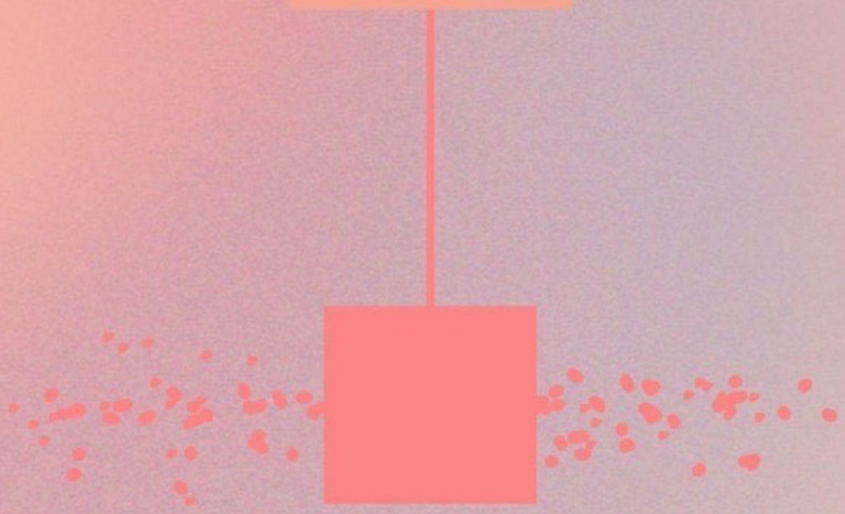
Reduced Order Modeling for Spatially Varying Radiative Interfaces

by

Nina Bagchus

MSc Applied Mathematics - Computational Science & Engineering

In collaboration with ASML



Daily Supervisor:	Shobhit Jain
Responsible Supervisor:	Martin van Gijzen
ASML Supervisor:	Abdullah Waseem

Cover: Graphic image made by Lotte Vollebregt

TU Delft

ASML

Abstract

Radiative heat transfer between moving geometries is critical in many high-temperature engineering systems, yet existing simulation approaches are often too computationally expensive. This study addresses this challenge by developing and testing a reduced order modeling framework for a transient, coupled, nonlinear radiative heat transfer problem with spatially varying subsystems.

We implemented a finite element model of two geometries in Python as a test problem, incorporating the nonlinear radiation boundaries, and varying view factors due to the motion between the subsystems. Different model order reduction techniques are applied on the test problem, each facing distinct limitations. The spatially varying nonlinearity caused most of the complexities. This nonlinearity is solely on the boundary, hence we use substructuring to isolate the nonlinearity. The linear internal dynamics can be reduced with well-established reduction techniques, like the Craig-Bampton method. In addition, the nonlinear boundary term is approximated by a feedforward neural network that was trained on simulation data to approximate nonlinear radiation exchange as a function of the temperatures on the radiative boundaries and relative position, enabling us to ignore the view factor computations, which are computationally very expensive.

The proposed reduced order model (ROM), combining Craig-Bampton reduction for internal degrees of freedom with a neural network approximation for interface radiation, achieves significant computational savings compared to the full order model (FOM). Results show that with decoupling internal and interface dynamics, the Craig-Bampton basis reduces the system matrices sizes while preserving accuracy in the full solution. On the interface, the neural network provides an efficient approximation for the nonlinear, spatially varying radiation operator, reducing the computational costs from quadratic $\mathcal{O}(NM)$ scaling (FOM) to a linear $\mathcal{O}(N + M)$ dependence, where N and M are the number of elements on the radiative boundaries of two components. In time integration, the Newton iterations remain identical in terms of the setup to the full model, but the reduced residual and Jacobian evaluations are significantly faster as the matrix sizes are reduced. Although both the Craig-Bampton reduction and neural network require additional offline computations (modal analysis and training), these are one-time costs. The online benefits are significant: the ROM reproduces the full solution very accurately while enabling much faster simulations, making it very useful for repeated evaluations, parametric-, and design studies.

Contents

List of Figures

Nomenclature

Abbreviations

FOM	Full Order Model
ROM	Reduced Order Model
POD	Proper Orthogonal Decomposition
SVD	Singular Value Decomposition
DEIM	Discrete Empirical Interpolation Method
VFM	View Factor Matrix
RHS	Right-Hand side
FEM	Finite Element Method
MOR	Model Order Reduction

Table 1: Simulation parameters (2D case)

Parameter	Symbol	Value
Length A	m	0.5
Height A	m	0.15
Length B	m	0.1
Height B	m	0.03
Thermal conductivity	k	237 $W/(m \cdot K)$
Density	ρ	2700 kg/m^3
Specific heat capacity	c_p	900 $J/(kg \cdot K)$
Heat flux	g	200 W/m^2
Emissivity	ϵ	1
Heat transfer coefficient A	κ_A	5 $W/(m^2 \cdot K)$
Heat transfer coefficient B	κ_B	5 $W/(m^2 \cdot K)$
Step size	h	0.05 m
Time step (fast motion)	Δt	1 s
Time step (slow motion)	Δt	80 s
Temperature	T_0	295 K

Table 2: Simulation parameters (3D ANSYS case)

Parameter	Symbol	Value
Length A	m	0.5
Height A	m	0.15
Width A	m	0.5
Length B	m	0.1
Height B	m	0.03
Width B	m	0.1
Material		Aluminium Alloy
Emissivity	ϵ	1
Heat transfer coefficient A	κ_A	20 $W/(m^2 \cdot K)$
Heat transfer coefficient B	κ_B	2 $W/(m^2 \cdot K)$
Mesh		Linear Tetrahedrons (Patch Conforming)
Temperature (initial and ambient)	T_0	295 K

Chapter 1

Introduction

This chapter provides the context and motivation for the research presented in this thesis. We begin by outlining the problem statement. Finally, we discuss the limitations of the existing research, present the research questions and give an overview of the structure of the thesis.

1.1 Problem Statement

In many engineering and physical systems, heat transfer between separate geometries occurs primarily through radiative exchange. This study investigates such a case within the context of ASML, an international company that develops machines to produce microchips. These microchips are used in a wide variety of everyday products, like telephones, laptops, cars or smartwatches.

Given that these chips require a nanometer precision, thermal management is very important in the development and maintenance of the machines. Minimal thermal deformation of internal components in the machine can cause the microchips to be defective. Therefore, accurate modeling of heat transfer in this machine is essential. The Full Order Model (FOM) of a thermal numerical model of the ASML machine is computationally very expensive, due to the high dimensionality and the complexity of it. Reduced Order Models (ROMs) are developed to reduce the costs of the simulations, while preserving the essential dynamics of the system.

The machine consists of numerous distinct parts that can be modeled and reduced separately prior to reconstructing them as one single model. For this study, we focus on a specific subsystem of the machine: the metroframe and the waferstage. This part plays a significant role in radiative heat transfer. The metroframe is a part that is a stationary part which is fixed, the waferstage moves under the metroframe, see Figure 1.1, where the smaller and lower geometry is the

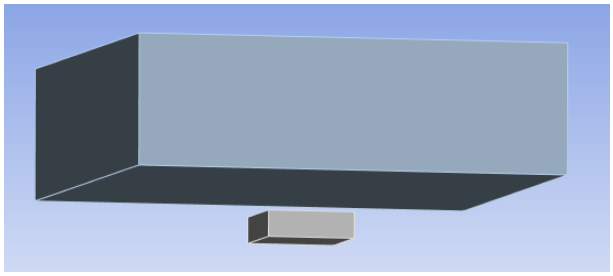


Figure 1.1: Simplified model of the metroframe and the waferstage

moving waferstage. Note that the two blocks are separated, they do not touch. In the actual machine, the waferstage is connected to a cable carrier which transports energy and data. This cable carrier heats the waferstage, causing it to radiate heat to its surroundings, including the metroframe.

The numerical model developed in this study for the metroframe-waferstage subsystem, contains two main difficulties, both mainly happening in the interface between the two geometries. The first is the nonlinear radiation boundary conditions that occur at the boundaries facing each other. The other component is the view factor matrix. The view factor matrix can be seen as a matrix that captures how much the elements or areas "see" each other. The view factor matrix contains all scalars, F_{ij}'s, with a number between 0 and 1 with a quantity that tells you how much element i sees element j. The number 0 means that the elements do not see each other, so a ray of heat will never hit that surface. A view factor of 1 means that all the energy that leaves an element will be absorbed by the other element [1].
The moving waferstage causes complexity in the computations of this view factor matrix. The view factor matrix will change every timestep because the elements "see" each other from a different point. During simulations of the heat distribution, the calculation of this view factor matrix is very expensive.

1.2 Background: Reduced Order Modeling

This study focuses on Model Order Reduction (MOR). This is a widely used technique to reduce high-dimensional mathematical models. MOR replaces a high-dimensional model with a low-dimensional approximation, the reduced order model (ROM), that preserves the dominant dynamics while reducing simulation time and the dimension of the FOM. Such high dimensionality typically arises when discretizing partial differential equations (PDEs), in this work it arises when discretizing the heat equation.

To solve the heat equation, the governing PDEs are discretized according to the Finite Element Method (FEM), i.e. the domain is partitioned into smaller subdomains. The heat equation is solved separately over the subdomains. This partitioning results in a large algebraic system. Accuracy requires fine meshes, and the number of degrees of freedom in the algebraic system grows rapidly with mesh refinement and domain size. Especially, for large, complex geometries or assemblies (e.g., ASMLs machines), it yields very high-dimensional and computationally expensive models, motivating MOR.

Numerous model-reduction methods exist, each suited to different aspects of a dynamical system. Choosing an appropriate method depends on the system's characteristics, whether it is linear or exhibits weak or strong nonlinearities; whether coupling is one- or two-way; whether there is time-dependent forcing (e.g., transient heat transfer); and whether a steady state exists. These are examples that are important for the scope of this thesis, there are in general a lot more characteristics for systems. In general, strong nonlinearities often require more complex nonlinear reduction techniques, whereas weak nonlinearities can sometimes be reduced with linear techniques. One-way coupling can simplify reduction by eliminating feedback, while two-way coupling introduces feedback that typically increases complexity.

Methodologically, it is useful to distinguish between physics-informed and data-driven elements, and between projection-based and non-projection approaches. Projection-based ROMs construct a low-dimensional subspace and project the governing equations onto it. Many of these are also data-driven in how the subspace is identified. Purely data-driven ROMs learn patterns and

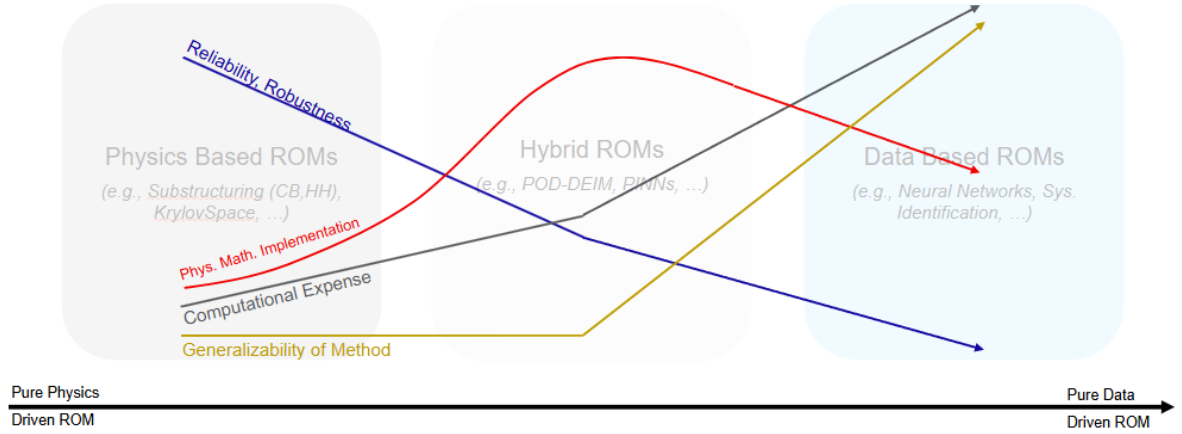


Figure 1.2: Overview of different MOR techniques. Including the advantages and disadvantages of the methods.

dominant dynamics directly from data or experiments. On the contrary, physics-informed ROMs are useful because they use the underlying physics to create a low-dimensional model, instead of relying only on data. This strategy does not require any data, which is advantageous, but they can be impractical when the equations are not general, i.e. they are unknown, too complex, or inaccessible. There are also hybrid strategies that combine both: physics-informed subspaces combined with data-driven approaches. An overview of these different MOR techniques and their advantages and disadvantages is given in Figure 1.2. In the figure, it becomes clear that the physics-informed ROMs are often relatively easy to implement, they are reliable because they are based on physics, and often have relatively low computational costs. In contrast, data-driven approaches are very general, meaning they can be applied across a wide range of domains with limited interpretability.

In this study, we are working with with a nonlinear heat transfer problem which shows a periodic steady state solution, due to the periodic movement of the waferstage. We begin with the classical, data-driven, projection-based method, the Proper Orthogonal Decomposition (POD) [2]. It uses the Singular Value Decomposition (SVD) to recognize the dominant modes - that capture the dominant spatial and temporal patterns - in the system and projects the full system onto a lower-dimensional subspace. This method only efficiently reduces a model if the system is linear, as the nonlinear term has to be evaluated in full space [3]. A strongly nonlinear system needs some additional techniques to capture the nonlinear behavior of the system.

A solution would be to linearize the nonlinear radiation boundary conditions to approximate it as linear boundary condition. This makes the system easier and very efficient to reduce if the ROM approximates the real solution accurate enough. For linear systems, there are already a lot of very efficient MOR techniques, like the POD method. Sometimes a data-driven SVD-based method, like the POD, is not efficient, because an SVD is relatively expensive to compute. There are a lot of other linear methods, like krylov methods [4], [5], which do not require the computation of the SVD and already have promising results. The difficulty of this study will be significantly reduced if the linearization of the nonlinear boundary conditions is accurate enough. Linearization is in practice hard to apply, as the test model is an idealized version of the real model with sharp corners, holes and non smooth movements of the waferstage. This motivates the investigation in nonlinear MOR techniques.

The Discrete Empirical Interpolation Method (DEIM) method is a popular extension to the POD method as it accelerates the computation of the nonlinearity [3]. When combined with POD,

it is proved to dramatically reduce the computational complexity of the POD for a nonlinear system [6]. DEIM uses, like POD, the SVD method to extract dominant modes of the nonlinear part of the system and creates a second basis with these modes. DEIM is most effective when the nonlinear term can be represented accurately with only a few modes. For systems with more complex nonlinearities, more modes may be required to achieve the same accuracy, and alternative reduction approaches may be more suitable.

We can also solve a general eigenvalue problem with the FEM matrices. This motivates the use of a modal decomposition. We can identify the small and large eigenvalues, which corresponds to the slow and fast modes of the system. The eigenvectors ϕ_i's form an orthogonal basis and allow projection on this basis to obtain a reduction of the system. To reduce the system, one chooses the smallest eigenvalues and creates a basis of the corresponding eigenvectors. Discarding some eigenvectors result in a lower-dimensional subspace onto which the full-order model (FOM) can be projected.

Usually, the eigenvalues provide information in the neighborhood of the equilibrium, but because the system has nonlinear behavior, the true dynamics can be very different. However, nonlinear systems often evolve on low-dimensional, curved surfaces in phase space known as invariant manifolds. Nonlinear Modal Decomposition is a method that provide guidelines of the construction of these manifolds and how to project the full-system onto it. Spectral Submanifolds (SSMs) are the smoothest invariant manifolds tangent to spectral subspaces of the linearized system [7]. They extend classical modal subspaces to nonlinear manifolds and capture the local nonlinear behavior near equilibria [8]. These methods require a steady state solution, which we do not have in our spatially varying model. There are some new studies that are also able to construct periodic SSMs, which are useful when the system does not have a constant steady state solution, but a periodic steady state solution [9], [10]. In the actual model, the behavior is less smooth compared to our test model, making it challenging to identify the periodic steady state.

As mentioned before, there are other reduction techniques that are more suitable for nonlinear systems. For example, data-driven neural networks are efficient in modeling nonlinear behavior [11]. Autoencoders are a popular method that learns projections from the high dimensional system to a low dimensional latent space. These methods do not rely on linear combinations of modes, but learn a nonlinear manifold that represents the important dynamics [12]. Autoencoding methods, like the use of Convolutional Autoencoders (CAEs) [13] or Variational Autoencoders (VAEs) [14] have already promising results. However, neural networks can be expensive to develop because they require large, representative datasets generated from the FOM. Moreover, a trained network is typically developed to a specific system; applying it to a different system generally requires new training data. By contrast, physics-informed approaches, which use the governing equations and constraints, are often more adaptable across different systems.

Furthermore, to align with the industry standard, in this thesis, we will reduce subsystems of the big assembly separately before assembling them together. Modal decomposition is a widely used method in the industry, therefore we have a strong focus on this method throughout this thesis.

The main MOR method for the interface is the use of partitioning the interface in so-called $N\_$ nodes which assume constant fluxes on these patches. This method is also called the virtual point transformation [15]. All the interface information is mapped to one $N\_$ node. Note that for a spatially varying problem, this assumption is not representative and can cause a significant error. A ROM for this specific problem has not yet been developed within existing workflows. Therefore, we will look beyond these workflows and propose an alternative approach.

Each method described above has its own strengths and limitations. The choice depends on the dynamics of the system:

- **Modal decomposition:** fits ASMLs workflow and provide a clear physics-informed ROM.
- **POD:** data-driven, effective for systems with weak nonlinearities.
- **DEIM:** additional method which is efficient for handling nonlinearities.
- **Autoencoders:** powerful for strong and complex nonlinearity.
- **NMD/SSM:** non data-driven technique for stronger nonlinearity and periodic behavior.
- **Linearized boundaries:** make all reduction methods easier to apply.

We do not aim to cover all possible methods, but we will explore methods or combinations of these methods to reduce the spatially varying nonlinear heat transfer problem. Along the way, practical constraints and other limitations shaped our choices to investigate in a certain range of MOR techniques.

This study also addresses another computational complex operation, namely the View Factor Matrix (VFM) computation. The view factor can be seen as a measurement of visibility. It computes how much a surface sees the other surface. In a FEM model, this results in a matrix (element-to-element measurement). Section 2.2 will explain this phenomena in detail. Due to the moving waferstage, the view factor from one element to another changes every timestep. This study also focuses on reducing the costs of computing that matrix. The study focuses on reducing the dimension of the matrix. One method will compute specific view factor matrix computations for different locations *offline* and store those. The VFM for the positions within the simulations that are not stored are approximated by an interpolation of the VFMs that are stored.

1.3 Research questions

This study focuses on Reduced Order Modeling for Spatially Varying Radiation Interfaces. The physical problem consists of heat transfer between two systems, which are coupled through the radiation boundary conditions. One of the systems is moving in space, resulting in a transient, nonlinear, coupled heat transfer system with a periodic steady-state solution if the moving system moves in a periodic matter.

Solving the system with the FEM method is computational very expensive due to the high dimensionality of the problem, the view factor matrix computations and the nonlinear boundary conditions. Furthermore, it is a coupled system where one component is moving relative to the other component. The main reserach question for this paper is: **Can we develop a ROM for the spatially varying radiative interface problem, that accurately approximates the solution of the FOM, while reducing the computational costs?**

Sub research aspects

1. Nonlinearity:

One of the sub-research questions focuses on the nonlinearity of the system. How much does this nonlinearity influence the system, e.g: it is only applied on the boundary, maybe the system is dominated by linear behavior? How good is the approximation when you linearize the boundary conditions? Extending the test problem to the real model will result in more difficult behavior of the system. Linearizing the system will probably not be accurate enough. The nonlinearity is therefore a strong component in this problem, and it influences the way of reduced-order modeling. First we will see how the linearized system behaves in comparison with the nonlinear system. After this, we will encoounter the nonlinearity in the problem and find a way to reduce the nonlinear system. There are various methods that have already been shown to be efficient, so we will try to see if those methods can be used in this specific problem.

2. View Factor matrix:

Another aspect of the ROM for this problem is the View Factor Matrix (VFM). Because of the high dimensionality and the moving waferstage, this VFM is computationally expensive. To reduce the computations of the view factor matrix, we can store data matrices at specific locations and interpolate the matrix during the time steps. Is this a good approximation for the real model, when the geometries become more complex?

3. Methods:

There is a wide variety of methods for reduced-order modeling available. The question is which one is best suited or which combination is best suited for this specific problem. Each method excels in different dynamical systems. It is important to know the dynamics of the system to know which method is best suited. This aspect is very important for this problem and results in the following research question: Which methods approximates the FOM the most accurate, while reducing the computation time efficiently?

4. Coupling of the systems:

The study contains two separate subsystems, the metroframe and the waferstage. They are connected with dynamics, but how strong is this connection? Can we solve the systems independently or do we have to solve the coupled system as a whole? In some reduced order modeling techniques, we create a basis, this basis can be a basis for the entire system or there can be two bases for the two different geometries. Also, the coupling between two systems can be one- or two-way coupling. One-way coupling reduces the complexity of the system, since the feedback is being neglected. For one way coupling, one subsystem is radiating all the heat and the other subsystem only absorbs heat. In two-way coupling, the later subsystem will eventually also radiate heat back.

1.4 Outline

Firstly, we will evaluate the methods on a test problem, described in chapter 2. Subsequently, the aim is to apply the model order reduction techniques, which are assumed to be the most promising for this problem, to a prototype of the metroframe-waferstage system developed in ANSYS, the $3D$ modeling software used within ASML. This 3D model resembles the actual real-life system more closely and allows for a more realistic evaluation of the Model Order Reduction approaches.

Chapter 3 will dive into the mathematical depth of the explained methods. Chapter 4 will go over all the results for the methods tested on our test problem, it also proposes the most promising algorithm for the problem. Chapter 5 shows the results of the most promising method on the

3D model in ANSYS. We will also discuss the limitations and options for further investigation in the Discussion in Chapter 6 and provide a conclusion for the proposed algorithm in Chapter 7. Finally, you can find all the Python- and MATLAB code in the Gitlab repository [16].

Chapter 2

Test Problem

Test problems are commonly used to reduce the complexity of a model. In this study, the test problem consists of two separate two-dimensional geometries without a common boundary, i.e. they are separated, but they are coupled through the radiation boundary conditions. The heat equation for this problem is solved within the framework of the Finite Element Method (FEM).

2.1 Radiation

The geometries are coupled through radiation boundary conditions. All bodies above the temperature of 0 K emit radiation. Thermal radiation is a volume phenomenon, i.e. the radiation is the result of excitation of all the particles of a body. However, radiation travels to the surface and is then emitted from the surface, radiation in interacting geometries is therefore considered as surface phenomenon [1].

The equation for radiation is given by the Stefan–Boltzmann law:

$$q = \sigma T^4, \tag{2.1}$$

where σ is the Stefan–Boltzmann constant, T is the temperature in Kelvin and q is energy emitted by radiation from the body in W/m^2. This is the equation for a black body, i.e. a perfect absorber and a perfect emitter. For a more realistic body, we need to take the emissivity into account. Eq. (2.1) becomes

$$q = \epsilon\sigma T^4, \tag{2.2}$$

where ϵ is the emissivity, the ratio of the energy emitted in comparison with a perfect emitter, which is a constant between zero and one.

For body i, we can describe the outward heat flow using Eq.(2.2): $q_i = \epsilon_i \sigma T_i^4$ where q_i is the energy emitted per m^2. Scaling this, with the area of the emitting surface, A_i, gives

$$Q_i = A_i \epsilon_i \sigma T_i^4$$

where Q_i is the energy emitted from surface A_i in W. From body j, which also emits energy, $q_j = \sigma T_j^4$ (assumption: black box), a part of the energy is absorbed by body i

$$q_{absorbed} = \alpha q_{received},$$

for α between zero and one. From the radiation emitted by the body j, the part absorbed by the body i is

$$q_{abs} = \alpha A_1 \sigma T_2^4.$$

Therefore, the net energy emitted (or absorbed) by body i is

$$Q_1 = A_1 \epsilon_1 \sigma T_1^4 - A_1 \alpha_1 \sigma T_2^4$$

For simplification, we will assume that $\alpha_1 = \epsilon_1$ (assumption), the equation will therefore simplify to

$$\boxed{Q_1 = A_1 \epsilon_1 \sigma (T_1^4 - T_2^4), \quad \text{W}} \tag{2.3}$$

Eq. (2.3) represents the derived expression for nonlinear radiative heat transfer between two surfaces. This formulation assumes that all the heat emitted by one surface *travels* completely to the other, i.e. the surfaces see each other with a view factor of one. However, this is not always the case, hence we should incorporate the view factor

2.2 View Factor

In radiation between two surfaces the view factor plays an important role. In a general enclosure with n surfaces, a surface i contributes to the heat exchange of the surrounding surfaces in the enclosure that are visible from surface i. For this matter, we introduce the view factor, the fraction of surface j that is visible by surface j. In a more mathematical way, it can be described as the amount of energy that leaves surface i and that is intercepted by surface j [17]:

$$F_{ij} = \frac{q_{i \to j}}{A_i J_i}, \tag{2.4}$$

where F_{ij} is the view factor from surface i to j, A_i the area of surface i and J_i the total radiosity.

Consider two differential patches dA_i and dA_j separated by a distance R. The radiant intensity leaving dA_i is dI_i, which is the intensity of radiation leaving surface i. The rate at which the energy leaves surface A_i and is intercepted by dA_j is expressed as

$$dq_{i \to j} \;=\; I_i \, \cos\theta_i \, d\Omega_{j \to i} \, dA_i,$$

where θ_i is the angle between the surface and the outward normal and $d\Omega_{j \to i}$ is the angle subtended by dA_j when viewed from dA_i[18]. $d\Omega_{j \to i} = \dfrac{dA_j \cos\theta_j}{R^2}$, hence we have

$$dq_{i \to j} = I_i \frac{\cos\theta_i \cos\theta_j}{R^2} \, dA_j \, dA_i.$$

For the total rate of the energy emitted by surface i and received by surface j, assuming it is a black body, i.e. $J_i = \pi I_i$, we can integrate over both surfaces

$$q_{i \to j} = J_i \int_{A_i} \int_{A_j} \frac{\cos\theta_i \cos\theta_j}{\pi R^2} \, dA_j \, dA_i$$

From the definition of the view factor in Eq. (2.4), we get the generalized equation for the view factor

$$\boxed{F_{A_i \to A_j} = \frac{1}{A_i} \int_{A_i} \int_{A_j} \frac{\cos\theta_i \cos\theta_j}{\pi R^2} \, dA_j \, dA_i \; .} \tag{2.5}$$

A detailed explanation and visualization for this equation can be found in chapter 4 of Modest [19] or in chapter 13 of Incropera and DeWit [18].

2.2.1 View Factor Equation in 2D

For a 2D problem, we can follow the same analogy: for a diffuse, line element dl_i the radiative intensity is I_i. The energy leaving dl_i toward a second element dl_j is

$$dq_{i\to j} = I_i \cos\theta_i \, d\varphi_{j\to i} \, dl_i.$$

The angle of dl_j and dl_i is

$$d\varphi_{j\to i} = \frac{\cos\theta_j dl_i}{R},$$

where R is the distance between the midpoints of i and j. Hence

$$dq_{i\to j} = I_i \frac{\cos\theta_i \cos\theta_j}{R} \, dl_j \, dl_i.$$

Following the exact same reasoning as above in the $3D$ case, we obtain the $2D$ formula for the view factor:

$$\boxed{F_{L_i\to L_j} = \frac{1}{L_i} \int_{L_i}\int_{L_j} \frac{\cos\theta_i \cos\theta_j}{\pi R} \, dl_j \, dl_i \, .} \tag{2.6}$$

The view factor is important in this study, because the waferstage is a moving geometry, which causes the geometric quantities to change every timestep. In each timestep, the surfaces face each other from a different angle and distance. In an FEM analysis, we work with differential areas, which causes the view factor matrix to have a very high dimension if there is a fine mesh. Also, in the real model, the geometry is not as ideal as in this test problem, there are corners or holes, causing the view factor computation to be more complex. All of this together causes a computationally complex situation.

The view factor matrix is calculated for each element on the boundary which radiates heat. In the test problem, this is the bottom of the metroframe and the top of the waferstage. To construct the view factor matrix, we loop over each element on these boundaries. Within this loop, each element will also loop over every other element that is on a radiation boundary to compute the view factor. The view factors of the elements on the same boundary are zero in this test problem. Note that in the real model, because of holes and corners, this does not have to be true. If there are 14 elements at the bottom of the metroframe and 11 on the top of the waferstage, the view factor matrix will be a 25×25 matrix. For a visualization of the view factor matrix, see Figure 2.1. Note that the white blocks are equal to zero; this shows that for element 14 (row 14) on surface A the view factor for the first 15 elements (first 15 columns), which are on the same surface A, are zero. For the last 11 columns (elements on surface B), the values are nonzero. Note that this structure of the matrix only applies for two facing surfaces. The diagonal will always be zero, since the view factor from one element to itself is always zero. The expressions in Eq. (2.5) and Eq. (2.6) assume that the surfaces are diffuse emitters and that there are no obstructions between them. It also satisfies the reciprocity relation:

$$A_i F_{ij} = A_j F_{ji}, \tag{2.7}$$

which makes computations easier. The effect of the view factor is visualized in Figure 2.2. A schematic drawing of the position of the waferstage (lower block) is given with the heat distribution in the metroframe (top block). Note that in Figures 2.2a and 2.2b, only the metroframe (top block) is given.

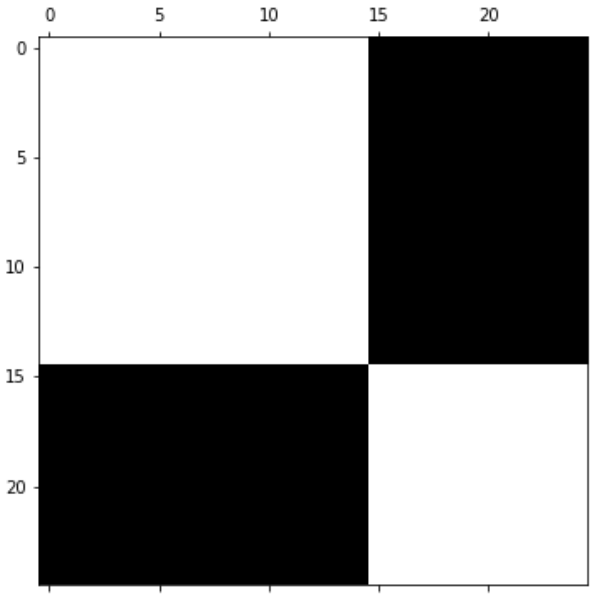
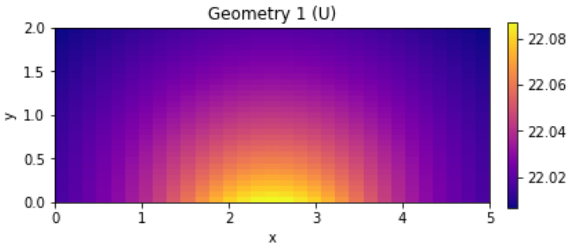
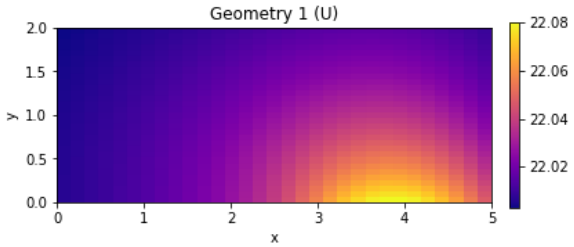


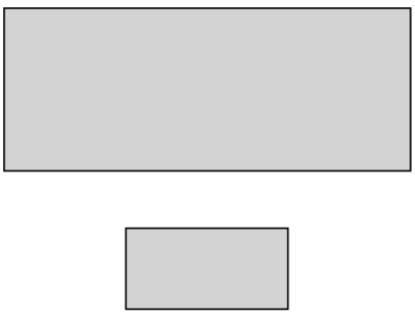
Figure 2.1: View factor matrix.



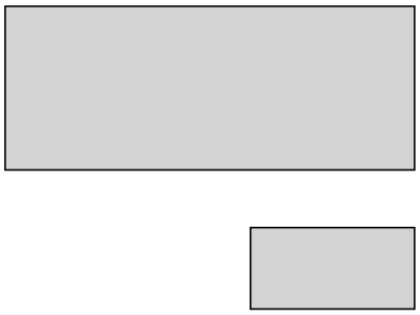
(a) Waferstage is placed in the middle



(b) Waferstage is placed at the right



(c) Schematic drawing (a)



(d) Schematic drawing (b)

Figure 2.2: Visualization of view factor

This view factor can be incorporated into Eq. 2.3. Also we can change the equation to the heat flux q_1 in W/m^2:

$$q_1 = F_{12}\epsilon_1\sigma(T_1^4 - T_2^4).$$

This equation gives the radiative boundary condition which can be incorporated into the discretized FEM equation of the two dimensional heat equation. This process is given in the next section.

2.3 Two-dimensional Heat Equation

To solve the transient heat equation, we start with the strong form of the energy equation:

$$\rho c_p \frac{\partial T}{\partial t} = \nabla \cdot (k\nabla T) \quad \text{in } \Omega,$$

where ρ is the material density, c_ρ the specific heat capacity, T the temperature field, k is the thermal conductivity and Ω is the computational domain.

Take the following boundary conditions into account:

- **Neumann boundary condition**

$$-k\frac{\partial T}{\partial n} = g_N \quad \text{on } \Gamma_N$$

- **Convection boundary condition (Newton's Law of Cooling)**

$$-k\frac{\partial T}{\partial n} = \kappa(T - T_\infty) \quad \text{on } \Gamma_C$$

  where:

  - κ is the convection heat transfer coefficient.
  - T_∞ is the ambient temperature.

- **Radiation boundary condition**

$$-k\frac{\partial T_i}{\partial n} = \sigma\epsilon F_{ij}(T_i^4 - T_j^4) \quad \text{on } \Gamma_R \tag{2.8}$$

  where:

  - $T_i(x, y)$ and $T_j(x, y)$ are the temperatures on surfaces i and j

In this study the temperature for component A, the metroframe (top geometry), is denoted by T_A and the temperature of the waferstage, component B is denoted by T_B. The radiation boundary condition from Eq. (2.8) for component A, in the continuous form, can be written as

$$-k\frac{\partial T_A}{\partial n} = \sigma\epsilon F_{AB}(T_A^4 - T_B^4),$$

but for simplicity we will omit the view factor term in the following part.

The following analysis focuses on component A: to derive the weak form, we multiply the strong form by a test function v and integrate over the domain Ω. Using integration by parts (Green's

theorem), the diffusion term is rewritten, and the natural boundary conditions are applied. The resulting weak form is: find all solutions for $T_A(t)$ with initial condition $T_A(0) = T_{A,0}$ so, s

$$\int_\Omega \rho c_p \frac{\partial T_A}{\partial t} v \, d\Omega + \int_\Omega k \nabla T_A \cdot \nabla v \, d\Omega + \int_{\Gamma_C} \kappa T_A v \, d\Gamma$$
$$= \int_{\Gamma_N} g_N v \, d\Gamma + \int_{\Gamma_C} \kappa T_\infty v \, d\Gamma - (\int_{\Gamma_R} \sigma \epsilon T_A^4 v \, d\Gamma - \int_{\Gamma_R} \sigma \epsilon T_B^4 v \, d\Gamma),$$

holds for all test functions $v \in V$. We then discretize the function space using piecewise linear basis functions. Let $V_h \subset V$ be this space. Then, the problem is to find $T_{A,h}(t) \in V_h$, with initial condition $T_{A,h}(0) = T_{A,0}$ such that the equation above holds for all $v \in V_h$ at each time t. In a coupled system we also need to find $T_{B,h}(t) \in V_h$ for component B.

For time integration, we apply a Crank-Nicolson scheme to the time derivative, this scheme is unconditionally stable. Also, to derive a system of equations, let $\{\phi_i\}_{i=1}^n$, be the basis of the space V_h, also known as the hat functions. Let the test function represent one of the basis functions and know that the following equation has to hold for all basis functions. Then, approximate T_A as the linear combination of the values on the nodes, i.e. $T_{A,h} = \sum_{j=1}^{n_u} T_{A,j}\phi_j$. With the same analogy, $T_{B,h} = \sum_{j=1}^{n_w} T_{B,j}\phi_j$. The discrete system for each test function ϕ_i, $i = 1, \ldots, n_u$, becomes:

$$M\dot{T_A} + KT_A = F - R(T_A, T_B)$$

where

$$M_{ij} = \int_\Omega \rho c_p \phi_j \phi_i \, d\Omega$$
$$K_{ij} = \int_\Omega k \nabla \phi_j \cdot \nabla \phi_i d\Omega + \int_{\Gamma_C} \kappa \phi_j \phi_i \, d\Gamma$$
$$R_i = \int_{\Gamma_R} \sigma \epsilon (\sum_{j=1}^{n1} T_{A,j}\phi_j)^4 \phi_i \, d\Gamma - \int_{\Gamma_R} \sigma \epsilon (\sum_{k=1}^{n2} T_{B,k}\phi_k)^4 \phi_i \, d\Gamma$$
$$F_i = \int_{\Gamma_N} g_N \phi_i \, d\Gamma + \int_{\Gamma_C} \kappa u_\infty \phi_i \, d\Gamma$$

To obtain the full coupled system, we get

$$\begin{pmatrix} M_A & 0 \\ 0 & M_B \end{pmatrix} \begin{pmatrix} \dot{T_A} \\ \dot{T_B} \end{pmatrix} + \begin{pmatrix} K_A & 0 \\ 0 & K_B \end{pmatrix} \begin{pmatrix} T_A \\ T_B \end{pmatrix} = \begin{pmatrix} F_u \\ F_w \end{pmatrix} - \begin{pmatrix} R_A(T_A, T_B) \\ R_B(T_A, T_B) \end{pmatrix}. \tag{2.9}$$

Eq. (2.9) presents the discretized form of the test problem. This formulation can be solved using finite element software, such as the FEM method. This concludes the setup of the test problem.

$$a(t) = V^T T(t)$$

Chapter 3

Model Reduction: Mathematical Framework

In this chapter, we present the mathematical framework of the reduced order modeling techniques used in this work. While Chapter 1 provided a conceptual overview, we now focus on the detailed derivation and theory of each method.

We begin with the modal decomposition and the Proper Orthogonal Decomposition (POD), which are very common projection based techniques. Next, we introduce the Discrete Empirical Interpolation Method (DEIM) to handle nonlinearities efficiently.

During the development of the ROM, it became evident that additional reduction techniques were needed to handle this systems with two separate subsystems with a nonlinear radiation boundary. As a result, this chapter also includes a detailed explanation of substructuring and the Craig-Bampton method, which were introduced at a later stage of the project.

3.1 Projection-Based Model Reduction

Projection-based Model reduction aims to approximate the high-dimensional solution of a dynamical system by projecting it onto a lower-dimensional subspace spanned by a reduced basis. Consider an uncoupled nonlinear full-order model of the form:

$$M\dot{T} + KT = F - R(T)$$

We approximate the solution in a reduced system as a linear combination of the vectors of the reduction basis :

$$T(t) \approx \sum_{i=1}^{r} v_i a_i(t) = Va(t), \tag{3.1}$$

where $V \in \mathbb{R}^{N\times r}$ is the reduced basis matrix and $a(t) \in \mathbb{R}^r$ are the reduced coordinates.

Substituting into the full-order equation and applying a Galerkin projection (multiplying by V^T from the left), we obtain the reduced-order model:

$$\underbrace{V^T M V}_{M_r}\,\dot{a} + \underbrace{V^T K V}_{K_r}\,a = \underbrace{V^T F}_{F_r} - \underbrace{V^T R(Va)}_{R_r(Va)} \tag{3.2}$$

This is a reduced system of size $r \ll N$ that captures the dominant dynamics of the full model. Projection-based methods like modal decomposition or POD are methods for creating such a reduction matrix V. For nonlinear terms $R(T)$, the evaluation of $R(Va)$ may still be expensive, and projection-based technique such as DEIM can be used to reduce computational cost

3.1.1 Modal Decomposition

When analyzing large dynamical systems, modal decomposition is a useful first step to understand the behavior of the system, especially when the system is linear or can be linearized. In the context of heat transfer, the spatial discretization of the governing equations are

$$M\dot{T} + KT = F - R(T),$$

where both M and K are constant if the nonlinearity is restricted to the boundary.

To better understand the dynamics of this system, assume that the forcing terms, F and $R(T)$, are zero. We can perform an eigenvalue decomposition of the generalized eigenproblem:

$$K\phi_i = \lambda_i M\phi_i, \quad \text{for } i = 1, \ldots, n \tag{3.3}$$

which yields eigenvalues λ_i and eigenvectors ϕ_i. These eigenvectors form an orthogonal basis. Using this basis, the solution $T(t)$ can be projected onto the eigenmodes, and the dynamics of each mode can be analyzed separately. The reduced basis will look like:

$$V_{MD} = \begin{bmatrix} \phi_1 & \phi_2 & \ldots & \phi_r \end{bmatrix} \in \mathbb{R}^{N \times r}.$$

The eigenvalues λ_i carry important information about the system:

- Negative real parts indicate stable modes that decay over time.
- Small magnitudes correspond to slow dynamics.
- Complex eigenvalues indicate oscillatory behavior.

This modal decomposition is especially useful in systems where a few modes dominate the long-term behavior. In such systems it helps to identify the slow subspace of the dynamics. We can neglect the other modes to efficiently reduce the system. The eigenvectors corresponding to the dominant eigenvalues, the dominant eigenmodes, form the reduces matrix V. Note that the eigenvalue problem assumes a zero forcing term. This is an assumption which is generally not true, like in our case. The behavior of the forcing terms is therefore not captured in the reduction basis and can result in larger errors or the necessity of more eigenmodes.

3.1.2 Proper Orthogonal Decomposition

Proper Orthogonal Decomposition (POD) is another projection-based technique. It is a data-driven technique that extracts the most dominant modes in the data set. The method creates the reduced basis V and again it consists of an orthonormal basis that tries to capture the behavior of the model optimally. The following analysis will focus on an arbitrary temperature T. This method, also tries to approximate the solution as

$$T(t) \approx \sum_{i=1}^{r} v_i a_i(t) = Va(t), \tag{3.4}$$

where ϕ_i are the POD modes, which we will construct and $a_i(t)$ are the time dependent coefficients, this is explained in more detail in Sabrina Kelbij Star et al., [20]. First, a snapshot

matrix is constructed with snapshots of the solution of a full-order system, $T(t)$. The snapshots are stored in a data matrix

$$X(t) = \begin{bmatrix} | & | & & | \\ T_1(t) & T_2(t) & \cdots & T_m(t) \\ | & | & & | \end{bmatrix},$$

where m is the number of snapshots. A Singular Value Decomposition (SVD) of the snapshot matrix is then performed:

$$S = U\Sigma W^T$$

where:

- $U \in \mathbb{R}^{N\times n}$ contains the left singular vectors (POD modes),
- $\Sigma \in \mathbb{R}^{n\times n}$ is a diagonal matrix of singular values,
- $W \in \mathbb{R}^{n\times n}$ contains the right singular vectors.

The columns of U corresponding to the largest singular values span the subspace in which the system evolves most significantly. By selecting the first r modes (where $r \ll N$), we define the reduced basis:

$$V_{\text{POD}} = \begin{bmatrix} u_1 & u_2 & \dots & u_r \end{bmatrix} \in \mathbb{R}^{N\times r}$$

where u_i are the dominant left singular vectors. This basis is then used to project the full-order model as described in the previous section.

3.1.3 DEIM

Consider the arbitrary nonlinear heat transfer equation

$$M\dot{T} + KT = F - R(T), \tag{3.5}$$

Using Eq. (3.1) to approximate the solution with the POD method and projecting Eq. (3.5) onto the reduced basis, $V_r \in R^{N\times r}$, using the Galerkin projection, like Eq. (3.2) gives

$$M_r\dot{a} + K_r a = F_r - V^T R(Va) \tag{3.6}$$

The linear terms are reduced successfully to a lower-dimensional space. However, the non-linear term $R(Va)$ is still computationally expensive. It requires evaluating the term in the full space ($\mathbb{R}^N$), whereas the POD method does not benefit in the sense of model reduction of the nonlinear term.

The Discrete Empirical Interpolation Method (DEIM) is developed to take care of the nonlinear part of the system. It approximates the nonlinear part as

$$R(Va(t)) \approx \Xi c(t),$$

where $\Xi \in \mathbb{R}^{n\times q}$ and $c(t) \in \mathbb{R}^{qx1}$ a coefficient vector. The POD-DEIM method uses, besides the common basis extracted from the snapshots of the full system, a second basis representing snapshots of the nonlinear term, Ξ [21]. It uses a DIEM interpolation matrix $P \in \mathbb{R}^{n\times q}$, which is made up of unit vector with specific interpolation points, such that

$$P^T R(Va(t)) \approx P^T \Xi c(t),$$

which gives

$$c(t) \approx (P^T \Xi^{-1}) P^T R(Va(t)),$$

so finally

$$R(Va(t)) \approx \Xi (P^T \Xi)^{-1} P^T R(Va(t)). \tag{3.7}$$

where $\Xi \in \mathbb{R}^{N \times q}$ is the reduced basis for the nonlinear term $R(Va(t))$ and $P \in \mathbb{R}^{N \times p}$ is a selection matrix that picks p important interpolation points.

Since $q \ll N$, and it only evaluates the nonlinear part at p selected indices, it significantly reduces the cost of evaluating $R(Va(t))$, making MOR efficient even for nonlinear systems [6].

The remaining task is to develop these matrices Ξ and P. Firstly, to construct the second basis, we take m snapshots from the nonlinear term R,

$$Y(t) = \begin{bmatrix} | & | & & | \\ R_1(t) & R_2(t) & \cdots & R_m(t) \\ | & | & & | \end{bmatrix}.$$

Taking the SVD of this matrix gives $Y = H \Sigma V^T$. Similarly as the POD method, we can take a rank-p approximation and form a new basis for Ξ, $\Xi_p = [\eta_1 \ \ \eta_2 \dots \eta_p]$.

After creating the second basis Ξ_p, we can now sample the matrix P by constructing measurements. To create the first interpolation point, choose $[\rho, \gamma_1] = \max|\eta_1|$, where γ_j is the index and ρ the extracted variable. Then $P_1 = [e_{\gamma_1}]$, which gives the first measurement matrix. For P_j, where $j = 2, 3, \dots, p$, use Algorithm 1.

Algorithm 1 P matrix for POD DEIM

```
P_1 = [e_{γ_1}]
for j = 1, 2, 3, ..., p do
    P_j^T H_j c_j = P_j^T η_{j+1}
    E_{j+1} = η_{j+1} − H_j c_j
    [ρ, γ_j] = max|E_{j+1}|
    P_{j+1} = [P_j  e_{γ_j}]
end for
```

Algorithm 1 gives the complete derivation of P, whereas we can approximate R as in Eq. (3.7). Inserting this approximation in Eq. (3.6) gives

$$M_r + K_r = F_r - V_r^T \Xi_q (P^T \Xi_q)^{-1} P^T R(V_r a(t)). \tag{3.8}$$

So, in the full model, the costs of evaluating the nonlinear part $R(U)$ is to a cost of $\mathcal{O}(n)$. In the POD-DEIM method, the nonlinear term is only evaluated at $p = q << N$ points which efficiently reduces the computational costs to $\mathcal{O}$(q) for the nonlinear term. The matrix multiplications

$$V_r^T \Xi_q (P^T \Xi_q)^{-1}$$

can be precomputed offline. So the evaluation of the nonlinear term is followed by a matrix-vector multiplications of order $\mathcal{O}(q)$ and $\mathcal{O}(qr)$. Therefore, the total cost of the nonlinear term is

$$\mathcal{O}(q + rq).$$

In contrast, evaluating the nonlinear term in the full-order model requires evaluating the nonlinear term in n entries, so of order $\mathcal{O}(n)$.

The terms M_r and K_r involve reduced mass and stiffness matrices of size $r \times r$, which can be precomputed offline. The online cost for evaluating each of these terms is thus $\mathcal{O}(r^2)$ for the multiplication with $a(t)$ or its derivative. In contrast, in the full-order model, the same operations involve matrices of size $n \times n$, resulting in a cost of $\mathcal{O}(n^2)$.

Combining all components of Eq. (3.8), the total cost per time step of the POD-DEIM reduced-order model is

$$\boxed{\mathcal{O}(r^2 + rq + q)},$$

whereas the full-order model typically has a cost of:

$$\boxed{\mathcal{O}(n^2 + n)}.$$

Since $r, q \ll n$, the POD-DEIM approach results in efficient reduction in computational cost.

The dynamics of the DEIM can also be combined with the classic modal decomposition. A bottleneck for this method is the accessibility of the nonlinear term. This method modifies the nonlinear term, a requirement is therefore that the nonlinear term is accessible and we can change it.

This concludes the projection-based methods that we investigate in this thesis. Furthermore, we investigate in structural changes within the discretized equation to isolate the nonlinear behavior of the system.

3.2 Substructuring

In the previous sections, we discussed some model order reduction techniques for the system. All these techniques had some limitations, e.g. the eigenvalue problem used in this method assumes a zero external force vector in the heat equation, whereas the test problem includes a nonlinear radiation term. Such behavior is not captured by the eigenvalues.

However, the nonlinearity that complicates the reduced-order modeling appears only on the boundary. This observation motivates the use of substructuring, a widely used technique that treats internal and boundary nodes separately. Substructuring enables the use of model reduction for internal nodes while keeping the boundary nodes in full space. This approach is explored by van Steen et al. [15] for model order reduction for interfaces. In the research of van Steen et al., model order reduction is only applied on the internal degrees of freedom, leaving the boundary nodes intact to facilitate coupling between subdomains.

In Eq (2.9) the full system is given. If we couple the two equation for the subsystems we get

$$\begin{pmatrix} M_A & 0 \\ 0 & M_B \end{pmatrix} \begin{pmatrix} \dot{T}_A \\ \dot{T}_B \end{pmatrix} + \begin{pmatrix} K_A & 0 \\ 0 & K_B \end{pmatrix} \begin{pmatrix} T_A \\ T_B \end{pmatrix} = \begin{pmatrix} F_A \\ F_B \end{pmatrix} - \begin{pmatrix} R_A(T_A, T_B) \\ R_B(T_A, T_B) \end{pmatrix} \tag{3.9}$$

Splitting this into internal and boundary nodes by permuting the matrices gives

$$\begin{pmatrix} M_{A,ii} & M_{A,ib} & 0 & 0 \\ M_{A,bi} & M_{A,bb} & 0 & 0 \\ 0 & 0 & M_{B,ii} & M_{B,ib} \\ 0 & 0 & M_{B,bi} & M_{B,bb} \end{pmatrix} \begin{pmatrix} \dot{T}_{A,i} \\ \dot{T}_{A,b} \\ \dot{T}_{B,i} \\ \dot{T}_{B,b} \end{pmatrix} + \begin{pmatrix} K_{A,ii} & K_{A,ib} & 0 & 0 \\ K_{A,bi} & K_{A,bb} & 0 & 0 \\ 0 & 0 & K_{B,ii} & K_{B,ib} \\ 0 & 0 & K_{B,bi} & K_{B,bb} \end{pmatrix} \begin{pmatrix} T_{A,i} \\ T_{A,b} \\ T_{B,i} \\ T_{B,b} \end{pmatrix} = \begin{pmatrix} 0 \\ F_{A,b} \\ 0 \\ F_{B,b} \end{pmatrix} - \begin{pmatrix} 0 \\ R_{A,b}(T_A, T_B) \\ 0 \\ R_{B,b}(T_A, T_B) \end{pmatrix}. \tag{3.10}$$

The zeros on the right-hand side occur because all the forces are applied on the boundary. This method allows us to treat the internal and external dynamics separately. There are different reduction techniques for the internal dynamics and for the interface dynamics, they are all listed and explained in detail by van Steen et al. [15]. We do not discuss all available substructuring and interface reduction techniques in this thesis, as the internal dynamics are governed by a classical linear system for which many effective and well-established methods exist. Instead, we focus on the nonlinear boundary system, for which a new framework is developed in this work: a neural network–based model reduction applied specifically to the radiative coupling. First, we will explain the linear reduction method Craig-Bampton which is used within this thesis.

3.2.1 Craig-Bampton

The Craig-Bampton method is a classic method for reducing the internal nodes with the eigenmodes of the system [22]. First we look at the problem for component A, i.e.

$$\begin{pmatrix} M_{A,ii} & M_{A,ib} \\ M_{A,bi} & M_{A,bb} \end{pmatrix} \begin{pmatrix} \dot{T}_{A,i} \\ \dot{T}_{A,b} \end{pmatrix} + \begin{pmatrix} K_{A,ii} & K_{A,ib} \\ K_{A,bi} & K_{A,bb} \end{pmatrix} \begin{pmatrix} T_{A,i} \\ T_{A,b} \end{pmatrix} = \begin{pmatrix} 0 \\ F_{A,b} \end{pmatrix} - \begin{pmatrix} 0 \\ R_{A,b}(T_A, T_B) \end{pmatrix}. \tag{3.11}$$

To reduce the number of degrees of freedom in the problem, we will reduce the internal nodes by expressing them in static and dynamic modes, i.e.

$$T_{A,i} = T_{A,s} + T_{A,d}.$$

The static solution for the internal nodes can be obtained by ignoring the time derivative component and considering the internal dynamics only, Eq. (3.11) becomes

$$K_{A,ii}T_{A,i} + K_{A,ib}T_{A,b} = 0.$$

Therefore, the static solution for the internal nodes can be expressed expressed as

$$T_{A,s} = -K_{A,ii}^{-1}K_{A,ib}T_{A,b}. \tag{3.12}$$

Note that the solution for the internal nodes can be expressed in terms of the solution of the boundary nodes in this scenario.

The dynamic part of the solution can be expressed by the eigenmodes. These can be obtained by solving

$$(K_{A,ii} - \omega^2 M_{A,ii})\phi_i = 0. \tag{3.13}$$

Note that Eq. (3.13) can be written as

$$T_{A,i} \approx T_{A,s} + T_{A,d} = \Psi T_{A,b} + \Phi \hat{T}_{A,i}, \text{ [15]}$$

where $\Psi := -K_{A,ii}K_{A,ib}$ from Eq. (3.12) and Φ contains the eigenmodes from the eigenvalue problem explained in Eq. (3.13). $\hat{T}_{A,i}$ is derived in the same manner as the regular modal decomposition technique.

We want the boundary nodes to stay in full space, i.e. after applying the reduction to the system, the boundary nodes should be the same. To reduce the internal nodes we apply the reduction matrix

$$Z_{CB} = \begin{pmatrix} \Phi & \Psi \\ 0 & I \end{pmatrix},$$

Note that with this reduction matrix, the boundary nodes are kept in full space:

$$\begin{pmatrix} T_{A,i} \\ T_{A,b} \end{pmatrix} = Z_{CB} \begin{pmatrix} \hat{T}_{A,i} \\ T_{A,b} \end{pmatrix}$$

This reduction matrix can be applied in the usual way for the system, i.e. for an arbitrary system this look like

$$Z^T M Z \hat{\dot{T}} + Z^T K Z \hat{T} = Z^T F - Z^T R(T). \tag{3.14}$$

Note that this projection keeps the same problem like we discussed with the POD method. The nonlinear term still needs to be evaluated in full space. Other methods like DEIM can be applied, but for this problem, we introduce also a new concept: a neural network specific for the nonlinear radiation boundary term.

The radiative heat flux is computed element-wise on the surface elements of the radiative boundary. Because the reduction strategy described above retains the boundary temperatures explicitly, there is no need to project the reduced solution back to the full space to recover the temperatures at the boundary nodes. This property motivates the use of a neural network that takes the boundary temperatures directly as input. The following section will explain in detail the neural network used in this thesis.

From Eq. (3.14), it follows that the nonlinear part can be written as

$$Z_{CB}^T R(T) = \begin{pmatrix} \Phi^T & 0 \\ \Psi^T & I \end{pmatrix} \begin{pmatrix} 0 \\ R_{A,b}(T_A, T_B) \end{pmatrix} = \begin{pmatrix} 0 \\ R_{A,b}(T_A, T_B) \end{pmatrix}, \tag{3.15}$$

indicating that the boundary remains unaffected by this reduction matrix. This is consistent with the purpose of the Craig-Bampton method: to retain the boundary degrees of freedom, thereby simplifying the coupling between components. Several methods for further reducing the boundary are known within ASML; see, the study of Van Steen et al [15]. The reduction techniques rely on applying another reduction matrix that only reduces the boundary nodes. The following matrix

$$V = \begin{pmatrix} I & 0 \\ 0 & \Xi \end{pmatrix},$$

keeps the already reduced internal nodes untouched and reduced the boundary:

$$\begin{pmatrix} \hat{T}_{A,i} \\ T_{A,b} \end{pmatrix} = \begin{pmatrix} I & 0 \\ 0 & \Xi \end{pmatrix} \begin{pmatrix} \hat{T}_{A,i} \\ \hat{T}_{A,b} \end{pmatrix}. \tag{3.16}$$

Applying this reduction matrix to the full system in the same manner as Eq. (3.14) and evaluating the nonlinear boundary term, like Eq. (3.15), gives:

$$V^T R(T) = \begin{pmatrix} I & 0 \\ 0 & \Xi^T \end{pmatrix} \begin{pmatrix} 0 \\ R_{A,b}(T_A, T_B) \end{pmatrix} = \begin{pmatrix} 0 \\ \Xi^T R_{A,b}(T_A, T_B) \end{pmatrix}$$

As mentioned above, methods for constructing this matrix are discussed in Van Steen et al. [15]. However, these existing methods do not address our specific challenge: reducing the nonlinear radiation term. The nonlinear function $R(T_A, T_B)$ requires the full temperature vectors of both components as input, since the function is evaluated element-wise. Unfortunately, this structure cannot be modified. This will be discussed in Section 4.2, where we face a similar limitation in ANSYS, where the element-wise definition of the radiation term is fixed and inaccessible. Consequently, methods such as the Discrete Empirical Interpolation Method (DEIM), which rely on structural changes to the nonlinear term, cannot be applied directly.

3.2.2 Neural Network for the Nonlinear Radiative Boundary Term

The existing interface reduction techniques do not resolve our problem. An alternative approach is to replace the nonlinear function R with an approximation that avoids the need for element-wise evaluation. A neural network is an approximation based on data, therefore we do not need access to the nonlinear radiation equations. In this work, we implement a deep feedforward neural network (also known as a multilayer perceptron, MLP) as an approximation for the nonlinear radiation term [23]. These kinds of neural network are a class of Artificial Neural Networks (ANNs), which are data-driven models trying to replicate biological neural systems. They are useful at approximating complex, nonlinear mappings between inputs and outputs when the underlying relationship is unknown or computationally expensive to evaluate.

In our case, a neural network can be trained to approximate the mapping $(T_A, T_B, x) \mapsto R$, where x denotes the spatial coordinate. This neural network creates a function which is a direct mapping from inputs to outputs. Instead of evaluating the full nonlinear finite element formulation at each timestep, the neural network takes as input the boundary temperatures of the radiative surfaces and the position of the waferstage. The output is the corresponding radiative heat flux on the boundary. By including the position as part of the input, we can neglect the expensive costs of the computation of the View Factor Matrix (VFM). This eliminates the need to recompute the VFM during the simulation, which significantly reduces the computational cost of the full simulation. Also, in Eq. (2.3), one can see that the function we want to approximate is dependent on the position through the VFM and the temperatures on the boundary. Therefore, these are the aspects that we want as input in our network.

The architecture of a feedforward neural network consists of layers of nodes (neurons), where each neuron computes a weighted sum of its inputs, applies a nonlinear activation function, and passes the result to the next layer, a detailed explanation can be found in the book of Goodfellow et al [23]. By including multiple layers, the network will be more accurate in approximating the nonlinear relationships between inputs and outputs.

In general, a feedforward neural network with L layers computes its output as

$$y = H^{(L)}\big(W^{(L-1)} H^{(L-1)}(\dots H^{(1)}(W^{(0)} x + b^{(0)}) \dots) + b^{(L-1)}\big) + b^{(L)}, \tag{3.17}$$

where x is the input vector, W is the the weight matrix and b is the bias vector. $H(\cdot)$ is a nonlinear activation function (e.g., ReLU, sigmoid, tanh) [23].

While other neural network models such as Convolutional Neural Networks (CNNs) or Recurrent Neural Networks offer additional capabilities. They all excel for different uses: CNNs work well with images or videos, and RNNs are suited for learning from past inputs in a sequence. However, they can be harder to run or need more structured data. The network used here is simpler, easier to understand, and still able to learn the nonlinear radiation relation from the simulation data.

Pseudoalgorithm of the Neural Network

In our problem, the goal is to approximate the nonlinear mapping

$$f : \underbrace{(T_{A,rad}, T_{B,rad}, x)}_{\mathbf{x}\in\mathbb{R}^d} \mapsto \underbrace{q_{\text{rad}}}_{\mathbf{y}\in\mathbb{R}^n},$$

where $T_{A,rad}$ and $T_{B,rad}$ represent temperature values on the radiative boundaries A and B, and x denotes spatial positioning of a moving geometry. The output q_{rad} is the nodal radiative heat flux at a designated radiation boundary (A or B). For both subsystems, we will create separate neural networks.

Another advantage of the neural network is regarding the input of the neural network. The input is a vector of the temperature on the nodes of the radiative boundary. The Craig-Bampton method does not reduce the boundary nodes, they remain untouched, i.e. in full space. This allows us to extract the full temperature vector on the radiative boundary without mapping the reduced temperature vector back to full space. This saves a matrix-vector multiplication in every timestep.

Feedforward Network Structure The model is a feedforward neural network with two hidden layers. More layers usually make the network more flexible and better able to capture complicated relationships, but they also make training slower and increase evaluation costs. Using too few layers, on the other hand, may lead to a model that does not capture the nonlinear behavior. Let $x \in \mathbb{R}^d$ be an input vector, and define the neural network mapping $f(x;\theta)$ as:

$$\begin{aligned} Z^{(1)} &= H\left(W^{(1)}x + b^{(1)}\right), \\ Z^{(2)} &= H\left(W^{(2)}Z^{(1)} + b^{(2)}\right), \\ \hat{y} &= W^{(3)}Z^{(2)} + b^{(3)}, \end{aligned}$$

where $W^{(k)}$ and $b^{(k)}$ are the weights and biases at layer k, $H(\cdot)$ is the activation function and $\hat{y}$ is the network's prediction of the radiative flux.

In this work, we choose the ReLU (Rectified Linear Unit) function,

$$H(x) = \max(0, x),$$

for both hidden layers due to its simplicity and ability to capture nonlinear dynamics. The training of the neural network consists of adjusting the parameter set $\theta = \{W^{(k)}, b^{(k)}\}$ to minimize a loss function that measures the difference between predictions and true outputs. A pseudo-algorithm for the process can be found in Algorithm 2.

Algorithm 2 Algorithm for the development of the Neural Network for predicting the radiation flux on the radiative boundary of component A.

Require: Snapshots: $T_A, q_A \in \Gamma_{\text{rad},A} \in \mathbb{R}^{g\times 1}$, $T_B \in \Gamma_{\text{rad},B} \in \mathbb{R}^{h\times 1}$ and $\mathbf{x} \in \mathbb{R}^{s\times 1}$ at times t_k:

1: **Build dataset** $D = \{(x, y)\}$ with $\mathbf{x}_k = [\, T_A(t_k),\ T_B(t_k),\ \mathbf{x}_k\,] \in \mathbb{R}^{(g+h+s)\times 1}$, $y_k = [q_A(t_k)] \in \mathbb{R}^{g\times 1}$.
2: **Split** D into train/test.
3: **Define network** $f_\theta : \mathbb{R}^{\text{in}} \to \mathbb{R}^{\text{out}}$ with n hidden layers and activation function.
4: **Train** by minimizing $L(\theta) = \text{MSE}(f_\theta(X), Y)$ on the training set.
5: **Test** on the test set.
6: **Validate**: given new (T_A, T_B, x) on $\Gamma_{\text{rad},A}$, output $\widehat{q}_A = f_{\theta^*}(x)$.

Jacobian The FOM of this problem is computed using the Newton method. This includes computing the Jacobian of the nonlinear radiation term. This was a part of the element wise function, which besides the radiation also created the Jacobian. This Jacobian can not be computed anymore since we do not have a known function anymore. This issue arose during the implementation, when we already noticed that the contribution to the Jacobian of the nonlinear term $R(T)$ is very small in contrast to the linear contributions. This suggested we can neglect the Jacobian of the nonlinear part, as the Newton method still converges if the Jacobian is not exact; it just needs some additional steps.

Note that this reduction approximates an element-wise function with a single input-output function. This is not a reduction based on a projection based method or it does not reduce the dimension of the system, but it does reduce the time of the simulation since it is a simple input-output function. Furthermore, it does not require the full temperature vector on the entire domain, but solely on the boundary. As they are kept in full space, we do not need to map the reduced temperature vector back to full space. Also, the computational expensive view factor matrix computations at each timestep can be neglected.

3.3 Linearization of the nonlinearity

As explained, linearization of the boundary conditions can be very efficient to reduce the system, as it enables the use of one of the many linear reduction techniques. In our continuous heat equation for component A, we want to linearize the radiative boundary condition

$$q_{\text{rad}} = c(T_A^4 - T_B^4).$$

Linearize around reference temperatures $\bar{T}_A$ and $\bar{T}_B$:

$$\begin{aligned} T_A^4 &\approx \bar{T}_A^4 + 4\bar{T}_A^3\,(T_A - \bar{T}_A), \\ T_B^4 &\approx \bar{T}_B^4 + 4\bar{T}_B^3\,(T_B - \bar{T}_B), \\ \Rightarrow \quad T_A^4 - T_B^4 &\approx (-3\bar{T}_A^4 + 3\bar{T}_B^4) + 4\bar{T}_A^3\,T_A - 4\bar{T}_B^3\,T_B. \end{aligned}$$

Hence

$$q_{\text{rad}} \approx c(-3\bar{T}_A^4 + 3\bar{T}_B^4) + 4c\,\bar{T}_A^3\,T_A - 4c\,\bar{T}_B^3\,T_B.$$

Hence the Robin-like boundary condition is:

$$q_{\text{rad}} \approx h_A\,T_A - h_B\,T_B + q_0, \qquad h_A = 4c\,\bar{T}_A^3,\ \ h_B = 4c\,\bar{T}_B^3,\ \ q_0 = c(-3\bar{T}_A^4 + 3\bar{T}_B^4).$$

For component B, we follow the same analogy.

Implementing this in the full system, given in Eq. (3.5), gives

$$\begin{aligned} M_A\dot{T}_A + (K_A + L_A)T_A - L_{AB}T_B &= F_A \\ M_B\dot{T}_B + (K_B + L_B)T_B - L_{BA}T_A &= F_B, \end{aligned} \tag{3.18}$$

where $L_A = h_A$, $L_{AB} = h_B$ and q_0 is incorporated in the right-hand side (RHS). L_B and L_{BA} are derived from following the same analogy for component B.

Note that within this constant c, the view factor is incorporated. As explained in Section 2.3, a Robin boundary condition appear on the RHS of the discretized FEM formulation and in the stiffness matrix. The view factor is not a constant, it will vary in time if the component B moves. Resulting in a varying stiffness and force vector in time.

This method is very tricky as we are testing it on an idealized version of the real world. The linearization can be much more accurate in this test problem compared to the actual model, where the movement is less smooth and there are gaps and corners in the geometries. Also, we need a steady state solution, or reference solution to linearize around, but as we already mentioned, we do no have that in our problem.

3.4 View Factor Matrix Reduction

In the test model presented in Section 1.4, the view factor matrix (VFM) is an $(n+m)\times(n+m)$ matrix, where:

- n denotes the number of boundary elements on geometry A,
- m denotes the number of boundary elements on geometry B.

The VFM depends directly on the spatial discretization of the domain. As the FEM mesh becomes finer, the number of boundary elements increases, and the VFM dimension grows accordingly. In addition, since the wafer stage moves in time, the view factors between boundary elements vary at each time step. As a result, the VFM must be recomputed dynamically, making this step computationally expensive.

In a numerical scheme for a real model, evaluating the VFM requires looping over all boundary element pairs, checking whether they are mutually visible (i.e., not obstructed), and computing the corresponding view factor. Assuming that all elements can see each other, the computational cost scales as:

$$\mathcal{O}\left((n+m)^2 + (n+m)\right)$$

In the idealized version, i.e. the test problem, we already know that every element sees all the elements of the other geometry and none on their own geometry, so the costs for computing the VFM will be lower. In the real model, elements on the same boundary can still see each other due to gaps.

One method to reduce the cost of view factor computation is to use interpolation of precomputed matrices. The idea is as follows:

1. Compute and store the VFM for a discrete set of relative positions between geometries A and B. Let these positions be denoted by $\{p_1, p_2, \ldots, p_N\}$, and their corresponding matrices be $\{VF_1, VF_2, \ldots, VF_N\}$.

2. During the simulations, when the wafer stage is at a new position $p \notin \{p_i\}$, approximate the VFM $VF(p)$ using interpolation, e.g.,

$$VF(p) \approx \sum_{i=1}^{N} w_i(p) VF_i$$

where $w_i(p)$ are interpolation weights satisfying $\sum_i w_i = 1$ and $w_i \geq 0$. We can use linear interpolation, but also other methods like cubic interpolation depending on how the view factor changes.

This method is particularly effective when the wafer moves in a predictable or structured manner, allowing for a structured sampling, which is the case.

Another way to reduce computational cost is by decreasing the size of the VFM. This is achieved by grouping boundary elements into clusters based on geometric similarity or similar radiative behavior.

For example, suppose we have:

- $n = 30$ elements on boundary A,
- $m = 20$ elements on boundary B.

The full VFM would be of size 50×50. If we identify 4 regions on A and 6 regions on B with internally similar view factor patterns, we can replace each group with a representative node. This reduces the VFM size to 10×10, leading to a reduction in computational costs. This is a widely used technique. However, due to the spatially varying waferstage, we cannot define any fixed regions since it will always change over time.

This chapter included the mathematical formulations of the reduction techniques used in this thesis. The next section will test the techniques on the test problems and provide the results of each technique.

Chapter 4

Numerical Results and Analysis

In this section, we present the results of applying several model order reduction (MOR) techniques to the nonlinear thermal system introduced in the previous chapters. The primary objective is to evaluate the effectiveness of various methods, ranging from classical projection-based techniques to data-driven neural networks, in capturing the dominant dynamics of the full-order model while significantly reducing computational complexity. This chapter will discuss the POD method first, followed by the DEIM which accelerates the computations of the nonlinear term. The chapter continues with the modal decomposition for the linearized system and the modal decomposition for the nonlinear system. This forms the basis for the last part of the chapter, namely substructuring. Substructuring is used to split the internal and boundary (interface) dynamics. This forms eventually the basis for our proposed algorithm. For each method, we analyze the accuracy of the reduced-order model and the simulation time in comparison with the full-order model (FOM).

The numerical results are calculated over a specific amount of time with a stepsize Δt. Using an implicit time integration scheme, namely Backward Euler, allowed us to pick any Δt, this is beneficial to compute the results over a longer period. The Newton-Rhapson method was used to iteratively approximate the time-dependent radiation term each timestep. The initial condition of both geometries and the ambient temperature is $295K$ ($22°C$) throughout the thesis. The ambient temperature is important for the convection boundary conditions which serves as cooling elements for the geometries. The cooling elements are applied at the top surface of geometry A and at the bottom surface of geometry B. The heat source is applied as an inward heat flux at the bottom surface of geometry B. Furthermore, the top surface of geometry B and the bottom surface of geometry A are the only radiation boundaries.

The movement is simulated with a sinusoidal oscillation from left to right, meaning that multiple measurements are required within one full cycle to accurately capture the dynamic behavior. To obtain a periodic steady-state solution, the simulation must span a sufficiently long time interval. Slow movements are easier to capture over longer time intervals, as larger timesteps can be used. As previously mentioned, to capture a sinusoidal motion accurately, multiple measurements within a single cycle are required. For slow behavior with big timesteps, this results in long cycle durations. Conversely, capturing fast motion requires smaller timesteps. For example, to accurately resolve a cycle, at least 10 measurements should be taken within one period, meaning the cycle time should be approximately:

$$\Delta t \times 10$$

where Δt is the chosen timestep. Table 1 shows the parameters used for the different motions. Fast motion is simulated with $\Delta t = 1$, hence a full cycle is completed within 10 seconds. Slow motion is modeled with timesteps of 80 seconds. This is the same throughout the rest of the chapter.

4.1 POD

The Proper Orthogonal Decomposition (POD) method, as introduced in Section 3.1.2, is a data-driven approach that constructs a reduced basis from simulation data. The corresponding Galerkin projection formulation for this method is presented in Eq. (3.8). The workflow is as follows:

1. Collect data from simulations over longer periods.
2. Apply the Singular Value Decomposition (SVD) on the data to obtain modes.
3. Plot the singular values to rank the importance of the modes.
4. Select the number of dominant modes to construct a projection basis.

Following this workflow yields a reduction matrix $\Phi \in \mathbb{R}^{r \times n}$, where r denotes the number of modes used in the reduction and n the total number of nodes in the full-order system.

For the POD method for our test problem, the two components A and B, we can apply two different approaches:

1. A single basis for both components.
2. Two separate bases; one for each component

An overview is given in Figure 4.1.

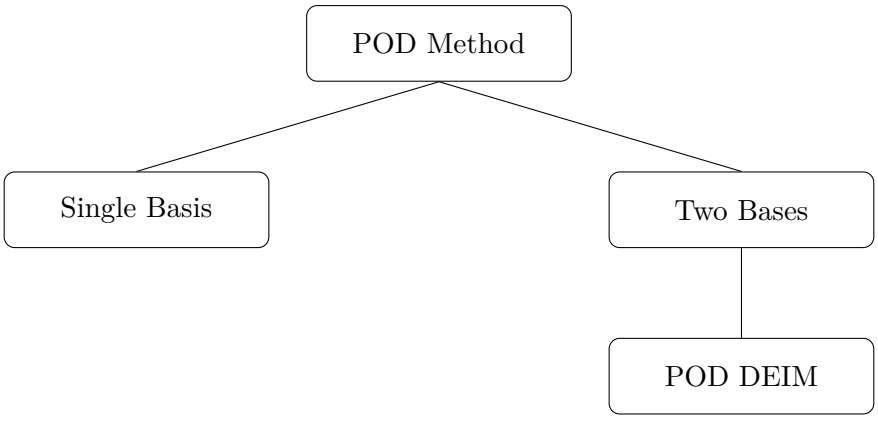


Figure 4.1: Overview POD strategies. As most industries uses component-wise reduction to be able to assemble them in different subsystemsyes, we will apply DEIM on the two bases strategy only.

In the first approach, **a single basis** is constructed for the entire coupled system. The snapshot matrices of both components are concatenated, i.e. $T = \begin{pmatrix} T_A \\ T_B \end{pmatrix}$ and the Proper Orthogonal Decomposition (POD) is applied globally. This results in one reduced basis Φ that spans the dynamics of the full system. For the reduction one can recall Eq. (3.2), i.e.

$$V_r^T M V_R \frac{da(t)}{dt} + V_r^T K V_r a(t) = V_r^T F - V_r^T R(V_r a(t)), \tag{4.1}$$

where $M = \begin{pmatrix} M_A & 0 \\ 0 & M_B \end{pmatrix}, K = \begin{pmatrix} K_A & 0 \\ 0 & K_B \end{pmatrix}, F = \begin{pmatrix} F_A \\ F_B \end{pmatrix}$ and $R = \begin{pmatrix} R_A(T_A, T_B) \\ R_B(T_A, T_B) \end{pmatrix}$.

This method captures correlated behavior between components and can be advantageous when the interaction between geometries is strong or tightly coupled.

Alternatively, each component can be treated as a separate subsystem. **Two independent bases** Φ_A and Φ_B are computed for each geometry based on its own snapshots, T_A and T_B. The overall reduced model is then assembled by coupling the individual reduced models. This is done in the same manner as Eq. (4.1), but with $\Phi = \begin{pmatrix} \Phi_A & 0 \\ 0 & \Phi_B \end{pmatrix}$.

This strategy allows for reusing the basis of the components in different configurations/simulations. This makes it suitable if this is a subsystem of a bigger structure.

The choice between a single or component-wise basis has implications on computational efficiency, accuracy, and flexibility. While a global (single) basis may better capture cross-domain dynamics, the component-wise approach supports substructural designs and is often preferable for systems with structured coupling.

4.1.1 Comparison Strategies

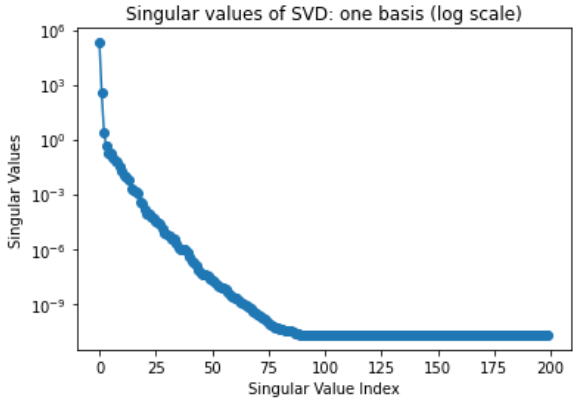
While constructing the reduced-order model, one can analyze the singular values obtained from the POD method to examine how much energy each mode captures. By plotting the decay of these singular values, we gain insight into the modal importance and it will allow us to determine how many modes are required to approximate the original system with sufficient accuracy.

A rapid decay in the singular values indicates that a small number of modes are sufficient to capture the dominant dynamics of the system. Conversely, a slow decay suggests that the system exhibits more complex behavior, requiring a larger number of modes for an accurate reduced representation. The number of modes r is typically chosen such that a predefined energy threshold (e.g., 99% of the cumulative energy) is captured.

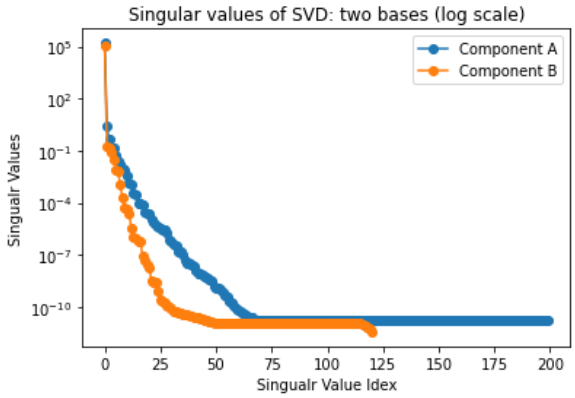
From Figure 4.2, we can analyze the singular values for both the global and component-wise POD approaches. One can obtain that in Figure 4.2, the clear drop in the graph includes more modes then the graphs for the separate subsystems (the *corner* is more to the right for the single basis). This is because the coupling terms introduce additional correlations across the domains. In essence, the coupling creates new patterns in the data. Therefore, the POD of the full coupled system must account for both the internal dynamics of each subsystem and the interaction effects between them, requiring more modes to accurately capture the dominant behavior.

4.1.2 Results POD Strategies

The results of the FOM are presented in Figure 4.3. In this figure, we show the spatial temperature at a specific timestep. We will compare the accuracy of reduced-order models using either a single global basis or separate bases for each component. In this part of the chapter, we will show spatial heat patterns as results. This is to give an idea how Reduced Order Modeling in this system works and to give a visualization of the heat pattern. Later on we will only show the temperature evolution over time evolutions. This gives a better overview of the results throughout the simulation.



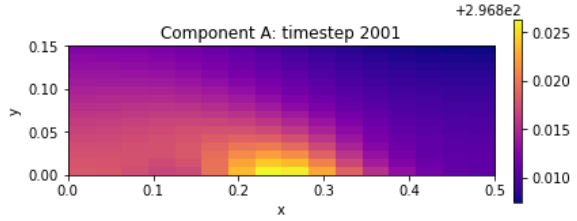
(a) Singular values for a single basis



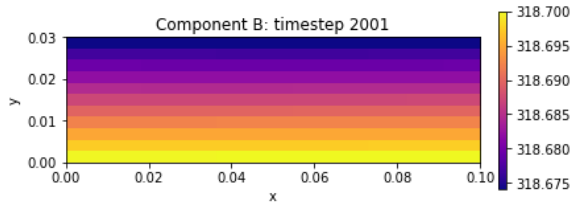
(b) Singular values for two separate bases.

Figure 4.2: Singular values of the POD method for a single or two bases. The singular values come from applying an SVD method to the data matrix.

The material properties and mesh are identical across all simulations in this thesis. The velocity of component B varies: slow motion allows larger time steps (faster runtime), while fast motion requires smaller time steps (slower runtime). We report the chosen velocity only when varying it leads to different results. Otherwise, we omit it for simplicity. The wafer stage motion is prescribed as a sinusoidal trajectory. To represent this motion in the simulation, the cycle is discretized into 11 timesteps. Hence, larger timesteps represents slower motion.



(a) Component A: FOM



(b) Component B: FOM

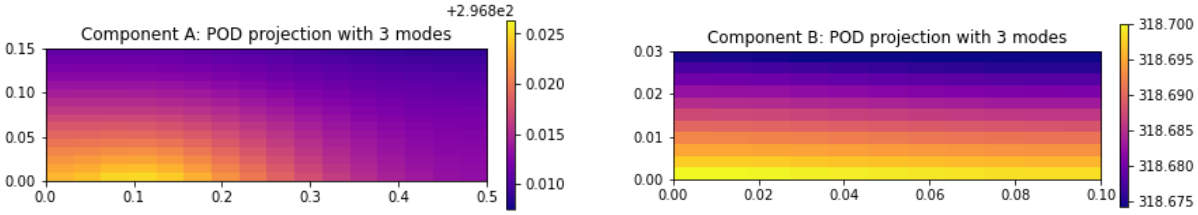
Figure 4.3: Results of the FOM for a specific timestep

In Figure 4.3, one can obtain a clear pattern: the wafer stage moves from left to right, leaving a thermal trail in its path.

One global basis. We first consider the case where one global basis is used for both components. The effect of the number of retained modes in the reduction basis Φ is illustrated in Figures 4.4 and 4.5, which show the temperature fields at a specific time index using three and five modes, respectively.

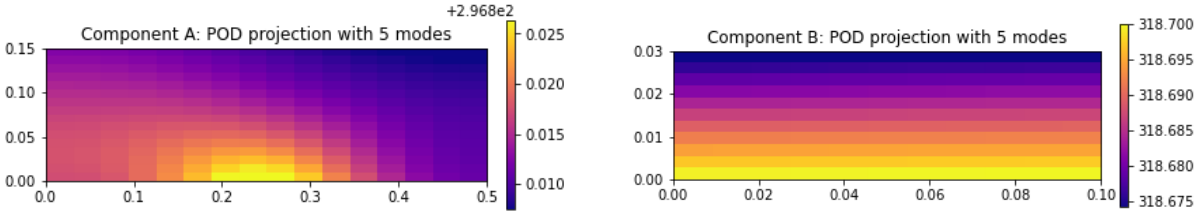
As the number of modes increases, the reduced-order solution converges toward the full-order reference. This is seen in the decreasing error in Figure 4.13. The improvement occurs because additional modes account for higher-order contributions to the system's dynamics, which are neglected in lower-dimensional models. Note that from these results, the results for including five modes seem to already be very accurate.

To evaluate the accuracy in time, we also consider the temperature over time at two representative nodes: one located at the boundary and one in the interior of component A. For the



(a) Component A: POD with 3 modes **(b)** Component B: POD with 3 modes

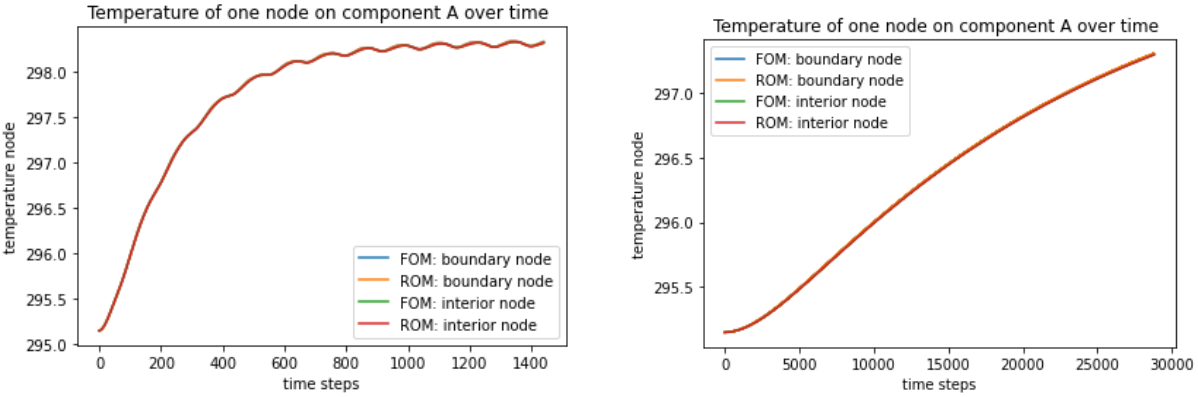
Figure 4.4: POD method with a single basis, conserves small part of the energy.



(a) Component A: POD with 5 modes **(b)** Component B: POD with 5 modes

Figure 4.5: POD method with a single basis, conserves part of the energy.

following analysis, we consider 7 modes in the reduced bases. To give an idea of the movement of the waferstage, Figure 4.6 plots the results of the temperature evolution over time. When component B moves relatively slow, see Figure 4.6a, the results are clear and easy to interpret. The

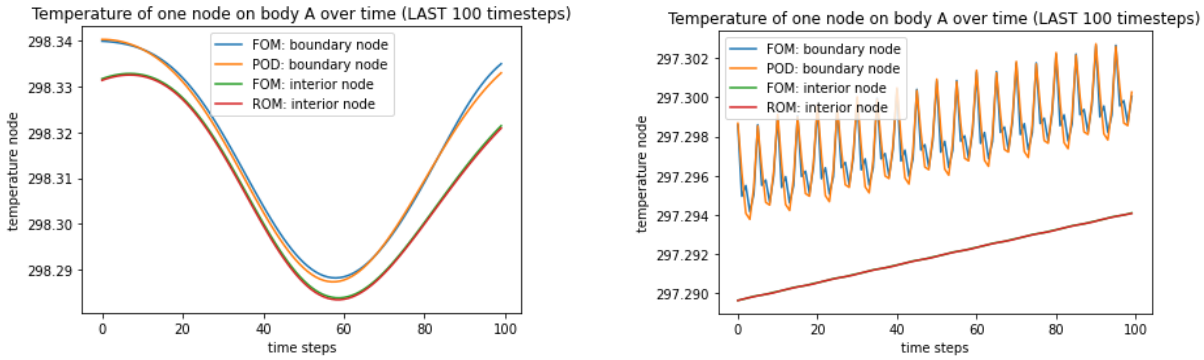


(a) Slow motion. $\Delta t = 100$. **(b)** Fast motion.$\Delta t = 1$.

Figure 4.6: Temperature over time on a boundary and interior node. The first figure gives the results for a slow motion of component B and the second gives the results for fast motion. Both result has the same heat source, an inward flux of 200 W/m^2. The eigenbasis consists of a single basis with 7 modes.

temperature of the boundary and internal node are very similar over time. However, this does not reflect the actual situation we are studying. In reality, the waferstage moves at a relatively high speed. Therefore, it is important to focus on fast motion as well. To accurately capture fast motion, small timesteps are required, which significantly increase the total simulation time. Due to the high speed, the temperature differences between the nodes become smaller, making it harder to observe clear variations in the full simulation results, see Figure 4.6b. For this reason, we often zoom in on a window of 100 timesteps throughout this thesis to better visualize the effects, see Figure 4.7. Note that the boundary node shows stronger deviations due to its direct contact with the nonlinear radiation interface. In contrast, the interior node remains more

stable, it experiences less deviation for faster motion, which is logical.



(a) Slow motion.

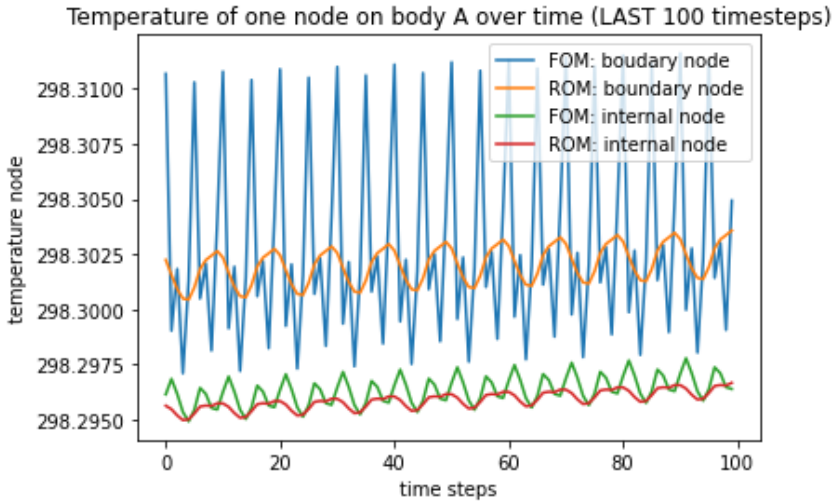
(b) Fast motion.

Figure 4.7: Temperature over time on a boundary and interior node for the last 100 timesteps of the full simulation. The first figure gives the results for a slow motion of component B and the second gives the results for fast motion. Both result has the same heat source, an inward flux of 200 W/m^2. The eigenbasis consists of a single basis with 7 modes.

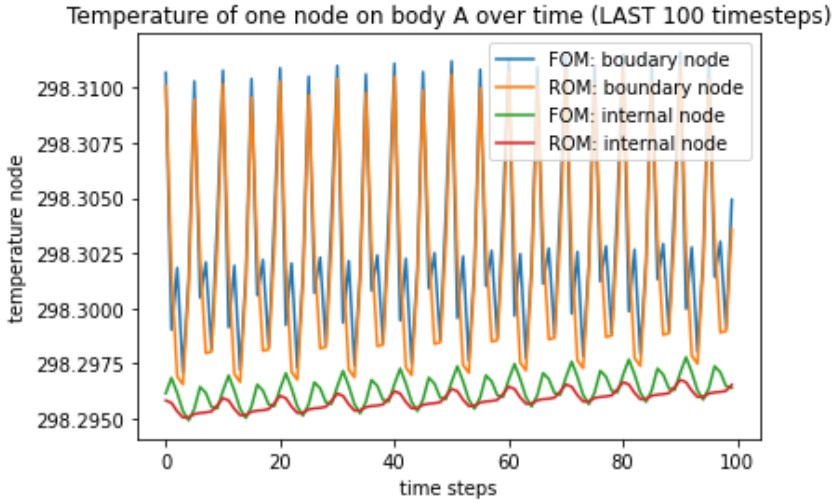
To visualize the effect of taking more modes into account, the last 100 timesteps of the full simulation for different numbers of modes with a relatively fast motion are given in Figure 4.8. The accuracy improves with more modes, as expected.

Figure 4.13 presents the error of the reduced model (with a single basis) as a function of the number of modes used. We again see a reduction in error with increasing modes. The dominant contribution to the error originates from component A, reflecting its stronger nonlinear behavior, hence we will focus on the results for component A throughout this thesis. Also throughout this thesis we calculate the error at a given time t_k: let $T_k \in \mathbb{R}^N$ be the reference (FOM) temperature vector and $\widehat{T}_k$ the reduced one. We calculate the *relative* L^2 error with:

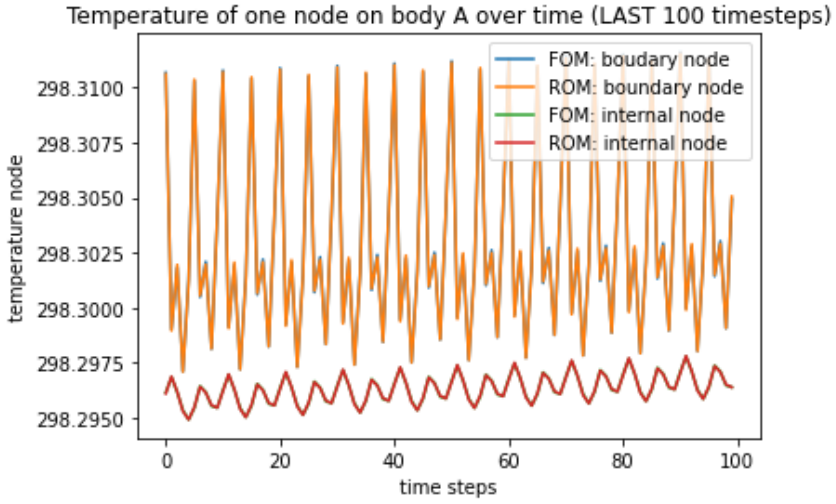
$$e_k^{\text{rel}} = \frac{\|T_k - \widehat{T}_k\|_2}{\|T_k\|_2}. \tag{4.2}$$



(a) Temperature over time for 3 modes.



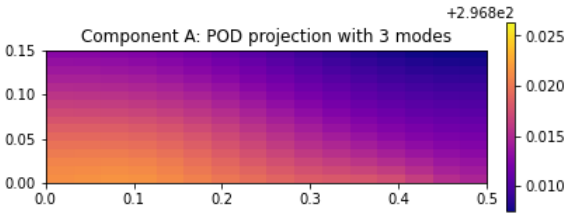
(b) Temperature over time for 5 modes.



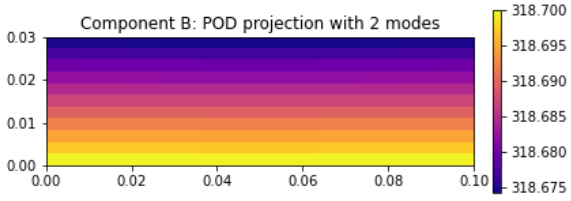
(c) Temperature over time for 10 modes.

Figure 4.8: Temperature over time of a boundary and an internal node in component A using **one basis** with varying number of modes.

Two separate bases. Next, we investigate a reduced model using two separate bases: one for component A and one for component B. Figure 4.11 shows the result with 60 modes for component A and 30 for component B - which contain approximately all energy (see Figure 4.2b) - while Figures 4.9 and 4.10 explore the effect of using fewer modes. Again, the accuracy improves with more modes.

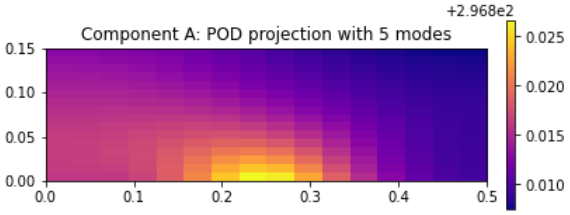


(a) Component A: POD with 3 modes.

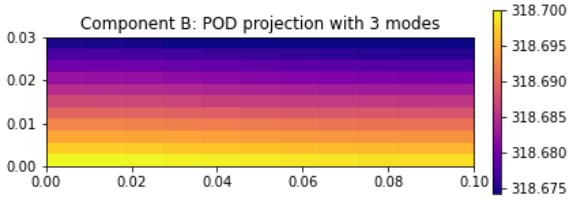


(b) Component B: POD with 2 modes.

Figure 4.9: POD method with two bases, conserves small part of the energy.

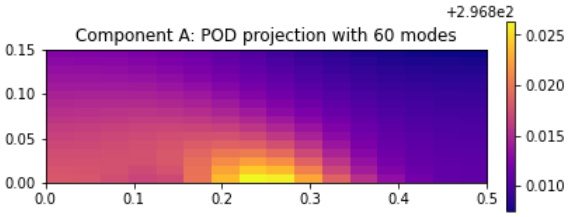


(a) Component A: POD with 5 modes.

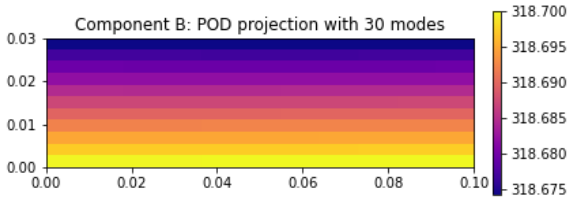


(b) Component B: POD with 3 modes.

Figure 4.10: POD method with two bases, conserves part of the energy.



(a) Component A: POD with 60 modes.



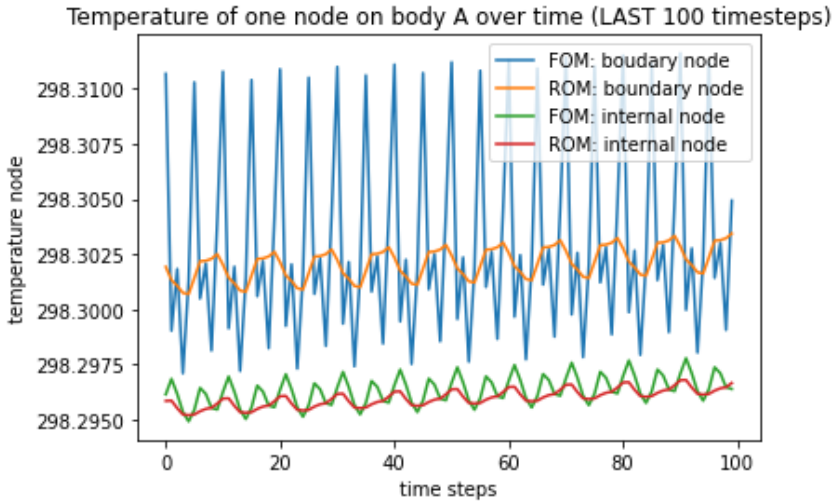
(b) Component B: POD with 30 modes.

Figure 4.11: POD method with two bases, conserve almost all the energy.

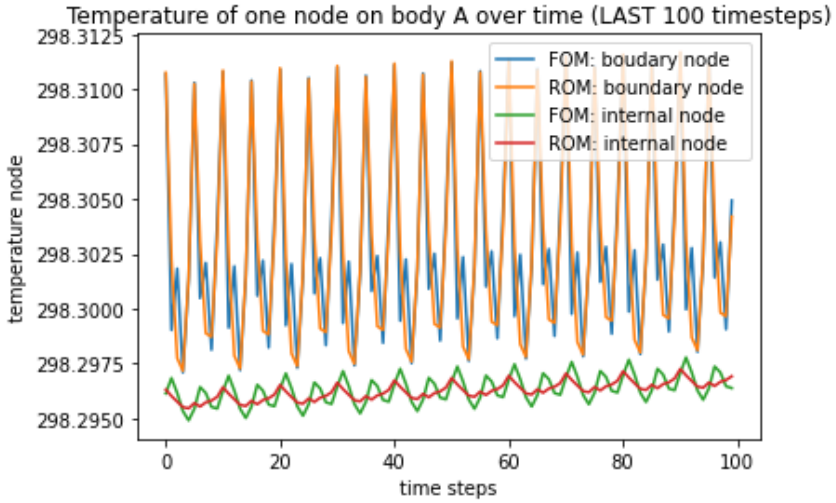
To assess time-dependent behavior, Figure 4.12 displays the temperature over time for the same representative nodes as before. The reduced solution closely follows the FOM when taking more modes into account.

Finally, Figure 4.13 shows the error decay with respect to the number of modes per component. The error is again computed as Eq. (4.2). As with the single global basis, a way to determine the required number of modes is to impose a maximum acceptable error threshold for the reduced model.

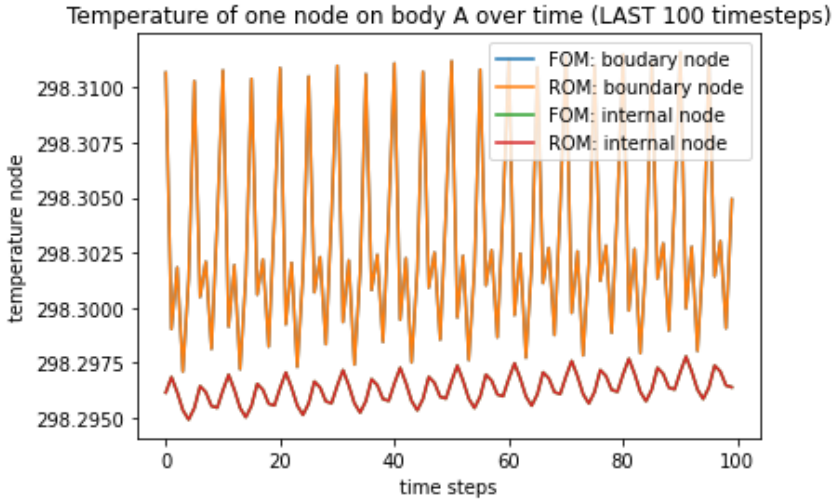
In conclusion, the POD method provides an effective model reduction technique, producing accurate results with a significantly lower-dimensional representation. However, as we know, its main limitation lies in the treatment of nonlinear terms, which still require evaluation in the full-order space. This drawback motivates the use of additional reduction strategies such as the Discrete Empirical Interpolation Method (DEIM), which enables efficient approximation of nonlinearities and further accelerates the reduced-order model.



(a) Temperature over time for 3 modes.



(b) Temperature over time for 5 modes.



(c) Temperature over time for 10 modes.

Figure 4.12: Temperature over time of a boundary and an internal node in component A using **two bases** with varying number of modes.

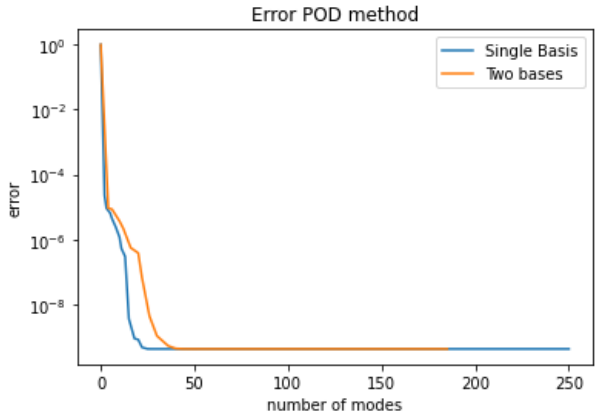


Figure 4.13: L2 error (Eq. (4.2)) of component A for the two strategies for the POD method based on a timestep compared to the FOM. The number of modes are the total number of modes used for both systems, i.e. when we use two separate bases, we add the number of modes in both systems.

4.2 POD DEIM

In the previous sections, we have used Proper Orthogonal Decomposition (POD) to construct reduced-order models of the full system. While the POD method is very effective for reducing the dimensionality of the linear part of the system, it does not necessarily reduce the evaluation of the nonlinear terms, this is also shown in Section 3.1.3. This section also shows how the Discrete Emperical Interpolation Method (DEIM) can be combined with the POD method to resolve this issue. The previous section showed that the system can be approached in two ways: using one global basis or two separate bases. Since both methods gave sufficient results, we now focus on using two separate bases. This fits better with a subsystem setup, which is made up of different connected subsystems. Using a separate basis for each part gives more flexibility and matches how the system is actually built.

In Section 3.1.3, we saw that applying the algorithm leads to evaluating $P^T R(\Phi_r a(t))$. However, this does not resolve the issue that we are facing. DEIM uses a trick: it swaps the order of evaluation and projection, computing $R(P^T \Phi_r a(t))$ instead. This means the nonlinear function is now evaluated only at selected locations. The full function is then approximated by interpolating from these values, reducing the computational cost significantly.

DEIM addresses this limitation by approximating the nonlinear term using a separate reduced basis and a small number of interpolation points. This enables a reduction in computational cost for the nonlinear part. This approach allows the nonlinear term to be evaluated only at a few selected entries, significantly reducing the cost of computing $R(T_A, T_B)$ during simulations.

4.2.1 Results DEIM

In the previous section, we saw that the results of the POD method for both strategies give promising results. The error for two separate bases is slightly larger, as seen in Figure 4.13. The absolute maximum difference in error is bounded by $10e^{-3}$, this can be compensated by taking more modes into account. To fit in the way of working of the industry (where it is common to have big assemblies and different subssystems), this result is acceptable. Therefore, the following results will only focus on a system with two separate bases.

In Figure 4.14, the temperature over time of a node on the boundary of component A can be obtained for the FOM, POD method and POD-DEIM method. The error is plotted in Figure

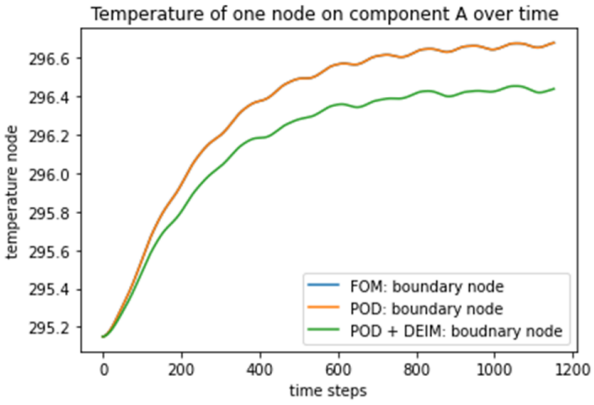


Figure 4.14: Temperature over time of a node in component A. POD-DEIM compared to the FOM and POD method. velocity ....

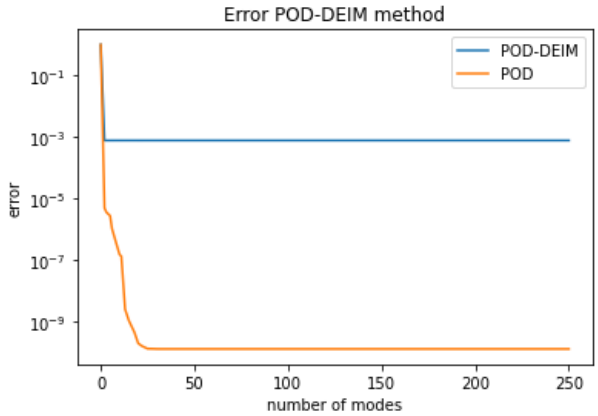


Figure 4.15: Error of the POD-DEIM method versus the number of eigenmodes. The error of the temperature of the last timestep compared to the FOM. The error of the POD method with two bases is also given as a reference. The L2-error is computed as (Eq. (4.2)).

4.15 and is computed as Eq. (4.2). It is clear that the POD DEIM offset is significant in comparison with the POD method. In this simulation, we took 50 POD modes, so the basis contains approximately all the dynamics, as seen in 4.13. Therefore, the error that occurs in this simulation comes only from the DEIM basis. For the POD DEIM method, we took as many interpolation points as the number of nodes on the nonlinear boundary, this would result in the most optimal solution. It is obvious that the error grows as time increases. This means that the error is accumulative, i.e. for every time integration step, the error is small, but after enough timesteps, it becomes significant. There are a few assumptions we make during the approximation with DEIM, e.g. we interchange the interpolation matrix and the radiative FEM term. The offset can come from one of these assumptions.

To conclude, the POD-DEIM method has some promising results for this problem in terms of reducing the evaluation of the nonlinear term, however it does give an offset in the results. To implement the POD-DEIM method, it is necessary to change the code of the nonlinear radiation term. However, this is not possible in the ANSYS environment within ASML. ANSYS does not provide information on the nonlinear term, therefore we cannot extract and adapt these calculations. Also, POD-DEIM is a data-driven approach which requires more data in real life complex problems in comparison to this test problem. This can be a bottleneck for this method.

Taken these disadvantages into account, we will try if there are other methods which also give promising results.

4.3 Modal Decomposition

To analyze and reduce the system dynamics, we consider two main approaches for eigenvalue (modal) decomposition:

1. **Linearized System:** where the nonlinear terms are linearized around a equilibrium state.
2. **Full Nonlinear System**

Each of these approaches can be applied using the same two possibilities as the previous section:

1. A **single basis** for the coupled system as a whole.
2. **Separate bases** for each subsystem individually.

An overview is given in Figure 4.16.

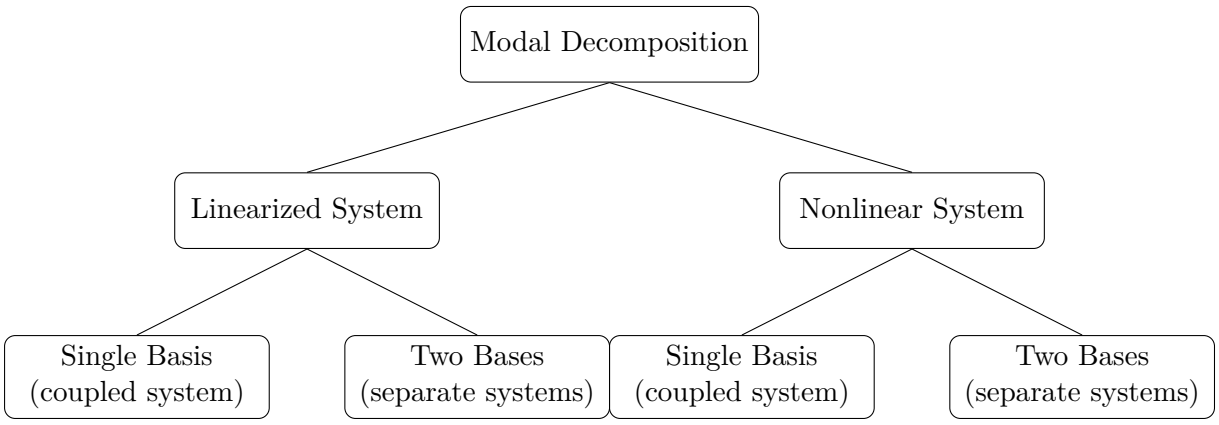


Figure 4.16: Overview modal decomposition strategies.

4.3.1 Linearized system

We begin by focusing on the linearized system. The linearization is carried out as described in Section 3.3, that is

$$\begin{aligned} M_A\dot{T}_A + (K_A + L_A)T_A - L_{AB}T_B &= F_A \\ M_B\dot{T}_B + (K_B + L_B)T_B - L_{BA}T_A &= F_B, \end{aligned} \tag{4.3}$$

which results in the following system

$$\begin{pmatrix} M_A & 0 \\ 0 & M_B \end{pmatrix} \begin{pmatrix} \dot{T}_A \\ \dot{T}_B \end{pmatrix} + \begin{pmatrix} K_A + L_A & -L_{AB} \\ -L_{BA} & K_B + L_B \end{pmatrix} \begin{pmatrix} T_A \\ T_B \end{pmatrix} = \begin{pmatrix} F_A \\ F_B \end{pmatrix} \tag{4.4}$$

Note that linearization usually requires an equilibrium state to expand around. To achieve such a state, we keep component B fixed so that a steady-state solution exists — one that does not change over time. If we linearize the real moving model, we get a time-dependent stiffness matrix due to the changing view factor and we would not have a steady state solution, instead the system will have a periodic steady state solution due to the periodic movement.

In this case, the view factor stays constant. The linearized model serves not only as a reference for the nonlinear modal decomposition, but also as a tool for gaining insight into the system's dynamics. Furthermore, it provides a basis for validating other methods and assessing uncertainty. In case unexpected behavior appears during simulations or model reduction, the linearized system offers a reliable fallback to help determine whether nonlinear effects are responsible for the observed issues.

Coupled system, one basis. The eigenvalue decomposition is discussed in Section 3.1.1. It creates a basis by solving the eigenvalue problem in Eq. (3.3), i.e. solving

$$K\phi_i = \lambda_i M\phi_i. \tag{4.5}$$

Solving this problem for a single eigenvalue is equivalent to solving this problem with

$$M = \begin{pmatrix} M_A & 0 \\ 0 & M_B \end{pmatrix}$$

and

$$K = \begin{pmatrix} K_A + L_A & -L_{AB} \\ -L_{BA} & K_B + L_B \end{pmatrix}.$$

After applying the eigenvalue problem, we obtain a set of eigenvectors ϕ_i and eigenvalues λ_i. We assemble the eigenvectors into the matrix $\Phi = [\phi_1, \phi_2, \ldots, \phi_n]$ and define the reduced temperature vector q by the transformation:

$$T = \Phi q. \tag{4.6}$$

Substituting into the original equation, we obtain:

$$M\Phi\dot{q} + K\Phi q = F. \tag{4.7}$$

Using the Galerkin projection, i.e. multiplying both sides from the left by $\Phi^\top$, we get:

$$\Phi^\top M\Phi = I, \quad \Phi^\top K\Phi = \Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n),$$

we obtain the decoupled system:

$$\dot{q} + \Lambda q = \Phi^\top F. \tag{4.8}$$

The method of selecting certain eigenvectors, ϕ_i, in the reduced matrix Φ can differ. A very common method is to consider the smallest r eigenvalues, see Figure 4.17, where the eigenvalues are ordered from the smallest eigenvalue as the first index and the largest eigenvalue as the last index. It is well known that eigenvalues with the smallest magnitudes correspond to the system's long-term behavior and therefore tend to dominate the systems behavior over time. In the figure, the eigenvalues are plotted on a logarithmic scale, where it becomes clear that the first eigenvalues contain the most energy. This makes it natural to consider an eigenbasis formed from the eigenvectors associated with the smallest r eigenvalues. This can, for example, be done in the same manner as in Section 4.1, by defining a certain threshold to determine the percentage of the energy conserved in the reduced model. The error of this ROM is shown in Figure 4.19, where the error is based on the comparison with the FOM at a certain timestep.

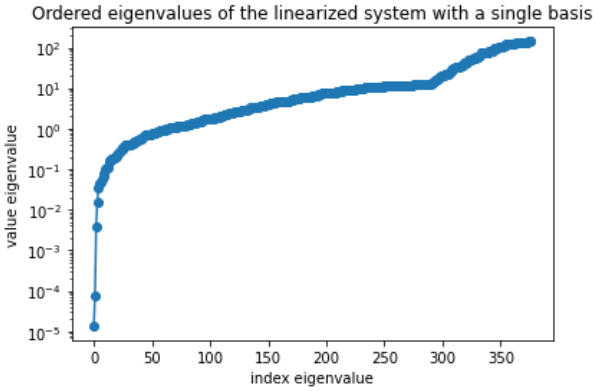


Figure 4.17: The eigenvalues of the global system in ascending order.

Another method is considering the projection of the right hand sight (RHS) onto the eigenbasis. This method is important in systems where there is a nonzero external forcing term. This term is not captured in the eigenvalue dynamics since it was assumed to be zero. Each q_i satisfies:

$$\dot{q}_i + \lambda_i q_i = \phi_i^\top F_i. \tag{4.9}$$

This shows that the excitation of each mode i depends directly on the projection $\phi_i^\top F$. If this projection is zero for some i, then the corresponding mode is not directly excited by the forcing.

To identify which modes are activated, we compute:

$$f_i := \phi_i^\top F. \tag{4.10}$$

This is done with all eigenvectors of the eigenvalue problem in the matrix Φ, where they are ordered again in the same manner as above, the eigenvector with the corresponding eigenvalue with the smallest magnitude is first, i.e. ϕ_1. A mode i is strongly excited if f_i is large in magnitude over time. The solution to this scalar ODE is:

$$q_i(t) = q_i(0)e^{-\lambda_i t} + \frac{f_i}{\lambda_i}\left(1 - e^{-\lambda_i t}\right), \tag{4.11}$$

and converges to a steady-state value $q_i^\infty = f_i/\lambda_i$ as $t \to \infty$. This reveals that the contribution of each mode to the solution is governed by its *excitation score*:

$$\text{excitation}_i := \left|\frac{f_i}{\lambda_i}\right| = \left|\frac{\phi_i^\top F}{\lambda_i}\right|. \tag{4.12}$$

In Figure 4.18, the excitation scores for each index are visualized. For this method, a mode should be included in a reduced basis if it exceeds a threshold:

$$\left|\frac{\phi_i^\top F}{\lambda_i}\right| \geq \varepsilon, \tag{4.13}$$

where ε is chosen based on desired accuracy.

The following question arises: if a mode i is not activated by the forcing term, i.e.,

$$f_i := \phi_i^\top F = 0$$

does this imply that mode i is unimportant and can be discarded?

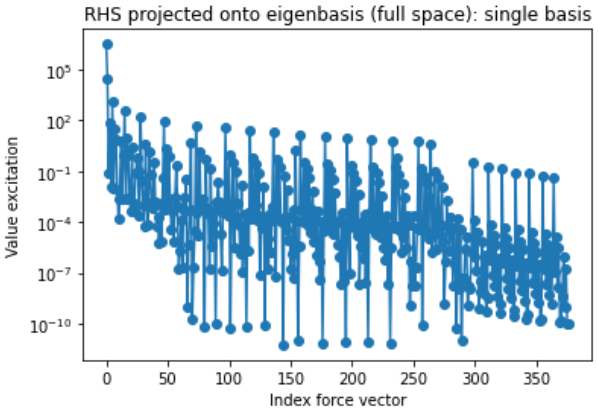


Figure 4.18: Projection of the RHS onto the full eigenbasis scaled with the eigenvalues for a single basis.

This is not necessarily the case. Even if $f_i = 0$, the modal amplitude q_i satisfies the homogeneous equation:

$$\dot{q}_i + \lambda_i q_i = 0,$$

with solution:

$$q_i = q_i(0)e^{-\lambda_i t}.$$

Thus, mode i contributes to the solution purely through its initial condition $q_i(0)$. Since,

$$t \to \infty,$$

this solution tends to zero. This means that the modes excited by the RHS contribute more dominant to the solution when approaching the steady state solution. Remember, we consider a stationary system, i.e. we have a constant steady state solution. However, for a transient solution, the modes contributing solely through the initial condition do play a role in the solution.

To visualize the excitation of the RHS, we plot the *excitation scores*, shown in Figure 4.18. From this figure, it becomes clear that not only the first r eigenmodes are significantly excited by the forcing term. Therefore, instead of blindly selecting the r smallest eigenmodes, we define a selection criterion based on the *excitation scores*. This leads to the following workflow:

1. Solve the eigenvalue problem.
2. Sort the eigenvalues.
3. Project the system's right-hand side (Eq. (4.3)) onto the eigenbasis by computing inner products.
4. Set a criterion, i.e. define ϵ to select the eigenmodes.

In Figure 4.19, the error of component A is given for this workflow. The error is computed as Eq. (4.2). The solution for the ROM with the modal reduction is compared to the FOM at a given timestep. It is compared to the other strategy: taking the eigenvectors for the basis corresponding to the smallest r eigenvectors. It becomes clear that increasing the number of modes according to the *smallest* r method gives certain jumps in the error at the positions where an eigenmode is excited. The jumps corresponds to including an eigenvector which excitation value is relatively high.

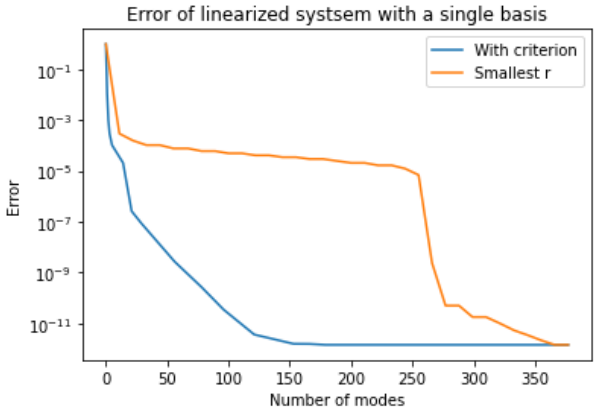


Figure 4.19: Error (component A): comparison between taking the eigenvectors corresponding to the smallest eigenvalues and between setting a criterion to select specific eigenvalues for a single basis. The error is computed as (Eq. (4.2)).

Remarks:

- We looked at the error at the last timstep, which is close to a steady state solution.
- A useful alternative is to shift the system to its steady-state solution. Let $T_\infty = K^{-1}F$ denote the steady-state temperature. Define the shifted variable $\tilde{T} := T - T_\infty$, which satisfies:

$$M\dot{\tilde{T}} + K\tilde{T} = 0, \tag{4.14}$$

  i.e., a homogeneous system with zero forcing. This transformation isolates the transient dynamics and avoids the need to project the forcing term. The solution becomes:

$$\tilde{q}_i(t) = \tilde{q}_i(0)e^{-\lambda_i t}. \tag{4.15}$$

Coupled system, two bases. Again, an alternative approach for constructing a reduced basis for the system in Eq. (4.3) is to consider the system as two separate subsystems. The benefits are discussed already in the previous section and this approach is also needed to fit in the way of working, since these components are part of a bigger ensemble.

Each subsystem has its own generalized eigenvalue decomposition:

$$\begin{aligned} K_A\phi_i^A &= \lambda_i^A M_A\phi_i^A, \quad \text{for } i = 1,\dots,n_A, \\ K_B\phi_j^B &= \lambda_j^B M_B\phi_j^B, \quad \text{for } j = 1,\dots,n_B, \end{aligned} \tag{4.16}$$

leading to two separate modal bases Φ_A and Φ_B.

We define the reduced temperature vectors as:

$$T_A = \Phi_A q_A, \quad T_B = \Phi_B q_B,$$

and substitute into the global system. Projecting onto the respective eigenbases yields a reduced system in modal coordinates:

$$\begin{bmatrix} \dot{q}_A \\ \dot{q}_B \end{bmatrix} + \begin{bmatrix} \Lambda_A & \Phi_A^\top K_{AB}\Phi_B \\ \Phi_B^\top K_{BA}\Phi_A & \Lambda_B \end{bmatrix} \begin{bmatrix} q_A \\ q_B \end{bmatrix} = \begin{bmatrix} \Phi_A^\top F_A \\ \Phi_B^\top F_B \end{bmatrix}, \tag{4.17}$$

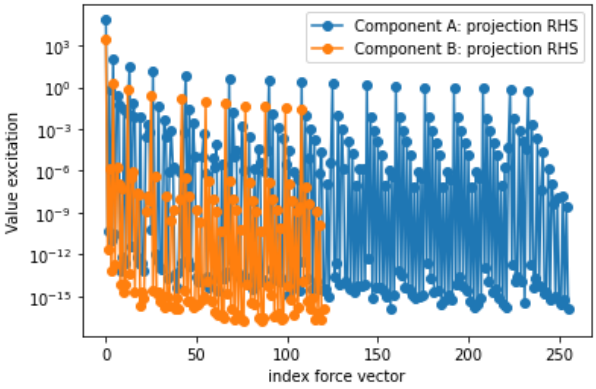


Figure 4.20: Projection of the RHS onto the full eigenbasis scaled with the corresponding eigenvalues (two different subspaces with two different eigenbases).

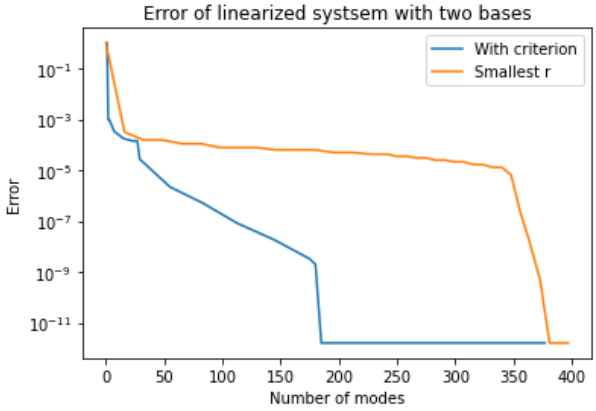


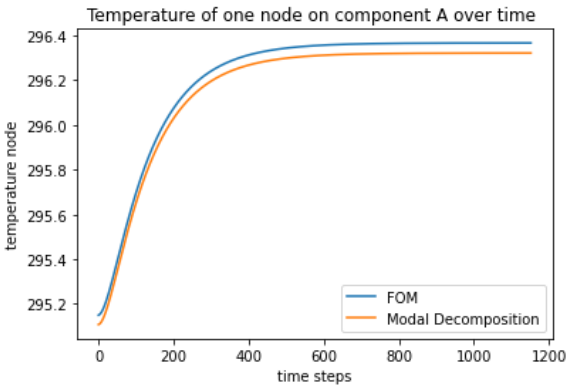
Figure 4.21: Error (component A): comparison between taking the eigenvectors corresponding to the smallest r eigenvalues and between setting a criterion to select specific eigenvalues for two bases. Note that this is the total number of modes, i.e. for two bases this means the number of modes of both reduced bases. The error is computed as (Eq. (4.2)).

where Λ_A and Λ_B are the diagonal matrices of eigenvalues for subsystems A and B.

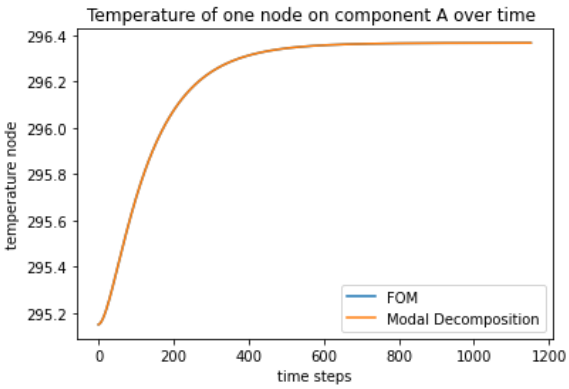
Following the same workflow as in the single-basis approach, the excitation values are shown in Figure 4.20. The errors, computed as Eq. (4.2), at a specific time step for both strategies (smallest r and the criterion-based method) are presented in Figure 4.21. From this figure, we observe that the error using the two-basis method is significantly larger compared to the global single-basis approach.

Equation (4.17) highlights that the eigenvalue problems in Equation (4.16) do not include the off-diagonal coupling terms in the full stiffness matrix. These terms represent the dynamic interaction between the two subsystems. As a result, more eigenmodes may be required to accurately capture the system's behavior.

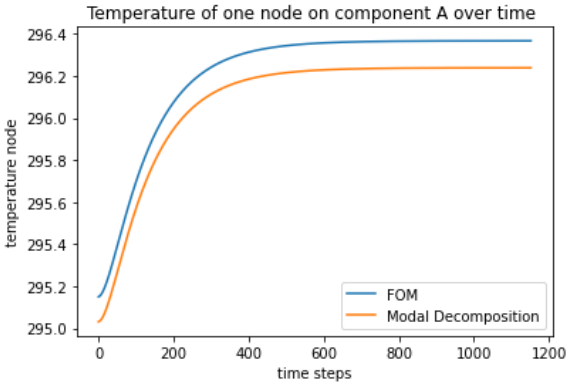
Results strategies Figure 4.22 clearly visualizes the benefits of the use of one single basis. The error is more significant for two bases, the reason for this is already discussed above. However, both methods significantly reduces the dimension of the system with a relatively small error. As mentioned before, to fit the way of working, we will continue this section with two separate bases.



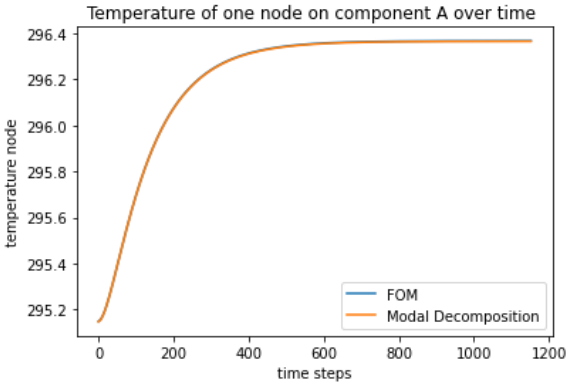
(a) Two separate bases, 29 modes.



(b) One single basis, 29 modes.



(c) Two separate bases, 17 modes.



(d) One single basis, 14 modes.

Figure 4.22: Modal decomposition: comparison between one or two bases for different mode counts.

4.3.2 Modal decomposition for the nonlinear system

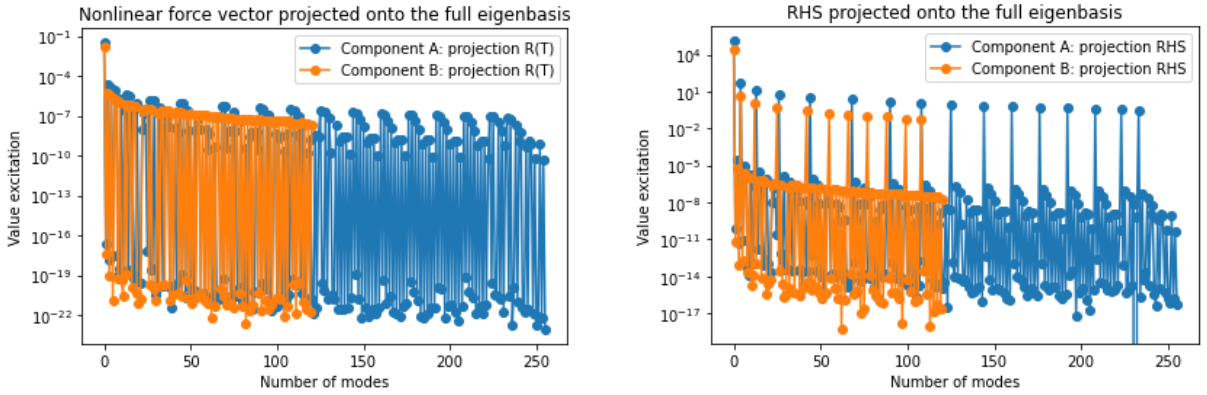
We now extend the analysis to systems with nonlinearities in the form:

$$M\dot{T} + KT = F - R(T), \tag{4.18}$$

The nonlinearity enters as a force of the right-hand side. Despite the introduction of nonlinearity, we still consider the same eigenvalue problem for the model reduction:

$$K\phi_i = \lambda_i M\phi_i, \quad \text{for } i = 1, \ldots, n \tag{4.19}$$

The following section assumes two bases for the two separate subsystems, constructed the same as above. For this analysis we consider a spatially varying simulation again. Unlike the linearized case, the projection of the nonlinear term onto these bases is time-dependent rather than a constant. As a result, we cannot precompute the important modes for this nonlinear term, nor can we shift the system to obtain a homogeneous system. Figure 4.23a shows the nonlinear term evaluated at the intial conditions, projected onto the full eigenbases. Figure 4.23b shows the full right-hand side of the system, namely $F - R(T)$ for both systems, projected onto their corresponding full eigenbasis.



(a) Full force vector, $F - R(T)$.

(b) Nonlinear term, $R(T)$

Figure 4.23: Comparison of the full right-hand side and the nonlinear component projected onto the full eigenbasis.

One can find that the constant forcing term yields larger excitation scores. To compare the mode-selection strategies, we will revisit the two methods again: first compare the results with the smallest r eigenvalues, for an increasing r. Hereafter, we will use a criterion for the full right-hand side, where the nonlinear term is evaluated at the initial condition. In Figure 4.24, the error for the two different approaches are given, computed as Eq. (4.2).

Furthermore, we can compare the linearized system with the nonlinear system. Because we cannot shift the equilibrium to obtain a homogeneous system for the nonlinear case, and the results in the linearized system for using a criterion were significantly better, we will compare the results for using a criterion for both methods. The results are given in Figure 4.25.

To visualize the results and errors, we will compare it to a spatial plot of the FOM, see Figure 4.26. Figure 4.27 represents the temperature evolution over time on one node, to obtain the error over time. The plot is again given for a full time simulation and the last 100 steps to get a closer look at the dynamics. For these plots, a criterion was set to select the number of modes. The approximation seems actually quite accurate, but if we plot the spatial temperature at the

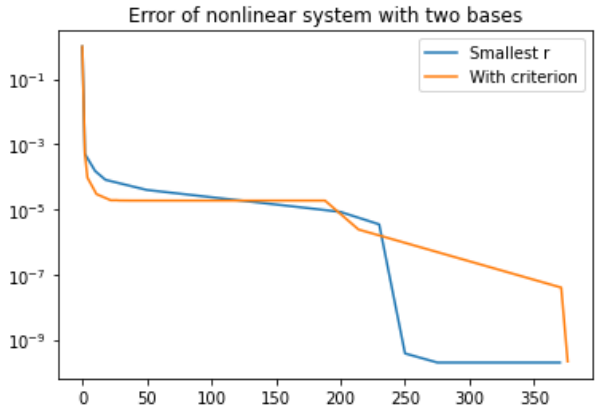


Figure 4.24: The error (Eq. (4.2)) of both strategies compared for a nonlinear system.

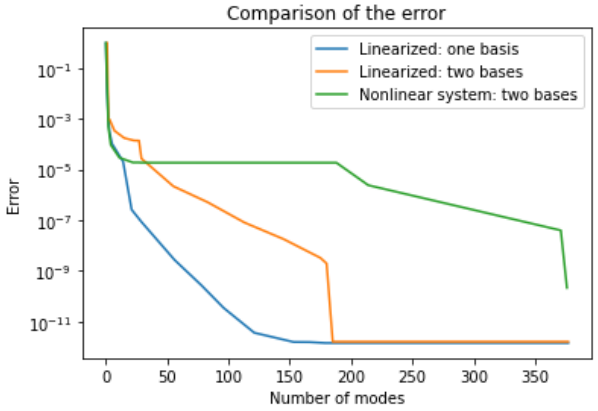
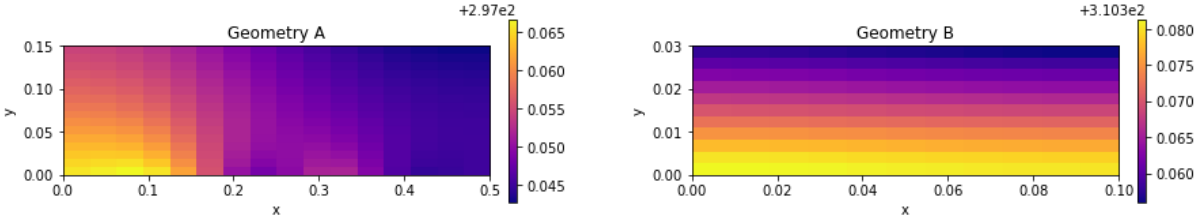


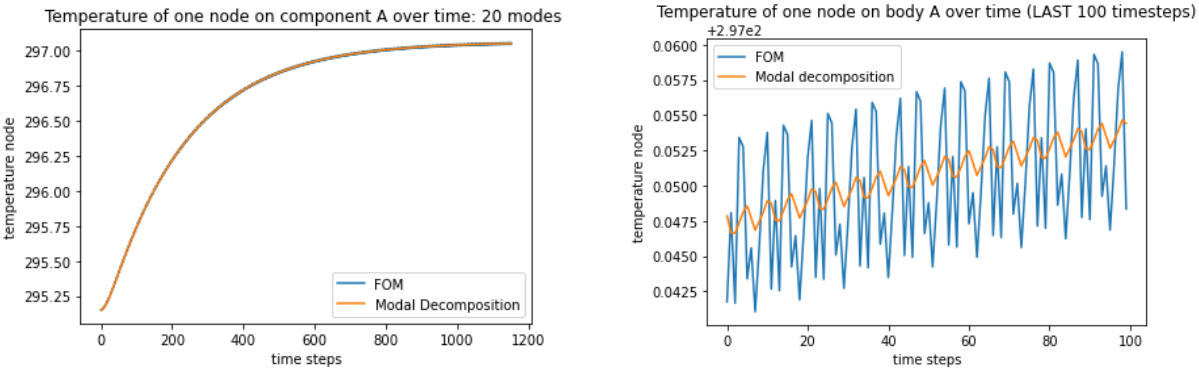
Figure 4.25: The comparison of the error (Eq. (4.2)) between the linearized and the nonlinear system using an increasing number of eigenmodes in the bases.

last timstep, see Figure 4.28, one can observe that this looks different then Figure 4.26. If we take a closer look at Figure 4.23, one can obtain that the excitation scores are much higher for the constant force vector then the nonlinear radiation term which include the view factor. These effects are not accurately incorporated in the modes used for these results.



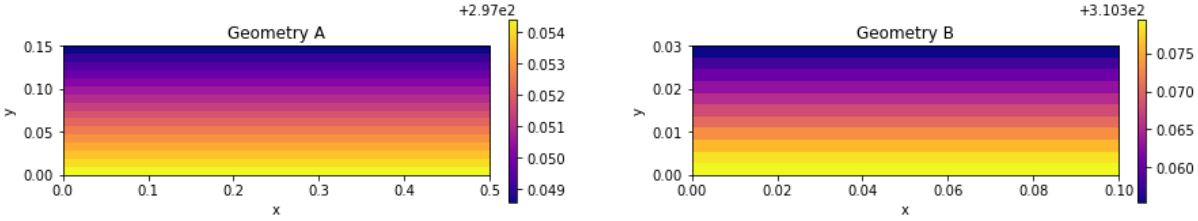
(a) Component A: FOM. **(b)** Component B: FOM.

Figure 4.26: FOM of the system. A reference for the results given in Figures 4.28 and 4.30.



(a) Full time simulation. **(b)** Last 100 timesteps.

Figure 4.27: Plots of the temperature over time on one node: FOM compared to the modal decomposition of a nonlinear system. The total number of modes is 27, that is for both systems. The eigenvectors are chosen based on a criterion.

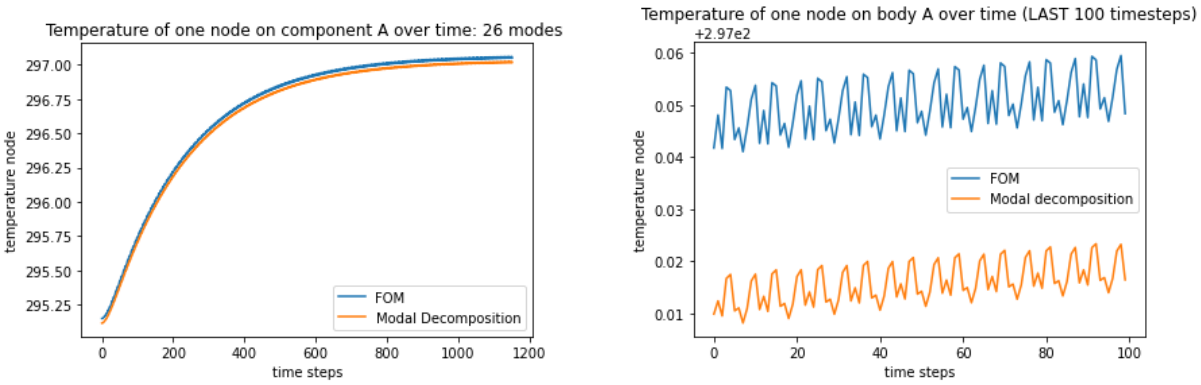


(a) Component A: Modal decomposition with 16 modes. **(b)** Component B: Modal decomposition with 11 modes.

Figure 4.28: Modal decomposition for the nonlinear system with two bases.

It is also possible to use the other strategy: taking the eigenvectors corresponding to the smallest r eigenvalues. The results for this are given in Figure 4.29 and Figure 4.30. One can also obtain that the error is quite significant within visualization in the later figure. In Figure 4.24, it can be seen that the error does not decrease much if we take some more modes into account.

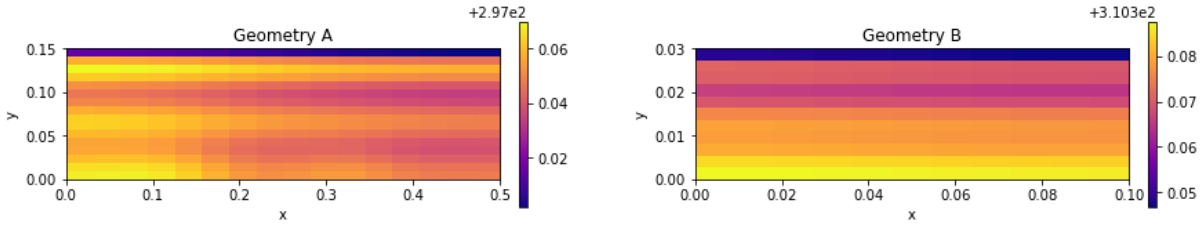
The results for the modal decomposition are at first sight really promising. The $L2$-error can be reduced to $> 10e^{-6}$ with a limited (~ 10) number of eigenmodes, see Figure 4.19. However, the results for the linearized system are only results for a stationary system. We can not move



(a) Full time simulation.

(b) Last 100 timesteps.

Figure 4.29: Plots of the temperature over time on one node: FOM compared to the modal decomposition of a nonlinear system. The total number of modes is 26, that is for both systems. The eigenvectors corresponding to the 13 smallest eigenvalues are taken for both systems.



(a) Component A: Modal decomposition with 10 modes.

(b) Component B: Modal decomposition with 10 modes.

Figure 4.30: Modal decomposition for the nonlinear system with two bases. The nonlinear behavior is not captured accurately.

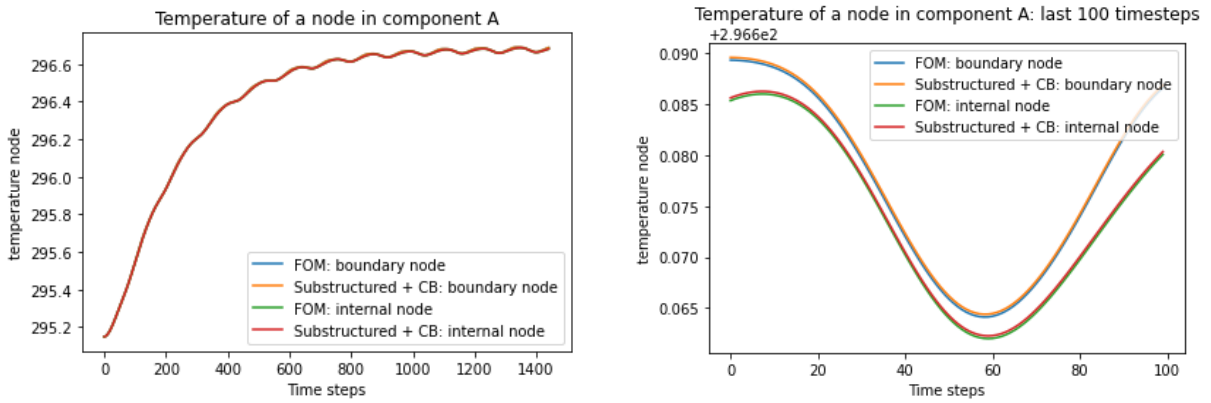
the waferstage due to limitations of the linearization, which is a big aspect of our problem. The modal decomposition for the nonlinear system does give a good basis, i.e. the error can be relatively small, around $10e^{-5}$ for more then 10 eigenmodes, in comparison with the FOM but, as we visualized, some spatial patterns are not captured within the eigenvalue problems. The results are therefore not accurate enough. As mentioned before, the nonlinearity appears only at the boundary, this motivates the use of substructuring, where we can apply the modal decomposition to the linear dynamics of the system.

4.4 Substructuring

As discussed in Section 3.2, the previous results motivate the use of substructuring. The constant mass- and stiffness matrix allow us to solve the generalized eigenvalue problem. However, the force vectors are nonzero, therefore the general eigenvalue problem does not capture the behavior appropriately. These forces are only applied on the boundary, motivating the use of substructuring. After substructuring, we can apply Craig-Bampton for the linear internal part and apply another technique on the boundary.

4.4.1 Internal Reduction: Craig-Bampton

The next step is to evaluate this method. As mentioned before, the velocity of the movement of the component B can vary. To obtain a periodic steady-state solution, the simulation must span a sufficiently long time interval. We start with a slow motion allowing us to quickly compute solutions over a longer time period.



(a) Full time period **(b)** Last 100 timesteps

Figure 4.31: Temperature of nodes over the full time period and the last 100 timesteps for the comparison of the FOM and the substructured system with Craig-Bampton reduction for **zero** eigenmodes.

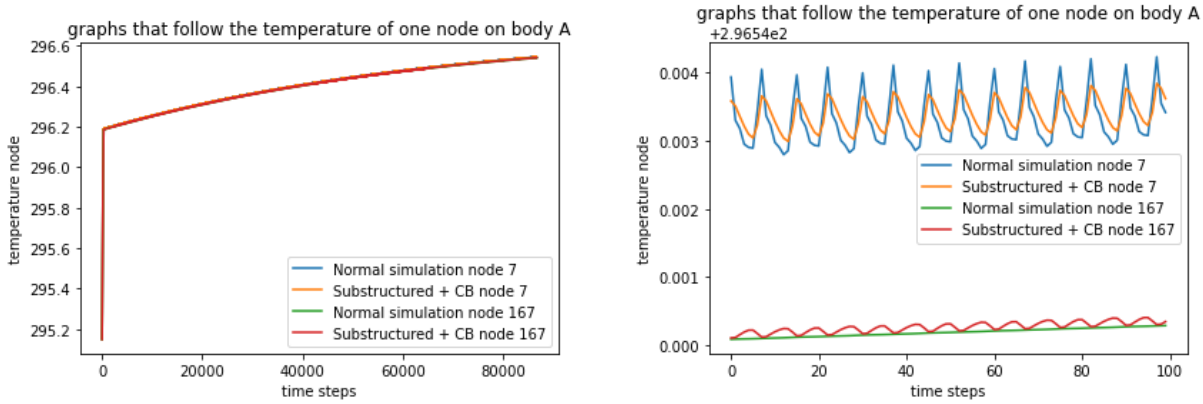
The simulation monitors the temperature evolution at two distinct nodes in component A: a boundary node and an internal node. As a result, the latter is expected to have a smaller temperature variation due to its distance from the radiative boundary.

The reduction to zero eigenvalues (i.e. $\Phi \equiv \mathbf{0}$) yields already an accurate result, see Figure 4.31. The results obtained in Figure 4.31a are the full time simulation of the FOM compared to the substructured system with the Craig-Bampton reduction with zero eigenmodes. Figure 4.31b show the last 100 timesteps to get a closer look at the results. The results are really accurate, they are overlapping very closely, meaning the ROM accurately captures the dynamics of the FOM. This insist that the internal dynamics can be fully described by the solution on the boundaries.

This conclusion follows from the decomposition of the solution into static and dynamic modes: the static modes are expressed as functions of the boundary nodes, while the dynamic modes are associated with the system's eigenmodes. Applying this reduction technique allows us to reduce the system so that only boundary solutions need to be computed. Effectively, this reduces a two-dimensional problem to a one-dimensional one.

However, it is important to note that the previous simulations involved a slowly moving geometry, enabling us to observe long-term behavior over a relatively short simulation time. To now assess the model with faster motion, we decrease the timestep. We first examine a simulation using four eigenmodes and compare its performance to the FOM.

As shown in Figure 4.32a, the reduced-order model accurately represents the FOM throughout the entire simulation period. However, the overall temperature variations remain relatively small compared to the total temperature scale. Figure 4.32b provides a detailed view of the final 100 time steps, where noticeably larger deviations between the two models become visible.



(a) Full time period

(b) Last 100 timesteps

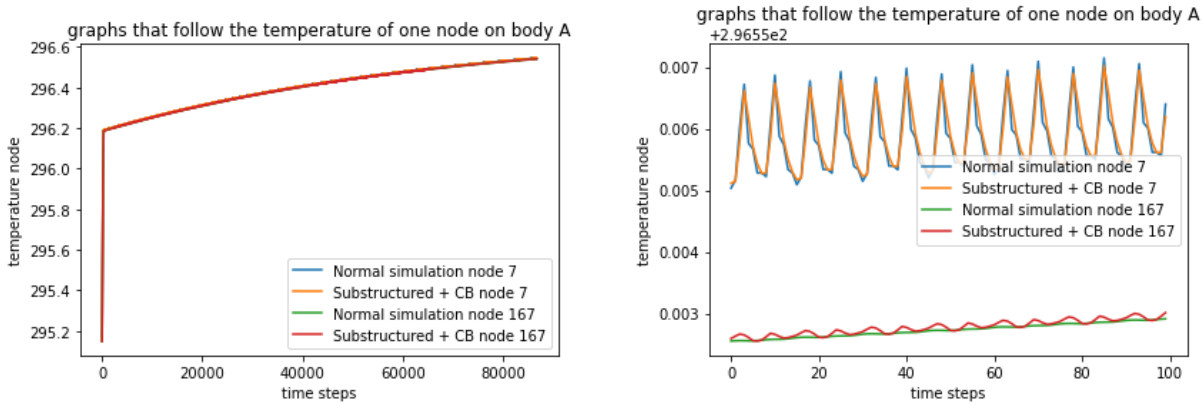
Figure 4.32: Temperature of nodes over the full time period and the last 100 timesteps with fast motion (four eigenmodes).

In both Figure 4.33 and Figure 4.34, the results are shown for an increasing number of eigenmodes. From the figures, it is evident that as the number of eigenmodes increases, ROM converges towards the Full Order Model (FOM), as expected. For the result with 12 eigenmodes, we can obtain nearly overlapping graphs, yielding a very accurate result. These results show that the system can still be significantly reduced for the interior nodes. However, it became clear that within the fast-motion case the transient behavior is stronger than in the slow-motion simulation. Since the Craig-Bampton eigenmodes represent this transient part, they become more important for accurate reduction when the system moves faster. This suggests that the motion of the system is important in determining the number of eigenmodes in the reduction.

4.4.2 Interface Reduction: Neural Networks

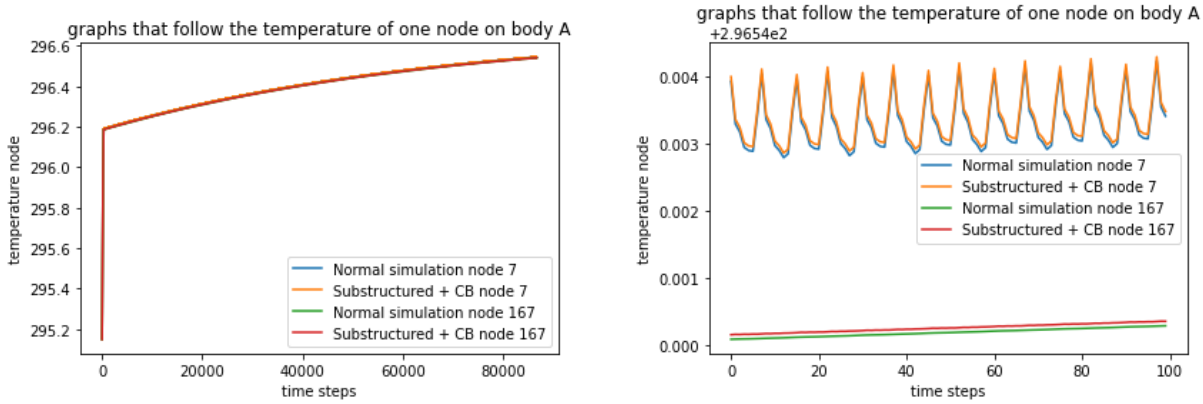
The data used to train the network was generated from long simulations of the reduced finite element model with the Craig-Bampton method. Specifically, we collected snapshots of the temperature on the radiation boundaries and the corresponding radiation outputs. These snapshot matrices were used to form the training dataset.

We implemented the neural network in TensorFlow, using a feedforward architecture with multiple hidden layers and a nonlinear activation function, see Section 3.2.2.



(a) Full time period **(b)** Last 100 timesteps

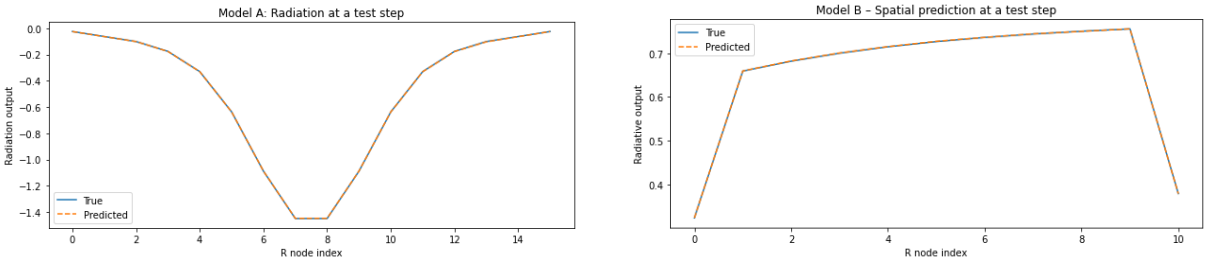
Figure 4.33: Temperature of nodes over the full time period and the last 100 timesteps with fast motion (10 eigenmodes).



(a) Full time period **(b)** Last 100 timesteps

Figure 4.34: Temperature of nodes over the full time period and the last 100 timesteps with fast motion with 12 eigenmodes

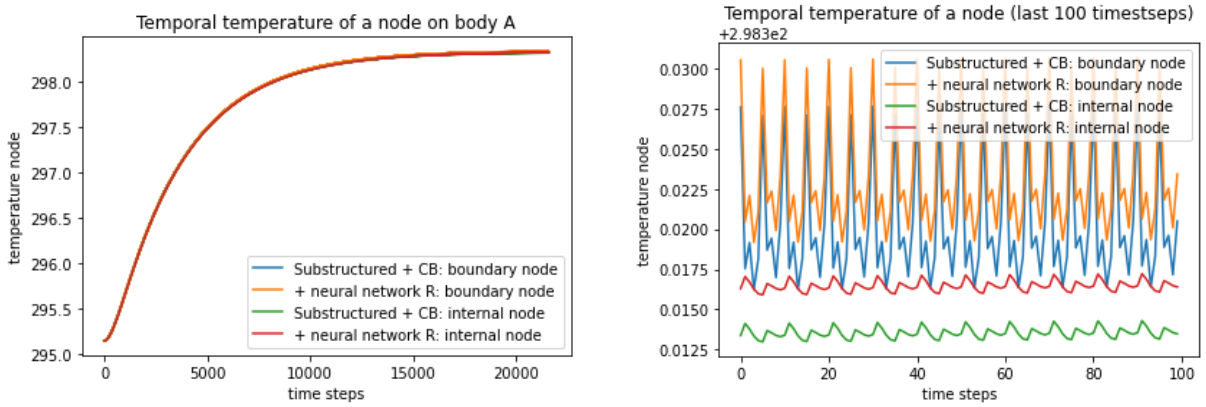
The neural network we created splits the data into a train and test set. This allows us to test the neural network on data it does not *know* yet. Figure 4.35a shows the results of specific test index for model A and Figure 4.35b shows it for model B.



(a) Neural network for component A **(b)** Neural network for component B

Figure 4.35: The results of the neural networks used in this work. A test sample is used to visualize the accuracy of the results from both models

In the results in this section, we included eight eigenmodes in Φ. In Figure 4.36, the temperature on a node of component A is given. The results give for a relatively fast motion. Again, since the temperature fluctuations are small, the deviations are not clear from the plot of the full time period. Therefore, the last 100 timesteps are given in Figure 4.36b. We obtain a very promising



(a) Full time period **(b)** Last 100 timesteps

Figure 4.36: Temperature of nodes over the full time period and the last 100 timesteps for the comparison of the substructured system with Craig-Bampton reduction and with the neural network.

result: the approximation with a neural network closely resembles the results without neural networks. Note that when zooming into a very small region of the graph, a slight offset appears. The offset in the last figure is a consequence of accumulative error. Due to the accumulative nature of numerical error, even a very small error can become significant after a large number of time steps. In this case, after approximately 20000 steps, the error, which was initially negligible, begins to noticeably affect the solution. The error in Figure 4.35a is barely visible, however, computing the absolute error gives a maximum value of

$$10e^{-4},$$

which will be significant after many timesteps. Unlike other reduction techniques based on eigenmodes, this model is based on a neural network architecture, and therefore increasing the number of eigenmodes is not applicable as a means of reducing the error. This offset is so small that we consider it acceptable and attribute it to numerical effects.

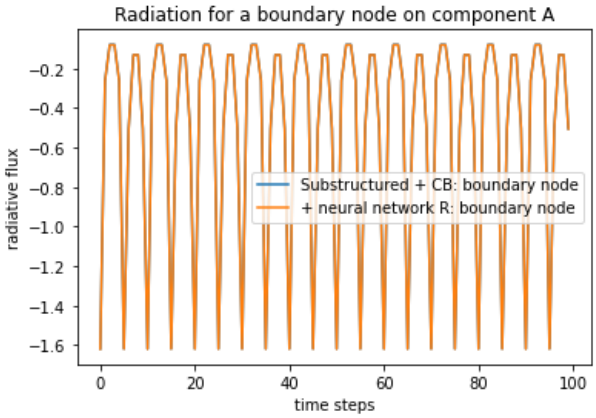
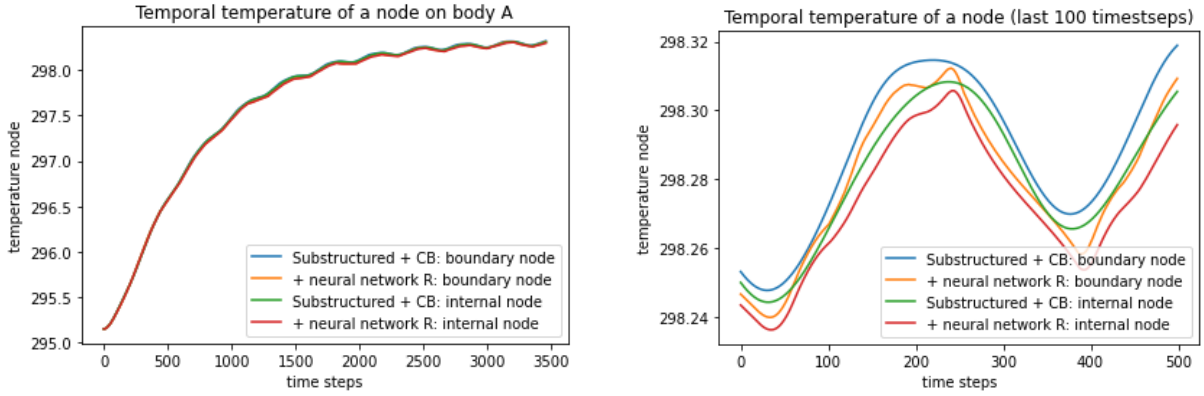


Figure 4.37: Radiation on a boundary node on component A

The results of the radiation term alone are shown in Figure 4.37. It can be seen that, in a full simulation, the radiation obtained from the standard computation of the nonlinear radiation term is approximately the same as the results from the neural network computation. However, we just saw that the very small error appearing in the approximation add up after many timesteps.

Figure 4.38 shows the results for a slow motion of component B. Also, Figure 4.39 shows the predicted radiation with the neural network again, compared to the full FEM computation of the radiation. As expected, the temperatures of the boundary node and the interior node are much closer to each other. These results are again very promising and provide motivation to extend the approach to a 3D case using ASML software.



(a) Full time period

(b) Last 100 timesteps

Figure 4.38: Temperature of nodes over the full time period and the last 100 timesteps for the comparison of the substructured system with Craig-Bampton reduction and with the neural Network.

The results in this section are very promising. We therefore propose the algorithm as an efficient technique to reduce the system: use substructuring to divide the internal and boundary dynamics. The nonlinearity is only appearing on the boundary, so we isolate the nonlinearity with the substructuring approach. The internal linear system can be reduced with a well-established known method, like the Craig-Bampton method. This method effectively reduces the system; for a stationary system, we can even ignore the transient behavior and reduce the system to a boundary problem only. The movement of the waferstage introduces transient internal behavior, the eigenmodes in the Craig-Bampton method capture this behavior and a few modes are

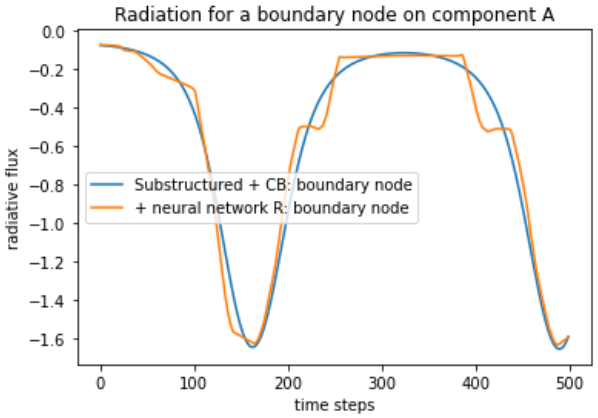


Figure 4.39: Radiation on a boundary node on component A. The results for relatively slow motion ($\Delta t = 80$).

needed to accurately capture this. For our proposed algorithm, we create a surrogate function for the nonlinear spatially varying term with a neural network. It creates a simple input-output function which is computationally very cheap. So, the nonlinear boundary term is not reduced in size, but within the complexity of the evaluation, and hence the runtime is reduced. Both components need separate neural networks, which is in line with the way of working of reducing component-wise: reduce all the subsystems separate so you can interconnect them with other sub assemblies, and within the whole assembly.

As mentioned before, the input of the neural network is the temperature on the radiative boundaries only. Craig-Bampton reduced the internal nodes only, leaving the boundary degrees of freedom in full space. This implies that we do not have to map the reduced temperature vector back to full space.

Also, the view factor matrix is incorporated within the neural network. This is an important feature of the neural network, as the view factor matrix computations are very expensive, especially for complex geometries.

Overall, this algorithm provides a promising reduction technique for the nonlinear, spatially varying, radiative heat problem and we will test it on a case study in ANSYS. An overview of the algorithm can be found in Figure 4.40 and Algorithm 3.

Algorithm 3 ROM workflow: Substructuring → Craig–Bampton (internal) → NN (interface radiation) → Time integration (Newton) → Reconstruct

Initialize FE model: e.g. M, K, loads F and $R(t)$, path $(x(t), y(t))$, initial state T_0, time discretization. **Offline**

1. **Substructuring:** split DOFs into interface and internal.
2. **Craig-Bampton (internal DOFs):** build CB basis; project to get reduced matrices.
3. **Neural network (interface radiation DOFs):** train NN to map interface temps + position → radiation flux.

Online

1. **Time stepping with Newton:** for each timestep,
   - evaluate interface radiation via NN,
   - assemble reduced residual/Jacobian, solve Newton update.
2. **Reconstruct full field:** map reduced solution back using the CB basis.

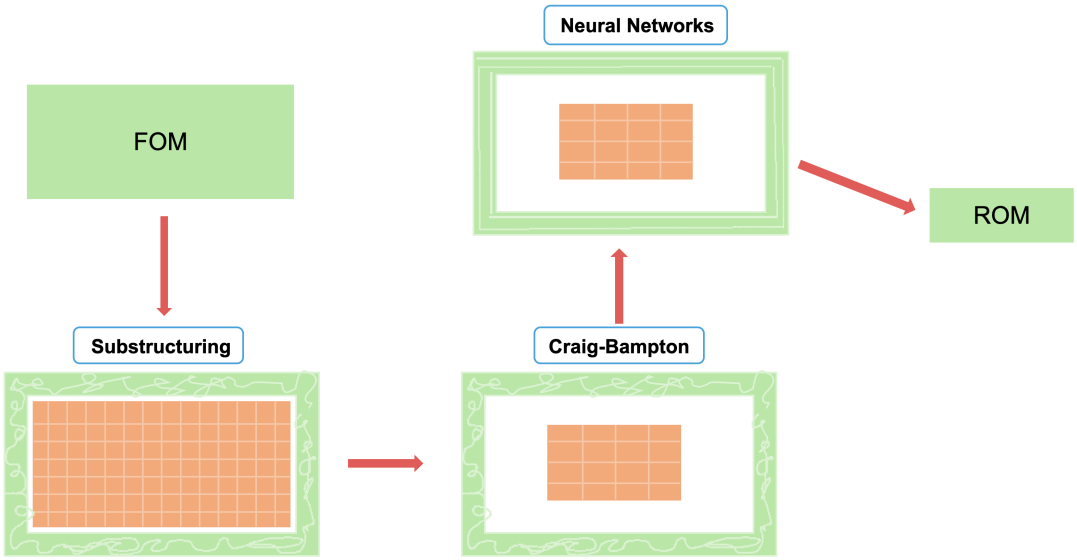


Figure 4.40: Algorithm ROM: (1) substructuring, (2) Craig-Bampton reduction of internal DOFs (reduction in matrix sizes) (3) neural network approximation of interface radiation (does not reduce size), (3) ROM

This algorithm can be reproduced within Python, the code can be found in the GitLab Repository [16].

Chapter 5

Case study in ANSYS: $3D$ model

This chapter applies the proposed algorithm from the previous chapter to a 3D case study in ANSYS. ANSYS is a $3D$ simulation platform with built-in finite-element solvers for PDEs such as the heat equation. The 3D model closely resembles the earlier test problem. Although the geometry remains an idealized, isolated model, it provides a realistic intermediate step and a suitable test model for evaluating the algorithm's performance.

5.1 Initializing $3D$ model

The $3D$ model in ANSYS is given in Figure 5.1. The radiative boundaries are the surfaces facing each other. Note that this is also an assumption: in the actual model, every boundary surface can radiate and absorb heat. Again, two relatively small, cooling Robin boundary conditions are applied at the top surface of component A and at the bottom surface of component B. Furthermore, a heat source is placed at the bottom surface of component B. The simulation parameters can be found in Table 2, these are the parameters that do not change throughout the simulations. The parameters that do change are given in Section 5.4.1. ANSYS can run a transient heat simulation and it can give the spatial temperatures defined on surfaces at specific intermediate timesteps, e.g. the temperature of the bottom surface of component A is given in Figure 5.2

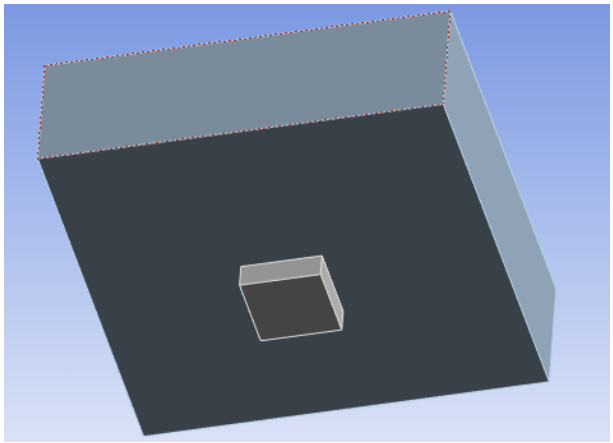


Figure 5.1: $3D$ model in ANSYS

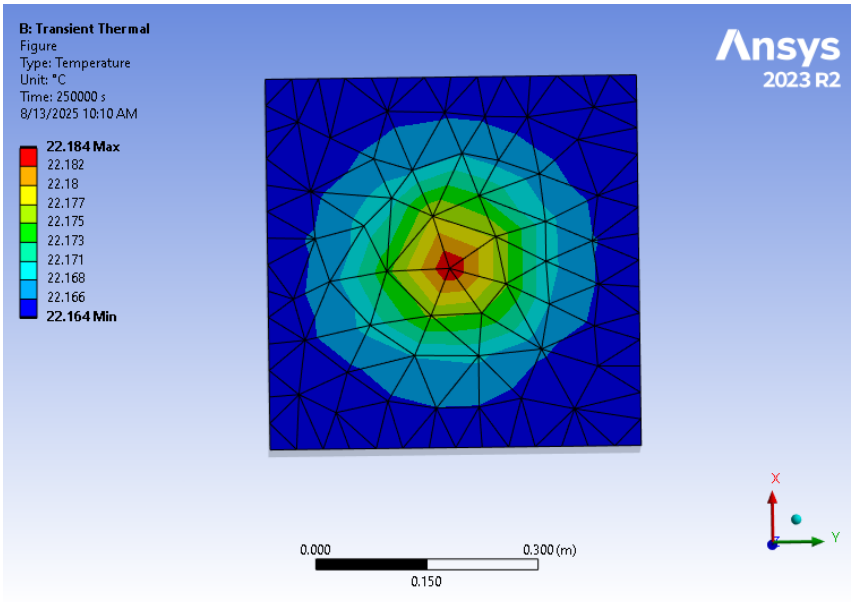


Figure 5.2: The temperature at the bottom of component A. The heat source is an inward flux of 135 W/m^2.

5.2 Collecting Data

We want to extract the data we need for a neural network: the temperatures on the radiative boundaries and the corresponding radiation fluxes on those boundaries. A snippet to export data is given in Appendix C, in Listing C.1. A simulation with a moving component is computationally expensive and time consuming to generate some samples. Hence, we start by restricting the study to stationary datasets: multiple static poses solved as separate runs, from which we built and evaluated the reduced models. In this way, we can quickly validate whether the proposed algorithm works for this case.

From ANSYS, we also extract other data needed for our ROM, like the global nodes of the whole assembly, the nodes of all the boundaries, the element connections, and the coordinates of the nodes. With this data, we can assemble our FEM matrices within MATLAB.

The fluxes on the radiation boundary that we extract are given on the nodes with the unit W/m^2, this means that we also have to adapt the data to Watt on that node, e.g. we can calculate the area of the surfaces attached to the nodes. However, these hand-made computations and adaptions can cause the FEM model in MATLAB to differ slightly from the model in ANSYS. A complete workflow is shown in Figure 5.3.

5.3 Neural Networks

The neural networks for this setup are the same as in the $2D$ case, a feedforward neural network with multiple layers. We train it in a similar manner and also test the results on a test set. The results are again very similar to the *real* radiative flux, see Figure 5.4. To gain more insight in the mathematical depth of the training of neural networks, the code of this training is provided in the Gitlab Repository [16].

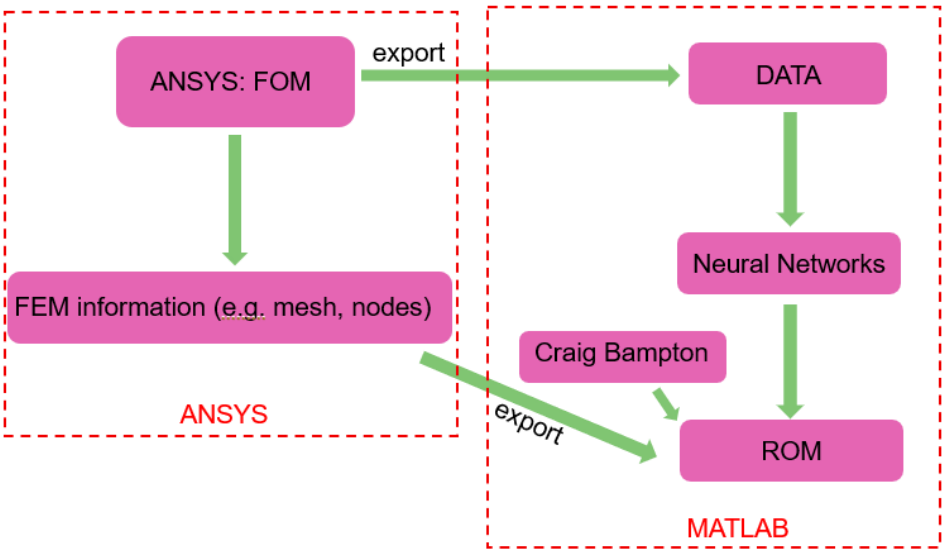
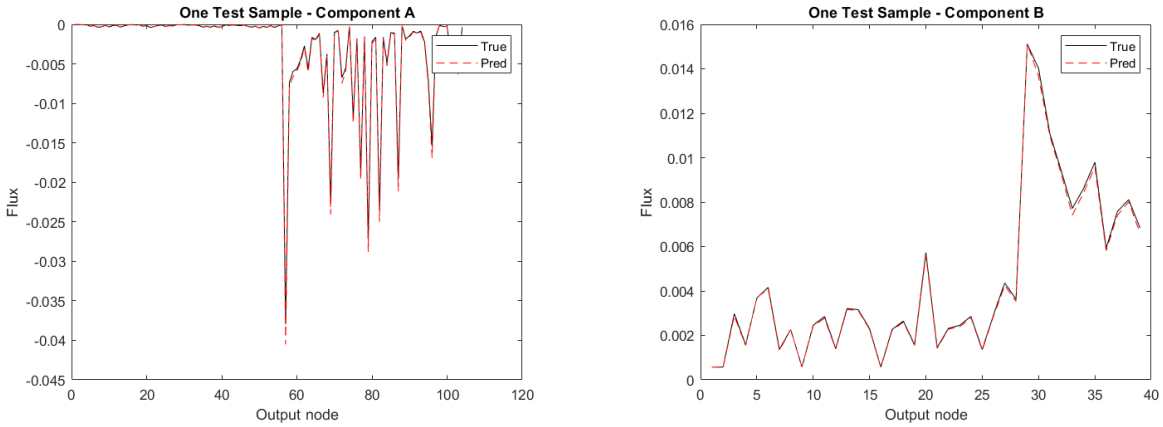


Figure 5.3: Workflow of Chapter 5; ANSYS and MATLAB.



(a) Neural network for component A.

(b) Neural network for component B.

Figure 5.4: Neural networks of both components. A test sample of the data set is visualized to see the effect of the neural network.

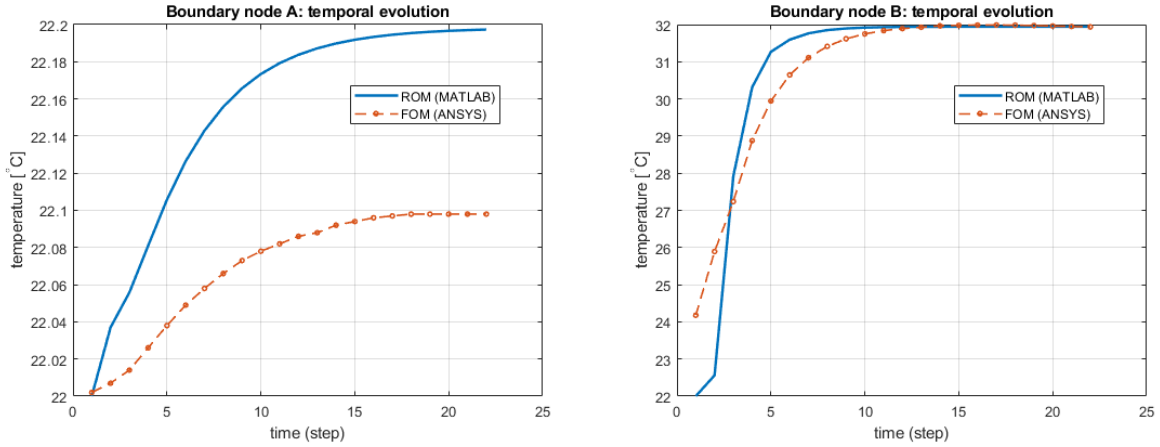
5.4 Results

This section discusses the results for the stationary problem and the limitations of further research.

5.4.1 Stationary model

To test the approximation of the model with Craig-Bampton and the neural networks from ANSYS data, we can compare a full simulation in ANSYS to our reduced simulation in MATLAB. In the following section we take into account 8 eigenmodes in component A and 3 eigenmodes in component B. The error does not reduce much when we include more modes. We compare runtime of a simulation over similar time simulations. To ensure a fair comparison for one case: material properties, mesh, and boundary conditions are the same. Also, both models use the same t_{start}, t_{end} and number of steps (Nt). The runtimes are measured excluding pre/post-processing and visualization. To obtain general results, we run the simulation for different cases. Within the cases, the heat source can change, i.e. the inward heat flux on the bottom surface of component B varies. t_{end} and Nt also varies throughout the cases:

- Case A: $g = 100\ W/m^2$, $t_{end} = 100{,}000s$, $Nt = 19$
- Case B: $g = 80\ W/m^2$, $t_{end} = 250{,}000s$, $Nt = 22$
- Case C: $g = 40\ W/m^2$, $t_{end} = 250{,}000s$, $Nt = 100$
- Case D: $g = 65\ W/m^2$, $t_{end} = 200{,}000s$, $Nt = 1000$
- Case E: $g = 130\ W/m^2$, $t_{end} = 10{,}000s$, $Nt = 10000$



(a) Component A: temperature of a boundary node **(b)** Component B: temperature of a boundary node

Figure 5.5: Temperature over time on the radiative surfaces for the FOM and the ROM.

For case A, we visualized the temperature over time of a node at the radiative boundary for components A and B compared to the FOM, see Figure 5.5. There is a clear offset between the temperatures of component A of the FOM and ROM. There are multiple explanations for this: modifying the radiative fluxes on the nodes in MATLAB can cause significant errors. Also, the mass- and stiffness matrix and the constant force vectors and matrices were not directly available within ANSYS. These are also computed separately out of the mesh, material properties and other information from ANSYS. These computations can cause significant errors in the results. However, the visualization and pattern of the temperature is quite accurate, see Figures 5.6 -

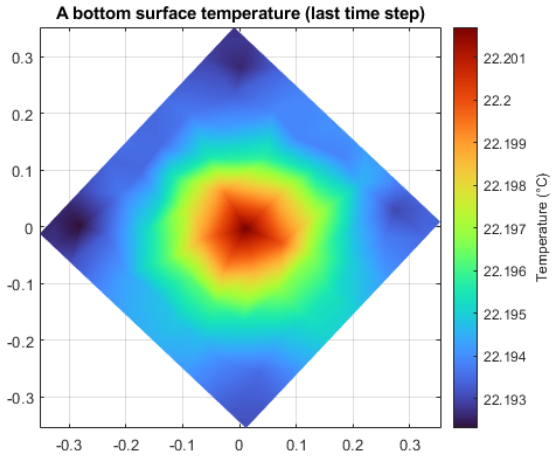
5.10. The runtime of a simulation is also significantly improved. The results for these runtimes are given in Table 5.1. Note that the average time for one timestep for the FOM in ANSYS is $0.124s$ and the time for the ROM in MATLAB is $0.0151s$. This means that the speed factor is given by:

$$\frac{0.124}{0.0151} \approx 8.2.$$

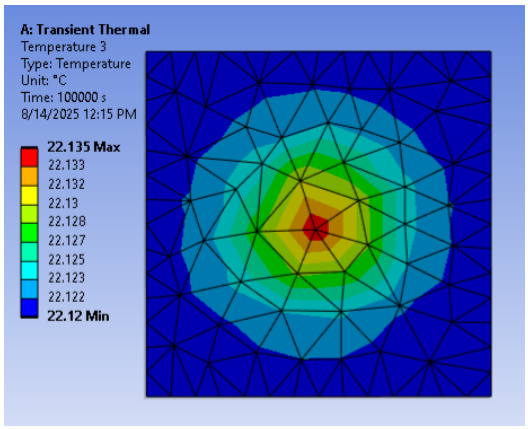
Hence, the ROM is 8.21 times faster compared to the FOM for the stationary setup.

Table 5.1: Runtime: FOM (ANSYS) vs. ROM (MATLAB). [Median time] over Nt timesteps.

Case	Num. Steps (N_t)	FOM time [s]	ROM time [s]	Rel. L^2 error	Abs. L^2 error
A	22	2.60 [0.19]	0.45 [0.0194]	3.47e-3	1.36
B	19	2.2 [0.16]	0.60 [0.03]	2.33e-3	0.91
C	100	8.6 [0.086]	1.37 [0.0137]	2.47e-3	0.96
D	1000	88.6 [0.0886]	6.48 [0.0065]	3.79e-3	1.48
E	10000	980.8 [0.098]	60.14 [0.006]	2.55e-3	0.99
Avg.	1	0.124	0.0151		

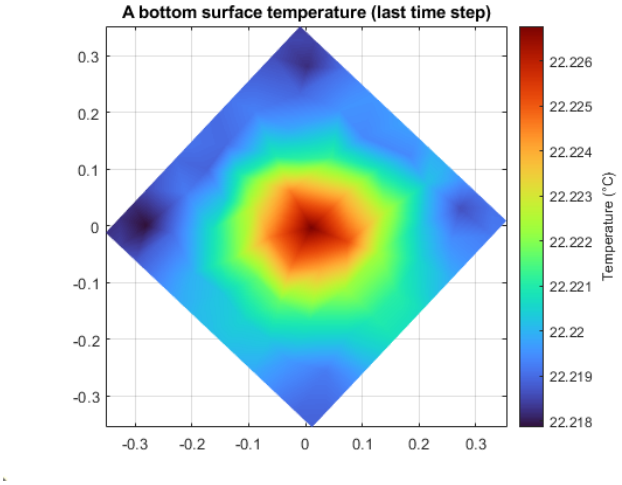


(a) MATLAB: ROM results at final timestep.

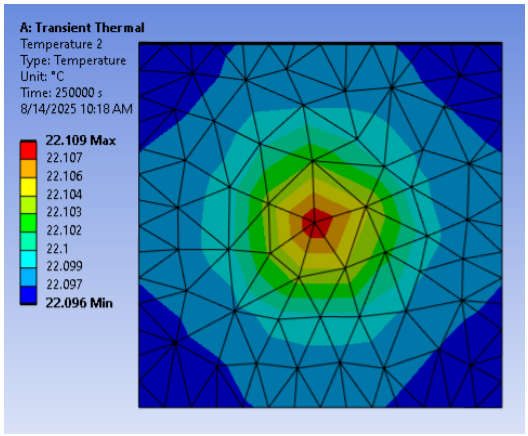


(b) ANSYS: FOM results at final timestep.

Figure 5.6: Case A: temperature on the bottom of component A at $t = 250{,}000\,\mathrm{s}$ ($Nt = 22$) with $80\,\mathrm{W/m^2}$ heating. ROM (MATLAB) vs. FOM (ANSYS).

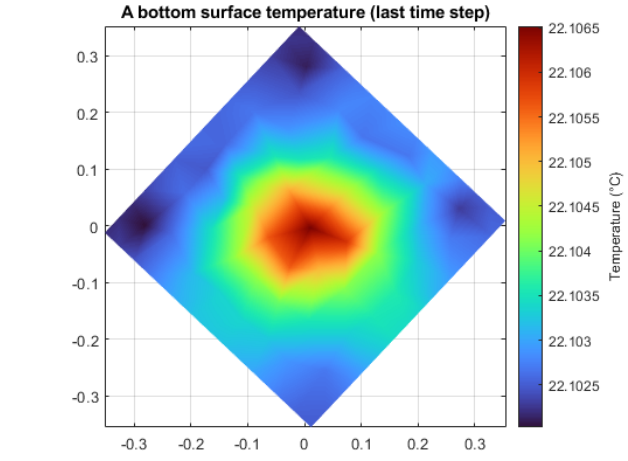


(a) MATLAB: ROM results at final timestep.

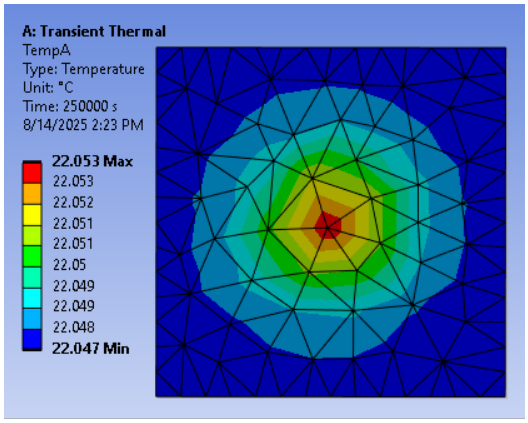


(b) ANSYS: FOM results at final timestep.

Figure 5.7: Case B: temperature on the bottom of component A at $t = 100{,}000\,\mathrm{s}$ ($Nt = 19$) with $100\,\mathrm{W/m^2}$ heating. ROM (MATLAB) vs. FOM (ANSYS).

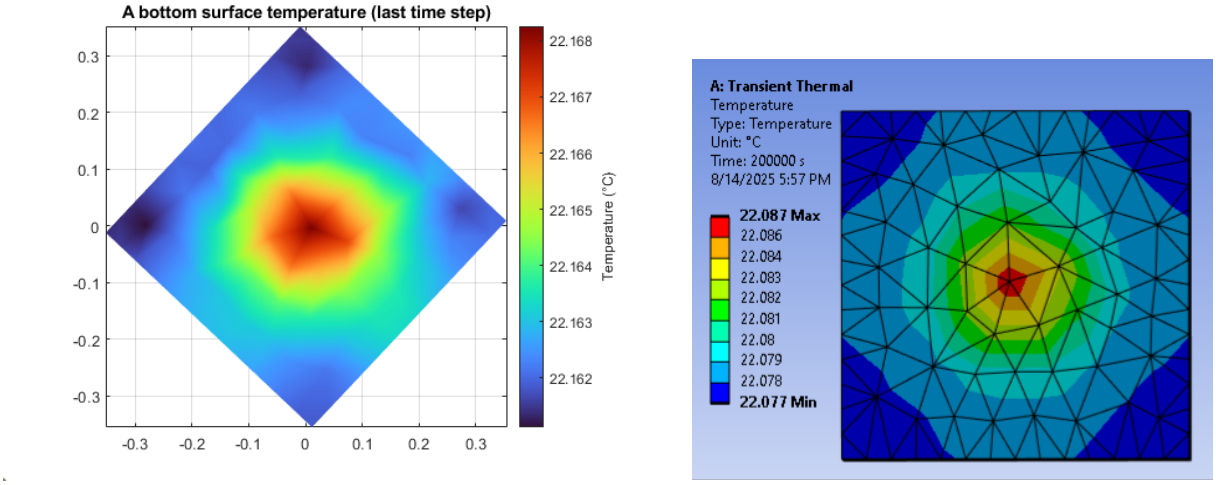


(a) MATLAB: ROM results at final timestep.



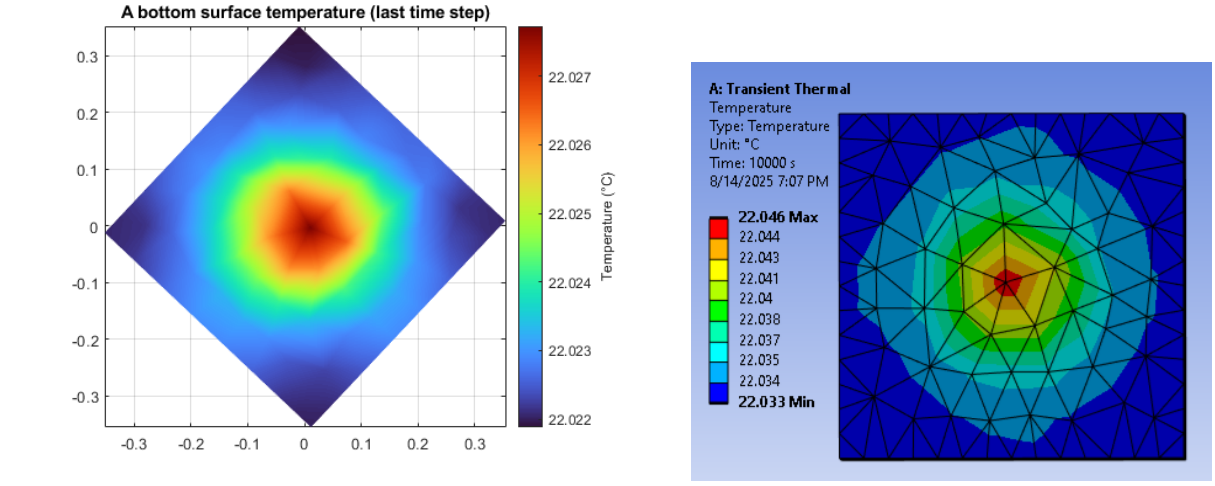
(b) ANSYS: FOM results at final timestep.

Figure 5.8: Case C: temperature on the bottom of component A at $t = 250{,}000\,\mathrm{s}$ with $80\,\mathrm{W/m^2}$ heating. ROM (MATLAB) vs. FOM (ANSYS).



(a) MATLAB: ROM results at final timestep. **(b)** ANSYS: FOM results at final timestep.

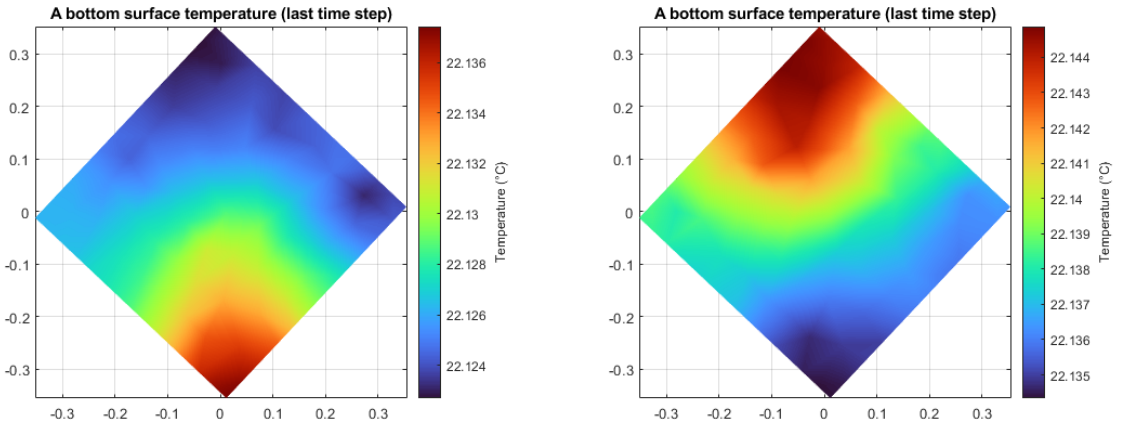
Figure 5.9: Case D: temperature on the bottom of component A at $t = 200{,}000\,\mathrm{s}$ with $65\,\mathrm{W/m^2}$ heating. ROM (MATLAB) vs. FOM (ANSYS).



(a) MATLAB: ROM results at final timestep. **(b)** ANSYS: FOM results at final timestep.

Figure 5.10: Case E: temperature on the bottom of component A at $t = 10{,}000\,\mathrm{s}$ with $130\,\mathrm{W/m^2}$ heating. ROM (MATLAB) vs. FOM (ANSYS).

The results can also be given for different positions, see Figure 5.11. Within our ROM, we can move the block, but the training data did not include transient behavior, due to limitations in ANSYS. This is for further research.



(a) Position at $(x, y) = (0.15; -0.15)$ **(b)** Position at $(x, y) = (-0.2; 0.15)$

Figure 5.11: ROM results for different positions of component B.

5.4.2 Moving component

The next step involves modeling a spatially varying component, which more closely reflects the core focus of this thesis. As discussed in the problem statement, the view factor matrix (VFM) is a computationally expensive component of the model. In the stationary case, the VFM is computed only once at the beginning of the simulation. Furthermore, the reported runtime includes only the time required for the time-dependent simulation; preprocessing steps such as the VFM computation are excluded.

Modeling a moving block for 10 timesteps took $12.8s$ for the FOM. In MATLAB, 10 timesteps give a total runtime of $0.33s$. This would mean the ROM is approximately

$$\frac{12.8}{0.33} \approx 39$$

times as fast as the FOM.

Furthermore, in ANSYS $2023R$, which is the software available within ASML, it is not possible to update the view factors each timestep, because they are fixed after the preprocessing steps. This means that component B contains a movement, but the radiation is fixed at the initial position of the moving component. As a result, we are unable to collect the correct data and cannot generate the ROM for this problem. However, the updated version of ANSYS supports this functionality, and the workflow to enable it is already in place. An APDL snippet can be added to the analysis branch in the project tree. The snippet is added in appendix C, Listing C.2. This means that the the speedfactor will be even higher then 39 as the view factor matrix computations are not included in this runtime. The single view factor calculation within this setup is $1.39s$. This means that calculating it every timestep will significantly increase the runtime.

Note that generating the data for the ROM can be very time consuming and expensive, especially in this moving setup. However, once the ROM is constructed, its evaluation becomes a computationally inexpensive operation. The input does not affect the evaluation cost compared

to the stationary case, as the VFM is embedded within the model. The ROM functions as a single input-output mapping, meaning it does not matter whether the spatial input remains constant across the timesteps or varies. This observation is very promising for further research. The costs of one evaluation of a neural network can also be calculated by running the neural network for an amount of iterations and dividing the total time by the number of iterations. The total time for an evaluation is: $1.8367e-5s$.

The average simulation time for one timestep is 0.124 seconds, see Table 5.1. Including the view factor matrix computations would give a simulation time of

$$0.124 + 1.39 = 1.514$$

seconds. Comparing this to the ROM gives:

$$\frac{1.514}{0.0151} = 100.26.$$

This means, including the view factor matrix for a single timestep would mean that the ROM would be 100.26 times faster than the FOM.

To conclude, the proposed algorithm for this nonlinear spatially varying radiative heat transfer problem will significantly reduce the total runtime. Without the movement, which causes the dominant part of the total runtime, the ROM is already approximately 8.21 faster than the FOM. Including movement, this *speedfactor* already scales up to approximately 39. This is the case where the view factor matrix is excluded from the runtime. This matrix evaluation for a single computation is already 1.39 seconds. Including this computation every timestep will significantly increase the total runtime. The factor will even increase to 100.26. Meaning the ROM is approximately 100 times faster than the FOM. To give an idea: some simulation of complex geometries can take a couple of days, e.g. $3 \times 24 \times 60 \times 60 = 259.200$ seconds (3 days). Computing this with the proposed algorithm will take only 2592 seconds, or approximately 43 minutes. Therefore the speed up for the ROM is really promising.

The code to reproduce the ROM can be found in [16].

5.5 Computational Complexity

The runtime we mentioned in the previous paragraph is only focused on this specific model. To gain more insight in the actual computational benefits of the ROM, we will look at the computational complexity of the two models. Firstly, we zoom into the comparison of the evaluation of the radiative term. Eventually, also the overall reduction is considered, as the Craig-Bampton method also reduces the system matrices.

A simulation consists of offline and online costs. Offline costs refer to precomputed models that are run beforehand, like predefining the material properties, mesh, training data or static mass- and stiffness matrices. It consists of all computations that do not change in time. In contrast, online costs are executed during the simulation to compute solutions. These costs must be kept minimal to enable fast evaluations of different simulations.

Neural Network. The most promising reduction technique in the algorithm is the neural network. The nonlinear, spatially varying radiative FEM assembly is the computational bottleneck of this problem. The online costs are especially very expensive as the view factor matrix

needs to be computed every timestep. The costs were already discussed in Section 3.4. For every timestep, the costs are

$$\mathcal{O}\left((N+M)^2 + (N+M)\right),$$

where N is the number of radiative interface elements on component A and M the number of radiative interface elements on component B.

To compute the computational costs of the neural network, we have to revisit the online evaluation. A detailed explanation for the computational complexity is given in Appendix A. The online costs are

$$\begin{aligned}\text{FLOPs}_{\text{fwd}} \approx\ & (2H_{t1}N_{\text{in}} + H_{t1}) + (2H_{t2}H_{t1} + H_{t2}) \\ & + (2H_{p1}D_p + H_{p1}) + (2H_{p2}H_{p1} + H_{p2}) \\ & + (2H_3(H_{t2} + H_{p2}) + H_3) + (2H_4H_3 + H_4) \\ & + (2N_{\text{out}}H_4 + N_{\text{out}}),\end{aligned} \tag{5.1}$$

where $N_{in} = N + M$ and $D_p = 2$ (2D case). The order then grows as $\mathcal{O}(N + M)$.

Radiation Assembly The FEM assembly is modeled with a specific code, where radiative heat transfer between two components (A and B) is modeled. This can be found in Appendix B. The core of the algorithm consists of a loop over all boundary elements of component A and component B. For each pair of radiation boundary elements $(i, j) \in \Gamma_A$ and $(k, l) \in \Gamma_B$, the function computes view factors, evaluates nonlinear radiation terms (e.g. T_A^4, T_B^4), computes and assembles the Jacobian and assembles contributions into global matrices.

Let N and M denote the number of boundary elements on components A and B, respectively. The total number of floating point operations (FLOPs) required by the algorithm scales as:

$$\text{FLOPs} = \mathcal{O}(N \cdot M)$$

This quadratic scaling comes from the pairwise interaction between all boundary elements of the two components, i.e. we need to evaluate all combinations.

A detailed estimate shows that each $(i, j) - (m, n)$ pair requires approximately 144 floating point operations, this can be derived from computing the flops of all the steps in the FEM code.

N	**M**	**FEM FLOPs** $(N \cdot M)$	**NN FLOPs** $(N + M)$
10	10	1,900	20
100	100	190,000	200
1,000	1,000	19,000,000	2,000
10,000	10,000	1,900,000,000	20,000
100	1,000	1,900,000	1,100
1,000	10,000	190,000,000	11,000

Table 5.2: Comparison of estimated FLOPs for radiation FEM assembly (quadratic scaling) and the neural network approximation (linear scaling). The radiation model scales as $N \times M$, while the neural network scales as $N + M$, making it computationally more efficient for large scale problems.

Table 5.2 and Figure 5.12 give an overview of the comparison of the growth of FLOPs when scaled as $\mathcal{O}(N \cdot M)$ or $\mathcal{O}(N + M)$.

The ROM achieves a computational complexity of only $\mathcal{O}(N + M)$ for the evaluation of the neural network, which is significantly more efficient than the $\mathcal{O}(N \cdot M)$ growth of a full model,

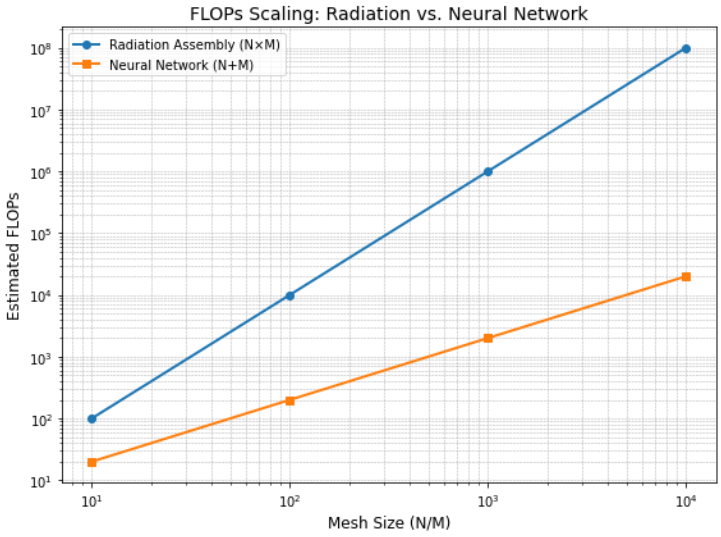


Figure 5.12: Log-log plot comparing the computational complexity of radiation assembly (FEM) and a neural network approximation. The radiation method scales quadratically as $\mathcal{O}(N \cdot M)$ due to pairwise interactions between boundary elements, while the neural network scales linearly as $\mathcal{O}(N + M)$. This demonstrates the significant computational advantage of the ROM for large scale simulations.

making the ROM far better suited for problems with large input sizes. Furthermore, the computational costs for the view factor matrix every timestep should also be taken into account, this significantly increases the complexity of the FOM. Of course, the full online costs of the simulation involve more than these radiation evaluations alone (e.g. Newton iterations, assembly, and residual/Jacobian updates), but these algorithmic steps remain unchanged in structure. The only difference within the ROM is the decrease in the size of the system matrices due to the Craig-Bampton method, leading to a further reduction in terms of FLOPs (e.g. matrix multiplications). A dominant feature in the proposed algorithm is the multiplication of a square system matrix with a vector. For a matrix of size $n \times n$, one matrix-vector product requires n multiplications and $(n-1)$ additions per row, i.e.,

$$\text{FLOPs} \;= n\,(n + (n-1)) = 2n^2 - n \;\approx\; 2n^2.$$

Hence, the cost scales quadratically with the number of degrees of freedom n. For a ROM, the full dimension n is replaced by a much smaller reduced basis size $r \ll n$. The corresponding matrix-vector cost becomes

$$\text{FLOPs}_{\text{ROM}} \approx 2r^2,$$

yielding a significant speedup, especially in time-dependent settings where such products are performed at every time step.

A drawback of both the neural network and Craig-Bampton reduction is that they require additional offline computations (e.g. training in the case of the neural network, or modal analysis and projections (matrix multiplications) in the case of Craig-Bampton). These precomputations can be significant, but they are one time costs. Once completed, the online phase is where the ROM gives significant advantages. The reduction in online complexity is very important to enable fast tests and simulations, making the ROM approach computationally better for repeated simulations or large scale parametric studies.

Chapter 6

Discussion

This chapter revisits the problem statement and subsequently addresses the sub-aspects of the main research question, as introduced in Section 1.3.

The central objective of this study is to develop a fast and accurate reduced-order model (ROM) for a spatially varying, nonlinear radiative heat transfer assembly. The full-order model becomes high-dimensional due to the fine finite element discretization of a complex geometry, combined with the computational cost of evaluating the nonlinear radiative boundary condition.

6.1 Sub research aspects:

1. Nonlinearity: The nonlinearity of the model was a big subject of this study. The nonlinearity of the model significantly influences other aspects of this research. Specifically, linear reduction methods failed to produce sufficiently accurate results due to the nonlinear contributions. This limitation led us to adopt a structural reduction approach: substructuring. Van Steen et al. [15] did extensive research on substructuring techniques. However, their work did not address the challenge of nonlinear interface reduction, which is the central focus of this thesis. Here, we aim to fill that gap by developing an algorithm that effectively handles nonlinearities at the interfaces. A neural network can deal with nonlinearity efficiently because the nonlinear activation functions in its layers let it learn complex relationships directly from the data, without needing to predefine them.

The linearization of the boundary conditions introduced additional complications. Specifically, the constants that appeared in both the stiffness matrix and the forcing term were not truly constant, they varied over time. The linearization is therefore not an accurate approximation of the moving nonlinear boundary condition. However, there are also other linear time-varying reduction techniques that we did not address in this thesis. This can be interesting for further research.

2: View Factor Matrix: The nonlinear boundary conditions also contain another complex, time consuming computation, namely the view factor matrix (VFM). The VFM changes every timestep due to moving waferstage and calculating this high-dimensional matrix is inefficient. Approximating the View Factor Matrix (VFM) through interpolation of precomputed VFMs reduced computational costs. However, in the proposed algorithm, this interpolation becomes redundant, as the VFM is already embedded in the training data. Consequently, neural networks implicitly learn the VFM during training, eliminating the need for separate interpolation.

3: Methods: Within this research, we investigated several reduction techniques on the test problem explained in Section 1.4. All the techniques excel at different aspects: some physics-informed methods are easy applicable, while data-driven ROMs need training data, which can be expensive to generate. However, physics-informed techniques are not always applicable due to limitations in the software or the complexity of the PDEs we want to solve.

We began with a data-driven technique, the POD method, where we efficiently reduced the linear part of the system. The nonlinear part was unfortunately not reduced due to the evaluation in full space for this term. This limitation can be handled with a nonlinear extension: DEIM. However, limitations within the ASML software, whereas we could not access the radiation calculation to adjust this, led us to exploring other methods.

A common reduction technique for linear systems is modal decomposition. It uses the eigenvectors of the general eigenvalue problem to construct a reduced basis. The results for the linearized model were very promising. However, the linearized model was not able to generate a spatially varying system. We evaluated the results of using the linear eigenbasis for our nonlinear problem. This gave some promising results, but the eigenbasis did not capture the nonlinear behavior accurate enough.

Again, the nonlinear radiation term is only applied on the boundary nodes and elements in a FEM model. This motivates the use of substructuring. Substructuring allows us to treat the internal dynamics, which are linear, separate from the nonlinear boundary dynamics. The nonlinear boundary part of the system was a new challenge, especially because we did not have any access to the radiation calculation in the simulation software. Modifying the nonlinear term, like applying DEIM, was therefore not an option. To address this, we adopted a neural network based approach to approximate the nonlinear term, which turned out to be an effective and promising reduction technique.

Coupling of the systems: The assembly consists of two subsystems, for which bases can be generated either globally or separately. While both approaches yielded accurate results, using a single global basis provided higher accuracy, as it captures the full cross-subsystem dynamics. However, to align with the subsystems workflow, we adopted the use of two separate bases, one for each subsystem. Despite the slightly bigger error, this approach still produced sufficiently accurate results, validating its practical applicability.

The sub-aspects are closely connected and can all be summarized as **our proposed algorithm**, see Figure 4.40 and Algorithm 3: The model contains strong nonlinearities on the radiative boundaries of the system. The isolation of the nonlinearity in the boundaries resulted in the use of substructuring. Reducing the linear internal dynamics can be done with known linear reduction methods, like the Craig-Bampton. The nonlinear, spatially separated, radiative interface between the two components caused the difficulty. A neural network can be used to generate a surrogate model for the radiation calculation. Generating data for this problem is possible within the software we use. The first, most expensive step is to create a training set. However, once the neural network is trained, the evaluation during ROM simulation is very fast, which is a significant reduction in simulation time compared to the FOM in ANSYS. An overview of the workflow of the algorithm can be found in Figure 5.3.

6.2 Outlook

Despite the promising results, several limitations emerged during the course of this study. First of all, the scope of model order reduction explored in this thesis is limited compared to the

broader landscape of available techniques. Several well-established methods, such as balanced truncation, rational Krylov/IRKA approaches, Physics-Informed Neural Networks, and purely data-driven autoencoder approximations, were not investigated.

The chosen methodological path was shaped by the specific requirements and limitations of the model: constraints on data availability, access to system operators, and the need to align with the subsystem-coupled workflow. These factors influenced and also justified the focus on the selected techniques. Future work could revisit other methods, like the ones above. Or for example, further research could investigate the Nonlinear Modal Decomposition with Subspectral Submanifolds, for periodic steady state solutions. This category of techniques, which rely on periodic steady-state or quasiperiodic solutions, remains unexplored throughout this thesis. In the actual model, the wafer stage moves in a periodic manner; however, the period is characterized by abrupt, non-smooth transitions, making it probably difficult to identify and work with these periodic steady-state solutions.

Another relevant class of methods addresses systems whose stiffness matrix varies in time, $K(t)$. In our case, this appears after linearizing the problem with changing view factors (moving geometry). There already some known methods for this class that successfully reduced this class of problems. Like parametric MOR (pMOR) [24], [25], POD-DEIM or again, the Subspectral Submanifolds methods [26]. This is also an interesting path to follow for further research.

ANSYS imposed several practical limitations that prevented us to apply the originally proposed algorithm in $3D$. We were not able to extract transient thermal behavior for a moving geometry as the view factor matrix can only be computed in the preprocessing step. Also, we could not modify the solvers internal code or even access the code, so integrating DEIM, which requires a change in the evaluation of the nonlinear term, was not feasible.

The data we needed to build and validate the method (e.g., consistent snapshots of the nonlinear term) was not easily accessible, we had to modify it to obtain the correct training input. Also, the ANSYS built-in structure did not allow us to make a transient spatially varying setup. ANSYS expects a stationary fixed geometry when evaluating a transient thermal simulation, because it fixes the VFM throughout the simulation. However, the results obtained in Chapter 5 are promising for a stationary nonlinear radiative model. The results on the test model proved that it is also feasible on a transient system. Therefore, if we can generate the correct training input, this will be a valid algorithm for a nonlinear spatially varying system. The algorithm and workflow is easily adjustable when this will be available.

Compared with the idealized test geometry, the real machine from ASML is substantially more complex. It consists of multiple interacting components with tight tolerances. Material properties (coatings, emissivities) and other boundary conditions further increase the modeling complexity. In this test model, the metroframe only heats because of the waferstage, while in the actual model, there are a lot of other components that influence this. To model this properly we need very fine meshes, which makes the models huge, computationally expensive and it also makes it harder to extract the correct data. This motivates the use of the proposed algorithm, as the evaluation of the neural network remains extremely fast. The only added complexity lies in the data generation process, but this is a one-time effort.

Chapter 7

Conclusion

The main research question of this thesis is: can we develop a reduced order model for a spatially varying radiative interface? We propose a reduced order model for a system with a spatially varying radiative interface, which enables efficient simulations while preserving the dynamics of the problem. The system consists of a moving component that heats up and radiates heat to its surroundings, including another component. The radiative boundary condition is a nonlinear spatially varying boundary condition, including the computation of a view factor matrix every timestep. These computations are very expensive.

While linear reduction methods did not accurately approximated the behavior of the full order model (FOM), the internal dynamics of the system are fully linear. Therefore, the nonlinearity was isolated at the boundary with the use of substructuring, where we partitioned the problem in internal and boundary dynamics. The internal dynamics are fully linear and can be reduced with well established linear reduction techniques like the Craig-Bampton method.

The nonlinearity, isolated on the boundary of the model, introduced a new challenge. Although existing literature offers numerous interface reduction techniques, they do not address the type of nonlinear, spatially separated and varying radiation problem encountered here.

This study proposes a new method: a neural network for this nonlinear term in the discretized FEM formulation. The surrogate is an input-output function, which takes the temperature on the radiative boundaries and the relative position as input. The output is the radiative flux on the boundary. Both assemblies have there own neural network which fit our desired workflow, i.e. to create ROMs per sub-assembly separately.

Substructuring and the Craig-Bampton method leave the boundary nodes *untouched*. It only reduces the internal nodes and therefore we do not need to map the reduced temperature solution back to full space every timestep, this is a real benefit in comparison to other techniques. The output is a vector consisting of the radiative fluxes on the designated boundaries, so it can be integrated directly into the FEM model.

Neural networks offer a powerful and efficient framework for Reduced Order Modeling (ROM), particularly in large scale simulations where computational cost is critical. By replacing traditional FEM radiation assembly, which scales quadratically as $\mathcal{O}(N \cdot M)$, with a neural network based model that scales linearly as $\mathcal{O}(N+M)$, significant speedups can be achieved in the online phase. N and M are the number of radiative interface elements.

Furthermore, the internal dynamics of the system are reduced using the Craig-Bampton method, which projects the full-order model onto a lower-dimensional subspace. This not only simplifies

the dynamic behavior but also reduces the size of the system matrices involved in the simulation. Combined with the neural network surrogate, this leads to a computational cheap and efficient ROM that retains essential physical accuracy while enabling fast evaluations of the full solution.

Lastly, the number of eigenmodes required in the Craig-Bampton reduction basis varied depending on the type of motion. For slow motion, no eigenmodes were necessary. In contrast, faster motion required more eigenmodes, although the graphs showed that even a small number captured most of the dynamic behavior. Using just 12 eigenmodes, the ROM closely approximates the FOM. This is a significant reduction in the sizes of the system matrices, resulting in very efficient computations, as matrix-vector multiplication are of order $\mathcal{O}(n^2)$, where n is the size of the system matrix.

Overall, the computational efficiency and the accuracy of the results are highly promising for this ROM.

Bibliography

[1] M. Thirumaleshwar. *Fundamentals of Heat and Mass Transfer*. Pearson India, India, 2006. Includes Mathcad-based solutions to problems.

[2] Philip Holmes. *Turbulence, coherent structures, dynamical systems and symmetry*. Cambridge university press, 2012.

[3] Carmen Gräßle, Michael Hinze, and Stefan Volkwein. Model order reduction by proper orthogonal decomposition, 2020.

[4] Xiaofei Liu, Hu Wang, Xiaolong Yu, and Chengjing Wang. A krylov-based proper orthogonal decomposition method for elastodynamics problems with isogeometric analysis. *Engineering Analysis with Boundary Elements*, 133:71–83, 2021.

[5] Heiko KF Panzer. *Model order reduction by Krylov subspace methods with global error bounds and automatic choice of parameters*. PhD thesis, Technische Universität München, 2014.

[6] Saifon Chaturantabut and Danny C Sorensen. Nonlinear model reduction via discrete empirical interpolation. *SIAM Journal on Scientific Computing*, 32(5):2737–2764, 2010.

[7] Cyril Touzé and Attilio Frangi. *Model Order Reduction for Design, Analysis and Control of Nonlinear Vibratory Systems*, volume 614. Springer Nature, 2025.

[8] George Haller and Sten Ponsioen. Nonlinear normal modes and spectral submanifolds: existence, uniqueness and use in model reduction. *Nonlinear dynamics*, 86:1493–1534, 2016.

[9] Hongming Liang, Shobhit Jain, and Mingwu Li. Bifurcation analysis of quasi-periodic orbits of mechanical systems with 1: 2 internal resonance via spectral submanifolds. *Nonlinear Dynamics*, 113(11):12609–12640, 2025.

[10] Thomas Breunung and George Haller. Explicit backbone curves from spectral submanifolds of forced-damped nonlinear mechanical systems. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 474(2213):20180083, 2018.

[11] Gerben Izaak Beintema. Data–driven learning of nonlinear dynamic systems: A deep neural state–space approach. 2024.

[12] Anthony Gruber, Max Gunzburger, Lili Ju, and Zhu Wang. A comparison of neural network architectures for data-driven reduced-order modeling. *Computer Methods in Applied Mechanics and Engineering*, 393:114764, 2022.

[13] Hans van Malsen. Convolutional autoencoder based reduced order modelling for physics problems-master thesis report. 2022.

[14] Diederik P Kingma, Max Welling, et al. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

[15] JJ van Steen, RHB Fey, L Iapichino, B Besselink, MLJ Verhees, and AM Steenhoek. Comparison of model order reduction techniques for interface dynamics. 2019.

[16] Nina Bagchus. Gitlab snippet: Code reduced order modeling for spatially varying radiative interfaces. `https://gitlab.tudelft.nl/-/snippets/375`, 2025. TU Delft, accessed on 2025-09-16.

[17] John R Howell, M Pinar Mengüç, Kyle Daun, and Robert Siegel. *Thermal radiation heat transfer*. CRC press, 2020.

[18] Frank P Incropera, David P DeWitt, Theodore L Bergman, Adrienne S Lavine, et al. *Fundamentals of heat and mass transfer*, volume 6. Wiley New York, 1996.

[19] Michael F Modest and Sandip Mazumder. *Radiative heat transfer*. Academic press, 2021.

[20] Sabrina Kelbij Star, Joris Degroote, Jan Vierendeels, Gert Van den Eynde, and Francesco Belloni. Reduced order modelling using a pod-based identification method for parameterized pdes. In *7th European Conference on Computational Fluid Dynamics*, 2018.

[21] Denis Sipp, Miguel Fosas de Pando, and Peter J Schmid. Nonlinear model reduction: a comparison between pod-galerkin and pod-deim methods. *Computers & Fluids*, 208:104628, 2020.

[22] Christian Soize and S Mziou. Dynamic substructuring in the medium-frequency range. *AIAA journal*, 41(6):1113–1118, 2003.

[23] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[24] Anthony T Patera, Gianluigi Rozza, et al. Reduced basis approximation and a posteriori error estimation for parametrized partial differential equations, 2007.

[25] Bernard Haasdonk. Reduced basis methods for parametrized pdes–a tutorial introduction for stationary and instationary problems. *Model reduction and approximation: theory and algorithms*, 15:65, 2017.

[26] Thomas Thurnher, George Haller, and Shobhit Jain. Nonautonomous spectral submanifolds for model reduction of nonlinear mechanical systems under parametric resonance. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 34(7), 2024.

Appendix A

Computational Complexity of the Neural Network

Let $r_A \in \mathbb{R}^N$ and $r_B \in \mathbb{R}^M$ be reduced-temperature inputs. Define the concatenated temperature input

$$N_{\text{in}} = \begin{bmatrix} r_A \\ r_B \end{bmatrix} \in \mathbb{R}^{N+M},$$

and a location input $\text{loc} \in \mathbb{R}^D$ (typically $D = 2$ for 2D).

Network architecture We use two different *towers* (temperature and position) followed by the head:

$$\begin{aligned} &\text{Temp: } N_{\text{in}} \;\to\; H_{t1} \;\to\; H_{t2}, \\ &\text{Pos : } D \;\to\; H_{p1} \;\to\; H_{p2}, \\ &\text{Fusion: } (H_{t2}+H_{p2}) \;\to\; H_3 \;\to\; H_4 \;\to\; N_{\text{out}}. \end{aligned}$$

Each layer computes

$$Z = Wx + b, \qquad H = \text{ReLU}(Z) = \max(0, Z),$$

and the $\max(0, \cdot)$ activation is not counted as FLOPs.

Weight shapes

$$\begin{aligned} &W_{t1} \in \mathbb{R}^{H_{t1} \times N_{\text{in}}}, \qquad W_{t2} \in \mathbb{R}^{H_{t2} \times H_{t1}}, \\ &W_{p1} \in \mathbb{R}^{H_{p1} \times D_p}, \qquad W_{p2} \in \mathbb{R}^{H_{p2} \times H_{p1}}, \\ &W_3 \in \mathbb{R}^{H_3 \times (H_{t2}+H_{p2})}, \qquad W_4 \in \mathbb{R}^{H_4 \times H_3}, \\ &W_{\text{out}} \in \mathbb{R}^{N_{\text{out}} \times H_4}. \end{aligned}$$

FLOP model for a dense layer For a single layer with $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, $x \in \mathbb{R}^{d_{\text{in}}}$, $z = Wx + b$ costs

$$\text{FLOPs}(Wx) \approx 2\, d_{\text{out}}\, d_{\text{in}}, \qquad \text{FLOPs}(+b) \approx d_{\text{out}}.$$

FLOPs per forward pass Summing the layers:

$$\begin{aligned}\text{FLOPs}_{\text{fwd}} \approx\ & \underbrace{(2H_{t1}N_{\text{in}} + H_{t1})}_{W_{t1}} + \underbrace{(2H_{t2}H_{t1} + H_{t2})}_{W_{t2}} \\ & + \underbrace{(2H_{p1}D_p + H_{p1})}_{W_{p1}} + \underbrace{(2H_{p2}H_{p1} + H_{p2})}_{W_{p2}} \\ & + \underbrace{(2H_3(H_{t2} + H_{p2}) + H_3)}_{W_3} + \underbrace{(2H_4H_3 + H_4)}_{W_4} \\ & + \underbrace{(2N_{\text{out}}H_4 + N_{\text{out}})}_{W_{\text{out}}}.\end{aligned}$$

Most of these parameters are fixed numbers. Only $N_{in} = N_{out}$ can vary with the mesh and model. It will grow linearly, i.e.

$$\mathcal{O}(N + M).$$

Appendix B

Radiation Code

Listing B.1: Python script for FEM radiation term

```
def radiation_bc_with_connectionU(U, W, num_nodes_U, num_nodes_W,
    radiation_edges_U, radiation_edges_W, nodes_U, nodes_W, sigma,
    epsilon, VF, ne_u, ne_w ):
    """
    Compute the nonlinear vector R(U,W) for geometry A(!!).

    Parameters:
    - U: solution vector
    - num_nodes: include the numbers of nodes for the corresponding
       geometry
    - radiation_edges: nodes of the edges which have radiation ([n1, n2
       ])
    - nodes: nodes of the corresponding geometry

    Returns:
    - R: Nonlinear residual vector (R+C)
    - J_nonlinear: The Jacobian of R_1 with respect to u
    - J_nonlinear_2: The Jacobian of C_1 with respect to w
    """
    R = np.zeros(num_nodes_U)  # Initialize  vector
    J_nonlinear = np.zeros((num_nodes_U,num_nodes_U))
    J_nonlinear_2 = np.zeros((num_nodes_U,num_nodes_W))

    # 1D linear shape functions in reference space (for edge with two
       nodes)
    def shape_functions_1D(xi):
        return np.array([(1 - xi) / 2, (1 + xi) / 2])
    # loop over elements on geometry 1 with U
    for elem1, edge1 in enumerate(radiation_edges_U):
        n1, n2 = edge1  # Get global node indices for this boundary edge
        x1, y1 = nodes_U[n1]
        x2, y2 = nodes_U[n2]

        # Compute 1D Jacobian (edge length in physical space)
        J_edge = 0.5 * np.sqrt((x2 - x1) ** 2 + (y2 - y1) ** 2)

        # Extract local U values
        U_local = np.array([U[n1], U[n2]])
```

```
#print(U_local)
# Initialize local residual contribution
R_e = np.zeros(2)
J_nonlinear_e = np.zeros((2,2))

# Perform quadrature integration over the boundary edge
for q, w in enumerate(quadrature_weights1d):
    xi_q = quadrature_points1d[q]  # Quadrature point in
       reference space
    phi = shape_functions_1D(xi_q)  # Evaluate shape functions
       at quadrature point
    #print(phi)

    # Compute Uh at quadrature point
    Uh_q = np.dot(U_local, phi)  # U_h = sum_j U_j * phi_j
    #print(Uh_q)
    # Compute nonlinear term
    nonlinear_termU = (Uh_q ** 4)
    #print(nonlinear_termU)
    VF_sum = 0

    #nonlinear term for the jacobian
    nonlinear_term_jacU = 4*(Uh_q)**3


    for elem2, edge2 in enumerate(radiation_edges_W):
        J_nonlinear_2_e = np.zeros((2,2))
        n1_W, n2_W = edge2  # Get global node indices for this
           boundary edge

        W_local = np.array([W[n1_W], W[n2_W]])
        Wh_q = np.dot(W_local, phi)
        #print(Wh_q)
        nonlinear_termW = (Wh_q ** 4)

        #derivative wrt w
        nonlinear_term_jacW = 4*(Wh_q**3)
        J_nonlinear_2_e[:] += -w*sigma *epsilon *
           nonlinear_term_jacW * np.outer(phi,phi) * J_edge* VF[
           elem1,elem2+ne_u]#*view factor
        #print(J_nonlinear_2_e)

        if VF[elem1,elem2+ne_u] == 0:
            print("VF is zero!!")
            print('u: e_u:', elem1, 'e_w:', elem2)

        VF_sum += (nonlinear_termU - nonlinear_termW)* VF[elem1,
           elem2+ne_u]#*view factor
        J_nonlinear_e[:] += w*sigma *epsilon *
           nonlinear_term_jacU * np.outer(phi,phi) * J_edge* VF[
           elem1,elem2+ne_u]

        # it should be in the rows of the nodes of elements e
           and the columns of element l
        J_nonlinear_2[n1, n1_W] += J_nonlinear_2_e[0, 0]
        J_nonlinear_2[n1, n2_W] += J_nonlinear_2_e[0, 1]
```

```
                J_nonlinear_2[n2, n1_W] += J_nonlinear_2_e[1, 0]
                J_nonlinear_2[n2, n2_W] += J_nonlinear_2_e[1, 1]

            R_e[:] += w * sigma * epsilon * VF_sum * phi[:] * J_edge  #
                Integral sum



        # Assemble local residual into global residual vector
        R[n1] += R_e[0]
        R[n2] += R_e[1]
        if 0> elem1 <3:
            print(R)
        J_nonlinear[n1, n1] += J_nonlinear_e[0, 0]
        J_nonlinear[n1, n2] += J_nonlinear_e[0, 1]
        J_nonlinear[n2, n1] += J_nonlinear_e[1, 0]
        J_nonlinear[n2, n2] += J_nonlinear_e[1, 1]

    return R, J_nonlinear, J_nonlinear_2

def radiation_bc_with_connectionW(W, U, num_nodes_W, num_nodes_U,
    radiation_edges_W, radiation_edges_U, nodes_W, nodes_U, sigma,
    epsilon, VF, ne_u, ne_w):
    """
    Compute the nonlinear vector R(U,W) for geometry A(!!).

    Parameters:
    - U: solution vector
    - num_nodes: include the numbers of nodes for the corresponding
       geometry
    - radiation_edges: nodes of the edges which have radiation ([n1, n2
       ])
    - nodes: nodes of the corresponding geometry

    Returns:
    - R: Nonlinear residual vector
    - J_nonlinear: The Jacobian of R_2 with respect to w
    - J_nonlinear_2: The Jacobian of C_2 with respect to u
    """
    R = np.zeros(num_nodes_W)  # Initialize  vector
    J_nonlinear_Rw = np.zeros((num_nodes_W,num_nodes_W))
    J_nonlinear_Cu = np.zeros((num_nodes_W,num_nodes_U))

    # 1D linear shape functions in reference space (for edge with two
       nodes)
    def shape_functions_1D(xi):
        return np.array([(1 - xi) / 2, (1 + xi) / 2])
    # loop over elements on geometry 1 with U
    for elem2, edge2 in enumerate(radiation_edges_W):
        n1, n2 = edge2  # Get global node indices for this boundary edge
        x1, y1 = nodes_W[n1]
        x2, y2 = nodes_W[n2]

        # Compute 1D Jacobian (edge length in physical space)
        J_edge = 0.5 * np.sqrt((x2 - x1) ** 2 + (y2 - y1) ** 2)
```

```
# Extract local U values
W_local = np.array([W[n1], W[n2]])

# Initialize local residual contribution
R_e = np.zeros(2)
J_nonlinear_Rw_e = np.zeros((2,2))



# Perform quadrature integration over the boundary edge
for q, w in enumerate(quadrature_weights1d):
    xi_q = quadrature_points1d[q]  # Quadrature point in
       reference space
    phi = shape_functions_1D(xi_q)  # Evaluate shape functions
       at quadrature point
    #print(phi)

    # Compute Uh at quadrature point
    Wh_q = np.dot(W_local, phi)  # U_h = sum_j U_j * phi_j
    #print(Wh_q)

    # Compute nonlinear term
    nonlinear_termW = (Wh_q ** 4)
    VF_sum = 0

    #nonlinear term for the jacobian
    nonlinear_term_jacW = 4*(Wh_q)**3



    for elem1, edge1 in enumerate(radiation_edges_U):
        J_nonlinear_Cu_e = np.zeros((2,2))
        n1_U, n2_U = edge1  # Get global node indices for this
           boundary edge
        U_local = np.array([U[n1_U], U[n2_U]])
        Uh_q = np.dot(U_local, phi)
        nonlinear_termU = (Uh_q ** 4)

        #derivative wrt w
        nonlinear_term_jacU = 4*(Uh_q**3)
        J_nonlinear_Cu_e[:] += -w*sigma *epsilon *
           nonlinear_term_jacU * np.outer(phi,phi) * J_edge* VF[
           elem2+ne_u,elem1]#*view factor
        if VF[elem2+ne_u,elem1] == 0:
            print('w: e_u:', elem1, 'e_w:', elem2)
            print("VF is zero!!")

        VF_sum += (nonlinear_termW - nonlinear_termU)* VF[elem2+
           ne_u,elem1]#*view factor

        J_nonlinear_Rw_e[:] += w*sigma *epsilon *
           nonlinear_term_jacW * np.outer(phi,phi) * J_edge * VF
           [elem2+ne_u,elem1]

        J_nonlinear_Cu[n1, n1_U] += J_nonlinear_Cu_e[0, 0]
        J_nonlinear_Cu[n1, n2_U] += J_nonlinear_Cu_e[0, 1]
```

```
                J_nonlinear_Cu[n2, n1_U] += J_nonlinear_Cu_e[1, 0]
                J_nonlinear_Cu[n2, n2_U] += J_nonlinear_Cu_e[1, 1]

            # Contribution to local residual
            #print(w * nonlinear_term * phi * J_edge)
            R_e[:] += w * sigma * epsilon * VF_sum * phi[:] * J_edge  #
                Integral sum

            # this does not go good yet

            # it should be in the rows of the nodes of elements e and
                the columns of element l


        # Assemble local residual into global residual vector
        R[n1] += R_e[0]
        R[n2] += R_e[1]

        J_nonlinear_Rw[n1, n1] += J_nonlinear_Rw_e[0, 0]
        J_nonlinear_Rw[n1, n2] += J_nonlinear_Rw_e[0, 1]
        J_nonlinear_Rw[n2, n1] += J_nonlinear_Rw_e[1, 0]
        J_nonlinear_Rw[n2, n2] += J_nonlinear_Rw_e[1, 1]

        #plt.figure(figsize=(6,6))
        #plt.spy(J_nonlinear_Cu)
        #plt.show()

        #plt.figure(figsize=(6,6))
        #plt.spy(J_nonlinear_Rw)
        #plt.show()
    return R, J_nonlinear_Rw, J_nonlinear_Cu
```

Appendix C

Code: ANSYS snippets

Listing C.1: Python script for FEM radiation term

```
import wbjn
dpn = wbjn.ExecuteCommand(ExtAPI, "returnValue(a+Parameters.
    GetActiveDesignPoint().Name)", a = "DP")


def after_post(this, solution):# Do not edit this line
    """
    Called after post processing.
    Keyword Arguments :
        this -- the datamodel object instance of the python code object
            you are currently editing in the tree
        solution -- Solution
    """
    export_dir = r"C:\Users\nbagchus\SIMS_DATA_NN"

    # Get solution result objects
    solu = Model.Analyses[0].Solution #solution
    all_results = solu.Children
    # Time-step result indices


    #a = 99

    # Loop through and export
    for i in range(5,26):
        result = all_results[i]
        #filename = os.path.join(export_dir, str(i) + "flxA_2.txt")
        filemame = export_dir + "\\TEMPA_" + str(dpn) + "_" + str(i-4) +
            ".txt"
        #print(f"Exporting time step {i} ? {filename}")
        result.ExportToTextFile(filemame)
    for i in range(26,47):
        result = all_results[i]
        #filename = os.path.join(export_dir, str(i) + "flxA_1.txt")
        filemame = export_dir + "\\TEMPB_" + str(dpn) + "_" + str(i-25)
            + ".txt"
        #print(f"Exporting time step {i} ? {filename}")
        result.ExportToTextFile(filemame)
```

```
    for i in range(47,68):
        result = all_results[i]
        #filename = os.path.join(export_dir, str(i) + "flxA_1.txt")
        filemame = export_dir + "\\RADA_" + str(dpn) + "_" + str(i-46) +
            ".txt"
        #print(f"Exporting time step {i} ? {filename}")
        result.ExportToTextFile(filemame)
    for i in range(68,89):
        result = all_results[i]
        #filename = os.path.join(export_dir, str(i) + "flxA_1.txt")
        filemame = export_dir + "\\RADB_" + str(dpn) + "_" + str(i-67) +
            ".txt"
        #print(f"Exporting time step {i} ? {filename}")
        result.ExportToTextFile(filemame)
    # To access properties created using the Property Provider, please
       use the following command.
    # this.GetCustomPropertyByPath("your_property_group_name/
       your_property_name")

    # To access scoping properties use the following to access geometry
       scoping and named selection respectively:
    # this.GetCustomPropertyByPath("your_property_group_name/
       your_property_name/Geometry Selection")
    # this.GetCustomPropertyByPath("your_property_group_name/
       your_property_name/Named Selection")

    # To access properties created using the Property Provider, please
       use the following command.
    # this.GetCustomPropertyByPath("your_property_group_name/
       your_property_name")

    # To access scoping properties use the following to access geometry
       scoping and named selection respectively:
    # this.GetCustomPropertyByPath("your_property_group_name/
       your_property_name/Geometry Selection")
    # this.GetCustomPropertyByPath("your_property_group_name/
       your_property_name/Named Selection")

    pass
```

Listing C.2: APDL snippet changing View Factor

```
/PREP7
!the element type number for surf251/252 generated from radiation bc can
    be extracted from checking the input file (ds.dat)
et_rsurf = 3
keyopt,et_rsurf,1,1
/SOLU
vfup,define,on
```