



**POLITECNICO**  
**MILANO 1863**

SCHOOL OF INDUSTRIAL AND INFORMATION ENGINEERING  
Master of Science Degree - Aeronautical Engineering

**A non-intrusive reduced order method  
based on Gaussian Process and  
Bayesian Optimization for hot rolling  
process**

**Advisor:**

Prof. Federico Piscaglia

**Co-advisor:**

Dr. Davide Baroli

**Candidate:**

Juan Jose Sandoval Sotelo 918598

**Academic Year 2021**

This page was intentionally left blank.

# Agradecimientos

*Todo se lo debo a mis padres mis hermanas, mi novia, mis amigos, y a las personas que de alguna manera hicieron de estos 2 años una experiencia única. Mil gracias, sin ustedes esto no hubiese sido posible.*

I would like also to thank professor Federico Piscaglia and my supervisor Davide Baroli because of the huge opportunity the gave to me. Thank to you I learned many interesting things and a new way of developing things.

# Acknowledgment

This result is part of a project that has received funding by Deutsche 5 Forschungsgemeinschaft (DFG, German Research Foundation) under Germany s Excellence Strategy EXC-2023 6 Internet of Production 390621612.

# Abstract

A Full Order Model (FOM) structural analysis of a classic hot rolling manufacturing process involves large deformations, thermal-plasticity, and mechanical contact, ending up as a high-non-linear and complex model, that requires high computational resources and prohibited cost for further performance of optimization of industrial process. For that reason, we instigate a technique, which is formerly developed in the aerodynamics community with successful outcomes, the non-intrusive reduced order methods. In such a framework, we choose to construct the reduced space by the Proper Orthogonal Decomposition (POD) method and parametrizing the multiphysics coupled problem concerning the variation of thermal mechanics properties of work-pieces. Subsequently, it is investigated the Gaussian Process Regression(GPR) used to construct a mapping between the parameters of the model and the projection of snapshot space onto the POD space. The novelty contribution presented in this work concerns the exploitation of Bayesian optimization in the Gaussian process to enrich the surrogate response surface and the instigation of a non-intrusive surrogate approach to highly nonlinear and evolving in time coupled problems. The numerical tests performed on hot rolling test case show efficient performance of 1000 times faster in comparison to the finite element-based forward simulation and without losing significantly the accuracy of the prediction.

# Sommario

In general, l'analisi strutturale Full Order Model (FOM) di un classico processo di produzione di laminazione a caldo coinvolge fenomeni fisici come grandi deformazioni, termoplasticità e contatto meccanico, risultando in un modello complesso e non lineare che richiede una grande quantità di risorse computazionali e un costo proibito per ulteriori prestazioni di ottimizzazione nel processo industriale. Per questo motivo, implementiamo una tecnica, che è stata precedentemente sviluppata nella comunità dell'aerodinamica con buoni risultati: Metodi di ordine ridotto non intrusivi. In tale quadro, scegliamo di costruire lo spazio ridotto con il metodo della Decomposizione Ortogonale Propria (Proper Orthogonal Decomposition oppure POD in inglese) e parametrizziamo il problema multifisico riguardante la variazione delle proprietà termomeccaniche dei "work-piece". Successivamente, si aggiunge al modello un processo gaussiano utilizzato per costruire una mappatura tra i parametri del modello e la proiezione dello "snapshot" sullo spazio POD. Il contributo di novità presentato in questo lavoro riguarda lo sfruttamento dell'ottimizzazione bayesiana nel processo gaussiano per arricchire la superficie di risposta surrogata e l'istigazione di un approccio surrogato non intrusivo a problemi accoppiati altamente non lineari e in evoluzione nel tempo. I test numerici eseguiti mostrano prestazioni efficienti di 1000 volte più veloci rispetto alla simulazione in elementi finiti e con un'alta accuratezza.

# Table of contents

- Acknowledgment . . . . . II**
- Abstract . . . . . III**
- Sommario . . . . . IV**
- List of figures . . . . . IX**
- List of tables . . . . . IX**
- Introduction . . . . . X**
- Introduction . . . . . XII**
- 1 State of Art . . . . . 1**
  - 1.1 Metal Forming Process . . . . . 1
    - 1.1.1 Hot Rolling . . . . . 2*
    - 1.1.2 Multiphysics modeling . . . . . 2*
  - 1.2 Hot Rolling FEM model . . . . . 8
    - 1.2.1 Enriched Fast Rolling Model . . . . . 8*
  - 1.3 Reduced Order Methods . . . . . 10
    - 1.3.1 Intrusive and non-intrusive ROM . . . . . 11*
    - 1.3.2 The Proper orthogonal decomposition(POD) . . . . . 17*
  - 1.4 Equation-free based interpolant: Gaussian process, ANN  
and radial basis function . . . . . 23
    - 1.4.1 GPR . . . . . 24*
    - 1.4.2 ANN . . . . . 26*

1.5 Bayesian Optimization . . . . .	29
1.5.1 Procedure . . . . .	30
<b>2 Full Order Model Digital Twin . . . . .</b>	<b>32</b>
2.1 Introduction . . . . .	32
2.2 Geometry definition . . . . .	34
2.3 Material Properties . . . . .	35
2.4 Interactions . . . . .	36
2.4.1 Surface to surface contact . . . . .	36
2.4.2 Surface radiation . . . . .	38
2.5 Multi-physics . . . . .	38
2.6 Meshing . . . . .	38
2.7 Boundary Conditions . . . . .	39
2.8 Results . . . . .	41
<b>3 Proper Orthogonal Decomposition . . . . .</b>	<b>44</b>
3.0.1 Gram Matrix . . . . .	45
3.0.2 SVD . . . . .	45
3.1 Truncation . . . . .	46
3.2 POD . . . . .	47
3.3 POD Results . . . . .	51
<b>4 Gaussian Process Prediction . . . . .</b>	<b>52</b>
4.1 Gaussian Process . . . . .	52
4.2 Bayesian Optimization . . . . .	57
4.3 Results . . . . .	58
<b>5 Conclusions and Future works . . . . .</b>	<b>75</b>
<b>Bibliography . . . . .</b>	<b>76</b>

# List of Figures

1.1.1 Rolling Procedure . . . . .	1
1.3.1 surrogate models [1] . . . . .	11
1.3.2 Reduced order Model Scheme . . . . .	12
1.3.3 Reduced Basis Procedure . . . . .	13
1.3.4 energy-rank POD . . . . .	17
1.3.5 SVD for number of row greater or equal than number of columns . . . . .	21
1.3.6 SVD for number of row greater or equal than number of columns . . . . .	21
1.4.1 GPR after training data. (Image taken from Mathworks.) . . . . .	24
1.4.2 A Gaussian distribution showing percentage of values within a certain standard deviation from the mean. (Courtesy of Andres Quesda, M.D., Department of Pathology and Laboratory Medicine, University of Texas-Houston Medical School.) . . . . .	25
1.4.3 GPR algorithm . . . . .	26
1.4.4 Propagation function . . . . .	27
1.4.5 Sigmoid function . . . . .	28
1.4.6 feedforward ANN . . . . .	28
2.1.1 Abaqus Workflow . . . . .	33
2.2.1 Geometry definition . . . . .	34
2.4.1 Master and slave surfaces. Only the master surface is the one that can penetrate the slave surface. Image taken from the Abaqus Manual [2] . . . . .	37
2.6.1 Mesh . . . . .	39
2.7.1 Geometry Reference . . . . .	41
2.8.1 Deformed body . . . . .	42
2.8.2 Displacement results . . . . .	42
2.8.3 Displacement results plot . . . . .	43
3.0.1 Chapter Workflow: The displacement that comes from the FEM model is taken as the input to construct the ROM by the next steps: Gram Matrix construction, SVD computation, SVD truncation, and POD construction. . . . .	44
3.1.1 Energy ranking for the modes obtained from the Gram Matrix. The modes were normalized with respect to the total sum of the basis for better visualization . . . . .	47

3.3.1 Computation of the error for every snapshot with respect to the recovered by using the POD . . . . .	51
4.0.1 Chapter Work Flow . . . . .	52
4.1.1 Results for the basis 1, a quite good approximation solution is found with the Gaussian Process. . . . .	54
4.1.2 Results for the basis 2: a similar but scaled behavior is obtained. . . . .	54
4.1.3 Results for the basis 3 . . . . .	55
4.1.4 Results for the basis 4 . . . . .	55
4.1.5 Results for the basis 5 . . . . .	56
4.1.6 Results for the basis 6 . . . . .	56
4.3.1 Results node 1 . . . . .	59
4.3.2 Results node 2 . . . . .	59
4.3.3 Results node 3 . . . . .	60
4.3.4 Results node 4 . . . . .	60
4.3.5 Results node 5 . . . . .	61
4.3.6 Results node 6 . . . . .	61
4.3.7 Results node 7 . . . . .	62
4.3.8 Results node 8 . . . . .	62
4.3.9 Results node 9 . . . . .	63
4.3.10 Results node 10 . . . . .	63
4.3.11 Results node 11 . . . . .	64
4.3.12 Results node 12 . . . . .	64
4.3.13 Results node 13 . . . . .	65
4.3.14 Results node 14 . . . . .	65
4.3.15 Results node 15 . . . . .	66
4.3.16 Results node 16 . . . . .	66
4.3.17 Results node 17 . . . . .	67
4.3.18 Results node 18 . . . . .	67
4.3.19 Results node 19 . . . . .	68
4.3.20 Results node 20 . . . . .	68
4.3.21 Results node 21 . . . . .	69
4.3.22 Results node 22 . . . . .	69
4.3.23 Results node 23 . . . . .	70
4.3.24 Results node 24 . . . . .	70
4.3.25 Results node 25 . . . . .	71
4.3.26 Results node 26 . . . . .	71
4.3.27 Results node 27 . . . . .	72
4.3.28 Results node 28 . . . . .	72

4.3.2	Results node 29 . . . . .	73
4.3.3	Results node 30 . . . . .	73
4.3.3	Results node 31 . . . . .	74
4.3.3	Results node 32 . . . . .	74

## List of Tables

2.2.1	Dimensions . . . . .	34
2.3.1	Material Properties . . . . .	36

# Introduction

During the last decades, huge efforts on obtaining more precise solutions have ended up in more refined and complex Differential equations. In this thesis, we will focus on Partial Differential Equations (PDE) that equally required improvements in computer technology and numerical methods to address advances in engineering applications.

The one here presented is the computational metal forming technique called Hot Rolling. This manufacturing process is estimated to be used in around 95% of the pre-processed metal to then being shaped as beams, sheet, plate, etc. As the market becomes every day more competitive, the necessity of continually developing new ways of faster computing is of crucial importance.

Part of the problem to obtain faster solutions is the fact that due to multiple sources of strong non-linearities inherent in metal forming problems such as thermal-plasticity in the bulk material, contact-mechanics in the surface of both contacts and large deformations, make the design analysis of large scale models turns into very complex PDEs.[3]

In order to solve the PDEs, two ways can be taken, either obtaining an analytical solution or a numerical solution. However, the first option usually becomes unfeasible as integrated multi-physics makes them more complex. As a result, and thanks to the advances in terms of less computational times for solving numerical methods, the second option has become the preferred choice of scientists.

The most used numerical methods to solve PDEs are the finite element method (FEM), finite volume methods (FVM) and finite difference methods (FDM), every one of them has its advantages and disadvantages concerning the others, however, in general, the main issue of selecting numerical methods is directly related with how refined the discretization of the continuous problem has to be because a range of degrees of freedom of  $[10^6 - 10^9]$  is often obtained.

For that reason, the solution of the high-fidelity model or Full Order Model (FOM) might take a considerable amount of time to be solved, making it sometimes unfeasible to handle. For example, in the field of aeronautics, to run a NON-parametrized Computational Fluid Dynamics (CFD) of a 3D-combustor, assuming that a High-Performance Computations (HPC) center is not available, the solution might take days or even weeks to be computed, then a parameterized simulation is not practical in any point of view.

For this reason, the development of faster methods to counteract this drawback of numerical methods is in continuous growth. In particular, reduced-order modelling (ROM) techniques has become a rapidly growing discipline [4][5][6][7][8][9][10][11][12][9][13][14][15] because it provides the potential to simulate physical and dynamic systems with substantially increased computation efficiency while maintaining high accuracy [16].

Although nowadays many ROM methods are available for linear systems, for nonlinear

systems this is an active area of research [17]. One of them is, that has been selected for the purposes of this thesis, is the Proper Orthogonal Decomposition (POD) method. It has optimized capabilities of being an efficient means of deriving a reduced basis for high-dimensional nonlinear flow systems [18].

A Reducer Order Model (ROM) can be divided into two categories: intrusive and non-intrusive methods. On the first hand, and intrusive-ROM(IROM) is addressed of showing instability and poor efficiency for non-linear problems[17]. Also, the necessity of having available the Governing Equations to construct the ROM, makes it impossible to be implemented in commercial software. On the other hand, a non-intrusive ROM is independent of the original FOM. Not only because to obtain the solution for the Full Order Model, the simulation has to be run by using the commercial software Abaqus, but also because our defined problem is highly non-linear, a non-intrusive method is selected.

Then, the innovative part of this thesis is developed here: A model that can compute not only the same solutions of the Full Order Model but also predict highly accurate solutions but within a considerably less amount of time than with a Full Order Simulation. To do so, first, we have to represent the input-output solution defined by a function that does not take into account any of the physics of the model. This function is defined by two inputs: the input-output set and the Kernel. Nevertheless, the GP is enriched with a Bayesian Optimization method, to make it more accurate, since the GP, by itself, is not able to take into account high non-linearities.

The goal of this thesis is to implement a two stages simulation: online-offline simulation, through replacing the Full Order Model with a Reduced Basis Approximation, obtained from a non-intrusive POD enriched with a Gaussian Process Regression and optimized with a Bayesian Optimization, to then assess the reduction of time simulation for the online part. The core part to construct the Reduced Order Model (ROM) and POD is written in Python language, characterized by a fast implementation time, with the addition of the following python libraries: Numpy, Pandas, Fenics, Abapy, PETSc, SLEPc, GPy, and emukit.

Concerning the previously introduced background, this thesis has been organized with the following structure: Chapter 1: The state of the art, as a way to present the background required to do and understand this thesis. Chapter 2: This is about the implementation of the Finite Element Model in the software ABAQUS CAE 2017 from Dassault Systemes to obtain the displacement to then construct the snapshot matrix. Please note that the goal of this chapter is not to do an extensive analysis of the results find in here but rather using them as the input for the next chapter, therefore a very specific step-by-step setup is not intended to be shown. Chapter 3: The implementation of the non-intrusive Proper Orthogonal Decomposition (POD) to obtain the Reduced Order Model(ROM), POD basis, and the reduced coordinates are presented. Chapter 4: The construction and

implementation of the Gaussian Process Regression(GPR) enriched with the Bayesian Optimization, taking as inputs the reduced coordinates and the snapshots projection, is demonstrated.

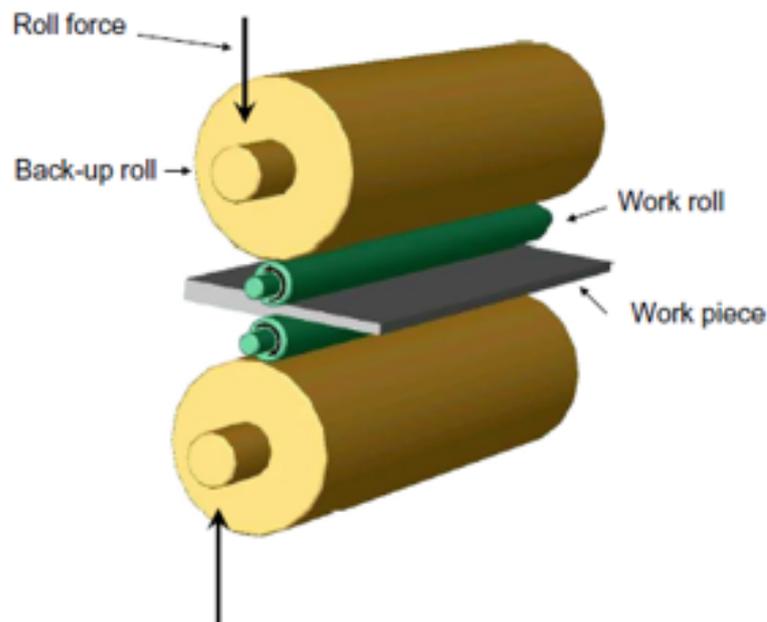
This thesis is part of a project that has received funding by Deutsche 5 Forschungsgemeinschaft (DFG, German Research Foundation) under Germany s Excellence Strategy EXC-2023 6 Internet of Production 390621612.

# State of Art

## 1.1 Metal Forming Process

Rolling Process is a widely and basic manufacturing process where more than 95% of metals (ferrous, non-ferrous, and alloys) have to go through in order to be usable in industrial applications: beams, sheet, plate, etc[19]. The main components are the work-piece, the work-rollers, one per each side and, the back-up roll. Moreover, the hot rolling can be performed in both hot and cold. This thesis empathizes on hot rolling only.

The working principle is the next: Initially, the work-piece needs to be reduced in thickness to transform the metal into an engineering piece. Afterward, it is pushed in a machine by a pusher to avoid the material to go back, then two work-rolls or rollers, induced by a torque, apply a compression load by constraining it to pass through a gap, smaller than the total thickness of the roller, between the rotating rolls, that is ideally fixed at a constant distance from each other by a set of more massive back-up rolls. The smaller the work-rolls, the less the energy used, but the bigger/massive the back-up rollers are. This process is repeated as many times as needed until the desired thickness is achieved, however, the gap has to be reduced every time, and this is done employing a certain number of sets of work-rolls and back-up rolls or by an adjustable single set. An exemplification of this process is shown in the figure 1.1.1.



**Fig. 1.1.1:** Rolling Procedure

### 1.1.1 Hot Rolling

The hot rolling process for high-scale-production companies is usually carried out in a traditional hot strip mill. However, there are several configurations. For more information about the other stages, please refer to the book of John G. Lenard in the bibliography section.[20].

The general stages, carried out at the IBF Aachen Laboratory, for this process are:

1. **Reheating furnaces:** The work-piece is heated up to a temperature higher than the re-crystallization temperature, normally in a range of 1200°C-1500°C, for two reasons. First, at the higher temperature the elastic modulus for a metal decreases then less force has to be applied to the work-piece for the same material but at room temperature. Second, because plastic deformation can be effectuated without increasing the deformation stress that can take place, making the material hard and brittle, therefore, harder to be deformed or even with non-desirable defects.

During this step a non-homogeneous temperature distribution in different sections of the work-piece occur, due to the fast cooling of the edges. For this reason, a post-processing procedure has to be conducted.

2. **Rough Rolling:** The work-piece is finally reduced in a range of 300-50 mm (depending on the number of passes), with a passing velocity from 1 m/s to 5 m/s.
3. **Cooling:** Depending on how the cooling process is performed the product microstructure of the product changes. Faster cooling means that the finished product will have a higher strength. Nevertheless, due to limitation conditions in the laboratory, this stage is performed by natural convection at room temperature.

### 1.1.2 Multiphysics modeling

The hot rolling process involves three physical models: First of all, high strain rates occur during rough rolling. Second, thermo-mechanical coupling comes about in three forms:

1. Heat transfer due to conduction when the roller (from now on the work-roller will be named in this way) and the work-piece interfaces are in contact and even more when the temperature differences are large.
2. Dissipation of heat due to friction work at the surfaces interface
3. Plastic work leads to heat dissipation

4. Heat transfer due to Radiation from the very hot work-piece ( 1200°C) to the room temperature roller( 25°C)

Third, mechanical contact is added to this modeling, since the physics included in the second stage involves it.

## Continuum thermo-mechanics and thermo-plasticity

There are many ways to implement thermo-elastic contact. The simplest way is based on node-to-segment contact approach and a penalty constrain enforcement is imposed. For more information about this method, refer to the reference section [21][22][23][24][25][26][27].

Another more sophisticated method is called Mortar Finite element method, as presented in reference [19]. For simplicity purposes, only the derivation of the continuum thermo-mechanics and thermo-plasticity, as well as the weak form are presented here.

Starting with the basic governing equation, with the following conventions:  $\Gamma \in R^3$  represents the continuum in reference configuration, while  $\partial\Omega$  refers to the surface of it. Additionally,  $\mathbf{x}$  refers to the current configuration of a material point  $\mathbf{X}$  that belongs to the continuum.

*Balance of mass:*

$$\dot{\rho}_0 = 0 \quad \text{in } \Omega \times (0, t_E] \quad (1.1.1)$$

*Balance of linear momentum:*

$$\text{Div } \mathbf{P} + \hat{\mathbf{b}}_0 = \rho_0 \ddot{\mathbf{u}} \quad \text{in } \Omega \times (0, t_E] \quad (1.1.2)$$

*Balance of angular momentum:*

$$\mathbf{P}\mathbf{F}^T = \mathbf{F}\mathbf{P}^T \quad \text{in } \Omega \times (0, t_E] \quad (1.1.3)$$

*Balance of energy:*

$$\dot{E} - \mathbf{P} : \dot{\mathbf{F}} + \text{Div } \mathbf{Q} - R = 0 \quad \text{in } \Omega \times (0, t_E] \quad (1.1.4)$$

*Clausius–Duhem inequality:*

$$\mathbf{P} : \dot{\mathbf{F}} - \dot{E} + T\dot{\eta} - \frac{1}{T}\mathbf{Q} \cdot \text{Grad } T \geq 0 \quad \text{in } \Omega \times (0, t_E] \quad (1.1.5)$$

Now, if the boundary conditions are introduced. Being  $\Gamma_u^D$  Dirichlet boundary conditions and  $\Gamma_u^N$  Neumann boundary conditions:

*Prescribed displacement:*

$$\mathbf{u} = \hat{\mathbf{u}} \quad \text{on } \Gamma_u^D \times (0, t_E] \quad (1.1.6)$$

*Prescribed traction force*

$$\mathbf{P}_N = \hat{\mathbf{t}}_0 \quad \text{on } \Gamma_u^N \times (0, t_E] \quad (1.1.7)$$

*Prescribed temperature*

$$T = \hat{T} \quad \text{on } \Gamma_T^D \times (0, t_E] \quad (1.1.8)$$

*Prescribed heat flux*

$$-\mathbf{Q}N = \hat{Q}_0 \quad \text{on } \Gamma_T^N \times (0, t_E] \quad (1.1.9)$$

*Initial displacement*

$$\mathbf{u} = \mathbf{u}_0 \quad \text{in } \Omega \times 0 \quad (1.1.10)$$

*Initial velocity*

$$\dot{\mathbf{u}} = \dot{\mathbf{u}}_0 \quad \text{in } \Omega \times 0 \quad (1.1.11)$$

*Initial Temperature*

$$T = T_0 \quad \text{in } \Omega \times 0 \quad (1.1.12)$$

Where  $\mathbf{u}$ ,  $\hat{\mathbf{t}}_0$ ,  $\hat{T}$  and  $\hat{Q}_0$  refer to prescribed displacement, Piola-Kirchoff tractions, temperature and heat flux per area in reference configuration, respectively. Furthermore,  $\mathbf{u}_0$ ,  $\dot{\mathbf{u}}_0$  and  $T_0$ , are the symbols for displacements, velocities and temperature at initial time, respectively.

Then, if Fourier Law, for any symmetric positive definite conductivity tensor  $\mathbf{K}$ , models the heat flux, and substitutes the tensor with the Jacobian determinant  $J = \det F$ , being

F the deformation gradient, and the scalar conductivity and assuming isotropic material, one can obtain:

$$\mathbf{Q} = -k_0 \mathbf{C}^{-1} \text{Grad } T \quad (1.1.13)$$

Where  $\mathbf{C}$  refers to the right Cauchy-Green tensor  $C = F^T F$

Some comments can be done for the last term of equation 1.1.5: By imposing the boundary condition 1.1.9 and taking into account that in the whole hot rolling process the temperature is positive, this term ends up being negative. As a result, this term can be removed from Clausius–Duhem inequality equation, and Clausius–Planck inequality is obtained:

$$\mathcal{D}_{\text{int}} := \mathbf{P} : \dot{\mathbf{F}} - \dot{E} + T\dot{\eta} \geq 0 \quad (1.1.14)$$

Afterward, as proposed in [28], the deformation gradient can be divided in two terms: the elastic contribution times the plastic contribution, as shown next:

$$\mathbf{F} = \mathbf{F}^e \mathbf{F}^p \quad (1.1.15)$$

Just as done as in the last step (proposed in [29]), the entropy can be separated into two terms:

$$\eta = \eta^e + \eta^p \quad (1.1.16)$$

Now, if Helmholtz free energy equation is defined in accordance with the last two formulations:

$$\mathcal{H} = \mathbf{E} - \mathbf{T}(\eta - \eta^p) \quad (1.1.17)$$

It is important to note that for finite strain applied on thermo-plasticity, equation (1.1.17) can be divided into three energetic contributions: elastic, hardening and thermal energy. As a result, equation (1.1.14) becomes:

$$\begin{aligned} \mathcal{D}_{\text{int}} = & \left( \mathbf{P} - \rho_0 \frac{\partial \psi^e}{\partial \mathbf{F}^e} \frac{\partial \mathbf{F}^e}{\partial \mathbf{F}} \right) : \dot{\mathbf{F}} + \left( -(\eta - \eta^p) + \rho_0 \frac{\partial \psi}{\partial T} \right) \dot{T} \\ & + \rho_0 \frac{\partial \psi^e}{\partial \mathbf{F}^e} \frac{\partial \mathbf{F}^e}{\partial \mathbf{F}^p} : \dot{\mathbf{F}}^p + \rho_0 \frac{\partial \psi^p}{\partial \alpha^i} \dot{\alpha}^i + T\dot{\eta}^p \geq 0 \end{aligned} \quad (1.1.18)$$

Where the term  $\alpha$  is defined as a strain-like scalar internal variable associated with isotropic hardening. Then, as  $\mathbf{F}, \dot{\mathbf{F}}, \mathbf{T}, \dot{T}$  may take arbitrary values, one can find the following constitutive equations:

$$\mathbf{P} = \rho_0 \frac{\psi^e}{\partial \mathbf{F}^e} \mathbf{F}^{p-T} \quad (1.1.19)$$

$$\mathbf{S} = 2\rho_0 \mathbf{F}^{p-1} \frac{\partial \psi^e}{\partial \mathbf{C}^e} \mathbf{F}^{p-T} \quad (1.1.20)$$

$$\eta^e = \eta - \eta^p = -\rho_0 \frac{\partial (\psi^e + \psi^p + \psi^\theta)}{\partial T} \quad (1.1.21)$$

To simplify last equation, Mandel stress tensor,  $\Sigma$ , and the plastic velocity gradient,  $\mathbf{L}^P$ , can be defined as in equation 1.1.22, and 1.1.23 by considering only the symmetrical part:

$$\Sigma = 2\mathbf{C}^e \partial \psi^e / \partial \mathbf{C}^e \quad (1.1.22)$$

$$\mathbf{L}^P = \dot{\mathbf{F}}^p \mathbf{F}^p - 1 \quad (1.1.23)$$

In the end, if the Clausius–Planck inequality (1.1.14), equation (1.1.21), and equation (1.1.18) are replaced in the Energy equation, equation (1.1.24) is obtained:

$$C_v \dot{T} = -\text{Div } \mathbf{Q} + R + \mathcal{D}_{\text{mech}} + H^{ep} \quad (1.1.24)$$

where  $H^{ep}$  is the elasto-plastic heating, and the specific heat capacity, per unit undeformed volume, is  $C_v = -\rho_0 T \frac{\partial^2 \psi}{\partial T \partial T}$ .

A last comment for this demonstration is the fact that the functions in charge of describing the evolution of *boldsymbol{F}^P*, and  $\alpha^i$  has not been introduced into here, however, this is out of the scope of this thesis. For more information regarding Hill orthotropic yield function, please refer to the reference [19].

## Thermo-mechanical contact modeling

The Hot rolling process can be simplified as two bodies in contact that are subjected to normal, and a tangential contact pressure [30]: The work-piece, being the slave surface, represented by  $\gamma^{C(1)}$ , and the roller, as the master surface,  $\gamma^{C(2)}$ . Both closed contours are defined by  $\Gamma^{(i)}$ ,  $i=1,2$  at reference condition. Then, imposing the same boundary conditions as in the last subsection (1.1.6-12), and enhancing the constraints of frictional contact at the contact surface when the roller and the work-piece are in contact  $\Gamma^{C(i)}$ , their surfaces can be decomposed into three subsets :

$$\partial\Omega^{(i)} = \Gamma_u^{D(i)} \cup \Gamma_u^{N(i)} \cup \Gamma^{C(i)} = \Gamma_T^{D(i)} \cup \Gamma_T^{N(i)} \cup \Gamma^{C(i)} \quad (1.1.25)$$

and

$$\emptyset = \Gamma_u^{D(l)} \cap \Gamma_u^{N(l)} = \Gamma_u^{D(i)} \cap \Gamma^{C(i)} = \Gamma_u^{N(i)} \cap \Gamma^{C(i)} = \Gamma_T^{D(i)} \cap \Gamma_T^{N(i)} = \Gamma_T^{D(i)} \cap \Gamma^{C(i)} = \Gamma_T^{N(i)} \cap \Gamma^{C(i)} \quad (1.1.26)$$

To define the normal gap  $g_n$ , two reference points have to be defined [31]:  $\mathbf{x}^{(1)}$  as an arbitrary point on the slave surface and  $\hat{\mathbf{x}}^{(1)}$  as its projection along the outward normal vector  $\mathbf{n}$ . In the same way, to define the relative tangential velocity  $\mathbf{v}_\tau$ , the material velocities of both points just defined are included in equation (1.1.28)

$$g_n = -\mathbf{n} \cdot [\mathbf{x}^{(1)} - \hat{\mathbf{x}}^{(2)}] \quad (1.1.27)$$

$$\mathbf{v}_\tau = (\mathbf{1} - \mathbf{n} \otimes \mathbf{n}) \cdot [\dot{\mathbf{x}}^{(1)} - \dot{\hat{\mathbf{x}}}^{(2)}] \quad (1.1.28)$$

The contact traction  $\mathbf{t}_c$  at the interface is decomposed in the same way to obtain the normal contact pressure  $p_n$  and the tangential contact traction  $\mathbf{t}_\tau$  [19]

$$p_n = \mathbf{n} \cdot \mathbf{t}_c^{(1)}, \quad \mathbf{t}_\tau = (\mathbf{1} - \mathbf{n} \otimes \mathbf{n}) \cdot \mathbf{t}_c^{(1)} \quad (1.1.29)$$

To define the mechanical constraints two conditions are added on the slave contact surface, :

1. Hertz–Signorini–Moreau condition for the normal direction (1.1.29)
2. Coulomb’s law of friction for the tangential direction (1.1.31)

$$g_n \geq 0, \quad p_n \leq 0, \quad p_n g_n = 0 \quad \text{on } \gamma_c^{(1)} \quad (1.1.30)$$

$$f^{fr} := \|\mathbf{t}_\tau\| - \mu(\theta_c) |p_n| \leq 0, \quad \mathbf{v}_\tau + \beta \mathbf{t}_\tau = 0, \quad \beta \geq 0, \quad f^{fr} \beta = 0 \quad \text{on } \gamma_c^{(1)} \quad (1.1.31)$$

In order to be consistent in normal and tangential direction, the following conditions are also added:

$$\dot{g}_n p_n = 0, \quad \dot{f}^{fr} \beta = 0 \quad (1.1.32)$$

To model  $\mu(\theta_c)$ , that represents the temperature dependent friction coefficient, the expression introduced in [32] is presented hereafter :

$$\mu(\theta_c) = \mu_0 \frac{(\theta_c - T_d)^2}{(T_d - T_0)^2} \quad (1.1.33)$$

Where  $\mu_0$  represents a reference friction coefficient at  $T_0$ , and  $T_d$  as the damage temperature, that is usually selected to be the lower melting temperature of the two contacting materials.

The last physical variable to be modeled for this sub-module is the local energy balance at the contact interface:

If the specific internal energy rate is assumed to be zero at all times, for simplicity only, then equation (1.1.34) is defined as the addition of the heat flux at the contact surfaces  $q_c^{(1)}$   $q_c^{(2)}$

$$\dot{e}_c = \mathbf{t}_\tau \mathbf{v}_\tau + q_c^{(1)} + q_c^{(2)} \quad (1.1.34)$$

For further information about possible choices for the heat fluxes at the contact interface, check the reference [32].

## 1.2 Hot Rolling FEM model

A wide variety of models that goes from very sophisticated approaches associating models at all scales, from the microstructure level of the product to the elastic behavior of the roll support, to very fast models used for on-line process control and automatic pass schedule optimization have been proposed to simulate the hot rolling process taking into account various interactions between the process, the evolution of the microstructure and the rolling mill. [33]

In the industry, a fast rolling model based on the slab method of elementary plasticity theory is widely used. Nevertheless, this model had to be enriched with a combination of thickness resolution of temperature with an advanced approach to also model shear strain.[34]

### 1.2.1 Enriched Fast Rolling Model

The fundamental calculation of the hot rolling procedure is the computation of force and torque. The process module's force calculation is based on the simplified roll force, defined on [35]

$$F = l_d \cdot b \cdot \sigma_m \cdot Q_p \quad (1.2.1)$$

where  $d_l$  is the length of the roll gap,  $b$  is the width of the rolling stock,  $\sigma_m$  is the mean flow stress and  $Q_p$  is a geometric factor that depends on the shape factor  $l_d/h_m$ , which requires to be calibrated on a FEM study.

Then, to calculate the Torque, Orowan [36] proposed the following relation.

$$M = F \cdot l_v \cdot m \quad (1.2.2)$$

Where  $l_v$  is a virtual distance from the center of the work roll to approximately half the length of the roll. This simple equation is corrected by a factor  $m$ , based on industrial forces and torques.

The kinematics of the process is simplified as a row of compression sequences. Moreover, if we assume that in the rolling direction the roll-stock is divided into slabs that are assumed to remain plane during compression, we find that the process is performed at uniaxial compression, then, it can be neglected the plane strain compression. As a result, one can obtain the next relation between the strain,  $\varepsilon$  and the initial  $h_i$  and final thickness of the slab  $h_f$ .

$$\varepsilon = \frac{2}{\sqrt{3}} \ln \left( \frac{h_i}{h_f} \right) \quad (1.2.3)$$

Now, to enrich this fast method two actions can be added: The implementation of Semi-empirical material equations and through-thickness resolution of temperature, strain and microstructure relations. On the first hand, the link between process variables and conditions in rolling to the materials rheological properties is carried out by the next equation:

$$\sigma = K \cdot \varepsilon^{(m_1+m_2 \cdot T)} \cdot \exp(T \cdot m_3) \cdot \varepsilon^{m_4} \cdot \exp(\varepsilon \cdot m_5) \quad (1.2.4)$$

where  $K, m_1, m_2, m_3, m_4$  and  $m_5$  represents material properties and  $T$  the temperature values.

Finally, to address temperature predictions, the classic Fourier partial differential equation (PDE) that takes into account heat flow and dissipation inside the roll stock as well as convection, radiation, and heat transfer to different surroundings is presented hereafter.

$$\rho \cdot c_p(T) \cdot \frac{\partial T}{\partial t} = \lambda(T) \cdot \left( \frac{\partial^2 T}{\partial y^2} \right) + W_Q \quad (1.2.5)$$

being  $\rho$ , the density,  $c_p$ , the specific heat,  $\lambda$ , the conductivity of the material,  $t$  is the time,  $y$  is the thickness direction, and  $W_Q$  represents the heat generated by dissipation.

### 1.3 Reduced Order Methods

The models to solve engineering problems become more refined and complex that its analytical solution is often compromised. To counteract this, numerical simulations are usually performed to obtain the solution. However, classical numerical approximations techniques, such as Finite Elements (FE), Finite Volumes, or spectral methods experience important challenges when dealing with engineering applications that require flow simulation and control, as in the case of aerodynamics, hydrodynamics, because although computational resources are in constant developing, multi-physics phenomena require  $1^6$  or  $1^7$  magnitudes of (DOF)degrees of freedom and several days of computations. Moreover, computers can perform only elementary operations in a very efficient way. For that reason, other mathematical operations (e.g integrates, matrix inversion, ...) require equivalent expression in terms of elementary operations.[37][6].

In the same way, if we are interested in solving problems that require parametric modeling (as in the case of this thesis), inverse identification, and process or shape optimization it is required the direct computation of a very large number of solutions of the concerned model for particular values of the problem parameters[38] [7]. If the number of parameters increases then both real-time and many-query simulations become unfeasible.

One way to deal with these issues is through the implementation of Reduced Orders Models (ROMs) that can reduce both the amount of CPU time and storage capacity. Usually, ROM techniques for a system with a high number degree of freedom are implemented in a projection framework based on reduced-space bases. However, there is a challenge when using this method: finding an accurate representation of the large-scale system but represented in the optimal value of reduced basis. Some well-known algorithms, such as optimal Hankel model reduction and balanced truncation, are carried out in polynomial time with the drawback of being impractical for large systems, as in the case of this thesis, because they require high computational resources that make them impractical.

To construct a ROM, first it is needed to construct the equation of motion for the model, then a result a Partial Differential Equation (PDE) is obtained. Next, if it is assumed that no analytical solution can be found, an approximation of the solution  $u(\mathbf{x}, t)$  can be defined:

For simplicity purposes a 1-D, linear and scalar PDE is presented here.

$$u(\mathbf{x}, t) = \sum_{i=1}^N U_i(t) N_i(\mathbf{x}) \quad (1.3.1)$$

where the solution is assumed to be separated in space and time. Then,  $U_i(t)$  is the solution in time and  $N_i(\mathbf{x})$  the solution in space. Moreover, the last one can be represented

by an associated shape function consistent with the boundary conditions of the model. The summation is defined from 1 to the number of nodes ( $N$ ) considered to approximate the field in the domain where the physical problem is defined. In other words, it depends on the discretization scheme, if it is defined as linear, then an algebraic problem of size  $N$  is obtained [4]

### 1.3.1 Intrusive and non-intrusive ROM

When implementing a ROM a selection has to be done because this will change with direction to be followed in the ROM implementation: is the code that computes the FOM or high-fidelity model an open-source or at least with access(not only this but they must be editable) to the governing equations or residual expressions? If the answer is positive an intrusive ROM might be an option, otherwise, the only option is a non-intrusive one. This selection is since the first option, as will be explained later, requires first to build a basis of a low-dimensional. This part requires the full-model operators, which means that the full-model operators need to be available either in an assembled form or through a routine that returns the action of the operators on a given vector.[39]. While the non-intrusive model takes the output of the FOM as its input. [1].

From the group of surrogate models, ROM is the only one that offers the possibility of having an intrusive, non-intrusive, or mixed approach. Moreover, a comparative study between interpolation and projection methods was carried out on heat transfer problems at reference [40]. There it was found that the POD projection method performs in a better way than the POD interpolation method for complex problems.

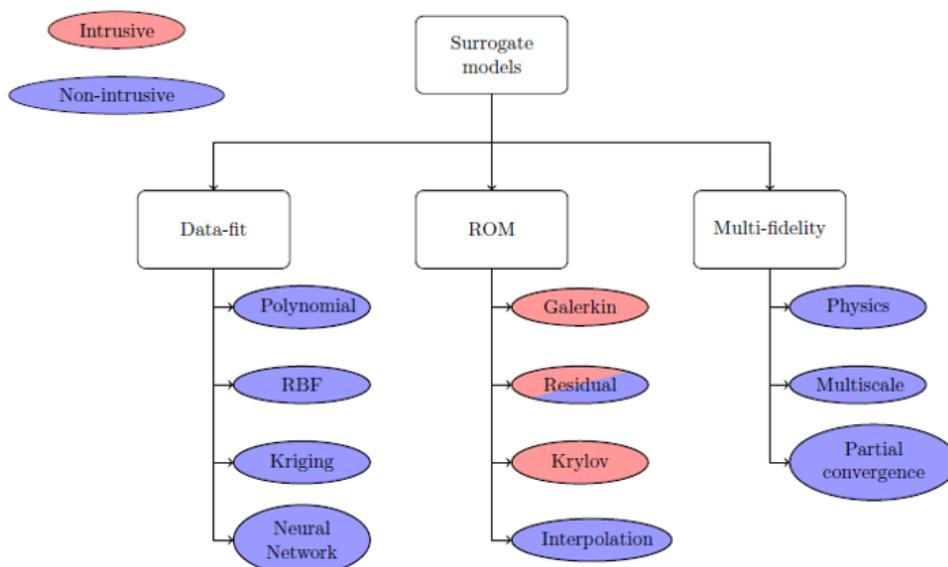


Fig. 1.3.1: surrogate models [1]

In general, as shown in fig 1.3.1, a surrogate model is divided into two stages. First, an offline section, where the snapshots are generated from the high-fidelity model. Then, a projection step is performed. Nonetheless, how this projection is done depends on whether an intrusive or non-intrusive ROM is applied. Second, an online section in charge of assembling and solving the Reduced Basis system followed by a posterior estimation of the error. Finally, the solution recovering is computed.

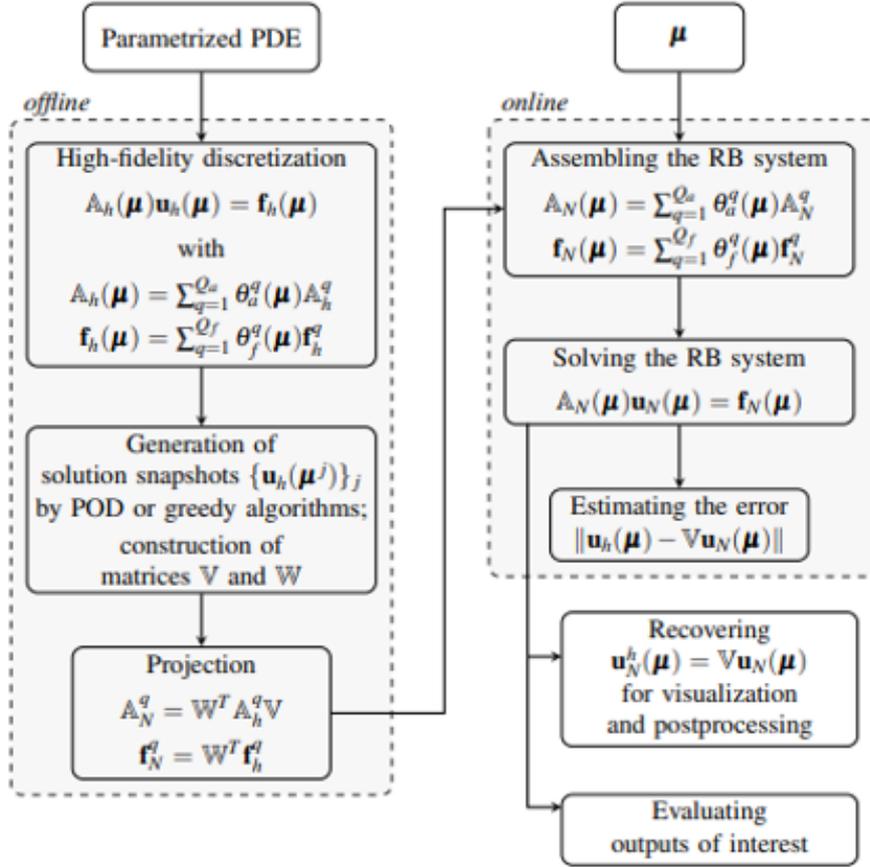


Fig. 1.3.2: Reduced order Model Scheme

## Intrusive Projection-based reduced methods

A reduced basis method for PDE looks for an approximated solution of the full order model (FOM), represented in equation (1.2.2), but, in a subspace of lower dimension, with the characteristic of being linear combinations among them, of the so-called basis functions generated by the snapshot matrix.

$$\mathbf{A}_h(\boldsymbol{\mu})\mathbf{u}_h(\boldsymbol{\mu}) = \mathbf{f}_h(\boldsymbol{\mu}) \quad (1.3.2)$$

The snapshot matrix (1.2.4) is built with a given set of solution from the FOM (1.2.3),

that correspond to  $N$  inputs varying over the parameter domain  $P$  (from now on these selected inputs will be called parameters, represented by  $\mu$  ).

$$\{u_h(\boldsymbol{\mu}^1), \dots, u_h(\boldsymbol{\mu}^N)\} \quad (1.3.3)$$

$$S_N = \mu^1, \dots, \mu^N \subset P \quad (1.3.4)$$

Then, the reduced problem is equivalent to the equation hereafter.

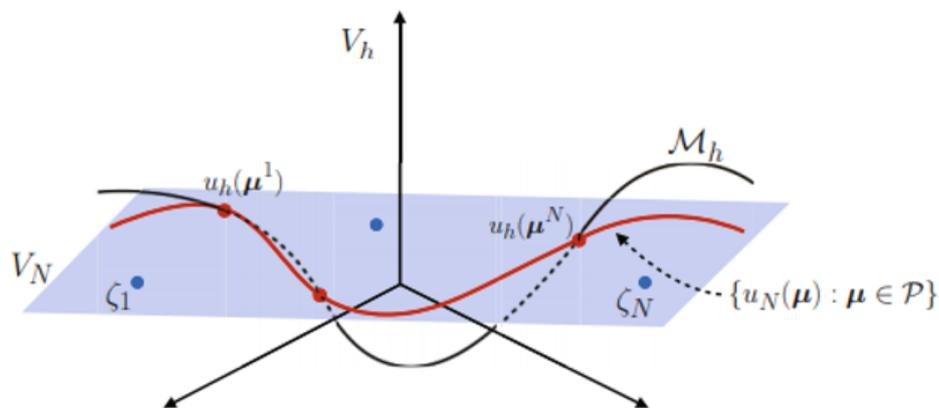
$$\mathbb{A}_N(\boldsymbol{\mu})\mathbf{u}_N(\boldsymbol{\mu}) = \mathbf{f}_N(\boldsymbol{\mu}) \quad (1.3.5)$$

In comparison to last equation  $\mathbf{A}_h$  is characterized for being a square matrix of dimension: number of degree of freedoms, while in the reduced space (1.2.5) it is represented in a subspace  $V_N \subset V_h$  of lower dimension  $N \ll N_h$ . Moreover,  $\mathbf{u}_h$  and  $\mathbf{u}_n$  are the vector of degree of freedoms for the FOM and ROM, respectively.

Then, a set of  $N$  functions are generated (the procedure is explained Afterward). These functions are called reduced basis and they span the reduced basis(RB) space, therefore, in general:  $\boldsymbol{\mu}^j \in P$  such that,  $\zeta_j = u_h(\boldsymbol{\mu}^j)$ .

$$V_N = \text{span} \{\zeta_1, \dots, \zeta_N\} \quad (1.3.6)$$

All this procedure can be summarized in the next image. From a FOM of 1 parameter (black line), the basis functions are computed (spanned in the reduced basis space in the blue plane), then a ROM is built (redline) to obtain the RB solutions.



**Fig. 1.3.3:** Reduced Basis Procedure

The last step, to find the RB solution, depends on the RB Method. Two approaches[5] are presented hereafter

## GALERKIN RB METHOD

If the parametrized PDE that governs the model is discretized, the strong formulation of the equation should be found for a given  $\boldsymbol{\mu} \in V$  :

$$L(\boldsymbol{\mu})u(\boldsymbol{\mu}) = f(\boldsymbol{\mu}) \quad \text{in } V' \quad (1.3.7)$$

Then, to implement a numerical approximation for the PDE, the weak formulation must be defined:

given  $\boldsymbol{\mu} \in P$ , find  $u(\boldsymbol{\mu}) \in V$  such that:

$$a(u(\boldsymbol{\mu}), v; \boldsymbol{\mu}) = f(v; \boldsymbol{\mu}) \quad \forall v \in V \quad (1.3.8)$$

Where  $a$  is the parametrized bilinear obtained from  $L(\boldsymbol{\mu})$ . For more details about this procedure, please refer to reference [5]. Then, the Galerkin reduced basis is implemented as shown hereafter. It is important to mention the following: this method is a project-based method, because the solution  $u_h(\boldsymbol{\mu})$  is projected onto the reduced basis  $V_N$  by means of the scalar product as shown next :

find  $u_N(\boldsymbol{\mu}) \in V_N$  such that

$$a(u_N(\boldsymbol{\mu}), v_N; \boldsymbol{\mu}) = f(v_N; \boldsymbol{\mu}) \quad \forall v_N \in V_N \quad (1.3.9)$$

Afterward, due to the orthogonal projection theorem[5], it is found that the Galerkin solution  $u_N(\boldsymbol{\mu}) \in V_N$  satisfies the property defined in equation 1.2.10 i.e. Based on the energy norm, the best combination of snapshots are automatically selected with this method:

$$u_N(\boldsymbol{\mu}) = \arg \min_{v \in V_N} \|u_h(\boldsymbol{\mu}) - v\|_{\boldsymbol{\mu}}^2 \quad (1.3.10)$$

## LEAST-SQUARED RB METHOD

To define this method, first Petrov-Galerkin reduced basis must be defined.

Continuing from the general RB solution (1.2.8), the reduced problem is defined as  $N$  equations coming from  $N$  independent conditions. As a result, the residual of the high-

fidelity problem is:

$$r(\boldsymbol{\mu}) = f(\boldsymbol{\mu}) - L(\boldsymbol{\mu})u_N(\boldsymbol{\mu}) \text{ of a subspace } W_N \subset V_h \quad (1.3.11)$$

Finally, the expression to define this RB method can be found in next equation:

find  $u_N(\boldsymbol{\mu}) \in V_N$  such that

$$a(u_N(\boldsymbol{\mu}), w_N; \boldsymbol{\mu}) = f(w_N; \boldsymbol{\mu}) \quad \forall w_N \in W_N \quad (1.3.12)$$

Next, as defined in [5], the last problem owns a special instance, but with respect to the former, this method allows choosing the test space. For further information refer to [5]. **Key issues** In references [41] and [42], it was studied the two principal issues that

an intrusive Reduced Order Model(ROM) addresses: lack of stability and non-linearity efficiency

On the first hand, stability issues were found on an intrusive POD method, mainly due to oscillation in the solutions after implementing a standard Bubnov-Galerkin projection of equations onto the reduced order space. As a result, non-linear terms are affected by those oscillations ending up in instability on the simulations.

To solve this issue, Xiao [36] proposed a cosine rule between the advection direction in Cartesian space-time and the direction of the gradient of the solution in a PetrovGalerkin method. As a consequence, the stability of the ROM solution was improved without tuning parameters. Other authors found that the stability depends on the choice of the inner product used to define the Galerkin projection. In 2008, [43] presented a stable symmetrical inner product that guarantees certain stability bounds for the linearized compressible Euler equations. Also, in [44] two stabilization methods for an intrusive POD were proposed.

On the other hand, a big issue that affects the selection of an intrusive ROM, is the Non-linearity reduction inefficiency because as seen in the Galerkin method, this method is limited to linear terms thus, after projecting onto the reduced space, the nonlinear term still depends on the original full system. As consequence, the reduced model still depends on the high dimensional size of the Full Order Model(FOM) system.

Some alternatives have been developed to speed up the computational time by using the discrete empirical interpolation method (DEIM). This method is a discrete variant of the empirical interpolation method (EIM) proposed in the context of reduced-basis model order reduction discretization of nonlinear PDE and consists of the reduction of the nonlinear components to reduce the computational complexity of the POD method.[36]

In consequence, factors of 10-100 speed up in CPU time over the original non-reduced model were found.

For the case of this thesis, the FEM model is represented by a high non-linear model, simulated on the commercial code ABAQUS that does not offer the possibility to access the governing equations. Thus, a non-intrusive scheme is selected instead.

## Non-intrusive reduced methods

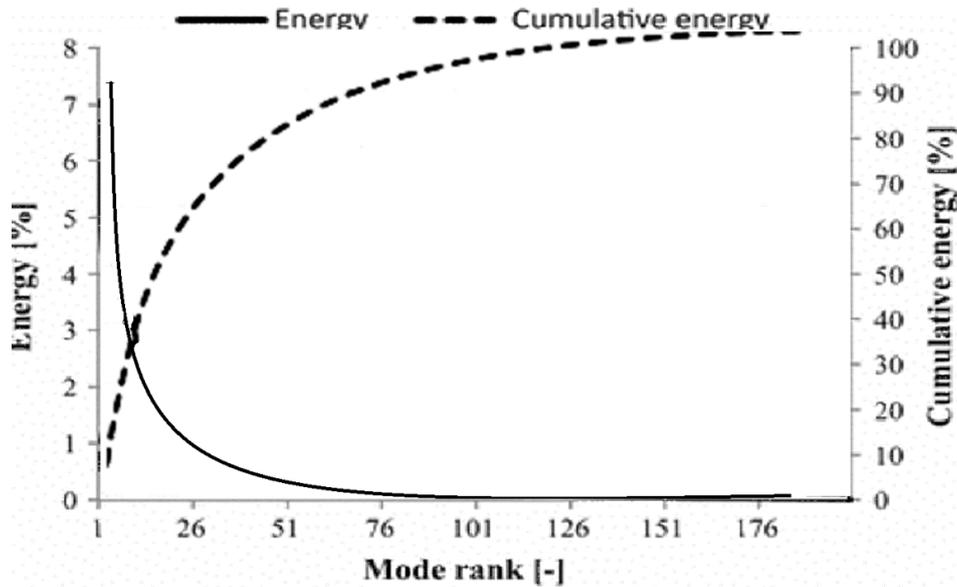
With respect to the last sub-section, non-intrusive reduced methods do not need access to the system of equations that model physics and do not modify the source code of the simulator. One of the first times these methods were used where at the reference [45], when two extensions to the proper orthogonal decomposition (POD) technique were considered for steady transonic aerodynamic applications.

On the first hand, to accurately obtain the variation of the inflow Mach number and the angle of attack, a procedure to couple a POD with an interpolation procedure. On the other hand, since some experimental data was incomplete/inaccurate, a POD taking this data as the input to complete/correct the data was added. In this method, the snapshot method as a way to determine the POD basis vectors was introduced by L. Sirovich [46]. It is mostly used for computational fluid dynamics (some examples of this applications can be found in the references [8] [9] [10] [11]) because it is a powerful tool to obtain RB for unsteady aerodynamic problems, from there it was possible not only to find very accurate solutions but fast. Furthermore, the same method was used in the reference [47] when POD was implemented to predict the flow temperature distribution in a regular cavity, with the novelty of varying Rayleigh number.

As time passes, more sophisticated POD-based models are implemented for parametric methods, as presented in [48], when the POD basis functions were used to characterize the geometric variation of the aerodynamic shape to then, obtain optimized results for fewer parameters. In reference, [49] an approach for obtaining the solution of the FOM in ROM form using a POD for unsteady aerodynamic model updating was presented. Moreover, a modification in the weakest point of a previous model was introduced to make it able to capture shocks and flow separation in transonic flow by imposing the minimization of experimental performances, and an aerodynamic model in reduced-order form via the POD approach. There it was possible to reassure the accuracy for such complex problems as for aeroelastic applications.

### 1.3.2 The Proper orthogonal decomposition(POD)

The POD is a linear method (this does not mean that modeled system must be linear) referred in the Euclidian Space based on the identification and extraction of coherent structures [1] with the aim of finding a proper orthonormal basis:the POD basis, by means of solving an eigenvalue problem to obtain the set of eigenvalues and corresponding eigenvectors to then sort them from more energetic to less energetic basis as in image 1.3.4.



**Fig. 1.3.4:** energy-rank POD

After that, the eigenvector and eigenvalue matrix can be truncated, based on a chosen criteria, to obtain a basis vectors, called modes, that represents the FOM but in ROM representation with considerable smaller matrix sizes, therefore POD contains more energy than any other basis. As a result, POD ends up being a representation of the function  $Z$  into  $k$  basis functions, such that:

for

$$\begin{aligned} z : \Omega \times \mathcal{T} &\rightarrow \mathbb{R} \\ x \times t &\mapsto f(x, t) \end{aligned} \quad (1.3.13)$$

$$z(x, t) \approx \sum_{k=1}^K a_k(t) \phi_k(x) \quad (1.3.14)$$

where  $\mathcal{T}$  is an interval of time,  $a_k$  is the scalar reduced coordinate associated with the  $k$ -th POD mode.

Then, to obtain an uniquely solution for the reduced coordinate and POD modes, with the characteristic for the POD modes to be optimal in the sense of most representative data, basis function must be ortho-normal by considering the inner product  $(\cdot)$  :

$$(\phi_i(x), \phi_j(x)) = \delta_{i,j}, \forall x \in \mathbb{R}^3, \forall i, j \in [1, K]^2 \quad (1.3.15)$$

where,

$$\delta_{i,j} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if otherwise} \end{cases}, \forall i, j \in [1, K]^2 \quad (1.3.16)$$

An interesting way of representing the reduced coordinates in terms of the basis function and the  $Z$  function is presented in the next equation. This representation will be crucial when we present the implementation of the Gaussian Process used as the input to obtain the reduced snapshots.

$$a_k(t) = (z(x, t), \phi_k(x)), \forall k \in [1, K] \quad (1.3.17)$$

As shown in reference [12], POD leads to a maximization problem, by maximizing the mean projection of the function  $z$  into the orthonormal basis function.

$$\begin{aligned} \max_{\phi_1, \dots, \phi_K} \sum_{k=1}^K \langle |(z(x, t), \phi_k(x))|^2 \rangle \\ \text{subject to } (\phi_i, \phi_j) = \delta_{i,j} \forall i, j \in [1, K]^2 \end{aligned} \quad (1.3.18)$$

where,

$$\delta_{i,j} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if otherwise} \end{cases}, \forall i, j \in [1, K]^2 \quad (1.3.19)$$

## Gram/Gramian Matrix

When the POD method is fed with the snapshot method, the first step is the assembling of the snapshot Gramian or Gram Matrix [43]. This is done with the aim of constructing an orthogonal basis.

Given a set  $V$  of  $m$  vectors (points in  $\mathbb{R}^n$ ), the Gram matrix  $\mathbf{G}$  is the matrix of all possible inner products of  $V$ , i.e.,

$$G_{ij} = \mathbf{v}_i^T \mathbf{v}_j \quad (1.3.20)$$

## Classical method, Singular Value Decomposition(SVD) and random SVD

This crucial part of the POD method is required to obtain the eigenvectors and their corresponding eigenvalue, to construct the reduced basis. To do so two big methods can be used to compute them: The traditional method or Singular Value Decomposition (SVD) in two different configurations, the traditional SVD, and the random SVD.

### CLASSICAL METHOD

Before entering in the details about this method, Fredholm equation has to be defined (for its derivation, please refer to [19]).

$$\int_D R(s, s') \phi(s') ds' = \lambda \phi(s) \quad (1.3.21)$$

Being,  $R$  the auto-correlation operator (linear, self-adjoint and non negative operator),  $\lambda$  is the eigenvalue matrix, and  $\phi$  are the basis functions.

After being defined last equation, the classical method requires a change of variable from the integration domain  $D$  of the inner product by  $\Sigma$  and  $X$  by the mesh coordinate  $\mathbf{x} \in \mathbf{R}^3$ . Also, if it is assumed that the average is represented by a temporal average, the Fredholm equation can be rewritten as:

$$\int_{\Omega} \frac{1}{T} \int_{\mathcal{T}} z(\mathbf{x}, t) z(\mathbf{x}', t) \phi(\mathbf{x}') dt d\mathbf{x}' = \lambda \phi(\mathbf{x}) \quad (1.3.22)$$

if the following discretization schemes substitutes the temporal and spatial integrals, as proposed in reference [12] as shown next;

$$\int_{\mathcal{T}} f(t) g(t) dt \approx \sum_{i=1}^{N_e} f(t_i) g(t_i) \quad (1.3.23)$$

$$\int_{\Omega} f(x) g(x) dx \approx \sum_{j=1}^d V_j f(x_j) g(x_j) \quad (1.3.24)$$

the Fredholm equation ends up as:

$$\frac{1}{N_t} \sum_{i=1}^{N_e} z(\mathbf{x}, t_i) \sum_{j=1}^d V_j z(x_j, t_i) \phi(x_j) = \lambda \phi(\mathbf{x}), \quad \forall \mathbf{x} \in \Omega \quad (1.3.25)$$

where  $V$  corresponds as a weighted volume of the correspondent cell of the mesh and  $d$  is the degree of freedoms.

Finally, If matrix notation is introduced the following general eigenvalue problem is obtained:

$$\hat{Z}\hat{Z}^T = \Phi^T \lambda \Phi \quad (1.3.26)$$

Where,

$$\hat{Z} = (M^{1/2})^T Z \text{ and } M \text{ is a diagonal matrix such as } M_{ii} = \frac{V_i}{N_t}$$

If the snapshot method wants to be used instead, the basis function can be defined as a linear combination of the snapshots  $a$ :

$$\phi(x) = \sum_{k=1}^{N_t} a(t_k) z(\mathbf{x}, t_k). \quad (1.3.27)$$

Then, the eigenvalue problem can be written as

$$\hat{Z}^T \hat{Z} = \mathbf{A}^T \lambda \mathbf{A}. \quad (1.3.28)$$

where,

$$\mathbf{A} = [a(t_1) \cdots a(t_{N_t})]^T$$

With respect to the traditional problem, snapshot is much more efficient because it eigenvalue problem is of size  $N_s$ , such that  $N_s \ll d$ .

## SVD

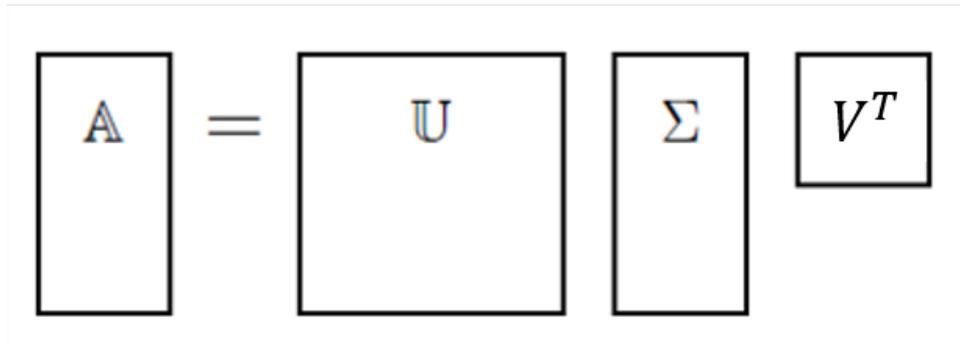
SVD is defined by [48] as a diagonalization process involving left and right multiplication by orthogonal, depending on the quantity of elements in the row and column.

For  $A \in R^{m \times n}$  :

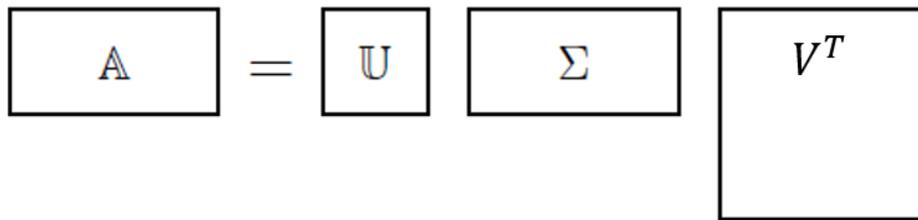
1. if  $m \geq n$ :
2. else:

This means that if  $A \in R^{m \times n}$  is real matrix, there exists two **orthogonal** matrices,  $U$  and  $V$  are respectively  $m \times m$  and  $n \times n$  unitary:

$$U = [\zeta_1 | \dots | \zeta_m] \in \mathbb{R}^{m \times m}, \quad V = [\psi_1 | \dots | \psi_n] \in \mathbb{R}^{n \times n} \quad (1.3.29)$$



**Fig. 1.3.5:** SVD for number of row greater or equal than number of columns



**Fig. 1.3.6:** SVD for number of row greater or equal than number of columns

that forms the following expression:

$$\mathbb{A} = \mathbb{U}\Sigma\mathbb{V}^T \quad (1.3.30)$$

having  $\Sigma$  as a diagonal matrix, such that:

$$\Sigma = \text{diag}(\sigma_1, \dots, \sigma_p) \in \mathbb{R}^{m \times n} \quad (1.3.31)$$

and

$$\Sigma_{ii} = \sigma_i, \forall i \in [1, \dots, \min(m, n)] \quad (1.3.32)$$

being  $\sigma_i$  the singular values: sorted in a descending order and non-negative.

Moreover, close relations between SVD and eigenvalue decomposition can be evidenced if it is applied to the function  $\mathbf{z}$  and multiplied by its transpose:

$$\hat{\mathbf{Z}}\hat{\mathbf{Z}}^T = \mathbb{U}\Sigma\mathbb{V}^T (\mathbb{U}\Sigma\mathbb{V}^T)^T = \mathbb{U}\Sigma\mathbb{V}^T\mathbb{V}\Sigma^T\mathbb{U}^T = \mathbb{U}\Sigma\Sigma^T\mathbb{U}^T \quad (1.3.33)$$

Now if equation (1.2.25) and last equation are compared, the following relations are found:

$$\begin{aligned} \mathbb{U} &= \phi^T \\ \Sigma\Sigma^T &= \lambda \end{aligned} \quad (1.3.34)$$

It is important to note that the diagonal matrix  $\Sigma$  (and there are the singular values) is

equal to the square root of the eigenvalue matrix.

A last vital aspect of this method is the link between the POD and the SVD in the sense of obtaining a linear system that relates them, by means of mathematical manipulations [1], then computing the truncated SVD of  $\hat{Z}$  gives a direct access to the values of the POD basis. **RANDOM SVD** The method here presented is an alternative for the traditional

SVD for high dimension parameter(s). For example, in reference [50], they draw many samples by projecting the parameter to the subspace spanned by its eigenvectors, what they call a Hessian-based parameter. However, the traditional SVD becomes unfeasible when the Hessian matrix becomes very large because of the large number of PDEs, therefore they employed a randomized SVD algorithm to compute the dominating eigenpairs of the Hessian. As a result, just some PDE solutions had to be found.

Also, in ref [51], they combine the standard Nyström method and the randomized SVD algorithm, to speed up the simulation since standard Nyström is highly efficient but requires a large enough number of columns to be sampled, while the randomized SVD algorithm is highly accurate but less efficient (in the computer engineering point of view). There, they found (through experiments) that the proposed algorithm is as accurate as the standard Nyström but with much lower time complexity.

## RANDOMIZED SVD ALGORITHM

This algorithm consists of two steps.

If  $\mathbf{A}$  is a matrix of size  $m \times n$ , the aim is to obtain matrices  $\mathbf{B}$  with size  $m \times k$  and  $\mathbf{C}$  with size  $k \times n$ , such that  $k \ll n$  and  $\mathbf{A} \approx \mathbf{B} \mathbf{C}$ . In other words, this procedure computes an approximate rank-2k factorization-SVD.

1. step 1:

- (a) Generate an  $n \times 2k$  Gaussian test matrix  $\Omega$ .
- (b) Optional step: if high accuracy is required, orthonormalization is required for each  $\mathbf{A}$  and  $\mathbf{A}^*$
- (c) Compute  $\mathbf{Y}$  by means of  $\mathbf{Y} = (\mathbf{A}\mathbf{A}^*)\mathbf{q}\mathbf{A}\Omega$  ; try to use as many digits as possible since  $\mathbf{Y}$  is vulnerable to round-off errors.
- (d) Construct a matrix  $\mathbf{Q}$  whose columns form an ortho-normal basis for the range of  $\mathbf{Y}$  .

2. step 2:

- (a) Compute the matrix  $B$  by means of :  $B = Q * A$
- (b) Apply the SVD for  $B$ , so then,  $B = \tilde{U}\Sigma V$
- (c) Compute  $U$  by means of:  $U = \hat{U} * Q$
- (d) return  $\Sigma, \hat{V}$  and  $U$

**SVD vs EIGEN DECOMPOSITION** Concerning the SVD, eigendecomposition has a better performance in the computation of the eigenvectors, but with the drawback of low precision for the smallest eigenvalues, and not sequential implementation is possible. On the other hand, SVD does perform better when precision for the smallest eigenvectors is computed, with the advantage that, SVD can implement a sequential computation, but paying a value of slow computation of the singular vectors. Moreover, for a matrix of rank  $k$ , and size  $m \times n$  a standard deterministic technique for computing SVD typically has a cost of  $(kmn)$  flops, while with randomized schemes it can produce an approximate SVD using only  $O(mn \log(k) + (m+n)k^2)$  flops. [52]

## 1.4 Equation-free based interpolant: Gaussian process, ANN and radial basis function

The innovative part of this thesis stays here because the aim is to replace the high-fidelity model, or also called ROM, that is driven by a set of governing equations, for a predictive method, based on a statistical approach, to find not only the same solution of the FOM but also predict a not prior known solution. In other words, with a new set of parameters (in this case, the elastic modulus and the density) being able to compute a correct solution. To do so there are three big methods: Gaussian Process Regression (GPR), Artificial Network (ANN), and Radial Basis Function (RBF).

Before entering into the subject , some concepts need to be understood first:

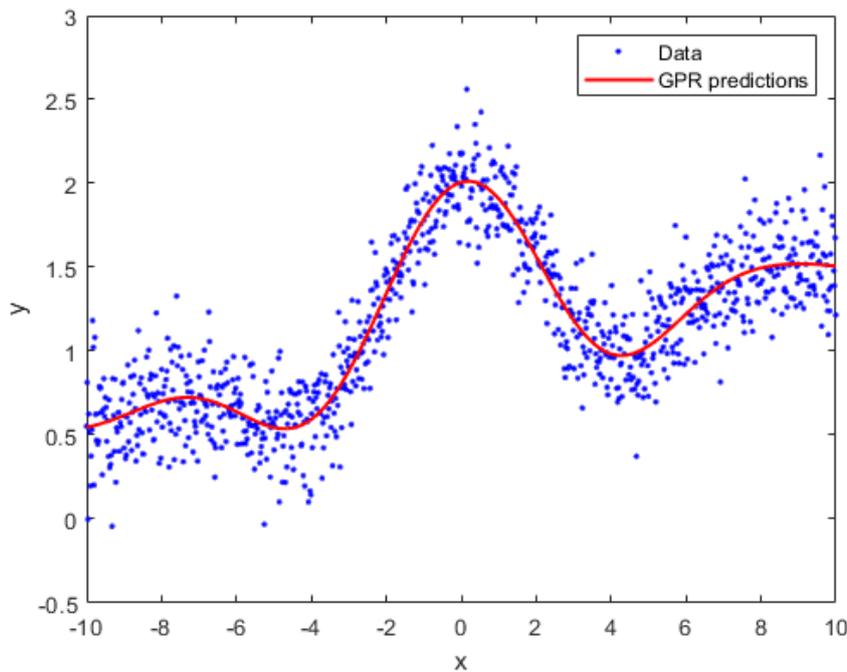
### Supervised and unsupervised learning

Supervised learning and unsupervised learning are the most general division that can be made about the types of learning in Machine learning.

On the first hand, in supervised learning the model with labeled data (already classified) is given as an input, to infer a learning algorithm. In the same way, a set of the desired output is also given for the learning process, then an error is computed between the predicted values and the one from the learning process. On the other hand, contrary to the former method, unsupervised learning receives data that has not been classified or categorized, so that the algorithm has to be able to underlay relationships or features from that input and group cases with similar features or characteristics. [53]

### 1.4.1 GPR

Gaussian Process Regression is a type of supervised learning in the form of regression (for continuous outputs), that constructs a model from a set of observation data: If  $D$  is the training set of  $M$  observations:  $\mathcal{D} = \{(\mathbf{x}_i, y_i) : i = 1, 2, \dots, M\}$ ,  $x_i$  is the input of  $d$  entries that lies in the input domain  $P$ , while  $Y_i$  is the output that corresponds to the input [54] [53].



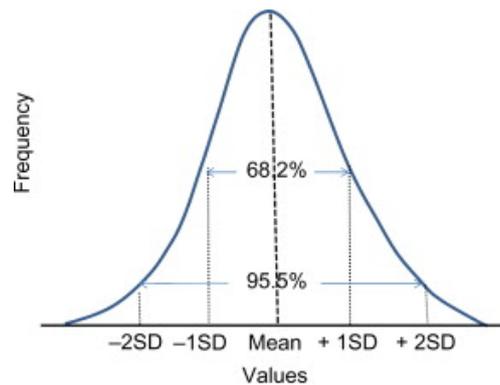
**Fig. 1.4.1:** GPR after training data. (Image taken from Mathworks.)

A distinction between a Gaussian distribution and Gaussian process needs to be clarified. The former one is represented by a set of vector and matrix: the mean function  $m(x)$ , and the covariance function  $k(x', x)$ , represented by a vector and a matrix, respectively. Therefore, this process describes probability distributions by functions, while the gaussian distribution is over vectors (a classic representation of this can be seen in the next image). In other words, Gaussian distribution is a natural generalization of the Gaussian probability distribution. [54]

The fundamental representation of a Gaussian process is presented hereafter, where the function  $f$  is distributed as a GP as function of the mean function  $m$  and covariance function  $k$ :

$$f \sim \mathcal{GP}(m, k(\mathbf{x}, \mathbf{x}')) \quad (1.4.1)$$

Then, from the random input set of data it is assumed that a finite number of them obeys a joint Gaussian distribution. Additionally, if a Gaussian noise is added to the model, in



**Fig. 1.4.2:** A Gaussian distribution showing percentage of values within a certain standard deviation from the mean. (Courtesy of Andres Quesada, M.D., Department of Pathology and Laboratory Medicine, University of Texas-Houston Medical School.)

order to take in account some data corruption, the next equation is obtained.

$$f(\mathbf{x}) \sim \text{GP}(0, \kappa(\mathbf{x}, \mathbf{x}')), \quad y = f(\mathbf{x}) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma_y^2) \quad (1.4.2)$$

## Kernel

In a GPR model, it is assumed that the set input-output follows a regression function (Kernel), that can be affected by noise. From there, a probabilistic distribution is found, then predictions are carried out by giving new inputs.

Said so, the importance of selecting or defining the correct kernel is evident. If this an unknown, a common one to start with, is the squared exponential kernel, function of the standard deviation  $\sigma_f$ , and a correlated length-scale  $\ell$ .

$$\kappa(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\frac{1}{2\ell^2} \|\mathbf{x} - \mathbf{x}'\|_{\mathbb{R}^d}^2\right) \quad (1.4.3)$$

Another interesting kernel characterized by considering a length scale  $\ell$  and the distance for being uncorrelated along  $x_m$  for each of the inputs dimensions is the automatic relevance determination(ARD) kernel.

$$\kappa(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\frac{1}{2} \sum_{m=1}^d \frac{(x_m - x'_m)^2}{\ell_m^2}\right) \quad (1.4.4)$$

## Hyper-parameters $\theta$

With respect to the parameters that are used to make the GP model to fit the training data, there are others that express “higher-level” properties of the model such as its complexity or how fast it should learn and make a significant difference on the predictive performance [53] [54]. For the case of SE kernel, the hyper-parameters repented by  $\theta$  are:

$$\boldsymbol{\theta} = \{\sigma_f, \ell, \sigma_y\} \quad (1.4.5)$$

and for the case of ARD SE kernel:

$$\boldsymbol{\theta} = \{\sigma_f, \ell_1, \dots, \ell_d, \sigma_y\} \quad (1.4.6)$$

Some authors [55][13][53][54], by using a standard gradient-based optimizer, can estimate it opt via the maximization problem:

$$\boldsymbol{\theta}_{\text{opt}} = \arg \max_{\boldsymbol{\theta}} \log p(\mathbf{y} | \mathbf{X}) = \arg \max_{\boldsymbol{\theta}} \left\{ -\frac{1}{2} \mathbf{y}^T \mathbf{K}_y^{-1}(\boldsymbol{\theta}) \mathbf{y} - \frac{1}{2} \log |\mathbf{K}_y(\boldsymbol{\theta})| - \frac{M}{2} \log(2\pi) \right\} \quad (1.4.7)$$

where  $p(y|X)$  is the conditional density function of the marginal likelihood ( $y$  given  $X$ ).

Finally, the algorithm that represents the GPR, proposed by [53] is presented herefter:

---

### Algorithm 3 GPR

**Input:** A training set of  $M$  observations  $\mathcal{D} = \{(\mathbf{x}_i, y_i) : i = 1, 2, \dots, M\}$ , a chosen kernel function  $\kappa(\cdot, \cdot)$ , test inputs  $\mathbf{X}^* \in \mathbb{R}^{d \times M^*}$

**Output:** Test outputs  $\mathbf{f}^* | \mathbf{X}^*, \mathbf{X}, \mathbf{y}$

- 1: Estimate the optimal hyperparameters  $\boldsymbol{\theta}_{\text{opt}}$  by maximizing the likelihood, in each iterative step of which one needs to
    - 2: Form a covariance matrix  $\mathbf{K}_y = \kappa(\mathbf{X}, \mathbf{X}) + \sigma_y^2 \mathbf{I}_M$ ;
    - 3: Calculate a vector  $\mathbf{a} \in \mathbb{R}^M$  such that  $\mathbf{K}_y \mathbf{a} = \mathbf{y}$ ;
    - 4: Calculate the likelihood  $\log p(\mathbf{y} | \mathbf{X}) = -\frac{1}{2} \mathbf{y}^T \mathbf{a} - \frac{1}{2} \log |\mathbf{K}_y| - \frac{M}{2} \log(2\pi)$ ;
    - 5: Calculate the gradient of the likelihood with respect to the hyperparameters;
    - 6: Set  $\mathbf{K}_y = \mathbf{K}_y(\boldsymbol{\theta}_{\text{opt}})$  and  $\mathbf{a} = \mathbf{a}(\boldsymbol{\theta}_{\text{opt}})$  for the optimal hyperparameters;
    - 7: Form correlation matrices  $\mathbf{K}^{**} = \kappa(\mathbf{X}^*, \mathbf{X}^*)$  and  $\mathbf{K}^* = \kappa(\mathbf{X}^*, \mathbf{X})$  for the optimal hyperparameters;
    - 8: Calculate a matrix  $\mathbf{A}^* \in \mathbb{R}^{M \times M^*}$  such that  $\mathbf{K}_y \mathbf{A}^* = \mathbf{K}^*$ ;
    - 9: Form the conditioning mean value vector  $\mathbf{m}^* = \mathbf{K}^{*T} \mathbf{a}$  and the corresponding covariance matrix  $\mathbf{C}^* = \mathbf{K}^{**} - \mathbf{K}^{*T} \mathbf{A}^*$ ;
  - 10: Define  $\mathbf{f}^* | \mathbf{X}^*, \mathbf{X}, \mathbf{y} \sim \mathcal{N}(\mathbf{m}^*, \mathbf{C}^*)$ .
- 

**Fig. 1.4.3:** GPR algorithm

## 1.4.2 ANN

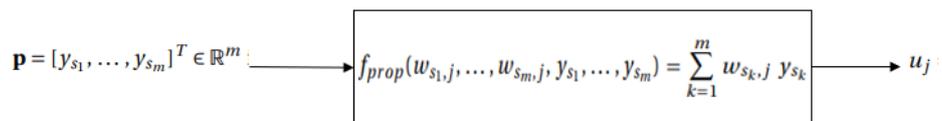
Artificial Neural Network(ANN) or neural network is computational model inspired by the ability the human brain has to organize and compute certain procedures (pattern recognition, perception, and motor control) based on observational data [14] [56]. This model consists of artificial neurons, that are a set of processing units, and a collection of weighted synaptic connections, that interconnects the neurons and follow the direction

imposed by the synapses. Hence, an artificial neural network is an oriented graph, with the neurons as nodes and the synapses as oriented edges, whose weights are adjusted by means of a training process to configure the network for a specific application. [57]

First, based on the last image, a neuron, named  $j$ , is connected with  $m$  neurons, that are sending data, and  $n$  that are receiving. such that:  $\{s_1, \dots, s_m\}$  and  $\{r_1, \dots, r_n\}$ , respectively. Then, the output of a generic neuron,  $\alpha$ , is defined as  $y_\alpha(t) \in R$ , at time  $t$  with a related weight  $w_{\alpha,\beta}$  from a generic neuron to another ( $\alpha \rightarrow \beta$ ). As a result, the previously defined neuron  $j$ , receives the weighted inputs : 1)  $w_{s_k,j}$  and 2)  $y_{s_k}(t)$ , where  $k$  goes from 1 to the number of sender neurons, at a time  $t$ . Afterward, this neuron, sends an output  $y_j(t + \Delta t)$  to the receiver neurons, and this receives it as an input of the form:  $w_{j,r_i} y_j(t + \Delta t)$ .

Moreover, to characterize the neuron  $j$  three functions are defined

1. The propagation function: It is in charged of converting a vectorial input  $\mathbf{p}$  into a net input i.e a scalar. This function is usually replaced by a weighted sum, and then it is converted into an adder [14].



**Fig. 1.4.4:** Propagation function

2. The activation function or squashing function, is a typically non linear functions in charge of limiting the output of a neuron by means of a threshold or bias  $\theta_j \in R$ .

$$a_j = f_{act} \left( \sum_{k=1}^m w_{s_k,j} y_{s_k} - \theta_j \right) \quad (1.4.8)$$

One of the most common choices to define the activation function is the Sigmoid function. This is because is a bounded, monotonic, differentiable real function that spans in a range between  $[0,1]$  i.e a positive derivate everywhere, and is also a suitable extension of the soft limiting nonlinearities used previously in neural networks.[58]

3. The output function, that is in charge of caculate the scalar output  $y_j \in R$ , based on  $a_j$ :

$$y_j = f_{out} (a_j) \quad (1.4.9)$$

If many neurons are interconnected, the neuronal network shown in the next image can be obtained. :

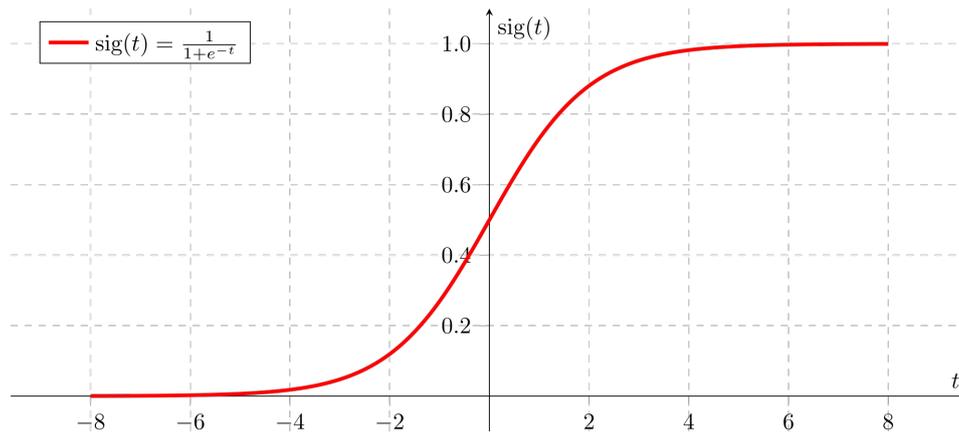


Fig. 1.4.5: Sigmoid function

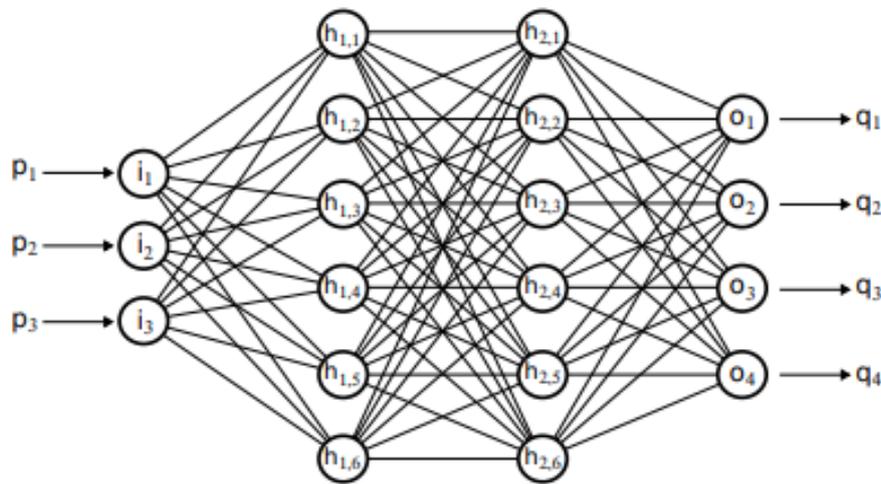


Fig. 1.4.6: feedforward ANN

This ANN is a feed-forward Neuronal Network, that consists of three-layer with three input neurons  $P_i$  for  $i = 1, 2, 3, 4$ , two hidden layers each one consisting of six neurons ( $h_{1.1}$  - $h_{1.6}$ ) and ( $h_{2.1}$  - $h_{2.6}$ ), and four output neurons  $Q_i$  for  $i = 1, 2, 3, 4$ . For further information of this type of ANN, please refer to [58] [57] [56] [14].

### RBF as activating function

A radial basis function (RBF) is a type of function used as an activation function for neuronal network with the characteristic of having the output symmetric around an associated center, named  $\mu_c$  [59].

In general a RBF is represented by:

$$\phi_c(\mathbf{x}) = \phi(\|\mathbf{x} - \mu_c\|; \sigma) \quad (1.4.10)$$

where  $\|\cdot\|$  represents a vector norm and  $\sigma$  width or scale parameter

For a scalar input, a typical RBF is :

$$f(x) = \exp\left(-\frac{(x-c)^2}{r^2}\right) \quad (1.4.11)$$

where  $c$  is the center, and  $r$  the radius.

Another possible choice is the multiquadric RBF , represented by :

$$f(x) = \frac{\sqrt{r^2 + (x-c)^2}}{r} \quad (1.4.12)$$

From these functions a RBF network can be implemented as input-output mapping. The simplest one consists of three-layers feed-forward neural network (similar to the one in 1.4.6 but with only one hidden layer) and it is able to provide a local presentation of a  $N$ -dimension space.

Radial basis function networks are distinguished from other neural networks due to their universal approximation and faster learning speed. An RBF network is a type of feed forward neural network composed of three layers, namely the input layer, the hidden layer and the output layer.

These methods are also used to approximate multivariate functions, especially in the absence of grid data.[60] Moreover, their main characteristic comes from their response decreases or increases monotonically with distance from a central point. With respect to other NN there are distinguished due to their universal approximation and faster learning speed. [61]

## 1.5 Bayesian Optimization

Bayesian optimization(BO) is a suite of techniques for machine learning optimization, in order to tune hyperparameters, or in general any "black-box" function but focusing not on finding a local optimum but a global one.

In general, BO is focused on solving the problem[62]:

$$\max_{x \in A} f(x) \quad (1.5.1)$$

The typically characteristics that the feasible set  $A$ , the input  $x$  and the objective function should have, are summarized hereafter.

For the input  $x$ :

1. it is in  $R^d$  , with a typical value of  $d$  lower than 20.

For the feasible set  $A$ :

1. Simple set, with easiness to assess relations.
2. Preferable a hyperrectangle  $\{x \in \mathbb{R}^d : a_i \leq x_i \leq b_i\}$  or the d-dimensional simplex  $\{x \in \mathbb{R}^d : \sum_i x_i = 1\}$ . For further information, please refer to [62]

For the function  $f$ :

1. Continuous
2. number of evaluations is limited, around a few hundred. Since, they take a significant amount of time.
3. not easy to optimize in the sense of it lack of special structure like concavity or linearity. i.e it is a black box
4. derivative free

Additionally, it is assumed that in almost all on BO, noise is independent across evaluations and Gaussian with constant variance. [62]

### 1.5.1 Procedure

BO is composed by two components: a Bayesian statistical model for modeling the objective function and an acquisition function for deciding where to sample next [62].

The algorithm can be divided in three main parts: an initialization, a while-loop and the solution.

1. Initialization:
  - (a) Place a Gaussian process prior on  $f$
  - (b) Observe  $f$  at  $n_0$  points according to an initial space-filling experimental design. for example a Design of Experiments
  - (c) set  $n$  at  $n_0$
2. while  $n \leq \text{max\_iteration}$ , some authors based the max iterations based on a budget of  $N$  function evaluations:
  - (a) Update the posterior probability distribution on  $f$  using all available data
  - (b) Identify the input  $x_n$  that maximizes of the acquisition function over  $x$ , where the acquisition function is calculated using the current posterior distribution.
  - (c) Observe  $y_n = f(x_n)$
  - (d)  $n+1$

3. evaluate  $x$  with: the largest  $x$  or the point with the largest posterior mean.

In simpler words, Bayesian Optimization is a way to optimize a prior Gaussian Process that is not able to accurately predict certain values because sudden non-linearities behaviors can be in the model. For example, based on the next image (taken from [62]), the GP was able to represent the linearities of the model, then by using the BO, an acquisition function is found, to accurately represent the model.

Due to the fact that the physics that governs this thesis model includes thermal-plasticity and contact-mechanics, a high non-linear function is expected, therefore a Gaussian Process might not be able to predict it. Therefore a Bayesian Optimization is a must to obtain the desired outputs.

# Full Order Model Digital Twin

## 2.1 Introduction

In this chapter, we focused our attention on the implementation of the FEM model for the Hot Rolling Simulation in the commercial code Abaqus. As it is presented in the next figure, 7 steps are followed to define the FEM model. Initially, in the geometry definition, a 2-D model is constructed instead of a 3-D since the contribution in the third coordinate is small enough to be neglected. Then, the material properties for every part defined in the last modulus are chosen. In this modulus, we can also select the temperature dependence and the parameters that want to be taken into account for the analysis. Now, to define the contact mechanics in the model and the surface radiation, the modulus interactions is carried out.

To define the multi-physics of our model, the modulus with the same name is activated. This part is crucial for the simulation characteristics because decisions as an explicit or implicit scheme, transient, static or dynamic simulation are taken. Then, the discretization of the geometry takes place in the meshing modulus. A posterior modulus, called Boundary conditions, allows us to apply the conditions at the initial conditions, such as temperature and structural constraints. Also, here the symmetry of the problem is defined. Finally, all the setup of the simulation, as parallel simulation, is carried out here.

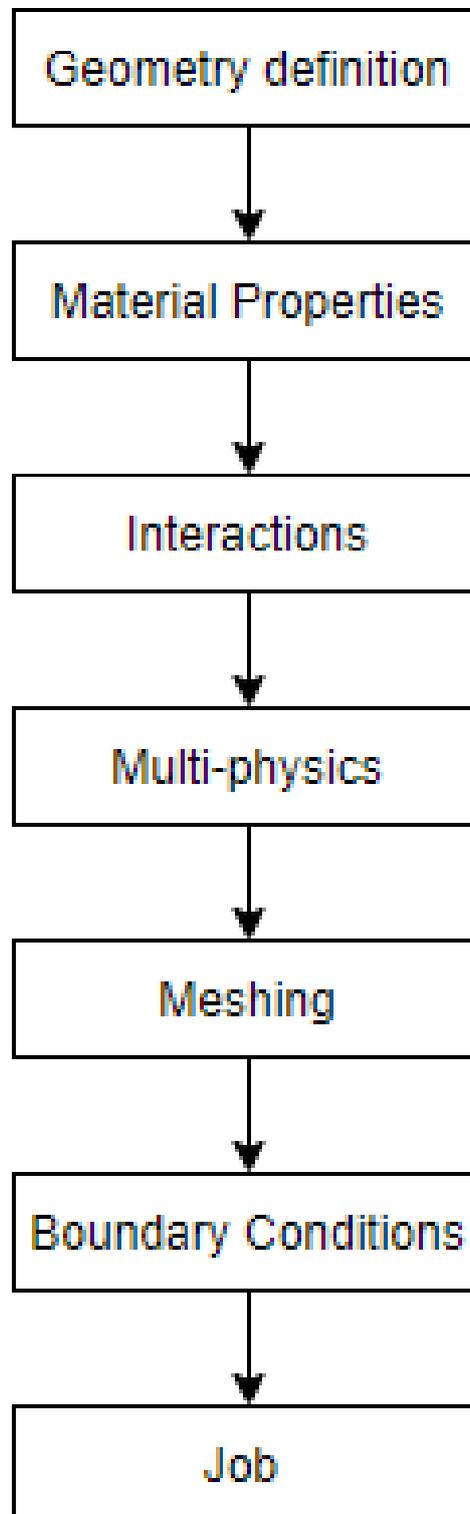
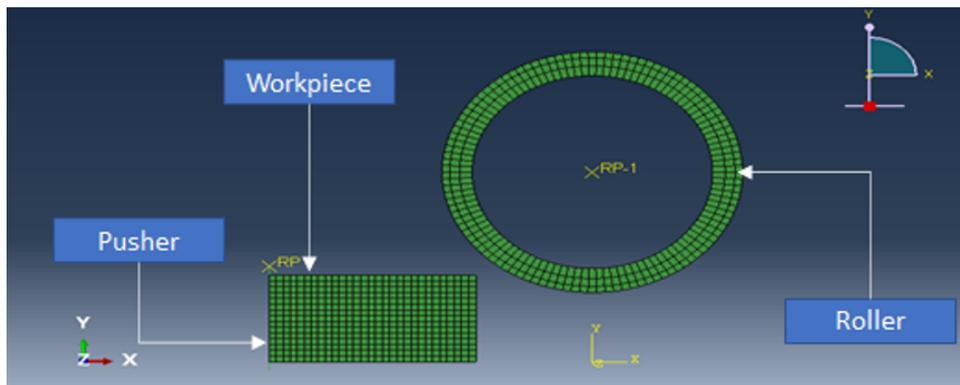


Fig. 2.1.1: Abaqus Workflow

## 2.2 Geometry definition

The first step to run a Job in Abaqus consists of defining the geometry of the hot rolling model. Initially, a FEM representation of this model might be thought of as a 3-D geometry, however, if we set ourselves on the axis representation of image 2.2.1, the Z direction represents a negligible contribution to the global deformation of the work-piece, thus, the geometry can be simplified to 2-D. In the same way, this model is symmetric in the x-axis. With these selections, a considerable amount of computational resources is saved. To summarize, we obtain a 2D simplification of the real model with the upper part only.

application of conventional contact algorithms (Node-To-Surface, Surface-To-Surface) to already meshed (discretized body) and fix all Degree of Freedoms of "discrete rigid" nodes. Additionally, we define the roller as a 2D discrete rigid because in the interaction modulus, when we choose contact-to-contact, the contact algorithm calculates the penetration of the materials taking in account that this body cannot deform.[2]



**Fig. 2.2.1:** Geometry definition

Concerning the measures of the model, the geometry is defined following the information given to us by the metal manufacturing laboratory of IBF Aachen RTWH and it is reported in table 2.2.1, corresponding with the names of image 2.2.1.

	Workpiece	Pusher	Roller
length or external diameter (cm)	170	—	160
thickness or internal diameter (cm)	11.17	11.17	128

**Tab. 2.2.1:** Dimensions

## 2.3 Material Properties

In the hot rolling process a wide range of temperature in a range of 20°C to 1200°C is presented at the work-piece. In the beginning, the work-piece is entered into the machine at the maximum temperature (Approx. 1200°C), then, at the end of the process, it is cooled down by natural convection at room temperature of 20°C. Similarly, as the roller is in contact with the work-piece, it experiments a heat exchange due to conduction and radiation, therefore a dependency of these parameters on both material properties with respect to temperature is necessary to add. To do so, the temperature-dependency flag in Abaqus is activated.

The multi-physics model selected here requires the conditions summarized in table 2.3.1. With respect to the inelastic heat fraction, a conservative default value of 0.9 is selected, but adding to the workpiece modeling not only yield stress and Plastic strain, as in the case of the roller, but also the Plastic Strain rate with temperature dependence. This last feature is added to account strain-rate dependent data.

	<b>Roller</b>	<b>Workpiece</b>
<b>density</b>	with temperature dependence	with temperature dependence
<b>conductivity</b>	with temperature dependence	with temperature dependence
<b>specific heat</b>	with temperature dependence	with temperature dependence
<b>elasticity</b>	Young Modulus, Poisson Modulus	Young Modulus, Poisson Modulus and temperature dependence
<b>inelastic heat fraction</b>	Fraction = 0.9	Fraction = 0.9
<b>plasticity</b>	Yield stress, Plastic strain	Yield stress, Plastic strain, Rate and temperature dependence

**Tab. 2.3.1:** Material Properties

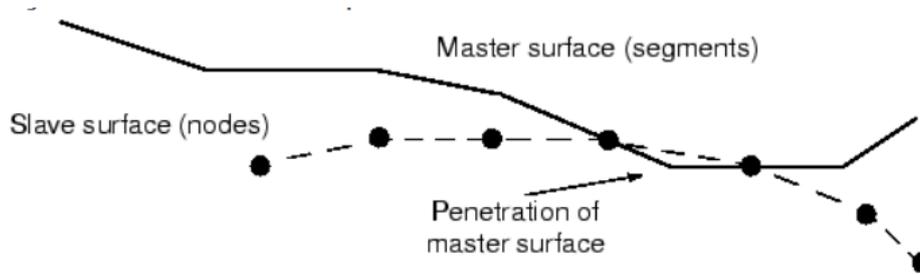
## 2.4 Interactions

In this module, two conditions are activated: Surface to surface contact and surface radiation. In the first case, the contact-mechanics are added to the model between the roller external surface and the entire work-piece in a finite-sliding formulation. For the work-piece the whole piece is included because of the change in the geometry during the rolling process. In the second case, the heat exchange due to the hot temperature of the workpiece is accounted for here.

### 2.4.1 Surface to surface contact

In order to model the contact properties to determine the role of the master surface and of the slave surface, the following flags are selected: Tangential behavior, Normal behavior, Thermal Conductance and Geometric properties. With this modeling, the force between the bodies at the rolling is calculated based on the virtual penetration, then a possible separation between the bodies can be taken into account [2].

For the hot rolling process is evident that the pusher is selected as the master surface, while work-piece is slave surface. Nonetheless, some considerations have to be taken: The slave surface should be the more finely meshed surface; and if the mesh densities are similar, the slave surface should be the surface with the softer underlying material. Moreover, The master surface only can penetrate the slave surface as shown next.



**Fig. 2.4.1:** Master and slave surfaces. Only the master surface is the one that can penetrate the slave surface. Image taken from the Abaqus Manual [2]

In the first case, the friction formulation is simplified in the way of just defining the friction coefficient, of 0.4, between the surface of the roller (external surface) and the upper part of the work-piece. In this way, it is modeling the traction between both surfaces and also it is restraining the slip between surfaces, otherwise, the shear stress will not be produced on the surface, then, the deformation field would not consider this contribution.

In the second case, the normal behavior is defined to select the separation and the way how the penetration is calculated so that the force between both forces is produced. Specifically, the Hard contact option is selected in order to minimize the penetration of the slave surface into the master surface at the constraint locations and does not allow the transfer of tensile stress across the interface, so that if the contact pressure is equal to zero, there is no contact between surfaces and separation between surfaces is produced. On the contrary, a finite value means the existence of contact pressure if the clearance between them reduces to zero.

In the third case, as the name suggests, thermal conductance allows the definition of conduction heat transfer between both surfaces when in contact, as defined in the following equation:

$$Q = k(\theta_A - \theta_B) \quad (2.4.1)$$

Where  $q$  is the heat flux per unit area crossing the interface from point A on one surface to point B on the other,  $\theta_A$  and  $\theta_B$  are the temperatures of the points on the surfaces, and  $k$  is the gap conductance. Point A is a node on the slave surface; and point B is the location on the master surface contacting the slave node or, if the surfaces are not in contact, the location on the master surface with a surface normal that intersects the slave node.

And in the last case, “Geometric properties” is selected in order to specify an out-of-plane surface thickness for every node to a unit.

### 2.4.2 Surface radiation

With this option selected, the heat exchange, due to the surface radiation contribution, is added to the energy equation employing the Stefan-Boltzmann law of radiation, as shown next. Here, based on data given by IBF Aachen RTWH, an emissivity  $e$  equals 0.25 and a room uniform temperature  $\theta_A$  of 25°C is defined. Finally,  $\sigma$  corresponds to the Boltzmann constant, while  $T$  is the surface temperature at each time step.

$$Q = e\sigma A (T^4 - T_A^4) \quad (2.4.2)$$

## 2.5 Multi-physics

In the first chapter, we explained that the hot rolling process involves the study of three big multi-physical models: Thermo-plasticity, which is involved in the work-piece when the roller(that cannot deform) is pushing it, triggering a plastic deformation after surpassing the elastic zone. contact-mechanics between the work-piece and the roller, because is due to the friction between both materials that this process can be carried out, and large-deformations (in the work-piece only), so that, a high-nonlinear model.

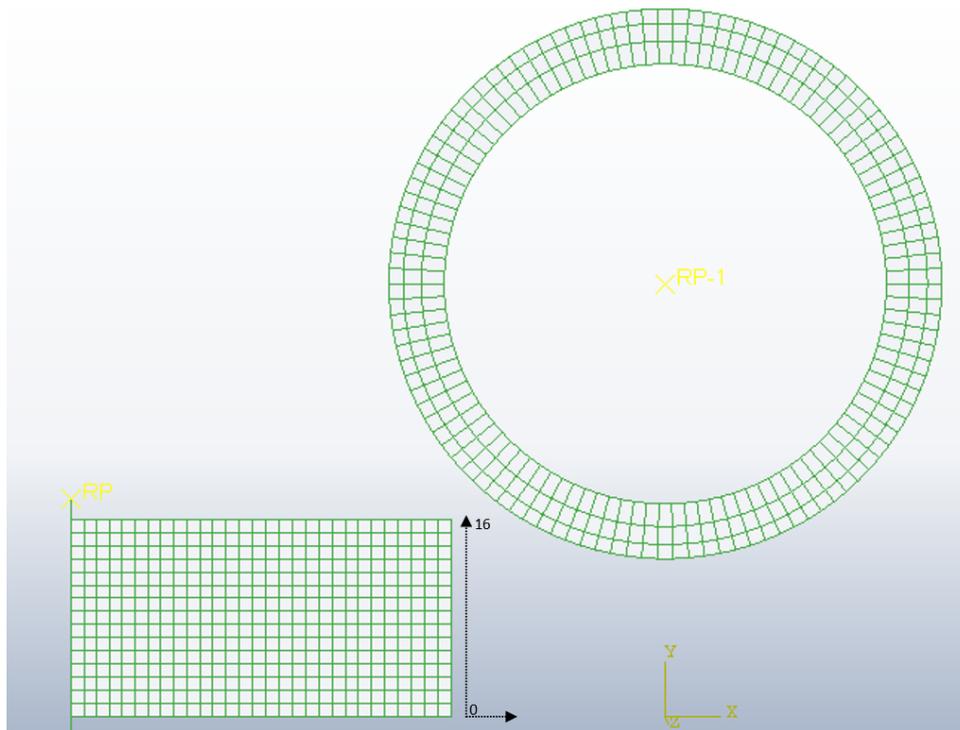
In Abaqus to setup the multi-physics, the panel-option called Step-manager is hit. For our thesis purposes, two approaches are available. The first one is the coupled temperature-displacement step, and the other is dynamic temperature-displacement. The former one is generally more costly than the static calculations and should be implemented when both large deformations and convergence problems are presented. For that reason, the coupled temperature-displacement step. Moreover, we selected a transient response, in a total analysis time of 16.5161 seconds. This last value is selected because of a previous analysis performed by our colleagues in IBF RWTH. Then, to address the non-linear effects of large deformations and displacements, the flag NLgeom is activated. Some other options that are out of the scope of this thesis are selected regarding the discretization scheme to solve the governing equations.

## 2.6 Meshing

Since the mesh set-up was already given by the IBF Aachen RTWH (already tested and found to be in accordance with the experimental tests), some general comments can be done about the mesh:

1. A quadratic, uniform, non-refined mesh has been set up for the work-piece as the main deformation is in the Y direction

2. A quadratic, uniform, non-refined mesh in the proximity of the outer diameter and adapted to the round shape of the roller has been set up.
3. Additionally, sixteen reference nodes are set up in each of the sixteen rows that is divided the work-piece, with the purpose of obtaining the graph displacement vs time.



**Fig. 2.6.1:** Mesh

## 2.7 Boundary Conditions

In accordance with the geometrical shape of the real problem, its definition in the FEM (Finite Element Method) could be thought as shown in image 1.1.1 (state of art section), however, it can be noted that the problem is symmetrical in the X-direction. This observation and the other ones, that are based on the convention defined in image 1.1.1, are shown next.

At time  $t=0$ , when the work-piece is lead towards the roller, the boundary conditions are:

$$Y = 0, X \in [0, L] : U_2 = U_{R1} = U_{R3} = 0 \quad (2.7.1)$$

$$Y = h, X = 0 : V_1 = V_2 = V_{R_3} = 0 \quad (2.7.2)$$

$$Y = L_r, X = D_r : V_1 = V_2 = V_{R_3} = 0 \quad (2.7.3)$$

At time  $t=t.\text{step}$ , when the hot rolling COUPLED TEMP-DISPLACEMENT simulation takes place, the boundary conditions are:

$$Y = 0, X \in [0, L] : U_2 = U_{R_1} = U_{R_3} = 0 \quad (2.7.4)$$

$$Y = L_r, X = D_r : V_{R_3} = 3.125 \quad (2.7.5)$$

$$Y = L_r, X = D_r : V_1 = V_2 = 0 \quad (2.7.6)$$

$$Y = h, X = 0 : V_1 = 200, V_2 = V_{R_3} = 0 \quad (2.7.7)$$

Where the equations (2.6.1) and (2.6.4) indicate the symmetry of the movement in the X-axis so that any rotation and translation in Y are restricted at the origin across the whole X-axis. The equation (2.6.2) and (2.6.3) sets an initial in still conditions for the pusher and the roller, while the equations (2.6.5) and (2.6.7) impose an initial rotation of 3.125 radians per second and a merely horizontal velocity of 200 mm/s to the pusher. Additionally, to restrict the translation in the roller, equation (2.6.6) is imposed.

Finally, for the roller and the work-piece an initial temperature of 25°C and 1200°C, respectively, are set up.

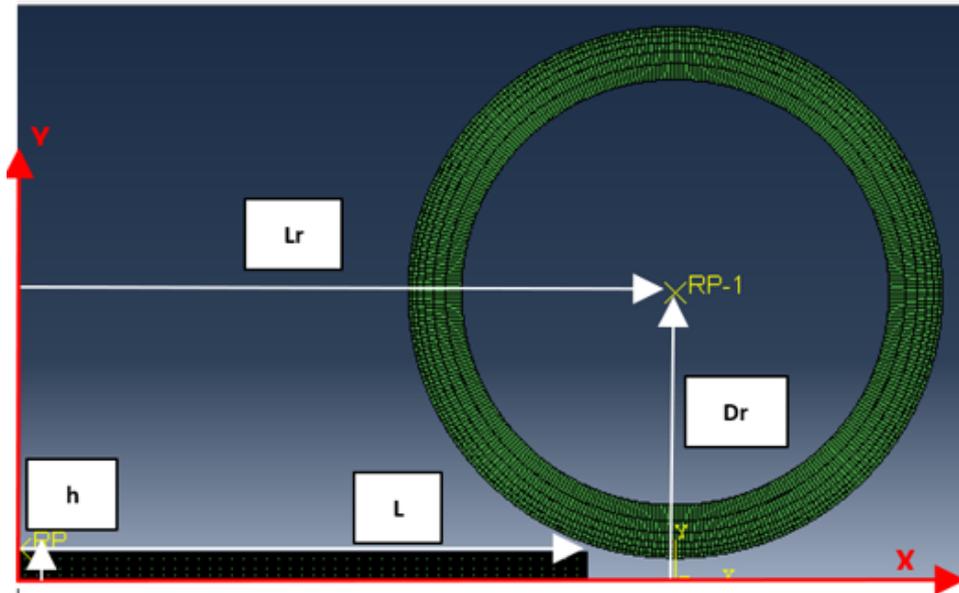


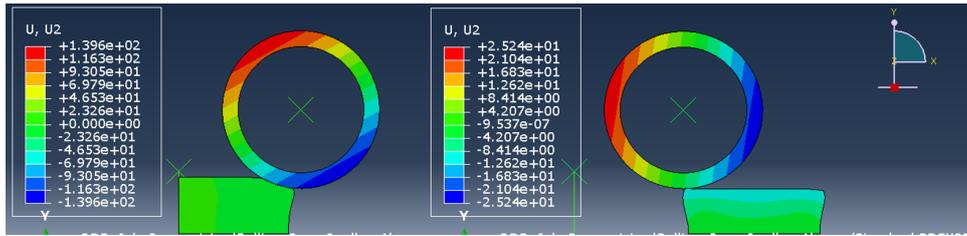
Fig. 2.7.1: Geometry Reference

## 2.8 Results

In ABAQUS the displacement can be obtained for every axis for every component ( $U_1, U_2, U_3$ ), or in the full shape  $U$ , nonetheless, concerning the X-direction ( $U_1$ ) it mainly corresponds to the translation displacement. Furthermore, since the geometry was defined in 2D, the component in the Z direction ( $U_1$ ) is not taken into account. As a result, only the vertical displacement ( $U_2$ ) is shown here (for the next chapters the displacement will consider this component only).

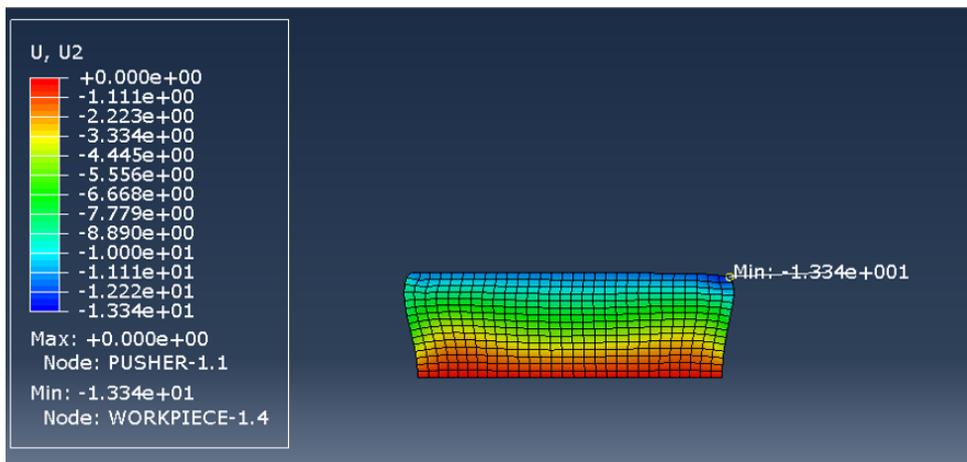
With respect to the roller, ideally it is rotating symmetrically, therefore, an also symmetrically displacement due to the centrifugal acceleration is expected to appear as shown in image 2.7.1, where the maximum values are obtained in the opposites for every time step. In the same way, the restriction, that the work-piece is applying to the roller, contributes to this result.

On the other hand, image 2.8.1 shows the displacement when the rolling process is starting and ending. At the beginning, the work-piece is being deformed as the local part that is in touch with the the roller takes the same shape as it rotates (counter clock-wise direction on the right hand convention), while the other extreme does not do it, as expected because the force applied by the roller to the work-piece in the left section of the work-piece goes in the same direction of the movement. On the other hand, near to the end of the rolling procedure, the right extreme is being deformed.

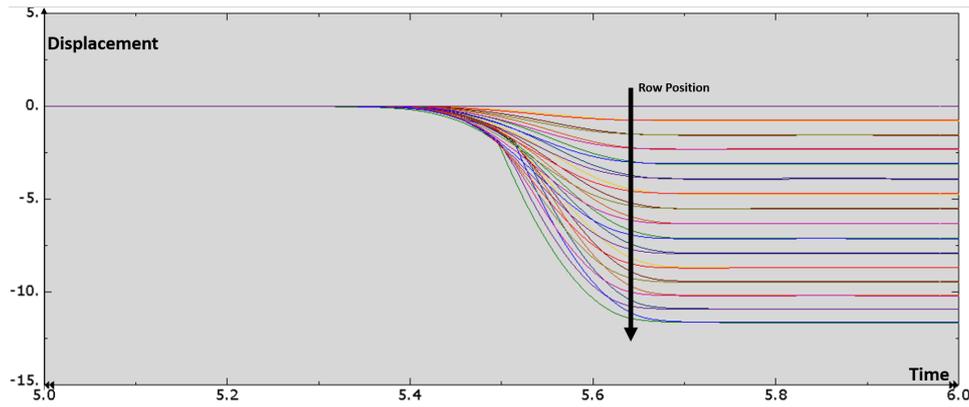


**Fig. 2.8.1:** Deformed body

Image 2.8.2 corresponds to the work-piece at the last time step. when the rolling process has already finished. The results show a distribution of the deformation in the surface. A very similar one is expected to find with the implementation of the non-intrusive reduced order model.



**Fig. 2.8.2:** Displacement results



**Fig. 2.8.3:** Displacement results plot

The reader may note the sixteen rows the work-piece was divided into and that corresponds to the 16 functions plotted (from top to bottom) in fig 2.8.3.

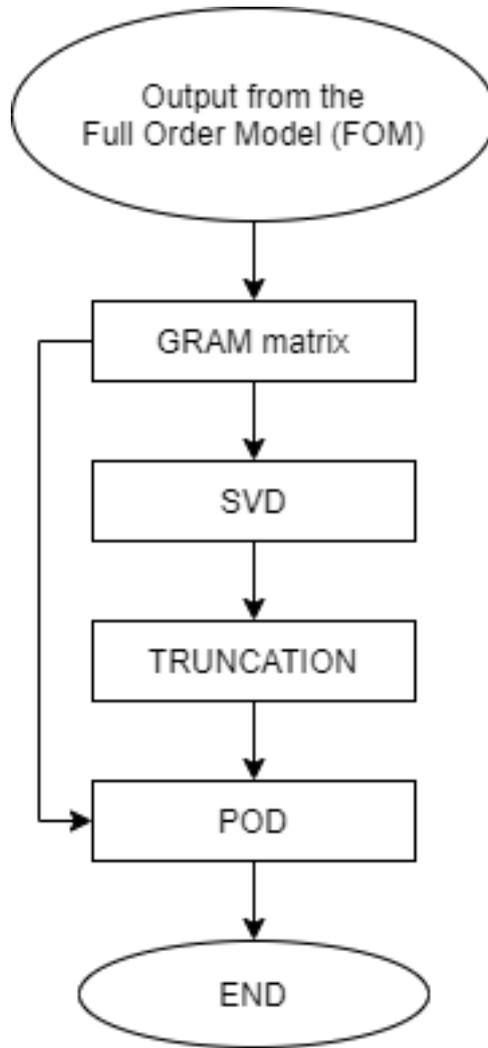
Furthermore, as can be seen in both images, the last row of the work-piece does not deform at all, and this happens because, if the other 15 functions are carefully observed, the slope of the function between 5.4 seconds and 5.6 seconds, decreases as further is located the row, then the last row arrangement of atoms keeps in the same orientation as at the beginning of the process, therefore the natural equilibrium due to the elasticity of the material is maintained.

In the same way, the other fifteen rows are under plastic-deformation, since they do not come back to the original position, and this means that the inter-atomic distance, that was keeping the equilibrium between the repulsive forces and the attractive forces, overpasses the 5 percentage allowed to make the material be in the elastic zone.

From the displacement obtained here, the first classification has to be done because the total analysis lasts for approximately 16 seconds, however only from the second 5 to 7 the thermo-plastic process is carried out, therefore only these values of the vertical displacements on these range of times are taken to the next step.

# Proper Orthogonal Decomposition

The workflow that summarizes the approach implemented on Python and that corresponds to this chapter is presented hereafter.



**Fig. 3.0.1:** Chapter Workflow: The displacement that comes from the FEM model is taken as the input to construct the ROM by the next steps: Gram Matrix construction, SVD computation, SVD truncation, and POD construction.

In the last chapter, the displacement for the Full Order Model (FOM) was computed by solving directly the governing equations. This output is used to build up the so-called snapshot matrix, where every row contains the displacement of all nodes at each time step.

Then, this output, defined by the variable  $S$  to avoid confusion, is used to calculate the gram matrix to obtain the output  $G$ . Next, the Single Value Decomposition (SVD) method is performed to compute  $\Sigma$ , the diagonal matrix with the singular values,  $U$ ,

the left singular vector, and  $V$ , the right singular vectors. These last variables are of vital importance to the main core of this chapter because by implementing an energetic method, those variables are truncated, based on a cutoff value of 0.999 (commonly used for this method). From there, 6 bases are found to represent the Full Order Model in the Reduced Basis approach, represented by  $\hat{\Sigma}$ ,  $\hat{U}$  and  $\hat{V}$ , to construct the Reduced Order Model (ROM). After that, by taking the gram matrix in the POD, the POD basis is computed.

$$S_N = \begin{bmatrix} U(\mu_0, t_0) & U(\mu_1, t_0) & \dots & U(\mu_n, t_0) \\ U(\mu_0, t_1) & U(\mu_1, t_1) & \dots & U(\mu_n, t_1) \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ U(\mu_0, t_m) & U(\mu_1, t_m) & \dots & U(\mu_n, t_m) \end{bmatrix} \quad (3.0.1)$$

For the specific case of this thesis, the size of the snapshot matrix is 3211 rows and 32 columns. The odd number 3211 comes up because the FEM analysis for every parameter takes a different time step to obtain a stable convergence. It is worth noticing that this type of method is not excluded only to a low number of nodes, but it has potential scalability.

### 3.0.1 Gram Matrix

As referred in [15], good practice in this approach is to study the linear independence of the snapshots  $U(\mu_0) U(\mu_1) \dots U(\mu_m)$ , because, this produces a large condition number of the associated solution matrix. To do so, the Gramian or Gram matrix is assembled by the Euclidean inner product, taking as the input the snapshot matrix. Then, an orthonormalization of the snapshots is computed, based on the next equation. It is worth noticing that  $\bar{S}$  refers to the complex conjugation of the snapshot matrix, however, the displacement is purely real.

$$G = \bar{S}^T S \quad (3.0.2)$$

### 3.0.2 SVD

After finding the Gram Matrix  $G$ , the SVD algorithm is applied (SVD was used instead of the SVD randomized because the size of the problem does not make it a good choice, however, as explained in the state of art chapter, SVD becomes unfeasible for very big problems) by the library ezyrb [63]. Moreover, this offline part is the one that takes most of the time, therefore PETSc and SLEPc libraries were used to run it in parallel.

In general what is done by using the SVD algorithm is to find the following relation:

$$G = U\Sigma V^T \quad (3.0.3)$$

However, to construct the POD it is necessary to extract only  $U$  and  $\Sigma$ .

### 3.1 Truncation

To continue with the development of the POD, an energetic approach is implemented. Where a ranking of the most energetic values based on each of the single values of the diagonal matrix  $\Sigma$ . So that, if the algorithm hereafter is applied, only til the criteria (commonly used for this purposes of 0.99) is reached, the loop finishes.

---

#### Algorithm 1: Energetic Rank Truncation

---

Recalling  $\Sigma$  and  $U$  from the SVD;

initialization  $N=0$ ;

**while**  $criteria \leq 0.999$  **do**

$criteria = \frac{\sum_{n=1}^N \Sigma}{\sum_{n=1}^{\infty} \Sigma};$   
 $N=N+1;$

**end**

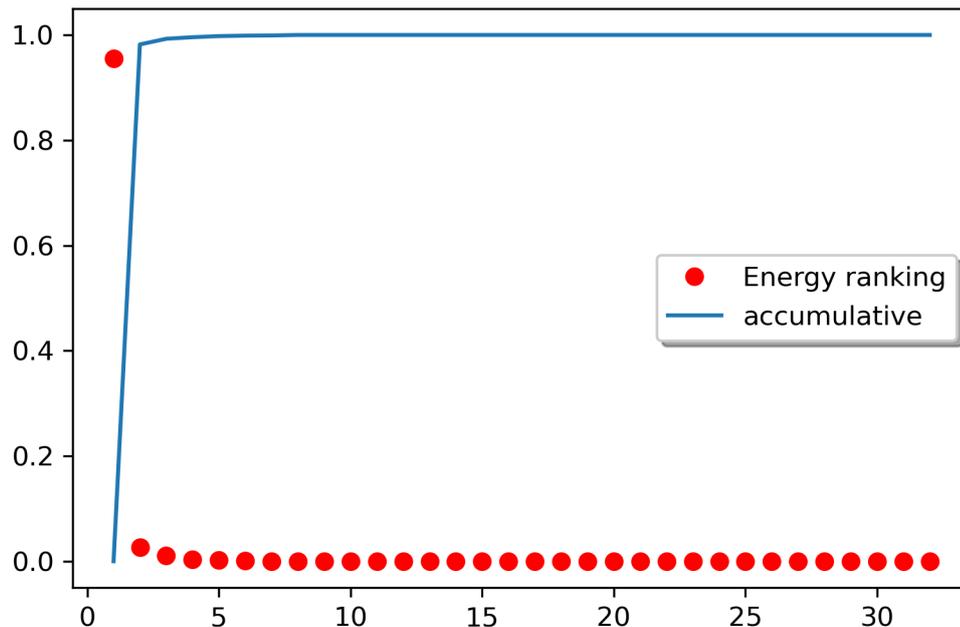
$\hat{\Sigma} = \Sigma[0 : N];$

$\hat{U} = U[0 : N];$

*Return*  $\hat{\Sigma}, \hat{U}$

---

From this small snippet of the whole code, a graphical representation of the basis energy ranking can be obtained. The red line represents the energy ranking of the normalized modes, and the continuous blue line, the accumulative energy.



**Fig. 3.1.1:** Energy ranking for the modes obtained from the Gram Matrix. The modes were normalized with respect to the total sum of the basis for better visualization

Based on it, our model shows us that just one basis represents most of the dynamics of the model, with a value of 0.98, however, to reach the constrained value of 0.999, 6 bases are needed. Moreover, from the third basis, the accumulative function is almost constant, then a low contribution from those modes is expected.

Then the truncated basis matrix can be constructed as represented in the last part of the snippet, by taking the modes from the highest to 6<sup>th</sup>.

## 3.2 POD

To construct the POD basis, the gram matrix, calculated on subsection 1 is recalled to then be multiplied by the reduced modes and obtain the POD basis. To do so, the following snippet shows the step by step procedure.

---

### Algorithm 2: POD basis construction

---

Recalling  $G$  (Gram matrix),  $\hat{\Sigma}$ ,  $\hat{U}$  and  $N$  ;

$V = G\hat{U}$ ;

**for**  $i = 1 : N$  **do**

$V[:, i] = \frac{V[:, i]}{\hat{\Sigma}(i)}$ ;

**end**

*return*  $V$ ;

---

As a result, the POD basis is found, represented by  $V$ . The last step we have to follow is the projection of the POD basis into the snapshot to reconstruct the POD. But before, to clarify what the code does, let exemplify a general dynamic problem that is represented by the stiffness matrix  $k$  and the displacement vector  $S$ :

$$\begin{pmatrix} F_1 \\ F_2 \\ F_3 \\ \cdot \\ \cdot \\ \cdot \\ F_N \end{pmatrix} = \begin{bmatrix} k_{11} & k_{12} & k_{13} & k_{14} \\ k_{21} & k_{22} & k_{23} & k_{24} \\ k_{31} & k_{32} & k_{33} & k_{34} \\ \dots & & & \\ \dots & & & \\ \dots & & & \\ k_{N1} & k_{N2} & k_{N3} & k_{N4} \end{bmatrix} \begin{pmatrix} S_1 \\ S_2 \\ S_3 \\ \cdot \\ \cdot \\ \cdot \\ S_N \end{pmatrix} \quad (3.2.1)$$

or in matrix form,

$$\{F\} = [K]\{S\} \quad (3.2.2)$$

The first step is the ROM construction by projecting the reduced basis in the stiffness matrix

$$[K_{ROM}] = [V]^T [K] [V] \quad (3.2.3)$$

Then, the same procedure is done with the force vector

$$[F_{ROM}] = [V]^T [F] \quad (3.2.4)$$

As a result, the displacement,  $S_\alpha$ , for the reduced order model can be computed by solving the linear system.

$$[F_{ROM}] = [K_{ROM}]\{S_\alpha\} \quad (3.2.5)$$

However, to find out the displacement in the high dimensional space, the displacement has to be projected back, as shown next.

$$\{S_{ROM}\} = [V]\{S_\alpha\} \quad (3.2.6)$$

Finally, the reduced coordinates in terms of the interpolated POD  $\alpha$  are calculated by also solving the following linear system:

$$[V]^T [V]\{\alpha\} = [V]^T [S] \quad (3.2.7)$$

Then, the error between the POD interpolation and the POD is computed by the norm between them, divided to the norm of the POD solution.

$$error_{\alpha} = \frac{|(\alpha - S_{\alpha})|}{|S_{\alpha}|} \quad (3.2.8)$$

Afterward, the error between the ROM and the FOM for the generalized problem is computed by:

$$error = \frac{|(\alpha - S)|}{|S|} \quad (3.2.9)$$

Now, based on a similar procedure but tailored for our system, the snippet shown in algorithm 3 is implemented.

Another interesting option to reconstruct the POD coefficient (actually, a better solution for this thesis), is the one proposed by [55], the snippet is shown as algorithm 4. It is important to mention that the basis  $V$  was normalized, due to the fact that, as studied in the reference [64], it helps to identify the subspace where the interpolation is stable.

**Algorithm 3:** Traditional POD reconstruction

---

```

Recalling V, S ;
Matrix = V.dot(V);
Matrix.convert("dense");
matrix_np = Matrix.getDenseArray();
reconstructSol = basis[0].duplicate();
error_norm_list = [];
error_normInf_list = [];
coeffmatrix = numpy.zeros([len(basis), len(S)]);
for i = 1 : len(S) do
    snapvec = vec(S[i]);
    snapProjvec = V.dotVec(snapvec);
    snapProjnp = snapProjvec.getArray();
    coeff = solve(matrix_np, snapProjnp);
    if verify == "True" then
        reconstructSol.set(0.0);
        reconstructSol.multiply(coeff, basis);
        error = snapvec - reconstructSol;
        error_normInf =  $\frac{\text{error.norm}(PETSc.NormType.NORM\_INFINITY)}{\text{snapvec.norm}(PETSc.NormType.NORM\_INFINITY)}$ ;
        error_norm =  $\frac{\text{error.norm}(PETSc.NormType.NORM\_2)}{\text{snapvec.norm}(PETSc.NormType.NORM\_2)}$ ;
    end
end
end

```

---

**Algorithm 4:** HG POD reconstruction

---

```

Recalling V,S ;
num_basis = V.getActiveColumns()[1];
coeffmatrix_HG = np.zeros([num_basis, len(S)]);
for i = 1 : len(S) do
    u = S[i];
    coordinatesPOD = V.dotVec(vec(u));
    coeffmatrix_HG[:, item] = coordinatesPOD.getArray();
    reconstructSol = vec(u).duplicate();
    reconstructSol.set(0.);
    V.multVec(1., 0., reconstructSol, coordinatesPOD.getArray());
    error = snapvec - reconstructSol;
    error_normInf =  $\frac{\text{error.norm}(PETSc.NormType.NORM\_INFINITY)}{\text{snapvec.norm}(PETSc.NormType.NORM\_INFINITY)}$ ;
    error_norm =  $\frac{\text{error.norm}(PETSc.NormType.NORM\_2)}{\text{snapvec.norm}(PETSc.NormType.NORM\_2)}$ ;
end
end

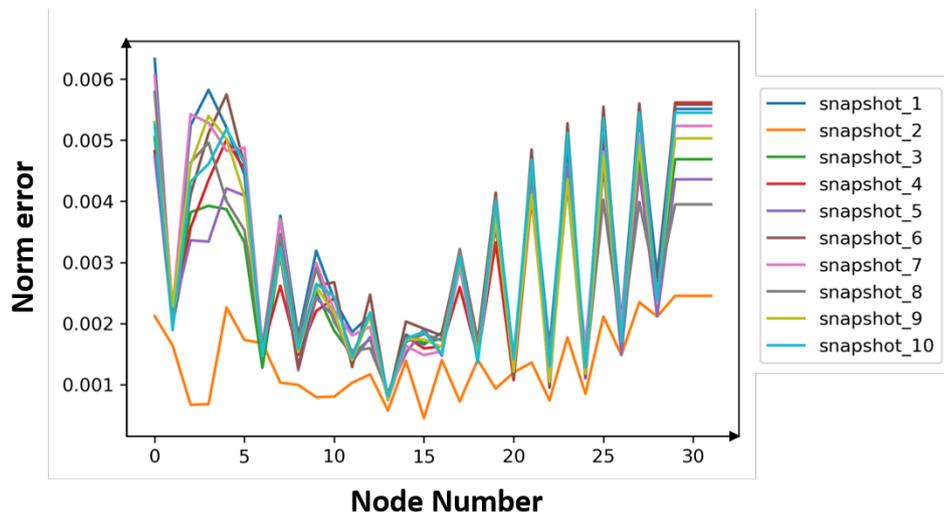
```

---

### 3.3 POD Results

Based on the procedure explained before, the reconstructed displacement is compared with respect to the one obtained by Abaqus, i.e Full Order Model, which is shown hereafter. The maximum error is 0.6%, with a minimum error of 0.01%, stressing out the fact that these results come from a reducer order model of 6 bases. Moreover, the highest accuracy is obtained in the core of the rolling process, where most of the non-linearities take place,

With these results we can confirm, as many other authors have published, that with a non-intrusive POD it is possible to obtain very accurate computations for numerous types of problems, taking the most important representation of the dynamics, with a considerable less amount of computational time.



**Fig. 3.3.1:** Computation of the error for every snapshot with respect to the recovered by using the POD

# Gaussian Process Prediction

In this chapter, the route that will be followed is based on the next image. First of all, the implementation of the Gaussian Process requires input from the POD, however, as described by many authors, the results that come from this implementation mainly depend on the input setting. For that reason, two different inputs were taken: the reduced coordinates and the projected snapshots. Then, the tuning of the kernel in the Gaussian Process is presented.

From the two results obtained in the Gaussian process, the first one, which we cataloged as the non-successful one, stops here, since no accurate results were found. While for the second case, the Bayesian optimization is implemented and the error between the FOM and our model is shown, followed by a discussion about them.

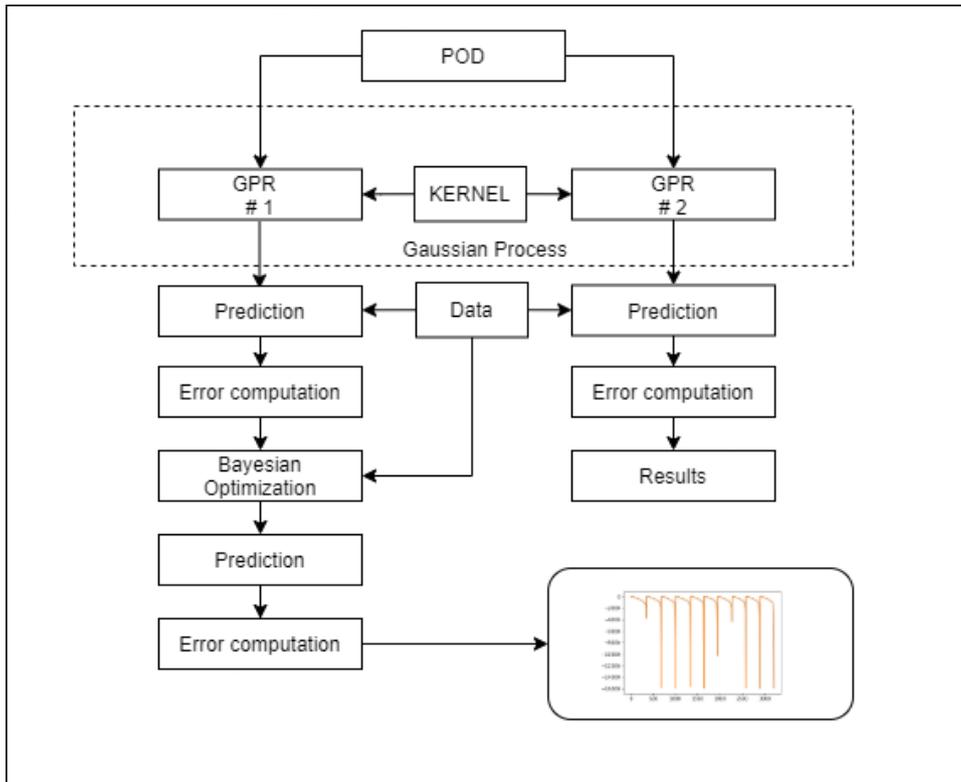


Fig. 4.0.1: Chapter Work Flow

## 4.1 Gaussian Process

To build up the Gaussian process regression, we used the Gaussian Process framework GPy available in python and created by the Sheffield machine learning group. However, two different inputs are introduced with the aim of obtaining the best prediction. This is done because some authors, as explained in the state of the art section, refer that the key

for a successfully Gaussian Process implementation is the correct setting of input-output. Thus, POD reduced coordinates and Projected Snapshots are selected.

The first step is to import the function

$$GPy.models.GPRegression(X\_data, Y\_data, kernel, normalizer = True$$

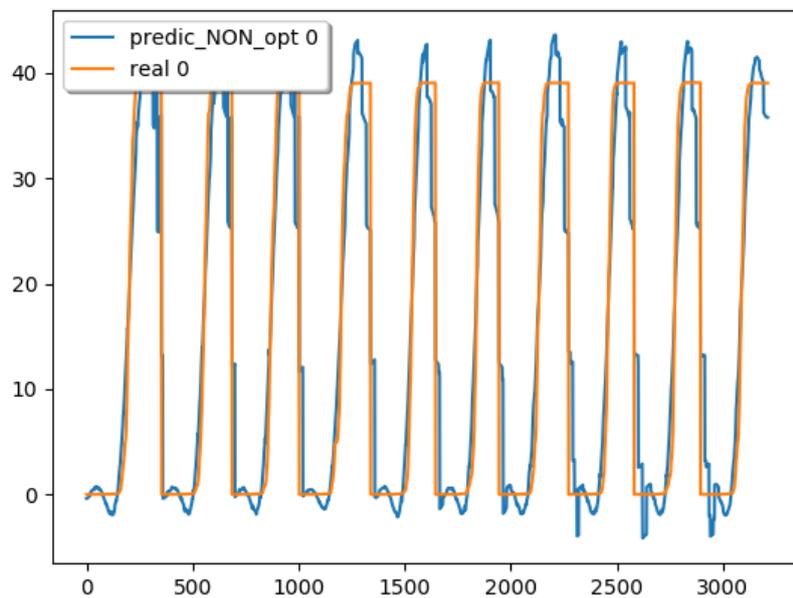
from the GPpy.  $X\_data$  refers to the input data that the GP needs to correlate with the output  $Y\_data$ . In our case, the input is the set of elastic and density parameters used in the FEM model, while the output for the GP1 is the POD reduced coordinates, and for GP2 the projected snapshots.

Then, to define the Kernel or regression function for the Gaussian Process Regression, we introduce a new function:  $GPy.kern.RBF(input\_dim = self.X.shape[1], ARD = True)$ . In this function, we enrich the Radial Basis Function (RBF) with an automatic relevance determination (ARD) structure, because high relevant input features can be effectively extracted to improve prediction accuracy and robustness. However, instead of using an integrand for our Kernel, we can replace it with a vanilla bayesian that permits to be surrogated by a GP function to then used the quadrature Bayesian Optimization. To do this a built-in function of the emukit library is needed:

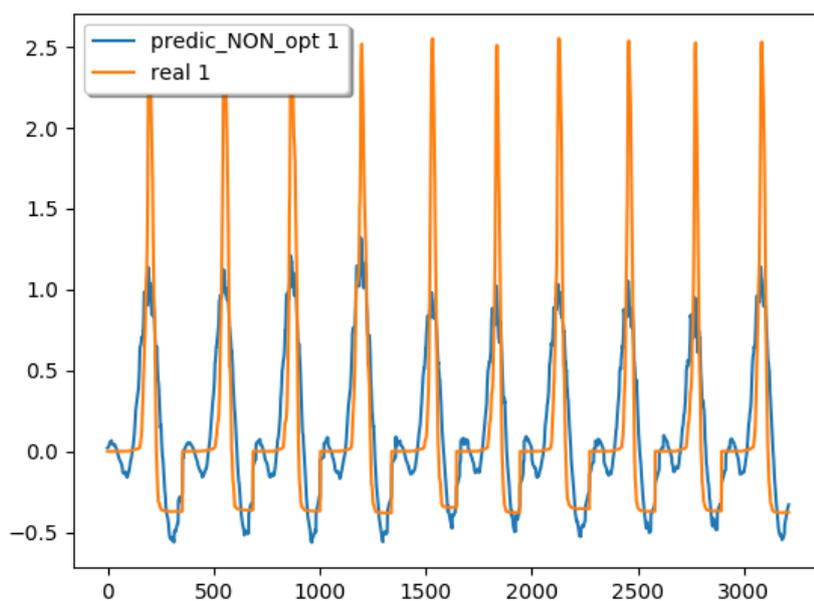
$$VanillaBayesianQuadrature(base\_gp, X\_sample, Y\_sample)$$

, where  $base\_gp$  is the base for the Gaussian Process in GPpy to obtain a Warped Gaussian process (WGP) regression i.e an extension of the standard GP that transforms the original non-Gaussian target series to a latent target series by estimating a warping function, such that the transformed series is well-modeled by a standard GP in latent space. In this way, WGP handles the non-Gaussian process and non-Gaussian noise. [65]

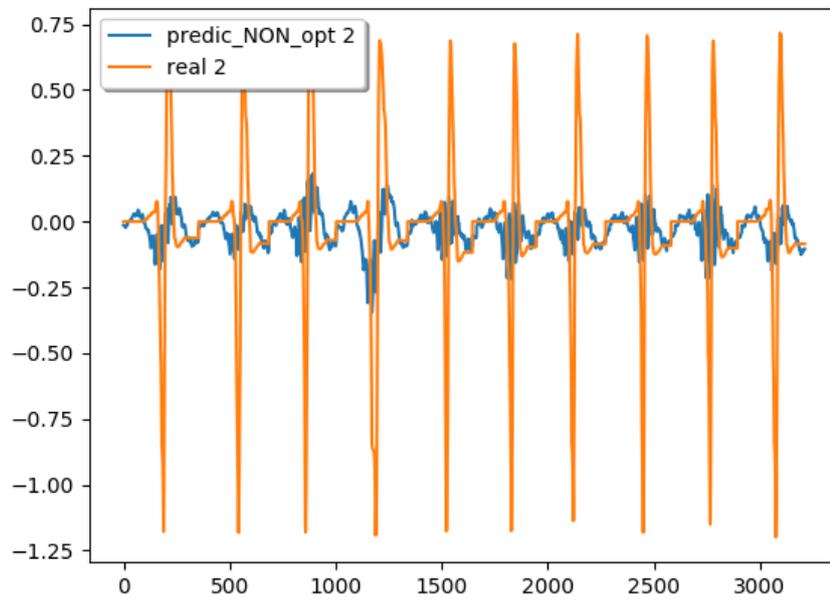
After running the code, what is obtained is a Gaussian process surrogate model that is placed upon the integrand which is then integrated directly. In other words, the function we want to integrate is substituted/surrogated by our Gaussian process. As a result, an input-output representation function is not given anymore for a multi-physics model, but instead for a Gaussian process. Unlike the second GP, for the first GP non-promising results are found, and some of them are shown hereafter. For that reason, we decided to continue only with the accurate one.



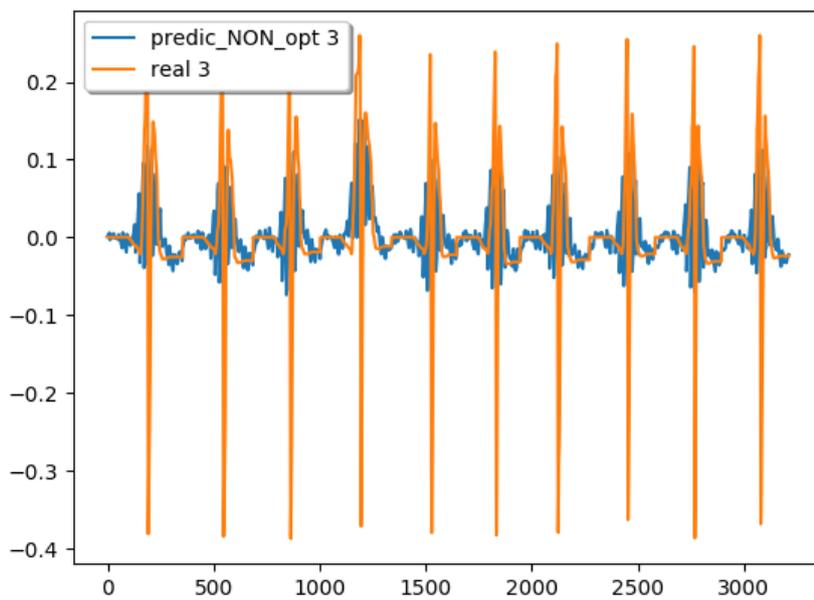
**Fig. 4.1.1:** Results for the basis 1, a quite good approximation solution is found with the Gaussian Process.



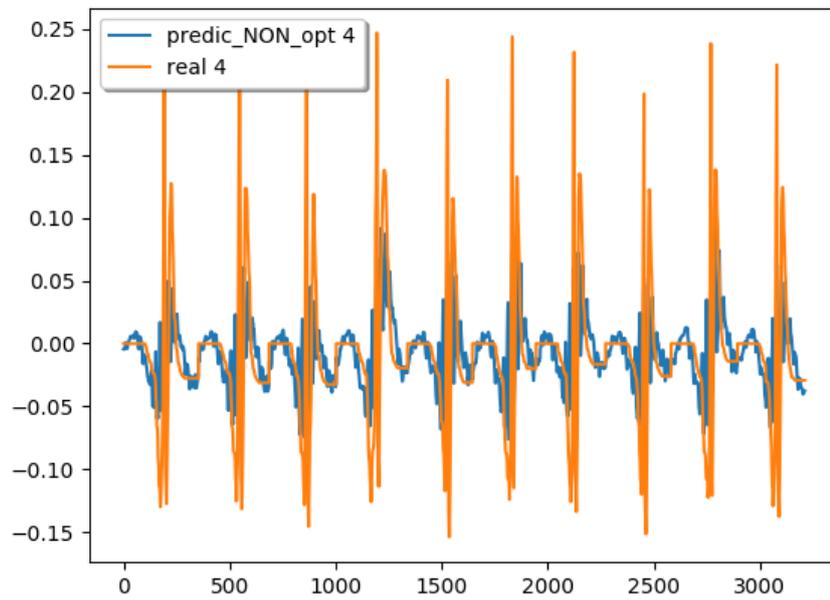
**Fig. 4.1.2:** Results for the basis 2: a similar but scaled behavior is obtained.



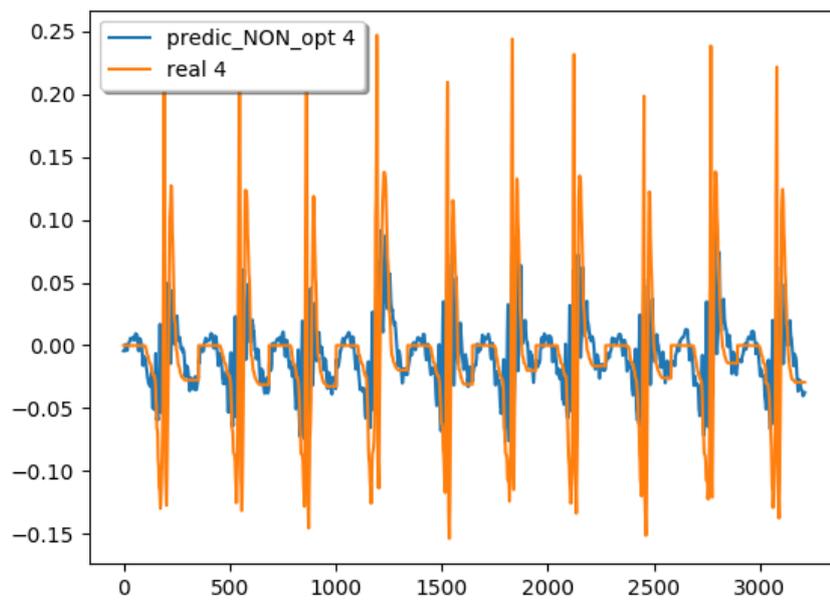
**Fig. 4.1.3:** Results for the basis 3



**Fig. 4.1.4:** Results for the basis 4



**Fig. 4.1.5:** Results for the basis 5



**Fig. 4.1.6:** Results for the basis 6

In the image 4.1.1 a very good approximation is computed by using the Gaussian Process. A better prediction might be carried out by the implementation of the Bayesian Optimization since global locals are detected by using it. As a result, the true maximum, which is not reached with the Gaussian process, might be computed. Nonetheless, the other 5 images show a very inaccurate prediction. These might be produced because non-correlated input-output is found by the Gaussian Process, then a good prediction is not possible with this setting. Moreover, because of the quite good results with GP2, as well as the easiness of recovering the displacement by projecting back the projected snapshots, we focus all our efforts on this one.

## 4.2 Bayesian Optimization

There are some tools to implement a Bayesian Optimization in a script, for our case, the library `ezyrb` is used. From there some built-in functions are imported: `VanillaBayesianQuadrature`, `RBFGPpy`, `QuadratureRBFLebesgueMeasure`, and `BaseGaussianProcessGPpy`. In the same way, from the library `Scipy`, the function `minimize` is imported.

After importing the external libraries, we start the implementation of the script by defining the class `GPRBayesOptim` that takes as input the Gaussian Process, defined in the last chapter. Before continuing with this implementation, some comments have to be done regarding the usage of a class in Python.

Python uses a class intending to provide all the standard features of Object-Oriented Programming. This means that multiple base classes are allowed with the option of overriding any methods of its base class or classes. As in C++, the method function is declared with an explicit first argument representing the object, which is provided implicitly by the call, however, built-in types can be used as base classes for extension by the user in Python only.

Said so, with the class defined before, a new local world is created, composed of three functions:

1. `__init__(self)` : This function has been created with the aim of initializing the code, utilizing the input "self". This variable is widely used in the programming world to refer to the same variable. As a result, 4 variables are defined with None elements: `X_sample`, `Y_sample`, `model` and `model_VBQ`.
2. `def fit(self, points, values, gpy_model, optimization_restart = 20)` : This function can take up to 5 inputs, but only 4 are necessary: Again the self function, but this time relocate the variables `X_sample` and `Y_sample`. Point and value parameter, where the coordinates and values of the observed points are defined, respectively. Then, `gpy_model` is taken as the output of the GPpy that was explained

in the last chapter. The 5<sup>th</sup> input is predefined when no input is given. It is an important parameter for the Bayesian Optimization because it refers to the number of restarts for the optimization to avoid bad local minima, starting from a sample of the probability of a set of distributions  $\theta$  if the model  $M$  happens. It is important to remember that the Gaussian process surrogate model is placed upon the integrand which is then integrated directly. In other words, the function that we want to integrate is substituted/surrogated by a Gaussian process.

Inside this function, there is a "sub-function" in charge of performing the optimization: `model.optimize_restarts(optimization_restart, verbose = False)` Here, the integrand of the function  $Y$  is represented by a GP process, then, employing a quadrature bayesian optimization, an optimization is reached with an optimization problem. Moreover, the kern used is a quadratic radial basis function, however, instead of using an integrand, it is replaced by a vanilla bayesian that permits to be surrogated by a GP function, to then used the quadrature bayesian optimization.

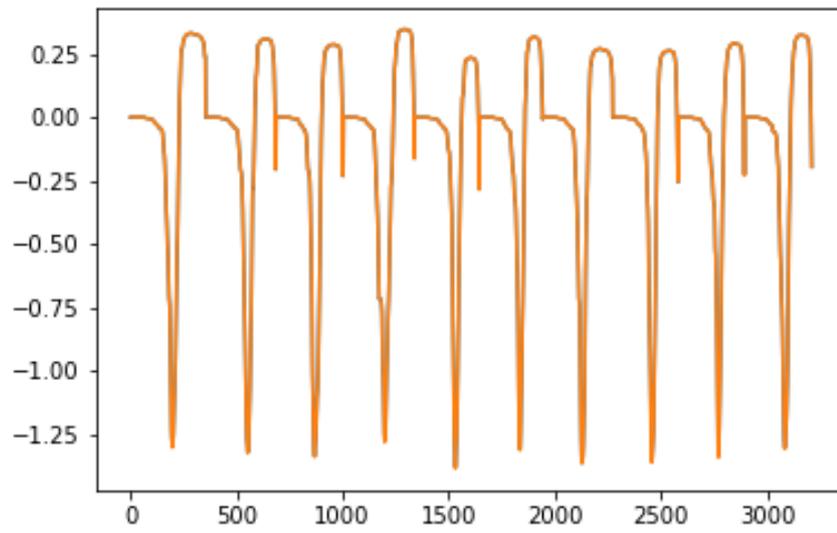
3. `def predict(self, new_points, return_variance = False, return_gradient = False)` :  
In this function the prediction part is performed. To do so, four inputs are necessary; `self` was already explained: `new_points` is a variable that refers to the new parameters of the density and elasticity sets that want to be predicted in the optimized GP-function. If the variance obtained in this model is required, then the flag from `False` to `True` has to change, as well as if the model gradients of mean and variance at a given point are of interest.

### 4.3 Results

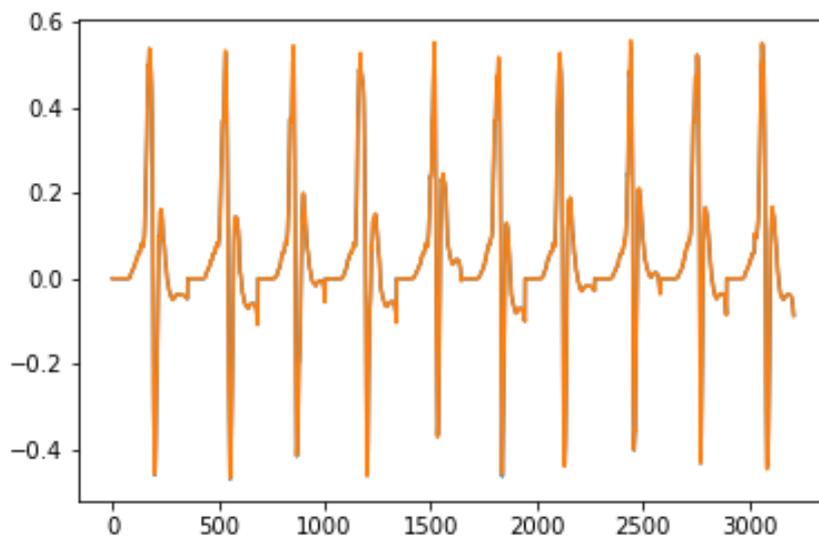
Based on the procedure shown before, we obtained a very accurate solution for every node of the FEM model. Also, with respect to the time simulation, after using the time function in Python, we obtain that our online model is 1000 times faster than the online model of the Full Order Model.

Moreover, if the average error is computed for every node, as the difference between the full order solution and the predictive solution, an error of less than 1% is found.





**Fig. 4.3.3:** Results node 3



**Fig. 4.3.4:** Results node 4

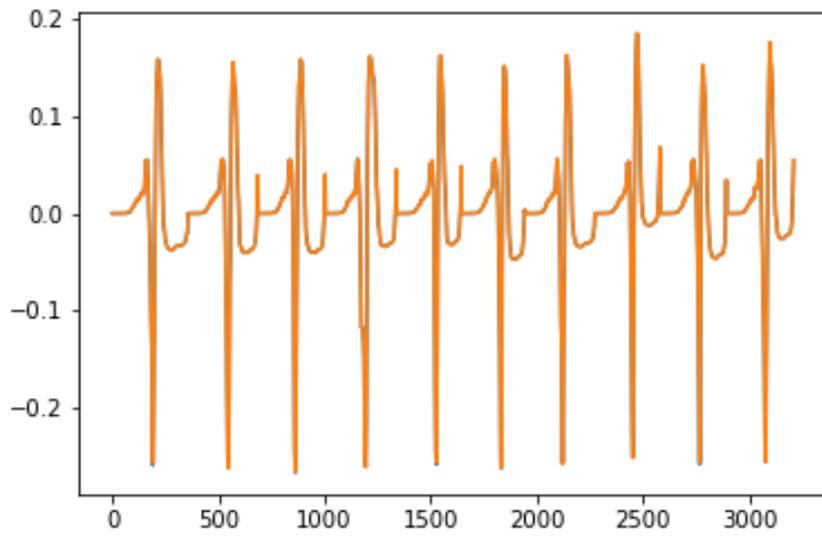


Fig. 4.3.5: Results node 5

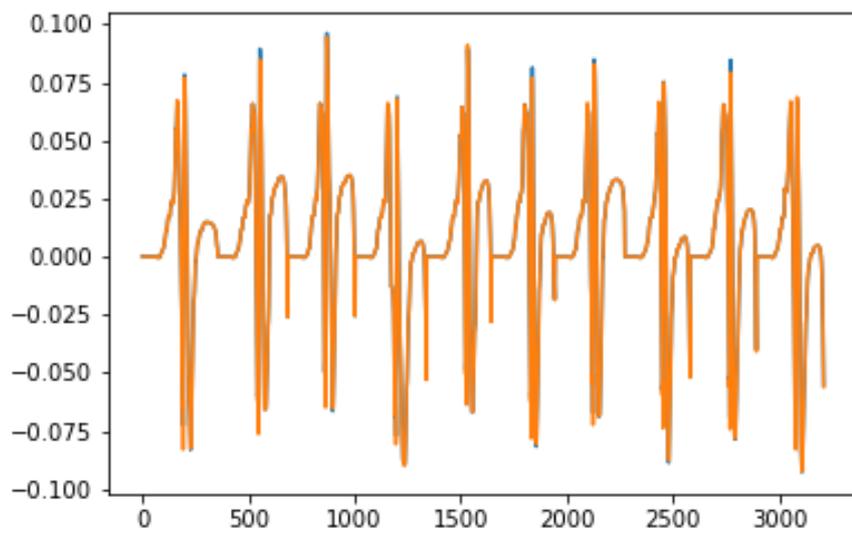


Fig. 4.3.6: Results node 6

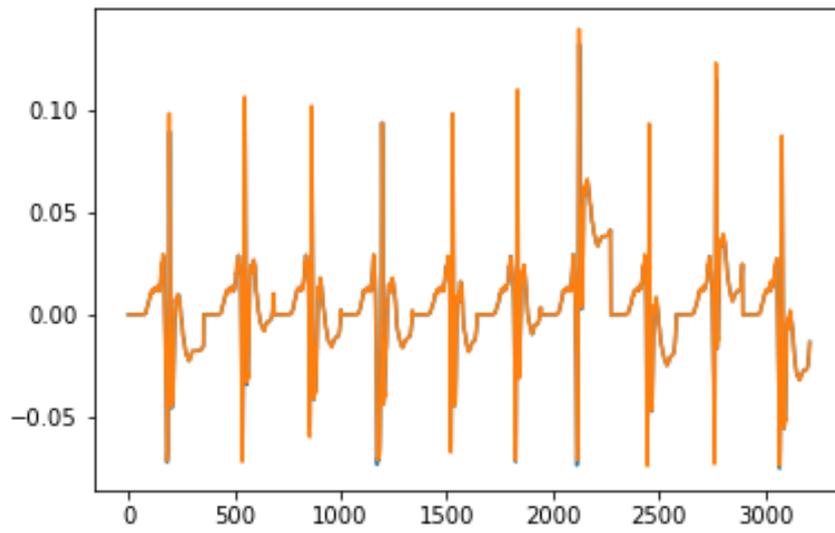


Fig. 4.3.7: Results node 7

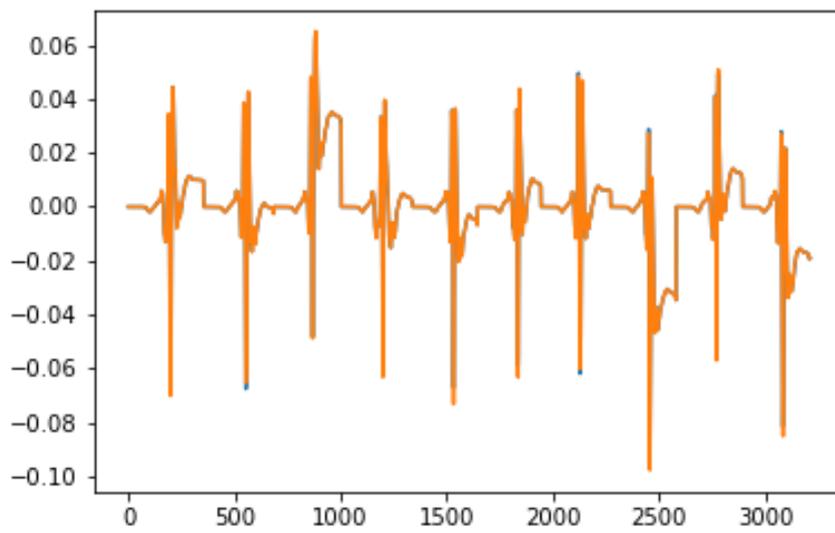


Fig. 4.3.8: Results node 8

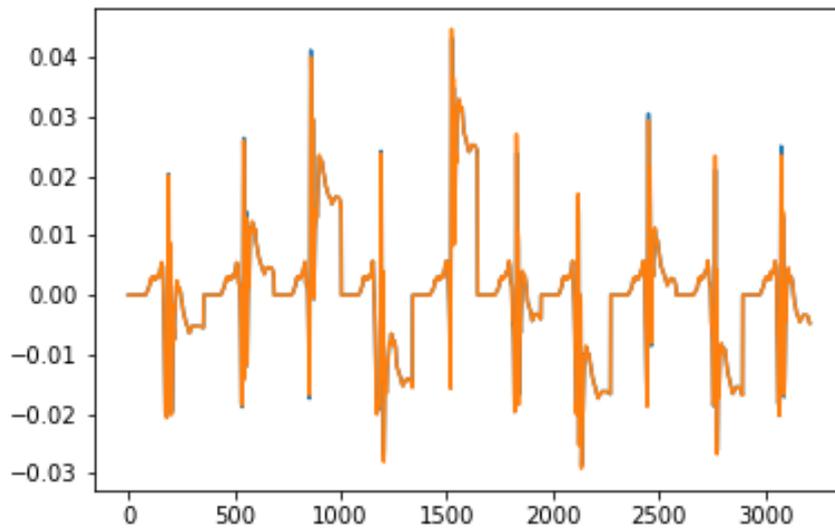


Fig. 4.3.9: Results node 9

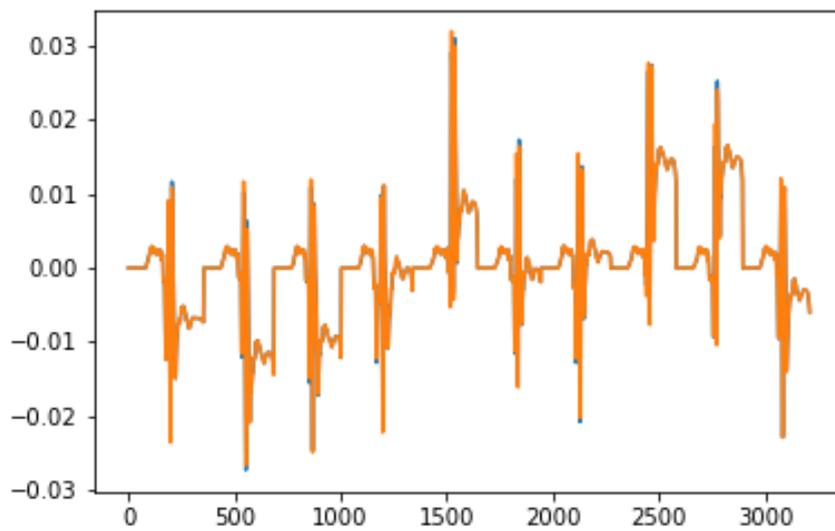


Fig. 4.3.10: Results node 10

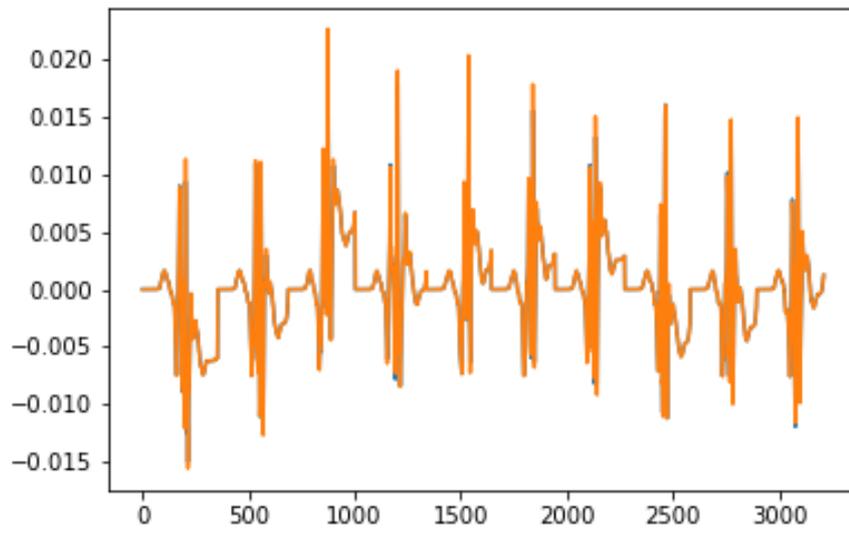


Fig. 4.3.11: Results node 11

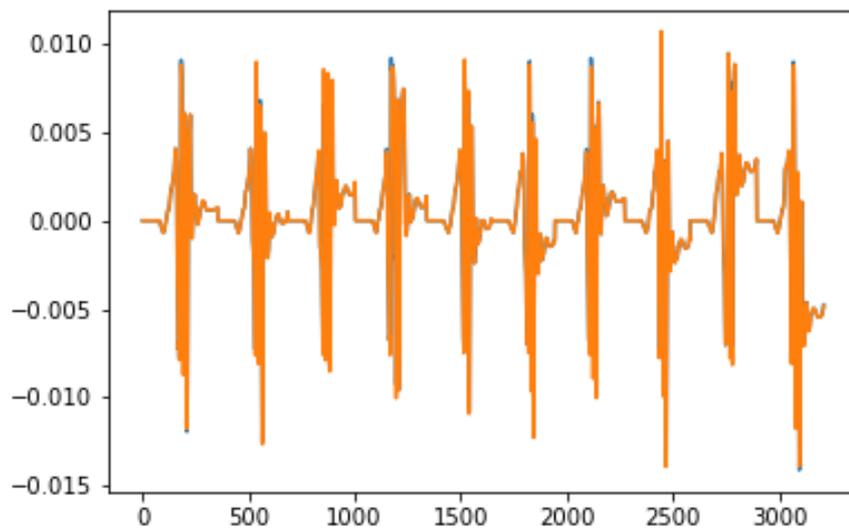
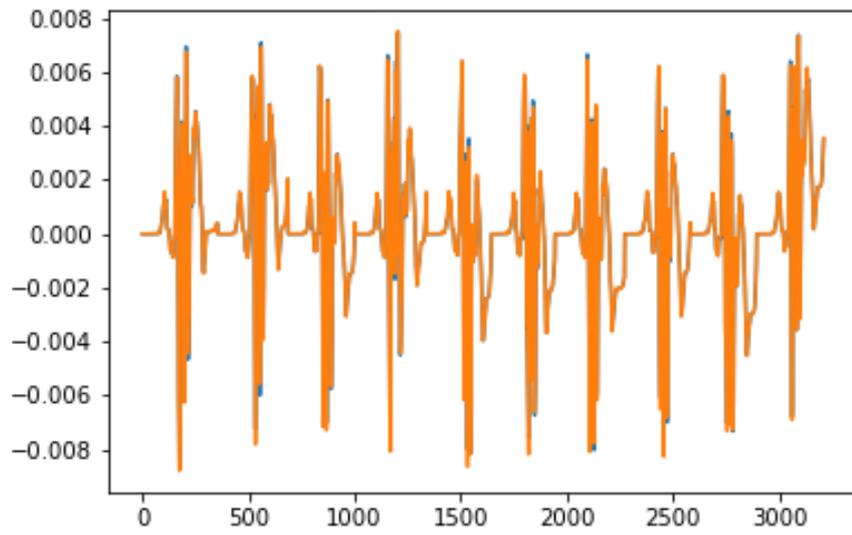
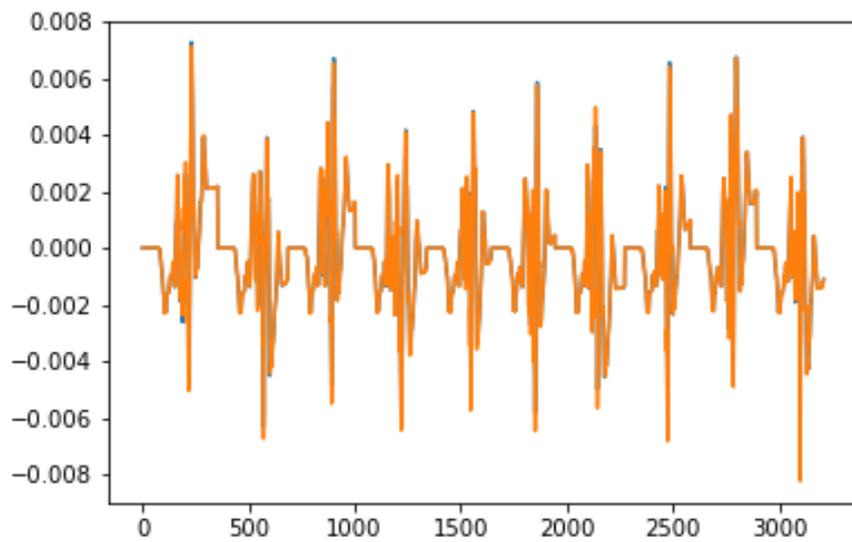


Fig. 4.3.12: Results node 12



**Fig. 4.3.13:** Results node 13



**Fig. 4.3.14:** Results node 14

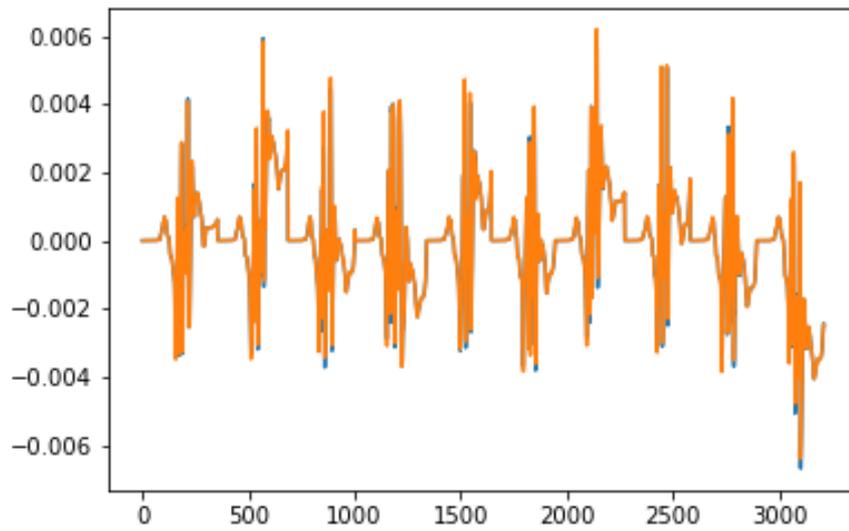


Fig. 4.3.15: Results node 15

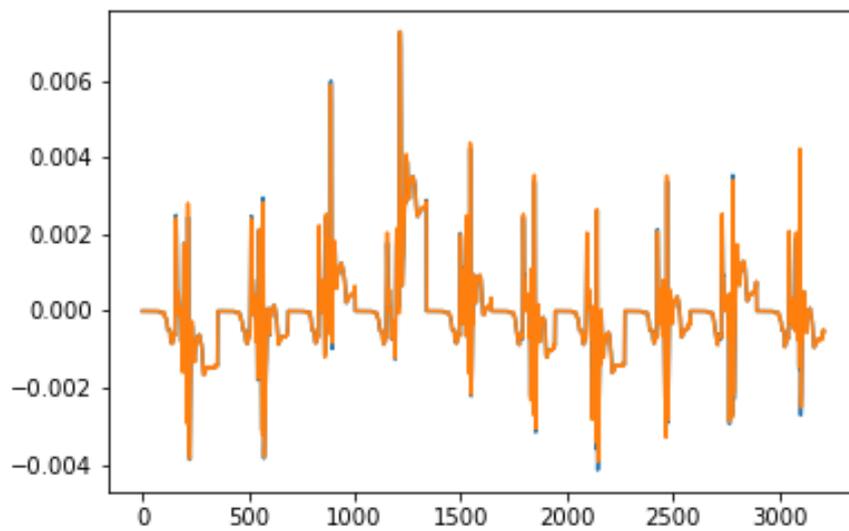


Fig. 4.3.16: Results node 16

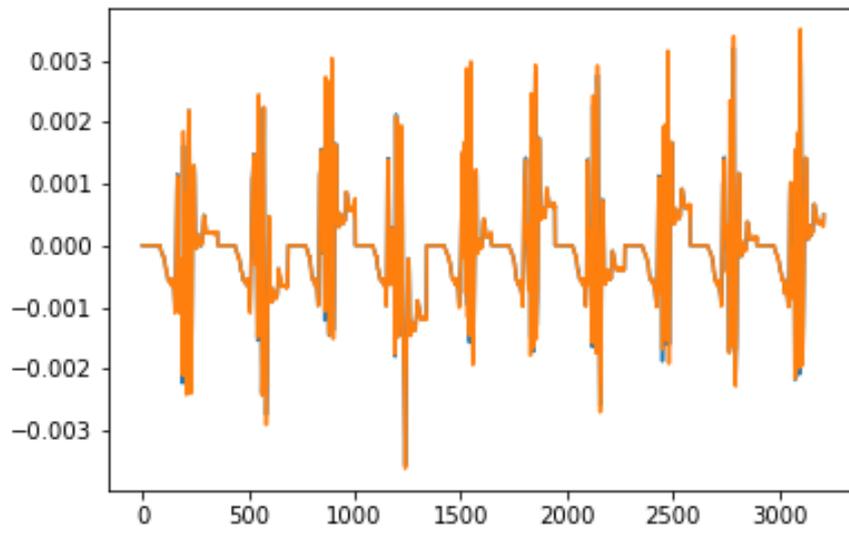


Fig. 4.3.17: Results node 17

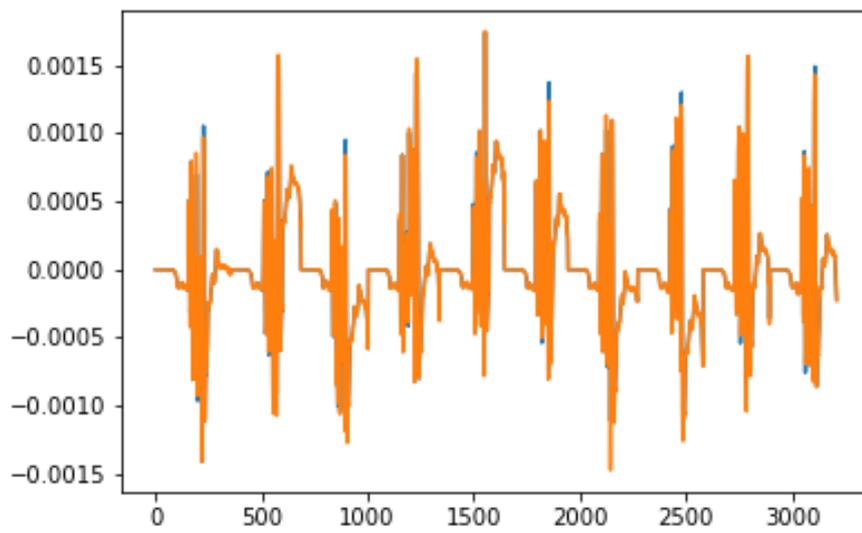
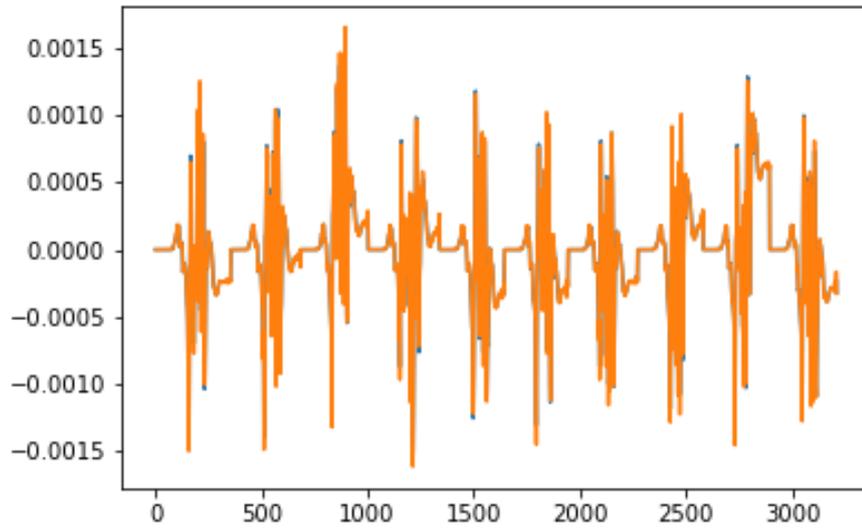
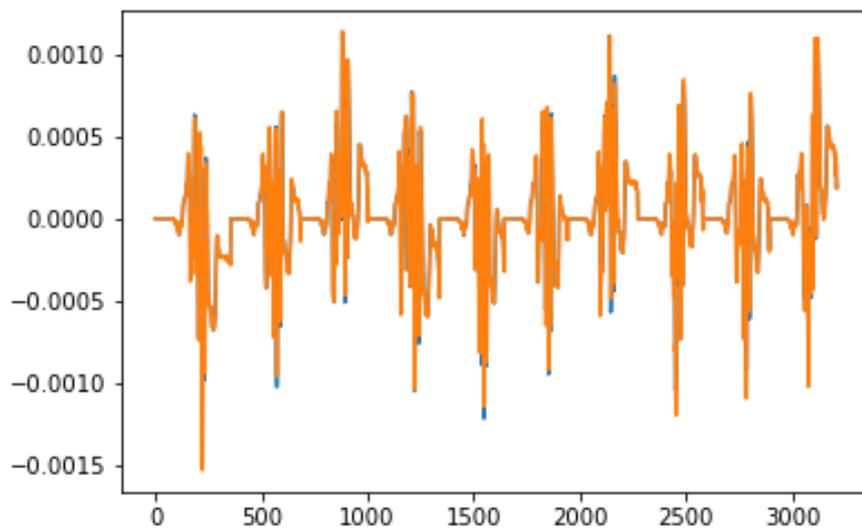


Fig. 4.3.18: Results node 18



**Fig. 4.3.19:** Results node 19



**Fig. 4.3.20:** Results node 20

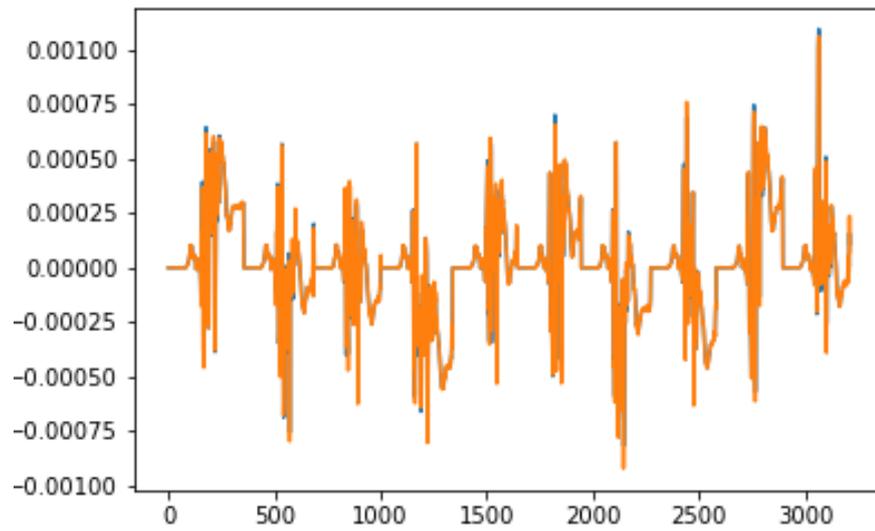


Fig. 4.3.21: Results node 21

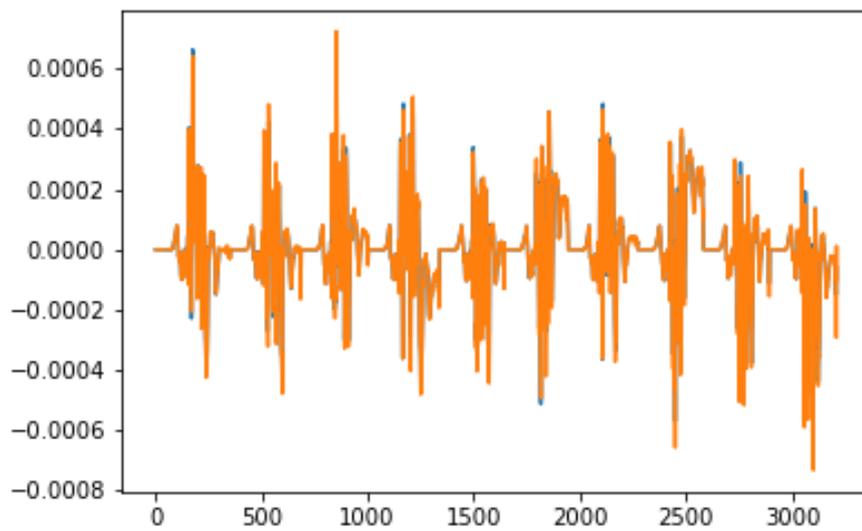
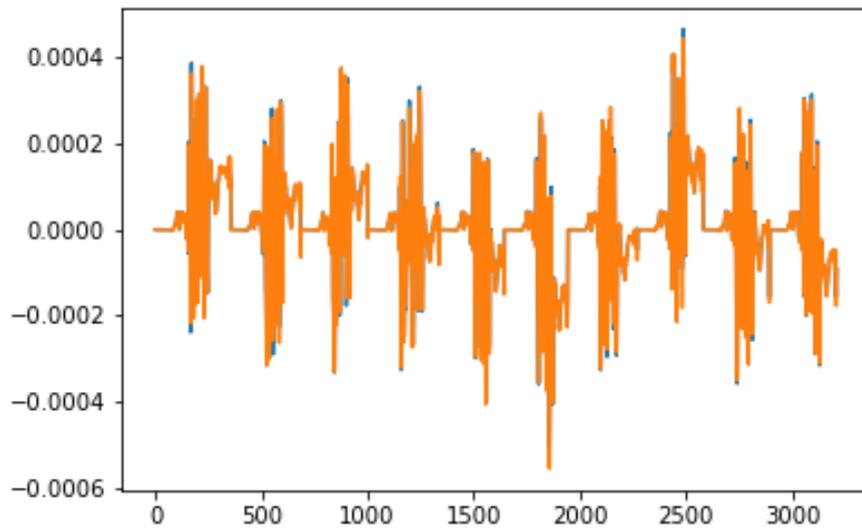
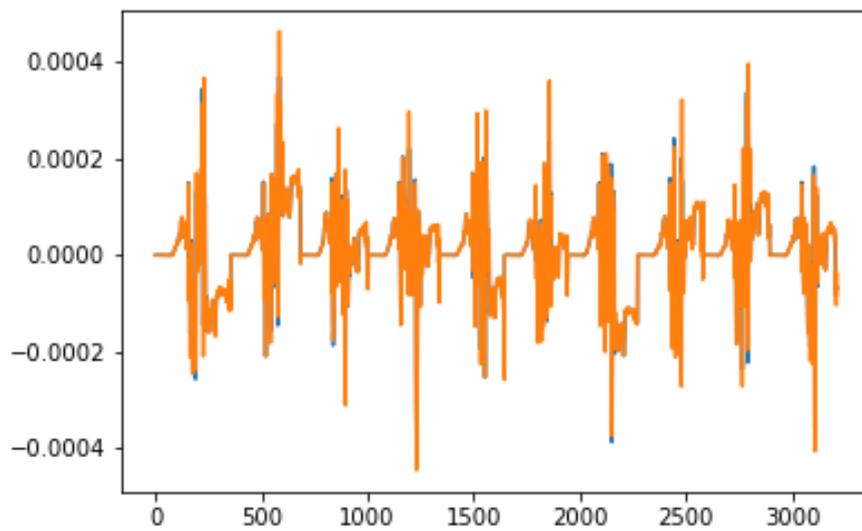


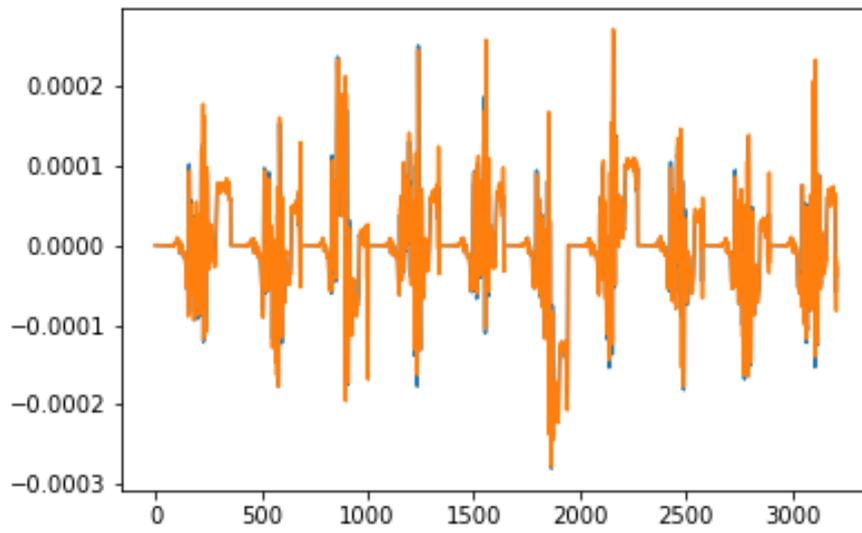
Fig. 4.3.22: Results node 22



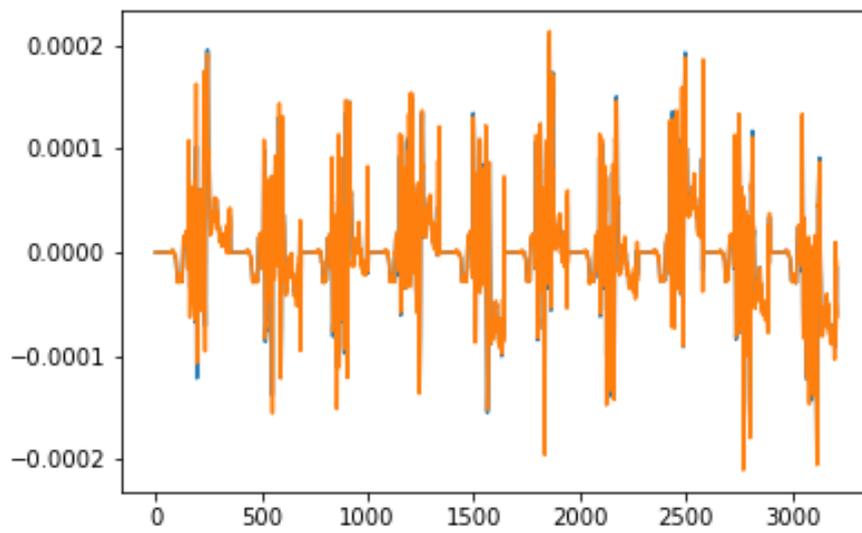
**Fig. 4.3.23:** Results node 23



**Fig. 4.3.24:** Results node 24



**Fig. 4.3.25:** Results node 25



**Fig. 4.3.26:** Results node 26

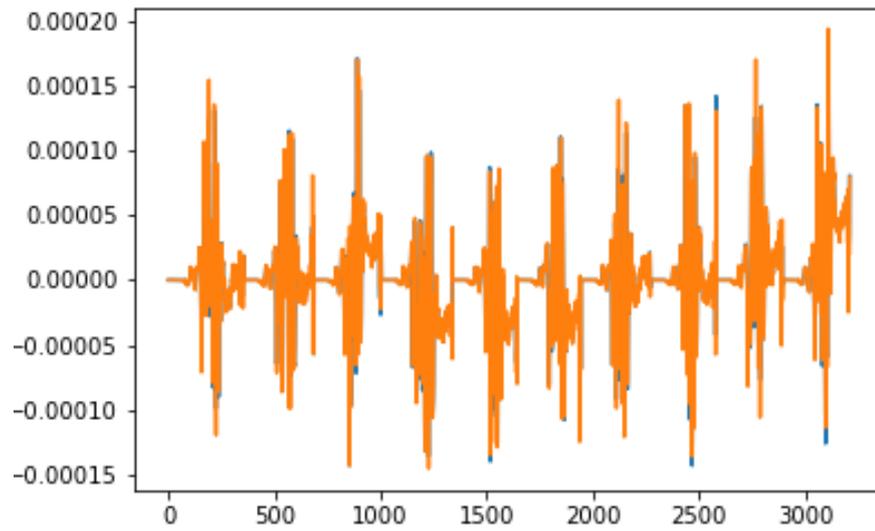


Fig. 4.3.27: Results node 27

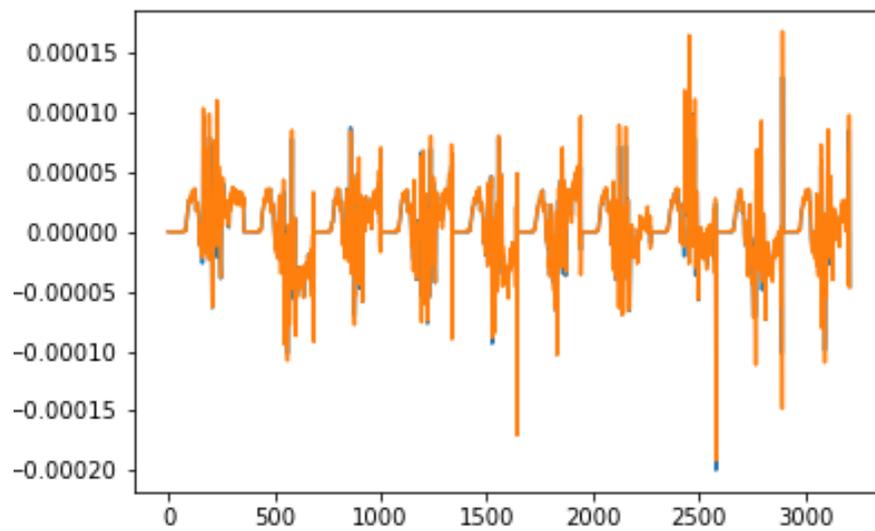
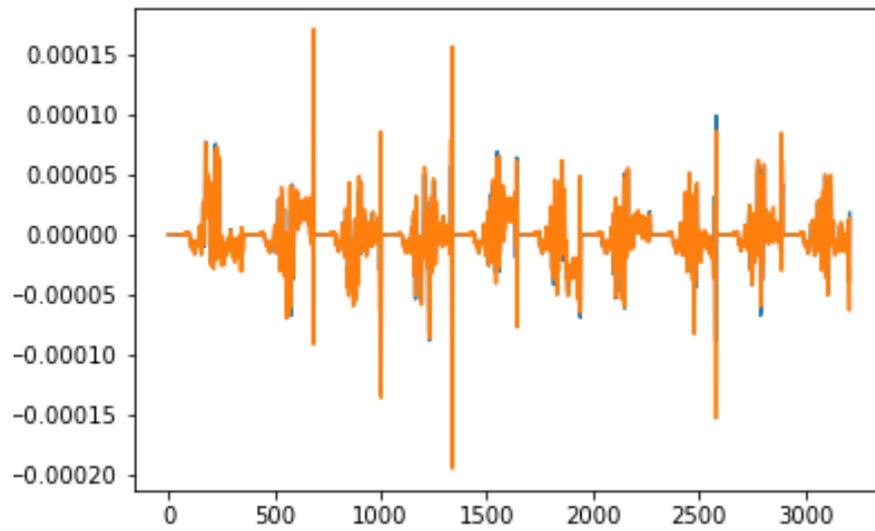
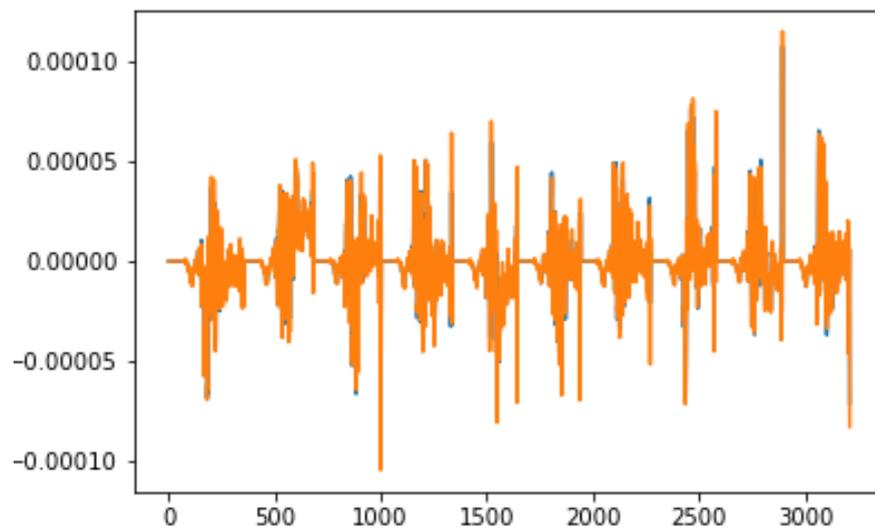


Fig. 4.3.28: Results node 28



**Fig. 4.3.29:** Results node 29



**Fig. 4.3.30:** Results node 30

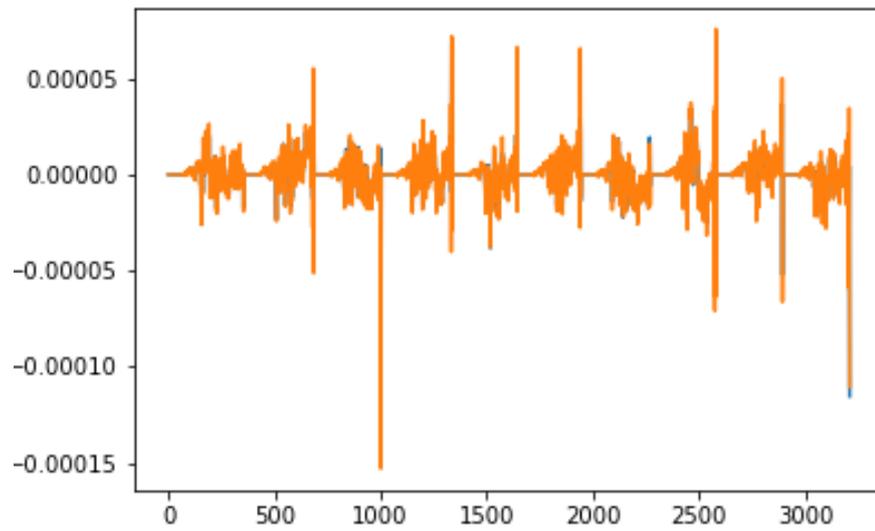


Fig. 4.3.31: Results node 31

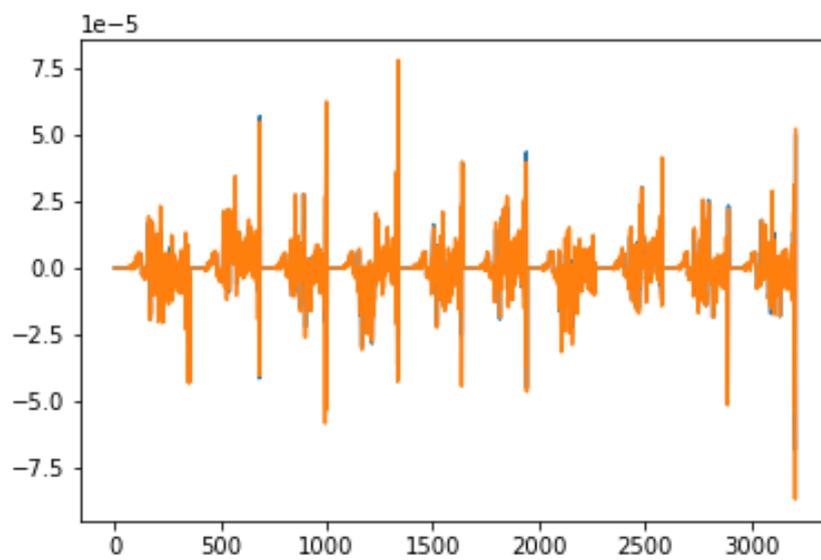


Fig. 4.3.32: Results node 32

# Conclusions and Future works

The Proper Orthogonal Decomposition(POD) implementation by itself generates very accurate results of errors less than 0.5% in less amount of time.

The Gaussian process shows an accurate results without being enriched with the Gaussian Process,but, some non-linearities are not correctly predicted.

The Proper Orthogonal Decomposition with Gaussian Process Regression improved with a Bayesian Optimization allows us to reproduce a multi-physics high-fidelity simulation with a high accuracy, that with respect to the FEM an average error of less than 1%. Moreover, the online procedure of our model with respect to the one of Abaqus is x1000 faster (real-time).

With the results obtained here, we plan to continue working on:

1. Implementation of a many-query control of rolling forming, by accessing a real time error.
2. Integration of data to obtain prediction of models with more Degree of Freedom (DOF) by bayesian inversion, which requires many query of prediction simulation.
3. Implementation of an Artificial Neuronal Network

# Bibliography

- [1] M Salvetti, C Corre, A Iollo, P Breilkopf, P Sagaut, J Jouhaud, Anne Gazaix, and C Louriou. Surrogate models coupled with machine learning to approximate complex physical phenomena involving aerodynamic and aerothermal simulations. *Doctoral Thesis L'UNIVERSITÉ DE TOULOUSE*, 2019.
- [2] Dassault Systèmes. User manual abaqus. *MIT*, 2016.
- [3] O. FRIDERIKOS, M. OLIVE, E. BARANGER, D. SAGRIS, and C. DAVID. A non-intrusive space-time interpolation from compact stiefel manifolds of parametrized rigid-viscoplastic fem problems. *HAL*, 2020.
- [4] Keiper W, Milde A, and Volkwein S. Reduced-order modeling (rom) for simulation and optimization. *Springer*, 2018.
- [5] Quarteroni, Manzoni A, and Negri F. Reduced basis methods for partial differential equations. *UNITEXT*, 2016.
- [6] David Amsallem and Charbel Farhat. Interpolation method for adapting reduced-order models and application to aeroelasticity. *AIAA*, 2012.
- [7] David Amsallem and Charbel Farhat. An online method for interpolating linear parametric reduced-order models. *SIAM Journal on Scientific Computing*, 2011.
- [8] E Dowell, K Hall, J Thomas, R Florea, B Epureanu, and J Heeg. Reduced order models in unsteady aerodynamics. *AIAA*, 1999.
- [9] K Hall, J Thomas, and E Dowell. Reduced order modeling of unsteady small disturbance flows using a frequency domain proper orthogonal decomposition technique. *AIAA*, 1999.
- [10] M Romanowski. Reduced order unsteady aerodynamic and aeroelastic models using karhunen loeve eigenmodes. *AIAA*, 1996.
- [11] P Beran and W Silva. Reduced order modeling: New approaches for computational physics. *AIAA*, 2001.
- [12] M. Fahl. Trust-region methods for flow control based on reduced order modeling. *PhD thesis, Trier university*, 2000.
- [13] Mengwu Guo and Jan S. Hesthaven. Data-driven reduced order modeling for time-dependent problems. *Comput. Methods Appl. Mech. Engrg.*, 2018.
- [14] J. S. Hesthaven and S. Ubbialia. Non-intrusive reduced order modeling of nonlinear problems using neural networks. *Journal of Computational Physics*, 2018.

- [15] Jan S. Hesthaven, Gianluigi Rozza, and Benjamin Stamm. Certified reduced basis methods for parametrized partial differential equations. *Springer*, 2016.
- [16] Schilder Wilhelmus H., van der Vorst Henk A., and Rommes Joost. Model order reduction: Theory, research aspects and applications. *Springer*, 2008.
- [17] Qian E, Kramer B, Peherstorfer B, and Willcox K. Lift & learn: Physics-informed machine learning for large-scale nonlinear dynamical systems. *Physica D: Nonlinear Phenomena*, 2020.
- [18] Dunhui Xiao. Non-intrusive reduced order models and their applications. *Imperial College Phd Thesis*, 2016.
- [19] Siddhartha Ray. Principles and applications of metal rolling. *Advanced Modeling and Simulation in Engineering Sciences*, pages 1–29, 2018.
- [20] John G Lenard. Primer on flat rolling sciences. *El Sevier*, pages 1–29, 2013.
- [21] Johansson L and Klarbring A. Thermoelastic frictional contact problems: modelling, finite element approximation and numerical realization. *Comput Methods Appl M*, 105:181–210, 1993.
- [22] Oanced V and Laursen T. A finite element formulation of thermo mechanical rate dependent frictional sliding. *Int J Numerical Methods Eng*, 1997.
- [23] Wriggers P and Miehe C. Contact constraints within coupled thermomechanical analysis:a finite element model. *Comput Methods Appl M*, 1994.
- [24] Wriggers P and Miehe C. Real contact mechanisms and finite element formulation:a coupled thermomechanical approach. *Int J Numer Methods Eng*, 35, 1992.
- [25] De Saracibar CA. Numerical analysis of coupled thermomechanical frictional contact problems. *Computational model and applications Arch Comput Methods E*, 5, 1998.
- [26] Pantuso D, Bathe KJ, and Bouzinov PA. A finite element procedure for the analysis of thermo-mechanical solids in contact. *Comput Struct*, 75, 2000.
- [27] Xing H and Makinouchi A. A three dimensional finite element modeling of thermomechanical frictional contact between finite deformation bodies using r-minimum strategy. *Comput Methods Appl M*, 191, 2002.
- [28] Lee EH. Finite-strain elastic—plastic theory with application to plane-wave analysis. *Appl Phys*, 38, 1967.

- [29] Alexander Seitz, Wolfgang A Wall, and Alexander Popp. A computational approach for thermo-elasto-plastic frictional contact based on a monolithic formulation using non-smooth nonlinear complementary functions. *Advanced Modeling and Simulation in Engineering Sciences volume*, 2018.
- [30] Wriggers P and Laursen TA. Computational contact mechanics. *Appl Phys*, 2006.
- [31] Laursen TA. Computational contact and impact mechanics. *Springer*, 2002.
- [32] Laursen TA. Computational contact and impact mechanics. *Springer*, 2002.
- [33] Gerhard Hirt, Markus Bambach, Simon Seuren, Thomas Henke, and Johannes Lohmar. Recent developments in modeling of hot rolling processes: Part i - fundamentals. *Institute of Metal Forming, RWTH Aachen University*, 2013.
- [34] Johannes Lohmar, Simon Seuren, Markus Bambach, and Gerhard Hirt. Design and application of an advanced fast rolling model with through thickness resolution for heavy plate rolling. *Institute of Metal Forming, RWTH Aachen University*, 2014.
- [35] R. B. Sims. Proceedings of the institution of mechanical engineers. 1954.
- [36] E. Orowan. Proceedings of the institution of mechanical engineers. 1943.
- [37] Chinesta F, Cueto E, and Abisset-Chavanne E. Virtual, digital and hybrid twins: A new paradigm in data-based engineering and engineered data. *Archives of Computational Methods in Engineering*, 2018.
- [38] Francisco Chinesta, Antonio Huerta, Gianluigi Rozza, and Karen Willcox. Model reduction methods. *American Cancer Society*, pages 1–36, 2017.
- [39] B Peherstorfer and K Willcox. Data-driven operator inference for non intrusive projection based model reduction. *Computer Methods in Applied Mechanics and Engineering*, 2016.
- [40] Y Wang, B Yu, Z Cao, W Zou, , and G Yu. A comparative study of pod interpolation and pod projection methods for fast and accurate prediction of heat transfer problems. *International Journal of Heat and Mass Transfer*, 2021.
- [41] Schlegel Michael and Noack Bernd R. On long-term boundedness of galerkin models. *Journal of Fluid Mechanics*, 2015.
- [42] Nguyen N.C. and Peraire J. An efficient reduced-order modeling approach for non-linear parametrized partial differential equations. *Int. J. Numer. Meth. Engng*, 2008.

- [43] Kalashnikova I and Barone M F. On the stability and convergence of a galerkin reduced order model (rom) of compressible flow with solid wall and far-field boundary treatment. *International Journal For Numerical Methods In Engineering*, 2009.
- [44] Iollo Angelo, Dervieux Alain, Desideri Jean-Antoine, and Lanteri Stephane. Two stable pod-based approximations to the navier–stokes equations. *Computing and Visualization in Science*, 2000.
- [45] Bui-Thanh Tan, Karen Willcox, and Murali Damodaran. Applications of proper orthogonal decomposition for inviscid transonic aerodynamics. *SAM*, 2003.
- [46] L Sirovich. Turbulence and the dynamics of coherent structures part 1 : Coherent structures. *Quarterly of Applied Mathematics*, 1987.
- [47] Hung V and Hien T. Modeling and control of physical processes using proper orthogonal decomposition. *El Sevier*, 2001.
- [48] Xiaojing Wuab, Weiwei Zhangb, Xuhao Pengb, and Ziyi Wangb. Benchmark aerodynamic shape optimization with the pod-based cst airfoil parametric method. *El Sevier*, 2015.
- [49] F Vetrano, F Mastroddi, and R Ohayon. Pod approach for unsteady aerodynamic model updating. *CEAS Aeronautical Journal*, 2015.
- [50] Peng Chen and Omar Ghattas. Hessian-based sampling for high-dimensional model reduction. *Arxiv*, 2018.
- [51] Mu Li, Wei Bi, James T. Kwok, and Bao-Liang Lu. Large-scale nyström kernel matrix approximation using randomized svd. *IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS*, 2015.
- [52] N. Halko, P. G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *Society for Industrial and Applied Mathematics*, 2011.
- [53] Michael W. Berry, Azlinah Mohamed, and Bee Wah Yap. Supervised and unsupervised learning for data science. *Springer*, 2020.
- [54] Carl Edward Rasmussen. Gaussian processes in machine learning. *Max Planck Institute for Biological Cybernetics*, 2004.
- [55] Mengwu Guo and Jan S. Hesthaven. Reduced order modeling for nonlinear structural analysis using gaussian process regression. *Computer Methods in Applied Mechanics and Engineering*, 2018.

- [56] Simon Haykin. Neural networks : A comprehensive foundation. *Pearson, Prentice Hall*, 2005.
- [57] Stergiou C and Siganos D. Neural networks. *Pearson, Prentice Hall*, 2013.
- [58] Han J and Moraga C. The influence of the sigmoid function parameters on the speed of backpropagation learning. *Research Group Computational Intelligence Dept. of Computer Science*, 1995.
- [59] J Ghosh and A Nag. An overview of radial basis function networks. *BH*, 2001.
- [60] M. D. Buhmann. Radial basis functions. *Cambridge University Press*, 2000.
- [61] Amir Sharif Ahmadian. Numerical models for submerged breakwaters. *BH*, 2016.
- [62] P. I. Frazier. Bayesian optimization. recent advances in optimization and modeling of contemporary problems. *INFORMS Tutorials in Operations Research*, 2018.
- [63] [mathlab.github.io/ezyrb/](https://mathlab.github.io/ezyrb/). Easy reduced basis method.
- [64] Orestis Friderikos, Emmanuel Baranger, Marc Olive, and David Néron. On the stability of pod basis interpolation via grassmann manifolds for parametric model order reduction in hyperelasticity. *HAL*, 2020.
- [65] Snelson E, Rasmussen C, and Ghahramani Z. Warped gaussian processes. *Advances in neural information processing systems 14*, 2003.