, “An introduction to the proper orthogonal decomposition,” *Curr. Sci.*, vol. 78, pp. 808–817, 2000.
- [88] 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.
- [89] 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.
- [90] I. Jolliffe, *Principal Component Analysis*, 1st ed. Berlin: Springer-Verlag, 2014.
- [91] 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.
- [92] M. Rathinam and L. Petzold, “A new look at proper orthogonal decomposition,” *SIAM J. Numer. Anal.*, vol. 41, no. 5, pp. 1893–1925, 2003.
- [93] 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.
- [94] L. Sirovich, “Turbulence and the dynamics of coherent structures. Part I: coherent structures,” *Quart. Appl. Math.*, vol. 45, no. 3, pp. 561–571, 1987.
- [95] J. Lucia, P. Beran, and W. Silva, “Reduced order modeling: new approaches for computational physics,” *JAerospace*, vol. 40, pp. 51–117, 2004.
- [96] S. Saks, *Theory of Integral*, 2nd ed. New York, NY, US: Hafner Publications, 1937.
- [97] R. Dudley, *Unifrom Central Limit Theorems*, 1st ed. New York, NY, US: Cambridge University Press, 2010.
- [98] A. Corhay and A. Rad, “Statistical properties of daily returns: Evidence from european stock markets,” *J. Bus. Finan. Account.*, vol. 21, no. 2, pp. 271–282, 1994.
- [99] H. Albrecher, A. Binder, V. Loutscham, and P. Mayer, *Introduction to Quantitative Methods for Financial Markets*, 1st ed. Berlin: Springer-Verlag, 2010.
- [100] M. Grigoriu, *Stochastic Calculus: Applications in Science and Engineering*, 1st ed. Basel: Birkhäuser, 2002.
- [101] R. Mahnke, J. Kaupuzs, and I. Lubashevsky, *Physics of Stochastic Processes: How Randomness Acts in Time*, 1st ed. Weinheim: WILEY-VCH Verlag GmbH and Co., 2008.



- [102] P. Wilmott and S. Howson, *The Mathematics of Financial Derivatives: A Student Introduction*, 1st ed. London: Cambridge University Press, 2002.
- [103] O. Vasicek, “An equilibrium characterization of the term structure,” *J. Financial Econ.*, vol. 5, no. 2, pp. 177–188, 1977.
- [104] 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.
- [105] J. Hull and A. White, “Pricing interest-rate-derivative securities,” *Rev. Financial Stud.*, vol. 3, no. 4, pp. 573–592, 1990.
- [106] ———, “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.
- [107] G. Golub and C. Reinsch, “Singular value decomposition and least squares solutions,” *Numer. Math.*, vol. 24, pp. 403–420, 1970.
- [108] S. Shreve, *Stochastic Calculus and Finance*, 1st ed. New York, US: Springer-Verlag, 2004.
- [109] S. Ross, *A First Course in Probability*, 8th ed. New Jersey, US: Prentice-Hall, 2010.
- [110] G. Darbellay, “A note on the volatility term structure in short rate models,” *ZAMM*, vol. 78, pp. 885–886, 1998.
- [111] H. Engl, “Calibration problems—an inverse problems view,” *Wilmott*, pp. 16–20, 2007.
- [112] MathConsult, “Calibration of interest rate models,” MathConsult GmbH, Linz, Austria, Report, 2009.
- [113] D. Anderson, J. Tannehill, and R. Pletcher, *Computational Fluid Mechanics and Heat Transfer*, 3rd ed. London: CRC Press, 2013.
- [114] J. Heinrich, P. Huyakorn, O. Zienkiewicz, and A. Mitchell, “An ‘upwind’ finite element scheme for two-dimensional convective transport equation,” *Int. J. Numer. Meth. Engng.*, vol. 11, pp. 131–143, 1977.
- [115] G. Sun and C. Trueman, “Efficient implementations of the crank-nicolson scheme for the finite-difference time-domain method,” *IEEE Trans. Microw. Theory Tech.*, vol. 11, pp. 131–143, 1977.
- [116] 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.
- [117] C. Prud’homme, D. Rovas, K. Veroy, L. Machiels, Y. Maday, A. Patera, and G. Turinici, “Reliable real-time solution of parametrized partial differential equations: Reduced-basis output bound methods,” *J. Fluids Eng.*, vol. 124, no. 1, pp. 70–80, 2001.
- [118] W. Oettli and W. Prager, “Compatibility of approximate solution of linear equations with given error bounds for coefficients and right-hand sides,” *Numer. Math.*, vol. 6, no. 1, pp. 405–409, 1964.
- [119] C. Brezinski, “Error estimates for the solution of linear systems,” *SIAM J. Sci. Comput.*, vol. 21, no. 2, pp. 764–781, 1999.
- [120] 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.
- [121] B. Notghi, M. Ahmadpoor, and J. Brigham, “Adaptive reduced-basis generation for reduced-order modeling for the solution of stochastic nondestructive evaluation problems,” *Comput. Methods. Appl. Mech. Engrg.*, vol. 310, pp. 172–188, 2016.
- [122] 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.
- [123] K. Veroy, C. Prud’homme, and A. Patera, “A posteriori error bounds for reduced-basis approximation of parametrized noncoercive and nonlinear elliptic partial differential equations,” in *Proceedings of the 16th AIAA Computational Fluid Dynamics Conference*, Orlando, United States, 2003, p. 3847.

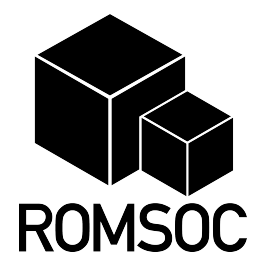


- [124] M. Grepl and A. Patera, “A posteriori error bounds for reduced-basis approximations of parametrized parabolic partial differential equations,” *M2AN Math. Model. Numer. Anal.*, vol. 39, pp. 157–181, 2005.
- [125] O. Alifanov, *Inverse Heat Transfer Problems*, 1st ed. Moscow Izdatel Mashinostroenie, 1988.
- [126] J. E. Marsden, *Elementary classical analysis*. W. H. Freeman San Francisco, 1974.
- [127] F. D. Moura Neto and A. J. da Silva Neto, *An Introduction to Inverse Problems with Applications*. Springer Publishing Company, Incorporated, 2012.
- [128] R. Eymard, T. Gallouët, 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.
- [129] 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.
- [130] H. Jasak, “Openfoam: Open source cfd in research and industry,” *International Journal of Naval Architecture and Ocean Engineering*, vol. 1, no. 2, pp. 89 – 94, 2009.
- [131] C. Botterbusch, S. Lucquin, P. Monticone, J. Drevet, A. Guignabert, and P. Meneroud, “Implantable pump system having an undulating membrane,” May 15 2018, uS Patent 9,968,720.
- [132] C. 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.
- [133] T. Snyder, A. Bourquin, F. Cornat, B. Burg, J. Biasetti, and C. Botterbusch, “CorWave LVAD development update,” *The Journal of Heart and Lung Transplantation*, vol. 38, no. 4, pp. 5341–5342, 2019.
- [134] T. S. M. Perschall, J.B. Drevet 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.
- [135] 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.
- [136] 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.
- [137] 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.
- [138] 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.
- [139] H. Si, “TetGen, a delaunay-based quality tetrahedral mesh generator,” *ACM Transactions on Mathematical Software (TOMS)*, vol. 41, no. 2, p. 11, 2015.
- [140] “Ispat guru,” <http://ispatguru.com/glossary-of-terms-used-for-a-blast-furnace/>, webpage.
- [141] M. Swartling, B. Sundelin, A. Tilliander, and P. G. Jönsson, “Heat transfer modelling of a blast furnace hearth,” *Steel research international*, vol. 81, no. 3, pp. 186–196, 2010. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/srin.200900145>
- [142] “Graftech international,” <http://archive.constantcontact.com/fs072/1101483160833/archive/1102208996928.html>, webpage.
- [143] 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>
- [144] M. S. Alnæs, J. Blechta, J. Hake, A. Johansson, B. Kehlet, A. Logg, C. Richardson, J. Ring, M. E. Rognes, and G. N. Wells, “The fenics project version 1.5,” *Archive of Numerical Software*, vol. 3, no. 100, 2015.
- [145] “Electricité de france, finite element `code_aster`, analysis of structures and thermomechanics for studies and research,” Open source on www.code-aster.org, 1989–2017.





The ROMSOC project

October 4, 2019

ROMSOC-D5.2-0.2

Horizon 2020