Acceleration of POD models with polynomial nonlinearities

Oscar Mauricio Agudelo[a,c], Jairo José Espinosa[b], Bart De Moor[a]

[a] *Department of Electrical Engineering (ESAT), Research Group SCD-SISTA,Katholieke Universiteit Leuven, Kasteelpark Arenberg 10, B-3001 Heverlee (Leuven), Belgium. E-mail:* {*mauricio.agudelo, bart.demoor*}*@esat.kuleuven.be*

[b] *Facultad de Minas, Universidad Nacional de Colombia, Carrera 80 No. 65-223, Medellin, Colombia. E-mail: jairo.espinosa@ieee.org*

[c] *Department of Automation and Electronics, Universidad Autónoma de Occidente, Calle 25 No 115-85, Cali, Colombia.*

Abstract

This paper presents a technique for accelerating the evaluation of nonlinear POD models with polynomial nonlinearities. Proper Orthogonal Decomposition (POD) and Galerkin projection are two well-known techniques that have been successfully employed for reducing the high-dimensionality of the discretized systems used to approximate Partial differential equations (PDEs). Although a large model-order reduction can be obtained with these techniques, the computational saving is small when we have to deal with nonlinear or Linear Time Variant (LTV) models. Nevertheless, if the high-dimensional systems are input affine and their nonlinearities are of polynomial type or can be easily approximated by low degree polynomials, then we can exploit the polynomial nature of the resulting POD models for generating compact and efficient formulations that are computationally less intensive. Moreover, if we use the feature selection theory for choosing the most relevant monomials of these POD models, we can significantly increase the computational gain. This is the approach that is followed in this article. Additionally, this paper discusses how to guarantee the local stability of POD models with polynomial nonlinearities.

Key words: Distributed Parameter systems, Model Reduction, Polynomial approximation.

1. Introduction

Proper Orthogonal Decomposition (POD), also known as Karhunen-Loève expansion or Principal Component Analysis (PCA), is a technique that has been applied in many physical systems modeled by Partial Differential Equations (PDEs) for deriving reduced order models. POD is a data driven method where a suitable set of orthonormal basis vectors are derived from simulation or experimental data. These basis vectors, which are organized in order of relevance, capture the spatial dynamics of the systems. The reduced order models are typically obtained by projecting (Galerkin projection) the high-dimensional models on the most relevant basis vectors.

Although we can reach a large model-order reduction with the POD technique, the computational saving offered by the reduced order models is small when nonlinear or Linear Time Variant (LTV) systems are considered. The reason of this limitation lies in the fact that we need the full spatial information from the original high-dimensional systems in order to evaluate the reduced order models. In [1, 2, 3] a method known as Missing Point Estimation (MPE) is proposed for tackling this problem. In this method the Galerkin projection is conducted only on some pre-selected state variables instead of the entire set. The remaining state variables are estimated by means of the POD basis vectors. It has been reported that this technique can save considerable computational effort. In [4], a Multi-Layer Perceptron (MLP) neural network is used to approximate the nonlinear vector function of the POD model of a heat transfer process in a one-dimensional bar. Given that the time for evaluating the trained MLP can be very short, we can save a substantial amount of time. However one of the main drawbacks of this approach is the local minima problem during the training of the neural network.

In this paper we propose a method for speeding up the evaluation of nonlinear POD models derived from input-affine high-dimensional systems with polynomial nonlinearities. It turns out that by exploiting the polynomial nature of these POD models, we can construct compact and efficient formulations

which save computation effort. Besides, by using sequential feature selection algorithms for choosing their most relevant monomials (suboptimal solution), we can notably increase the computational saving. Notice that this approach might also be applied to high-dimensional systems with non-polynomial nonlinearities, provided that these nonlinearities can be approximated by low degree polynomials.

Usually, the model reduction of an stable model by POD often results in a reduced order model that is stable as well. However, this is generally not guaranteed [5]. By using the Lyapunov's indirect method we can obtain an eigenvalue constraint that can be used for guaranteeing the local stability of a POD model with polynomial nonlinearities. Since this constraint leads to a non-smooth optimization problem, which is difficult to solve in its original form, in this paper we also present two manners of dealing with this constraint. In both cases, the local stability is guaranteed independently of the data used, although the quality of the reduced order model is still depending on the quality of the data.

This paper is organized as follows. Section 2 presents a description of the dynamical system that will be used to explain our technique, the nonlinear heat transfer problem in a one-dimensional bar. In Section 3 the derivation of a reduced order model for the bar by means of POD and Galerkin projection is discussed. Section 4 introduces our technique for accelerating the nonlinear POD model found in Section 3 by exploiting its polynomial nature. In Section 5 we discuss how the local stability of POD models with polynomial nonlinearities can be guaranteed. Section 6 shows some validation and simulation results, and finally Section 7 summarizes and concludes the paper.

2. Heat Transfer in a One-dimensional Bar

The system under study is the silicon bar shown in Figure 1. The bar has attached an actuator which provides a uniformly distributed heat flux $u(t)$ between $z = z_{\mathrm{a}}$ and $z = z_{\mathrm{b}}$. Additionally, an external heat flux $d(t)$ is applied

uniformly along the bar whose ends are kept at 25°C (ambient temperature) all the time.

If only temperature variations in the z-direction are considered, the dynamics of the temperature $T(z,t)$ of the bar can be modeled by the following nonlinear PDE:

$$\rho C_{\mathrm{p}} \frac{\partial T(z,t)}{\partial t} = \frac{\partial}{\partial z}\left(\kappa\left(T(z,t)\right)\frac{\partial T(z,t)}{\partial z}\right) + V(z,t) \tag{1}$$

with the following initial and boundary conditions:

$$\begin{aligned} T(z,t=0) &= 25^{\circ}\mathrm{C} \\ T(z=0,t) &= T(z=L,t) = 25^{\circ}\mathrm{C}. \end{aligned}$$

Here ρ is the material density in $[\mathrm{Kg}\cdot\mathrm{m}^{-3}]$, C_{p} is the heat capacity in $[\mathrm{J}\cdot\mathrm{Kg}^{-1}\cdot\mathrm{K}^{-1}]$, $\kappa\left(T\right)$ is the temperature dependent heat conductivity in $[\mathrm{J}\cdot\mathrm{s}^{-1}\cdot\mathrm{m}^{-1}\cdot\mathrm{K}^{-1}]$, t is the time in [s], z is the spatial coordinate in [m] and $V(z,t)$ is the heat source applied to the bar at position z and time t in $[\mathrm{W}\cdot\mathrm{m}^{-3}]$. $V(z,t)$ is defined as follows:

$$V(z,t) = \begin{cases} d(t)+u(t), & z_{\mathrm{a}} \le z \le z_{\mathrm{b}} \\ d(t), & \text{elsewhere.} \end{cases}$$

The relation between the temperature and the heat conductivity $\kappa\left(T\right)$ is described by a polynomial of degree 3,

$$\kappa(T) = \kappa_0 + \kappa_1 T + \kappa_2 T^2 + \kappa_3 T^3 \tag{2}$$

where $\kappa_0 = 36$, $\kappa_1 = -0.1116$, $\kappa_2 = 1.7298\times10^{-4}$ and $\kappa_3 = -1.78746\times10^{-7}$ are real coefficients in the appropriated units.

The length of the bar is $L = 0.1$ m and the remaining numerical values of the model parameters are: $\rho = 3970\ \mathrm{Kg}\cdot\mathrm{m}^{-3}$, $C_{\mathrm{p}} = 766\ \mathrm{J}\cdot\mathrm{Kg}^{-1}\cdot\mathrm{K}^{-1}$, $z_{\mathrm{a}} = 0.005$ m and $z_{\mathrm{b}} = 0.04$ m.

The operating ranges in $[\mathrm{W}\cdot\mathrm{m}^{-3}]$ of $d(t)$ and $u(t)$ are $-500\cdot10^3 \le d(t) \le 500\cdot10^3$ and $-1500\cdot10^3 \le u(t) \le 1500\cdot10^3$ respectively. Some of the previous

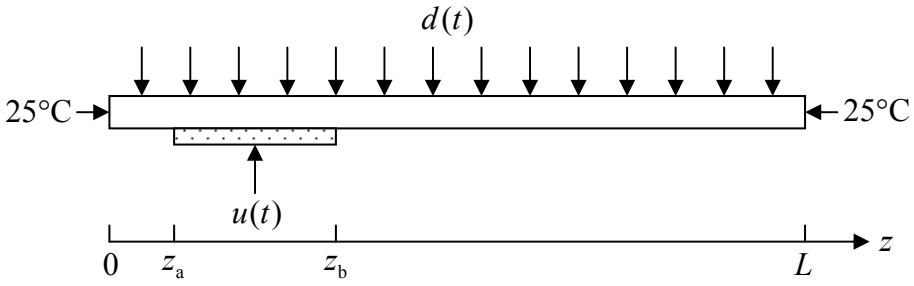

Figure 1: Silicon Bar.

numerical values were inspired on the values given in [6] .

For simulation purposes, it is necessary to reduce the infinity dimensionality of (1) by discretizing the spatial domain. To this end, the partial derivatives with respect to space were replaced by backward (the inner spatial derivative) and forward (the outer spatial derivative) difference approximations. This is equivalent to replace the second partial derivative with respect to space by a central difference approximation in the linear version of the heat equation where κ is kept constant. The discretized model of the bar is given by the following set of nonlinear ordinary differential equations:

$$\frac{dT_i}{dt} = c_1\Big(\kappa(T_{i+1})T_{i+1} - \big(\kappa(T_{i+1}) + \kappa(T_i)\big)T_i + \\ + \kappa(T_i)T_{i-1}\Big) + c_2 V_i \tag{3a}$$

$$\text{for } i = 1, \ldots, N-1,$$

with

$$c_1 = \frac{1}{\rho C_{\mathrm{p}} \left(\Delta z\right)^2} \\ c_2 = \frac{1}{\rho C_{\mathrm{p}}},\ T_0 = T_N = 25^{\circ}\mathrm{C} \tag{3b}$$

where N is the number of sections in which the bar is divided, Δz is the length of each section, and T_i and V_i are the temperature and heat flux at the point

$z_i = i\Delta z$.

If we define $\mathbf{T}(t) \in \mathbb{R}^{N-1} = [T_1(t), T_2(t), \ldots, T_{N-1}(t)]^T$ as the vector containing the temperature of the grid points from z_1 till z_{N-1} every time instant, we can write equation (3) as follows:

$$\dot{\mathbf{T}}(t) = \mathbf{F}\left(\mathbf{T}(t)\right) + \mathbf{B}_1 d(t) + \mathbf{B}_2 u(t) \tag{4}$$

where $\mathbf{F}\left(\mathbf{T}(t)\right) : \mathbb{R}^{N-1} \rightarrow \mathbb{R}^{N-1}$ is a vector-valued or vector function which contains the nonlinear terms of the model and whose ith component function is given by

$$F_i(\mathbf{T}(t)) = c_1\Big(\kappa(T_{i+1}(t))T_{i+1}(t) - \big(\kappa(T_{i+1}(t)) + \kappa(T_i(t))\big)T_i(t) + \kappa(T_i(t))T_{i-1}(t)\Big),$$

and $\mathbf{B}_1$ and $\mathbf{B}_2$ are vectors defined as: $\mathbf{B}_1 \in \mathbb{R}^{N-1} = [c_2, c_2, \ldots, c_2]^T$, $\mathbf{B}_2 \in \mathbb{R}^{N-1} = [0, \ldots, 0, c_2, \ldots, c_2, 0, \ldots, 0]^T$. The position of the nonzero elements in $\mathbf{B}_2$ corresponds to the position of the grid points that are in contact with the actuator.

The spatial domain was divided into $N = 500$ sections which means that (4) has $N - 1 = 499$ states. With such amount of states the design of a control system for the bar is not an easy task. In addition, the simulation of (4) demands a considerable amount of computational resources. In order to tackle this situation, in the next section a reduced order model (few number of equations and states) of the bar will be obtained by means of POD and Galerkin Projection.

3. Nonlinear POD model of the system

Given that the initial state of (4) does not provide information about the system dynamics, we are going to work with the temperature deviations with respect to the ambient temperature (25°C). Consequently, the vector $\mathbf{T}(t)$ is

split as follows:

$$\mathbf{T}(t) = \mathbf{T}^{\Delta}(t) + \mathbf{T}^{*}$$

where $\mathbf{T}^{\Delta}(t) \in \mathbb{R}^{N-1}$ is the vector containing the deviations of the temperature profile and $\mathbf{T}^{*} \in \mathbb{R}^{N-1}$ is a constant vector which contains the initial temperature profile of the bar (ambient temperature).

In proper orthogonal decomposition (POD) [7, 8, 9, 10, 11], we start by observing that $\mathbf{T}^{\Delta}(t) \in \mathbb{R}^{N-1}$ can be expanded as a sum of orthonormal basis vectors:

$$\mathbf{T}^{\Delta}(t) = \sum_{j=1}^{N-1} a_j(t)\boldsymbol{\varphi}_j \tag{5}$$

where $\boldsymbol{\varphi}_j \in \mathbb{R}^{N-1}\ \forall j = 1, \ldots, N-1$ is a set of orthonormal basis vectors (POD basis vectors or POD basis functions) in the discretized spatial domain, and $a_j(t) \in \mathbb{R}\ \forall j = 1, \ldots, N-1$ are the time-varying coefficients, or POD coefficients, associated to each basis vector. These POD basis vectors are ordered according to their relevance to $\mathbf{T}^{\Delta}(t)$.

The main dynamics of the system can be represented using the first n most relevant basis vectors, since $\boldsymbol{\varphi}_1, \boldsymbol{\varphi}_2, \ldots, \boldsymbol{\varphi}_n$ condensates the main spatial correlations. An nth order approximation of (5) is then given by means of the truncated sequence

$$\mathbf{T}_n^{\Delta}(t) = \sum_{j=1}^{n} a_j(t)\boldsymbol{\varphi}_j, \quad n \ll N-1. \tag{6}$$

An approximate (reduced order) model of $\mathbf{T}^{\Delta}(t)$ can be derived by building a model for the first n POD coefficients. This is the essence of model reduction by POD.

The POD basis vectors are determined from simulation or experimental data of the process. The dynamic model for the first n POD coefficients is commonly found by means of the Galerkin projection [1][11, 12, 13]. For the specific case of the linear systems, subspace identification techniques can be used for finding such a model [14][15].

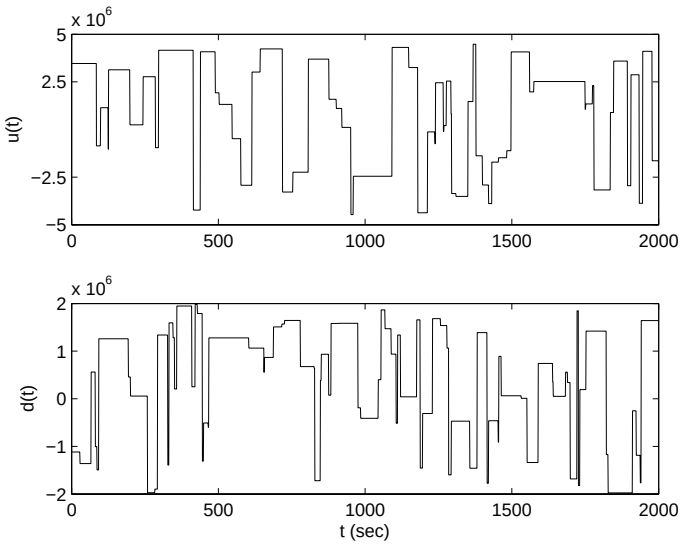

Figure 2: Pseudo Random Multilevel Noise Signals (PRMNS) used in the generation of the snapshot matrix $\mathbf{T}_{\text{snap}}$. Amplitudes in $[\text{W} \cdot \text{m}^{-3}]$.

In the next subsections we present in detail the steps followed for deriving the reduced order model of the bar.

3.1. Generation of the snapshot Matrix

We have built the snapshot matrix $\mathbf{T}_{\text{snap}} \in \mathbb{R}^{499 \times 2001}$ by collecting the evolution of the deviations of the temperature profile when Pseudo Random Multilevel Noise Signals (PRMNS) were applied to the process inputs $u(t)$ and $d(t)$,

$$\mathbf{T}_{\text{snap}} = \left[\mathbf{T}^{\Delta}(0), \mathbf{T}^{\Delta}(\Delta t), \ldots, \mathbf{T}^{\Delta}\left((N_{\text{d}} - 1)\Delta t\right)\right].$$

Here, $N_{\text{d}} = 2001$ is the number of samples gathered using a sampling time Δt of 1 s. The excitation signals can be observed in Figure 2. A commutation probability of 3% was set for the signals and the amplitudes in $[\text{W} \cdot \text{m}^{-3}]$ of $d(t)$ and $u(t)$ were restricted to the intervals $[-500 \cdot 10^3 \;\; 500 \cdot 10^3]$ and $[-1500 \cdot 10^3 \;\; 1500 \cdot 10^3]$ respectively.

3.2. Derivation of the POD basis Vectors

The POD basis vectors were derived by calculating the singular value decomposition (SVD) of $\mathbf{T}_{\text{snap}}$,

$$\mathbf{T}_{\text{snap}} = \mathbf{\Phi}\mathbf{\Sigma}\mathbf{\Psi}^T$$

where $\mathbf{\Phi} \in \mathbb{R}^{499\times 499}$ and $\mathbf{\Psi} \in \mathbb{R}^{2001\times 2001}$ are unitary matrices, and $\mathbf{\Sigma} \in \mathbb{R}^{499\times 2001}$ is a matrix containing the singular values of $\mathbf{T}_{\text{snap}}$ in a decreasing order on its main diagonal. The left singular vectors, that is the columns of $\mathbf{\Phi}$,

$$\mathbf{\Phi} \in \mathbb{R}^{499\times 499} = [\boldsymbol{\varphi}_1, \boldsymbol{\varphi}_2, \ldots, \boldsymbol{\varphi}_{499}]$$

are the POD basis vectors.

3.3. Selection of the most relevant POD basis vectors

We have made the selection by checking the singular values of $\mathbf{T}_{\text{snap}}$ (the larger the singular value the more relevant the basis vector is) and the truncation degree P_n of (6), which is defined as follows:

$$P_n = \frac{\sum_{j=1}^{n} \sigma_j^2}{\sum_{j=1}^{N-1} \sigma_j^2}; \quad n = 1, \ldots, N-1 \tag{7}$$

where σ_j is the jth singular value of $\mathbf{T}_{\text{snap}}$. The number of POD basis vectors can be chosen such that the fraction of the first singular values in (7) is large enough to capture most information in the data [1]. An ad-hoc criterion commonly applied is that n has to be determined for $P_n = 0.99$ [16]. The closer P_n to 1, or similarly the closer $1 - P_n$ to 0, the better the approximation of $\mathbf{T}_{\text{snap}}$ will be. In this problem, the first $n = 6$ basis vectors were selected. The 6th order approximation of $\mathbf{T}^{\Delta}(t)$ is given by

$$\mathbf{T}_n^{\Delta}(t) = \sum_{j=1}^{n} a_j(t)\boldsymbol{\varphi}_j = \mathbf{\Phi}_n \mathbf{a}(t), \tag{8}$$

where $\mathbf{\Phi}_n = [\boldsymbol{\varphi}_1, \ldots, \boldsymbol{\varphi}_6]$ and $\mathbf{a}(t) = [a_1(t), \ldots, a_6(t)]^T$.

3.4. Construction of the model for the first $n = 6$ POD coefficients

In order to derive a dynamic model for the POD coefficients, we have used the Galerkin projection. If we define a residual function $R(\mathbf{T}, \dot{\mathbf{T}})$ for equation (4) as follows:

$$R(\mathbf{T}, \dot{\mathbf{T}}) = \dot{\mathbf{T}}(t) - \mathbf{F}(\mathbf{T}(t)) - \mathbf{B}_1 d(t) - \mathbf{B}_2 u(t) \tag{9}$$

and we replace $\mathbf{T}(t)$ by its nth order approximation $\mathbf{T}_n(t) = \mathbf{T}_n^{\Delta} + \mathbf{T}^*$ in (9), the projection of $R(\mathbf{T}_n, \dot{\mathbf{T}}_n)$ on the space spanned by the basis vectors $\mathbf{\Phi}_n$ shall vanish. That is,

$$\left\langle R(\mathbf{T}_n, \dot{\mathbf{T}}_n), \boldsymbol{\varphi}_j \right\rangle = 0; \quad j = 1, \ldots, n = 6 \tag{10}$$

where $\langle ., . \rangle$ denotes the Euclidean inner product. If we replace $\mathbf{T}(t)$ by its nth order approximation $\mathbf{T}_n(t) = \mathbf{\Phi}_n \mathbf{a}(t) + \mathbf{T}^*$ in equation (4), and we apply the inner product criterion (10) to the resulting equation,

$$\mathbf{\Phi}_n^T \mathbf{\Phi}_n \dot{\mathbf{a}}(t) = \mathbf{\Phi}_n^T \mathbf{F}\left(\mathbf{\Phi}_n \mathbf{a}(t) + \mathbf{T}^*\right) + \mathbf{\Phi}_n^T \mathbf{B}_1 d(t) + \mathbf{\Phi}_n^T \mathbf{B}_2 u(t)$$

$$\dot{\mathbf{a}}(t) = \mathbf{\Phi}_n^T \mathbf{F}\left(\mathbf{\Phi}_n \mathbf{a}(t) + \mathbf{T}^*\right) + \mathbf{\Phi}_n^T \mathbf{B}_1 d(t) + \mathbf{\Phi}_n^T \mathbf{B}_2 u(t)$$

then we can find the reduced order model of the bar with only 6 states,

$$\dot{\mathbf{a}}(t) = \mathbf{\Phi}_n^T \mathbf{F}\left(\mathbf{\Phi}_n \mathbf{a}(t) + \mathbf{T}^*\right) + \tilde{\mathbf{B}}_1 d(t) + \tilde{\mathbf{B}}_2 u(t) \tag{11}$$

$$\mathbf{T}_n(t) = \mathbf{\Phi}_n \mathbf{a}(t) + \mathbf{T}^*$$

where $\tilde{\mathbf{B}}_1 = \mathbf{\Phi}_n^T \mathbf{B}_1$ and $\tilde{\mathbf{B}}_2 = \mathbf{\Phi}_n^T \mathbf{B}_2$. Finally if we define a new vector-valued function $\mathbf{f} : \mathbb{R}^6 \to \mathbb{R}^6$ as $\mathbf{f}\left(\mathbf{a}(t)\right) = \mathbf{\Phi}_n^T \mathbf{F}\left(\mathbf{\Phi}_n \mathbf{a}(t) + \mathbf{T}^*\right)$, then the reduced order model of the bar can be written more compactly as follows:

$$\dot{\mathbf{a}}(t) = \mathbf{f}\left(\mathbf{a}(t)\right) + \tilde{\mathbf{B}}_1 d(t) + \tilde{\mathbf{B}}_2 u(t) \tag{12a}$$

$$\mathbf{T}_n(t) = \mathbf{\Phi}_n \mathbf{a}(t) + \mathbf{T}^*. \tag{12b}$$

In general, it should be clear that we do not know the compact expression of $\mathbf{f}(\mathbf{a}(t))$ in (12a). So, in order to simulate the reduced order model, the ODE solver has to evaluate indirectly $\mathbf{f}(\mathbf{a}(t))$. Firstly, the solver has to map the state of the reduced order model $\mathbf{a}(t)$ into the original high-dimensional space by means of this linear transformation $\mathbf{T}_n^{\Delta}(t) = \mathbf{\Phi}_n \mathbf{a}(t)$. Secondly it has to evaluate the resulting high-dimensional state vector $\mathbf{T}_n^{\Delta}(t)$ in the vector function $\mathbf{F}(\mathbf{T}_n^{\Delta}(t) + \mathbf{T}^*)$ of (4), and finally it has to map the results of this evaluation to the low dimensional space by pre-multiplying them by $\mathbf{\Phi}_n^T$. The evaluation of $\mathbf{f}(\mathbf{a}(t))$ is done as many times as it is required by the ODE solver within each integration step. Hence the indirect evaluation of $\mathbf{f}(\mathbf{a}(t))$ is the bottleneck that limits severely the computational gain of the nonlinear POD model.

In order to cope with this problem, in a previous study [4], we used a Multi-Layer Perceptron (MLP) neural network for approximating the vector function $\mathbf{f}(\mathbf{a}(t))$. Given that the time required for evaluating the trained MLP can be very short, a significant saving of computational time can be obtained. In this approach both the computational gain and the capability of learning accurately complex nonlinear mappings are limited by the size of the network. The larger the MLP, the lesser is the computational gain but the better is the ability of learning complex nonlinear mappings and vice versa. Therefore, the size of the net has to be chosen in such a way that provides a good trade-off between accuracy and computational gain. One of the main difficulties of this approach is to find a suitable number of hidden neurons for the MLP. Additionally, since the neural networks suffer of the local minima problem, the training of the MLP has to be performed several times starting from different initial conditions in order to make sure that the network is not trapped in a local minimum. However the convergence to the global minimum can not be guaranteed. This problem gets worse as the number of neurons with nonlinear activation functions increases.

Another aspect that we have to bear in mind, is that the approximation of the vector function offered by the neural network is only valid within an specific range. Beyond this range the approximation fails, and the POD model based on the MLP is not reliable anymore.

In the following section we propose an alternative way of speeding up the evaluation of the POD model (12) by exploiting its polynomial nature.

4. Polynomial POD model

We start by noticing that the mth component function of $\mathbf{f}(\mathbf{a}(t)) = \mathbf{\Phi}_n^T \mathbf{F}(\mathbf{T}_n(t))$ can be written as

$$f_m(\mathbf{a}(t)) = \sum_{i=1}^{N-1} \varphi_{i,m} F_i(\mathbf{T}_n(t)) \tag{13}$$

where $\varphi_{i,m}$ is the ith entry of the basis vector $\boldsymbol{\varphi}_m$, $N = 500$ is the number of sections in which the bar was divided, $n = 6$ is the number of selected basis vectors, $\mathbf{T}_n(t)$ is the nth order approximation of $\mathbf{T}(t)$ and F_i is the ith component function of $\mathbf{F}(\mathbf{T}_n(t))$ defined as follows:

$$\begin{aligned} F_i(\mathbf{T}_n(t)) = c_1 \Big(& \kappa(\tilde{T}_{i+1}(t))\tilde{T}_{i+1}(t) - \big(\kappa(\tilde{T}_{i+1}(t)) + \\ & + \kappa(\tilde{T}_i(t))\big)\tilde{T}_i(t) + \kappa(\tilde{T}_i(t))\tilde{T}_{i-1}(t) \Big), \end{aligned} \tag{14}$$

where c_1 is a constant given by (3b), $\kappa(T) : \mathbb{R} \to \mathbb{R}$ is the scalar polynomial function defined in (2) and $\tilde{T}_i(t)$ is the ith entry of the vector $\mathbf{T}_n(t)$ that is given by

$$\tilde{T}_i(t) = \sum_{j=1}^{n} \varphi_{i,j} a_j(t) + T_i^* \tag{15}$$

where $\varphi_{i,j}$ is the ith entry of the basis vector $\boldsymbol{\varphi}_j$, a_j is the jth POD coefficient and T_i^* is the ith entry of $\mathbf{T}^*$ which contains the initial temperature profile of the bar. If we replace (15) in (14),

$$F_i\left(\mathbf{a}(t)\right) = c_1 \left(\kappa \left(\sum_{j=1}^{n} \varphi_{(i+1),j} a_j(t) + T_{i+1}^* \right) \times \right.$$

$$\times \left(\sum_{j=1}^{n} \varphi_{(i+1),j} a_j(t) + T^*_{i+1} \right) -$$

$$- \left(\kappa \left(\sum_{j=1}^{n} \varphi_{(i+1),j} a_j(t) + T^*_{i+1} \right) + \kappa \left(\sum_{j=1}^{n} \varphi_{i,j} a_j(t) + T^*_i \right) \right) \times$$

$$\times \left(\sum_{j=1}^{n} \varphi_{i,j} a_j(t) + T^*_i \right) +$$

$$+ \kappa \left(\sum_{j=1}^{n} \varphi_{i,j} a_j(t) + T^*_i \right) \left(\sum_{j=1}^{n} \varphi_{(i-1),j} a_j(t) + T^*_{i-1} \right) \Bigg) \tag{16}$$

then we obtain the component function F_i in terms of the POD coefficients $a_j(t)\ \forall j = 1, \ldots, n$ associated to the selected basis vectors. Notice that this function is a multivariate polynomial of degree $d_{\mathrm{p}} = 4$ in $a_j(t)\ \forall j = 1, \ldots, n$. Since $f_m(\mathbf{a}(t))$ (see (13)) is a linear combination of the functions $F_i(\mathbf{a}(t))\ \forall i = 1, \ldots, N-1$, then $f_m(\mathbf{a}(t))$ is also a multivariate polynomial of degree $d_{\mathrm{p}} = 4$ in $a_j(t)\ \forall j = 1, \ldots, n$.

After replacing (16) in (13) and simplifying the resulting expression, we obtain the following general representation for $f_m(\mathbf{a}(t))$,

$$\begin{aligned} f_m(\mathbf{a}(t)) = w_{m,0} + w_{m,1} a_1(t) + \cdots + w_{m,n} a_n(t) + \\ + w_{m,(n+1)} a_1^2(t) + w_{m,(n+2)} a_1 a_2 + \cdots + w_{m,(r-1)} a_n^{d_{\mathrm{p}}} \end{aligned} \tag{17}$$

where $w_{m,i}$ is the real coefficient of the ith monomial ($i = 0, 1, 2, \ldots, r-1$) of the function $f_m(\mathbf{a}(t))$, d_{p} is the largest possible degree for each monomial and r is the number of monomials which is given by the following formula [17]:

$$r = \sum_{j=0}^{d_{\mathrm{p}}} \left(\begin{array}{c} n+j-1 \\ j \end{array} \right) = \sum_{j=0}^{d_{\mathrm{p}}} \frac{(n+j-1)!}{j!\,(n-1)!}\,. \tag{18}$$

For this particular problem, the number of monomials per component function is equal to 210. The monomials (without coefficients) $a_{i_1} a_{i_2} \cdots a_{i_s}$ of degree s

$(1 \leq s \leq d_{\mathrm{p}})$ in (17) are built from the set $\mathcal{C}$,

$$\begin{aligned} \mathcal{C} = \big\{ \langle i_1, i_2, \cdots, i_s \rangle : 1 \leq i_1, i_2, \ldots, i_s \in \mathbb{Z}^+ \leq n, \\ i_1 \leq i_2 \leq \cdots \leq i_s \big\} \end{aligned} \tag{19}$$

which contains the possible combinations of the indexes $i_1, i_2, \ldots, i_s$.

By using the general representation of $\mathbf{f}(\mathbf{a}(t))$ defined in (17) through its component functions, we can recast the POD model (12) as follows:

$$\begin{aligned} \dot{\mathbf{a}}(t) &= \mathbf{f}\left(\mathbf{a}(t)\right) + \tilde{\mathbf{B}}_1 d(t) + \tilde{\mathbf{B}}_2 u(t) \\ \mathbf{T}_n(t) &= \mathbf{\Phi}_n \mathbf{a}(t) + \mathbf{T}^* \end{aligned} \tag{20}$$

with

$$\begin{aligned} \mathbf{f}\left(\mathbf{a}(t)\right) &= \left[f_1\left(\mathbf{a}(t)\right), \ldots, f_m\left(\mathbf{a}(t)\right), \ldots, f_n\left(\mathbf{a}(t)\right) \right]^T \\ f_m(\mathbf{a}(t)) &= w_{m,0} + w_{m,1} a_1(t) + \cdots + w_{m,n} a_n(t) + \\ &\quad + w_{m,(n+1)} a_1^2(t) + w_{m,(n+2)} a_1 a_2 + \cdots + w_{m,(r-1)} a_n^{d_{\mathrm{p}}}. \end{aligned}$$

From now on, this compact representation of (12) will be referred to as Polynomial POD model or P-POD model. Note that now the evaluation of $\mathbf{f}$ does not require the use of the high-dimensional vector function $\mathbf{F}$ anymore, and consequently we can expect to evaluate $\mathbf{f}$ much faster.

In general, if the discretized system used to approximate the PDE or PDEs is input affine and its nonlinearity is of polynomial type, then the vector function $\mathbf{f}(\mathbf{a}(t))$ can be represented efficiently by multivariate polynomials in terms of the POD coefficients. Notice that we might generate polynomial POD models from discretized systems with non-polynomial nonlinearities, provided that these nonlinearities can be approximated by low degree polynomials.

4.1. Calculation of the coefficients of the polynomials

If we define a vector $\mathbf{y}(t) \in \mathbb{R}^n$ as follows:

$$\mathbf{y}(t) = \dot{\mathbf{a}}(t) - \tilde{\mathbf{B}}_1 d(t) - \tilde{\mathbf{B}}_2 u(t) \tag{21}$$

then (12a) can be cast as $\mathbf{y}(t) = \mathbf{f}(\mathbf{a}(t))$.

The coefficients of the polynomials $f_m(\mathbf{a}(t))$ $\forall m = 1, \ldots, n$ in (20) are computed in such a way that the fitting to the data generated by (21) at the time instants $t = k\Delta t$, for $k = 0, 1, 2, \ldots, N_{\mathrm{d}-1}$,

$$\mathbf{y}_{N_\mathrm{d}} \in \mathbb{R}^{n \cdot N_d} = \begin{bmatrix} \mathbf{y}(0) \\ \mathbf{y}(1) \\ \vdots \\ \mathbf{y}(k) \\ \vdots \\ \mathbf{y}(N_\mathrm{d} - 1) \end{bmatrix}$$

is optimal in the least squares sense. The vector $\mathbf{y}_{N_\mathrm{d}}$ is built from the data gathered during the snapshots experiment (see Section 3.1) where N_d samples of the temperature profile were collected. This vector is assembled as follows. From the snapshots experiment we know the values of $u(k) = u(t = k\Delta t)$ and $d(k) = d(t = k\Delta t)$ for $k = 0, 1, 2, \ldots, N_\mathrm{d} - 1$, and from the SVD of the snapshot matrix we have the evolution of $\mathbf{a}(k) = \mathbf{a}(t = k\Delta t)$ for $k = 0, 1, 2, \ldots, N_\mathrm{d} - 1$ (obtained by taking the first $n = 6$ rows of $\mathbf{\Sigma}\mathbf{\Psi}^T$). By using (12a) and the values of $u(k)$, $d(k)$ and $\mathbf{a}(k)$ for $k = 0, 1, 2, \ldots, N_\mathrm{d} - 1$, we calculate $\dot{\mathbf{a}}(k) = \dot{\mathbf{a}}(t = k\Delta t)$ and afterwards $\mathbf{y}(k) = \mathbf{y}(t = k\Delta t)$ by means of (21) for the same time instants.

Notice that the mth component function of $\mathbf{f}(\mathbf{a}(t))$ defined at the sampling times $t = k\Delta t$ (with $k = 0, 1, 2, \ldots$) can be written as follows:

$$f_m\left(\mathbf{a}(k)\right) = f_m\left(\mathbf{a}(t = k\Delta t)\right) = \boldsymbol{\alpha}(k)\tilde{\mathbf{w}}_m \tag{22}$$

where $\boldsymbol{\alpha}(k) \in \mathbb{R}^{1\times r}$ is a row vector defined as

$$\boldsymbol{\alpha}(k) = \Big[1, a_1(k), a_2(k), \ldots, a_n(k), a_1^2(k), a_1(k)a_2(k), \ldots, a_n^2(k), \ldots, a_1^{d_{\mathrm{P}}}(k), a_1^{d_{\mathrm{P}}-1}(k)a_2(k), \ldots, a_n^{d_{\mathrm{P}}}(k)\Big],$$

and $\tilde{\mathbf{w}}_m \in \mathbb{R}^r$

$$\tilde{\mathbf{w}}_m = \big[w_{m,0}, w_{m,1}, w_{m,2}, \ldots, w_{m,(r-1)}\big]^T$$

is a vector containing the coefficients of the monomials. From here the vector function $\mathbf{f}(\mathbf{a}(k)) = \mathbf{f}(\mathbf{a}(t = k\Delta t))$ can be compactly formulated in this way

$$\mathbf{f}(\mathbf{a}(k)) = \boldsymbol{\Lambda}(k)\mathbf{w} \tag{23}$$

where

$$\boldsymbol{\Lambda}(k) \in \mathbb{R}^{n\times(r\cdot n)} = \begin{bmatrix} \boldsymbol{\alpha}(k) & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\alpha}(k) & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & \boldsymbol{\alpha}(k) \end{bmatrix},$$

and

$$\mathbf{w} \in \mathbb{R}^{r\cdot n} = \begin{bmatrix} \tilde{\mathbf{w}}_1 \\ \tilde{\mathbf{w}}_2 \\ \vdots \\ \tilde{\mathbf{w}}_m \\ \vdots \\ \tilde{\mathbf{w}}_n \end{bmatrix}$$

is a vector containing all the coefficients that we want to find.

The predictions of (23) regarding $\mathbf{y}(k)$ for $k = 0, 1, 2, \ldots, N_{\mathrm{d}} - 1$, are given by the following expression

$$\hat{\mathbf{y}}_{N_{\mathrm{d}}} = \boldsymbol{\Omega}\mathbf{w} \tag{24}$$

where $\hat{\mathbf{y}}_{N_\mathrm{d}} \in \mathbb{R}^{n \cdot N_\mathrm{d}}$ and $\mathbf{\Omega} \in \mathbb{R}^{(n \cdot N_\mathrm{d}) \times (r \cdot n)}$ are defined as

$$\hat{\mathbf{y}}_{N_\mathrm{d}} = \begin{bmatrix} \mathbf{f}(\mathbf{a}(0)) \\ \mathbf{f}(\mathbf{a}(1)) \\ \vdots \\ \mathbf{f}(\mathbf{a}(k)) \\ \vdots \\ \mathbf{f}(\mathbf{a}(N_\mathrm{d}-1)) \end{bmatrix}, \; \mathbf{\Omega} = \begin{bmatrix} \mathbf{\Lambda}(0) \\ \mathbf{\Lambda}(1) \\ \vdots \\ \mathbf{\Lambda}(k) \\ \vdots \\ \mathbf{\Lambda}(N_\mathrm{d}-1) \end{bmatrix}.$$

As it was mentioned before, the coefficients $\mathbf{w}$ of the vector function $\mathbf{f}$ are found by means of the least squares method in which the minimization problem

$$\min_{\mathbf{w}} \; J = \left(\mathbf{y}_{N_\mathrm{d}} - \hat{\mathbf{y}}_{N_\mathrm{d}}\right)^T \left(\mathbf{y}_{N_\mathrm{d}} - \hat{\mathbf{y}}_{N_\mathrm{d}}\right) \tag{25}$$

is solved. We can obtain an equivalent optimization problem in terms of $\mathbf{w}$ by substituting (24) in (25) and ignoring the resulting constant term,

$$\min_{\mathbf{w}} \; \frac{1}{2}\mathbf{w}^T \left(\mathbf{\Omega}^T\mathbf{\Omega}\right)\mathbf{w} - \left(\mathbf{y}_{N_\mathrm{d}}^T\mathbf{\Omega}\right)\mathbf{w}. \tag{26}$$

As it is well-known, the solution of this minimization problem can be found by solving this equation: $\left(\mathbf{\Omega}^T\mathbf{\Omega}\right)\mathbf{w} = \mathbf{\Omega}^T\mathbf{y}_{N_\mathrm{d}}$. In this study this equation is solved by using the QR (orthogonal, triangular) decomposition of $\mathbf{\Omega}^T\mathbf{\Omega}$ and back substitution. This procedure is numerically more stable than inverting the matrix product $\mathbf{\Omega}^T\mathbf{\Omega}$.

Finally, it is important to remark that in the calculation of the monomial coefficients the data ($\mathbf{a}(k)$, $\mathbf{y}(k)$, for $k = 0, 1, 2, \ldots, N_d - 1$) was normalized first (by dividing by the standard deviation) in order to avoid numerical problems. Consequently, at the moment of evaluating the derivative of the P-POD model we have to carry out the appropriated conversions. For the sake of clarity and simplicity in our derivations, we did not include the normalization part in the previous equations.

4.2. Reduction of the number of monomials

From (18) it is clear that the number of monomials of the P-POD model can be very large. In fact, this number increases exponentially with the number of POD coefficients, compromising the computational gain of the P-POD model. We can tackle this situation by properly selecting a reduced set of monomials whose combination provides a good approximation of the vector function $\mathbf{f}(\mathbf{a}(t))$. To this end, in this section we propose the use of the sequential feature selection methods [18, 19, 20, 21]. Here we will use the term "features" for referring to as the "monomials" of $\mathbf{f}\left(\mathbf{a}(t)\right)$.

The sequential feature selection methods have basically two components: the objective function, called the criterion, which the method seeks to minimize over all feasible feature subsets, and the sequential search algorithm. The algorithm adds or removes features from a candidate subset while evaluating the criterion. In general, an exhaustive evaluation of all possible feature subsets is infeasible, since we have to deal with $2^r = 2^{210} = 1.64 \times 10^{63}$ candidates. This is why a suboptimal search strategy is necessary for directing the feature selection process as it explores the space of all possible combination of features. Representative examples of sequential search algorithms include the Sequential Forward Selection (SFS), the Sequential Backward Selection (SBS), the Plus-L Minus-R Selection (LRS) and the Bidirectional Search (BDS) among others [18]. The first two search strategies, SFS and SBS, are the simplest greedy search algorithms that we can use. In the sequential forward selection, the features are sequentially added to an empty candidate set until a predetermined number of features has been added. In contrast, in the sequential backward selection, the features are sequentially removed from a full candidate set until a predefined number of features has been eliminated.

Let us define the set $\mathcal{S}$ as the set containing the positions of the selected features or monomials from vector $\boldsymbol{\alpha}(k)$ at certain moment. We will denote the vector that contains these features as $\boldsymbol{\alpha}^{\mathcal{S}}(k)$ and the matrix $\boldsymbol{\Omega}$ (see Section 4.1) constructed from $\boldsymbol{\alpha}^{\mathcal{S}}(k)$ and the data points as $\boldsymbol{\Omega}^{\mathcal{S}}$. If we designate the vector comprising the coefficients of the selected features as $\mathbf{w}^{\mathcal{S}}$, then the predictions

$\hat{\mathbf{y}}^{\mathcal{S}}_{N_\mathrm{d}} \in \mathbb{R}^{n \cdot N_\mathrm{d}}$ of the P-POD model with a reduced set of monomials about $\mathbf{y}(k)$ for $k = 0, 1, 2, \ldots, N_d - 1$, can be written as

$$\hat{\mathbf{y}}^{\mathcal{S}}_{N_\mathrm{d}} = \mathbf{\Omega}^{\mathcal{S}} \mathbf{w}^{\mathcal{S}}. \tag{27}$$

In this study, we use as objective function $J^{\mathcal{S}}$ or criterion the Sum Squared Error (SSE) between $\hat{\mathbf{y}}^{\mathcal{S}}_{N_\mathrm{d}}$ and $\mathbf{y}_{N_\mathrm{d}}$,

$$J^{\mathcal{S}} = \left(\mathbf{y}_{N_\mathrm{d}} - \hat{\mathbf{y}}^{\mathcal{S}}_{N_\mathrm{d}}\right)^T \left(\mathbf{y}_{N_\mathrm{d}} - \hat{\mathbf{y}}^{\mathcal{S}}_{N_\mathrm{d}}\right) \tag{28}$$

and as a search algorithm the sequential forward selection. We chose SFS instead of SBS because SFS is computationally less demanding due to the fact that the criterion function $J^{\mathcal{S}}$ is evaluated over smaller subsets of monomials.

For the evaluation of each candidate feature subset, a K-fold cross-validation scheme is used by the feature selection algorithm. In K-fold cross-validation the available data (the observations or data points are the rows of $\mathbf{y}_{N_\mathrm{d}}$ and $\mathbf{\Omega}^{\mathcal{S}}$) is partitioned in K samples. Of the K samples, a single sample is retained as the validation data for evaluating the candidate feature subset and computing its corresponding SSE, and the remaining $K - 1$ samples (training data) are used for calculating the coefficients of the monomials via least squares. The cross-validation process is then repeated K times (the folds), with each of the K samples used exactly once as the validation data. In order to generate a single goodness-of-fit measure of the candidate, the K SSEs are summed and the result divided by the number of observations or data points. Once this goodness-of-fit measure has been calculated for all the candidate feature subsets, the algorithm picks the subset with the smallest measure.

The advantage of using K-fold cross-validation is that all the observations in the dataset are eventually used for both training and validation. For this problem a 10-fold cross-validation scheme was used.

In the next section it will be shown that the local stability around the origin of the P-POD models depends only on the linear terms, and consequently

the algorithm was properly configured to include these terms in the candidate feature subsets.

After running the feature selection algorithm, $N_{\text{sm}} = 25$ features or monomials were finally chosen ($J^{\mathcal{S}} = 1.3727 \cdot 10^{-4}$). These selected features are pointed out by the set $\mathcal{S}^*$, which contains the suboptimal solution found by the algorithm. Once this set is established, the coefficients of the monomials $\mathbf{w}^{\mathcal{S}^*} \in \mathbb{R}^{N_{\text{sm}} \cdot n}$ are computed via least squares.

Lastly, the polynomial POD model with a reduced set of monomials, which will be referred to as P-POD-RS model, has the following form,

$$\dot{\mathbf{a}}(t) = \mathbf{f}^{\mathcal{S}^*}(\mathbf{a}(t)) + \tilde{\mathbf{B}}_1 d(t) + \tilde{\mathbf{B}}_2 u(t) \tag{29}$$

$$\mathbf{T}_n(t) = \mathbf{\Phi}_n \mathbf{a}(t) + \mathbf{T}^*$$

with

$$\mathbf{f}^{\mathcal{S}^*}(\mathbf{a}(t)) = \mathbf{\Lambda}^{\mathcal{S}^*}(t)\mathbf{w}^{\mathcal{S}^*}$$

$$\mathbf{\Lambda}^{\mathcal{S}^*}(t) \in \mathbb{R}^{n \times (N_{\text{sm}} \cdot n)} = \begin{bmatrix} \boldsymbol{\alpha}^{\mathcal{S}^*}(t) & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\alpha}^{\mathcal{S}^*}(t) & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & \boldsymbol{\alpha}^{\mathcal{S}^*}(t) \end{bmatrix}$$

where

$$\begin{aligned} \boldsymbol{\alpha}^{\mathcal{S}^*}(t) = \big[& a_1(t), a_2(t), a_3(t), a_4(t), a_5(t), a_6(t), a_1^2(t), \\ & a_1(t)a_2(t), a_1(t)a_3(t), a_1(t)a_4(t), a_1(t)a_5(t), a_1(t)a_6(t), \\ & a_2^2(t), a_2(t)a_3(t), a_2(t)a_4(t), a_2(t)a_5(t), a_2(t)a_6(t), \\ & a_3(t)^2, a_3(t)a_4(t), a_3(t)a_5(t), a_3(t)a_6(t), a_4^2(t), \\ & a_1(t)^2 a_2(t), a_1^4(t), a_1^2(t)a_2(t)a_6(t) \big] \end{aligned}$$

contains the 25 selected monomials.

5. Polynomial POD model with stability guarantee

A property that is desirable to preserve in the model reduction process is the stability of the original model. In this aspect, the model reduction of an stable model by POD often results in a reduced model that is stable as well. Nevertheless, this is generally not guaranteed. Eventually we can end up with an unstable model, specially when the quality of the available data is poor [5].

For explaining our methods, in this section we are going to use the polynomial POD model with the full set of monomials described by (20). However, these methods can also be applied in a very straightforward way to the polynomial POD model with a reduced set of monomials (see (29)).

Since global stability of a nonlinear system is in most cases hard or even impossible to prove, this study will be focused on the local stability of the P-POD model (20) around the origin. The stability of (20) is analyzed through its autonomous counterpart,

$$\dot{\mathbf{a}}(t) = \breve{\mathbf{f}}(\mathbf{a}(t)) = \mathbf{f}(\mathbf{a}(t))\big|_{w_{1,0}=w_{2,0}=\cdots=w_{n,0}=0}. \tag{30}$$

which is found by making $u(t)$, $d(t)$ and the constant terms $w_{1,0}, w_{2,0}, \ldots, w_{n,0}$ of the vector function $\mathbf{f}(\mathbf{a}(t))$ equal to zero. Notice that the constant monomials can be seen as the entries of an additional vector $\tilde{\mathbf{B}}_{\mathbf{3}} = [w_{1,0}, w_{2,0}, \ldots, w_{n,0}]^T$ multiplied by an extra input $u_{\mathrm{e}}(t)$ equal to one.

We can derive a stability condition for the autonomous system (30) by using the Lyapunov's indirect method. According to this method, the stability of the origen in (30) is inferred from the stability of the linearized system,

$$\delta\dot{\mathbf{a}}(t) = \mathbf{A}\delta\mathbf{a}(t). \tag{31}$$

Here $\delta\mathbf{a}(t) \in \mathbb{R}^n$ is the deviation variable and $\mathbf{A} \in \mathbb{R}^{n\times n}$ is the jacobian matrix

defined as follows:

$$\mathbf{A} = \left.\frac{\partial \breve{\mathbf{f}}}{\partial \mathbf{a}}\right|_{\mathbf{a}=\mathbf{0}} = \left.\begin{bmatrix} \frac{\partial \breve{f}_1}{\partial a_1} & \frac{\partial \breve{f}_1}{\partial a_2} & \cdots & \frac{\partial \breve{f}_1}{\partial a_n} \\ \frac{\partial \breve{f}_2}{\partial a_1} & \frac{\partial \breve{f}_2}{\partial a_2} & \cdots & \frac{\partial \breve{f}_2}{\partial a_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \breve{f}_n}{\partial a_1} & \frac{\partial \breve{f}_n}{\partial a_2} & \cdots & \frac{\partial \breve{f}_n}{\partial a_n} \end{bmatrix}\right|_{\mathbf{a}=\mathbf{0}}$$

$$= \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,n} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n,1} & w_{n,2} & \cdots & w_{n,n} \end{bmatrix}.$$

According to the Lyapunov's indirect method, the origin is asymptotically stable if $\mathbf{A}$ is Hurtwitz, i.e. all the eigenvalues of $\mathbf{A}$ have a negative real part, $\mathrm{Re}[\lambda_i(\mathbf{A})] < 0$, $\forall i = 1, \ldots, n$. Given that the jacobian matrix $\mathbf{A}$ is made of the coefficients of the linear terms, it is clear that these coefficients are the only ones that affect the stability of the P-POD model around the origin. If we add the eigenvalue constraint $\mathrm{Re}[\lambda_i(\mathbf{A}(\mathbf{w}))] < 0$ to the minimization problem (26), then we end up with the following non-convex optimization problem:

$$\min_{\mathbf{w}} \ \frac{1}{2}\mathbf{w}^T \left(\mathbf{\Omega}^T\mathbf{\Omega}\right)\mathbf{w} - \left(\mathbf{y}_{N_\mathrm{d}}^T\mathbf{\Omega}\right)\mathbf{w} \tag{32a}$$

subject to

$$\mathrm{Re}[\lambda_i(\mathbf{A}(\mathbf{w}))] < 0, \quad \text{for } i = 1, \ldots, n. \tag{32b}$$

The eigenvalue constraint makes problem (32) non-smooth [22], which in general is difficult to solve in its current form [23]. In order to overcome this situation, in this section we present two manners of dealing with this constraint. In our first approach, we replace the eigenvalue constraint by an LMI, which is a relaxation of the original constraint, providing a sufficient condition for the local stability of the P-POD model. In our second approach, the eigenvalue constraint is substituted by a nonlinear matrix equality and an LMI. Unlike the

first approach, these replacements give sufficient and necessary conditions for the local stability of the reduced order model. In both cases, the local stability is guaranteed independently of data used. Nevertheless, the quality of the reduced oder model still depends on the data used in the model reduction process.

5.1. Semidefinite problem formulation

A relaxation of the eigenvalue constraint (32b) is given in the following lemma:

Lemma 1 ([5]). *Let* $\mathbf{A}$ *be a square matrix. If the Hermitian part of* $\mathbf{A}$*, i.e.* $\frac{1}{2}\left(\mathbf{A}+\mathbf{A}^{\mathrm{H}}\right)$*, is negative definite, then* $\mathbf{A}$ *is Hurwitz.*

Proof. See [5].

By using Lemma 1, we can modify the optimization problem (32) as follows:

$$\min_{\mathbf{w}} \ \frac{1}{2}\mathbf{w}^T\left(\mathbf{\Omega}^T\mathbf{\Omega}\right)\mathbf{w} - \left(\mathbf{y}_{N_\mathrm{d}}^T\mathbf{\Omega}\right)\mathbf{w} \tag{33a}$$

subject to

$$-\frac{1}{2}\left(\mathbf{A}(\mathbf{w})+\mathbf{A}(\mathbf{w})^T\right)-\mu\mathbf{I}\succeq 0 \tag{33b}$$

where $\mu>0$ is a positive number used to guarantee the strictly positive definiteness of $-\frac{1}{2}\left(\mathbf{A}(\mathbf{w})+\mathbf{A}(\mathbf{w})^T\right)$. The inclusion of μ is necessary given that most solvers do not work with strict positive/negative definite constraints. The optimization problem (33) is nothing else than a SDP problem that can be solved efficiently by interior point methods, such as the one implemented in the Matlab Toolbox Sedumi [24].

It should be clear that not all the Hurtwitz matrices have negative-definite hermitian parts (the Lemma 1 only provides a sufficient condition for the local stability of the model), and therefore the stability constraint (33b) might be a little restrictive.

In the following subsection a non-restrictive replacement of the eigenvalue constraint (32b) will be discussed.

Table 1: Numerical Results

Model	Order $n = 3$		Order $n = 4$	
	J	λ_i	J	λ_i
P-POD-NC	-81	-2.8236 -0.6355 $+0.1010$	-173.1056	-0.4729 $-2.4805 \pm 1.4922i$ -29.1026
P-POD-SDP	-80.0536	-3.0384 -2.0838 -0.6233	-168.9984	-0.5544 $-4.0173 \pm 1.2011i$ -30.0831
P-POD-NSDP	-80.9989	-2.8239 -0.6347 -0.0046	-173.1056	-0.4729 $-2.4805 \pm 1.4922i$ -29.1026

J is the cost function value at the solution of the optimization problems (26), (33) and (34).

λ_i are the eigenvalues of the state matrix of the P-POD model.

$\mu = \tilde{\mu} = 10^{-8}$.

5.2. Nonlinear Semidefinite problem formulation

Alternatively as it was done in the previous subsection, we can substitute the eigenvalue constraint (32b) by the well-known Lyapunov 's matrix equality (this equality is derived from the application of the Lyapunov's direct method on linear time-invariant systems) presented in the next theorem.

Theorem 1. *Given the autonomous system,* $\delta\dot{\mathbf{a}}(t) = \mathbf{A}\delta\mathbf{a}(t)$*, the origin* $\delta\mathbf{a}(t) = 0$ *is asymptotically stable if and only if, for any symmetric positive definite matrix* $\mathbf{Q}$*, there exists a symmetric positive definite matrix* $\mathbf{P}$ *such that,*

$$\mathbf{A}^T\mathbf{P} + \mathbf{P}\mathbf{A} + \mathbf{Q} = \mathbf{0}.$$

Theorem 1 gives a sufficient and necessary condition for the stability of the P-POD model around the origin. By using this theorem we can adapt the

minimization problem (32) in the following way,

$$\min_{\mathbf{w},\mathbf{P}} \ \frac{1}{2}\mathbf{w}^T\left(\mathbf{\Omega}^T\mathbf{\Omega}\right)\mathbf{w} - \left(\mathbf{y}_{N_\mathrm{d}}^T\mathbf{\Omega}\right)\mathbf{w} \tag{34a}$$

subject to

$$\mathbf{A}(\mathbf{w})^T\mathbf{P} + \mathbf{P}\mathbf{A}(\mathbf{w}) + \mathbf{Q} = \mathbf{0} \tag{34b}$$

$$\mathbf{P} - \tilde{\mu}\mathbf{I} \succeq 0 \tag{34c}$$

where $\mathbf{Q} \in \mathbb{S}_{++}^n$ is a given positive definite matrix, and $\tilde{\mu}$ is a positive number used to guarantee the strictly positive definiteness of $\mathbf{P}$. Problem (34) is a nonlinear SDP problem which in general is non-convex. More precisely, we have an optimization problem that involves a Bilinear Matrix Inequality (BMI), $\mathbf{A}(\mathbf{w})^T\mathbf{P} + \mathbf{P}\mathbf{A}(\mathbf{w}) \prec 0$ or $\mathbf{A}(\mathbf{w})^T\mathbf{P} + \mathbf{P}\mathbf{A}(\mathbf{w}) + \tilde{\mu}\mathbf{I} \preceq 0$. We can solve problem (34) by using the Matlab toolbox PENBMI, which implements a general-purpose solver for BMI problems. The algorithm implemented in PENBMI is described in [25] and it is based on the augmented lagrangian method. It can be viewed as a generalization to nonlinear semidefinite problems of the penalty-barrier-multiplier method originally introduced in [26] for convex optimization [27].

In order to get a feasible starting point ($\mathbf{w}_0$, $\mathbf{P}_0$) for the BMI solver, and properly set the constraints (34b) and (34c), we can solve the convex optimization problem stated in (33). Thus, the starting point and constraints parameters might be fixed as follows:

- $\mathbf{w}_0$ is obtained directly of solving (33)
- $\mathbf{P}_0 = 0.5 \cdot \mathbf{I}$
- $\mathbf{Q} = -0.5\left(\mathbf{A}_0 + \mathbf{A}_0^T\right)$, where $\mathbf{A}_0$ is constructed from $\mathbf{w}_0$
- $0 < \tilde{\mu} \leq 0.5$.

Unlike (33b), the stability constraint given by (34b) and (34c) is non-restrictive, but leads to a non-convex optimization problem.

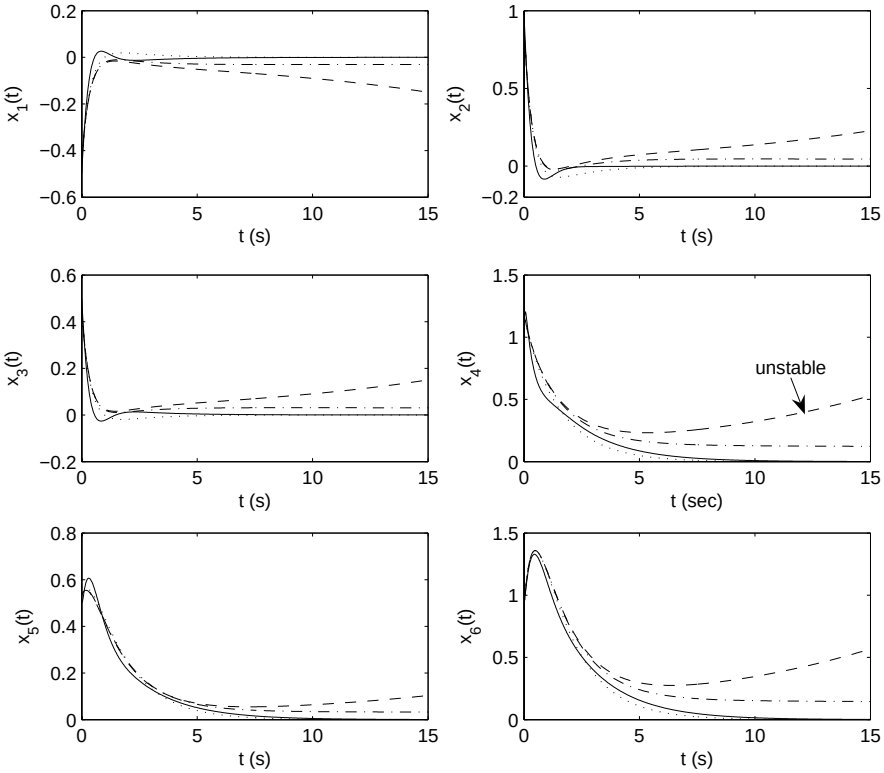

Figure 3: Response of the POD models of order 3 to the initial conditions $\mathbf{x}(t=0) = [-0.5, 1, 0.5, 1, 0.5, 1]^T$. Solid line - Full order model. Dashed line - P-POD-NC model. Dotted line - P-POD-SDP model. Dash-dotted line - P-POD-NSDP model.

5.3. Numerical example

Consider the following autonomous linear system (from [5]),

$$\dot{\mathbf{x}}(t) = \tilde{\mathbf{A}}\mathbf{x}(t) \tag{35}$$

where

$$\tilde{\mathbf{A}} = \begin{bmatrix} -2.3 & 0.5 & 0 & 0 & 2.3 & -0.8 \\ 1.5 & -2.5 & 0 & 0 & -4.5 & 1.5 \\ -22.8 & -0.5 & -25 & 0 & -2.3 & 0.8 \\ -33 & -0.6 & 15.6 & -25.8 & -2.6 & 13.5 \\ 8.4 & 1.0 & 3.7 & 2.8 & -0.5 & -1.4 \\ 25 & 0.9 & 11.1 & 8.4 & 3.9 & -6.0 \end{bmatrix}.$$

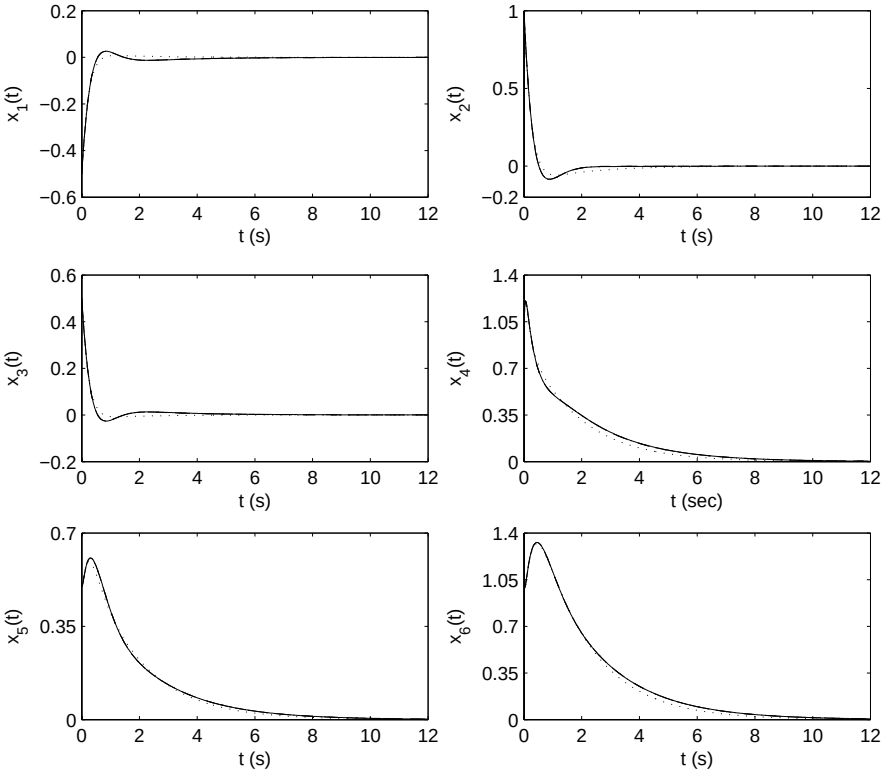

Figure 4: Response of the POD models of order 4 to the initial conditions $\mathbf{x}(t=0) = [-0.5, 1, 0.5, 1, 0.5, 1]^T$. Solid line - Full order model. Dashed line - P-POD-NC model. Dotted line - P-POD-SDP model. Dash-dotted line - P-POD-NSDP model.

The eigenvalues of $\tilde{\mathbf{A}}$ are -0.4704, -1.5865, $-2.5335 \pm 1.5514i$, -24.9998 and -29.9763 and clearly the model is stable. In this example we are interested in deriving a reduced order model (a P-POD model with only linear terms) of (35) by using POD. To this end, we first built the snapshot matrix,

$$\mathbf{X}_{\text{snap}} = \big[\mathbf{x}(0), \mathbf{x}(t = \Delta t), \ldots, \mathbf{x}(t = 480\Delta t)\big], \ \Delta t = 0.025\,\text{s}$$

from the evolution of the state vector $\mathbf{x}(t)$ starting from the initial condition $\mathbf{x}(t=0) = [-0.5, 1, 0.5, 1, 0.5, 1]^T$. Afterwards, we derived the POD basis vec-

tors,

$$\mathbf{\Phi} = \begin{bmatrix} -0.0401 & 0.3354 & 0.1237 & -0.2280 & 0.5646 & 0.7070 \\ 0.0573 & -0.7177 & 0.1541 & 0.3907 & 0.5525 & 0.0016 \\ 0.0402 & -0.3364 & -0.1234 & 0.2224 & -0.5661 & 0.7072 \\ 0.4758 & -0.3934 & -0.2819 & -0.7335 & 0.0370 & -0.0031 \\ 0.3238 & -0.0222 & 0.9099 & -0.1388 & -0.2180 & -0.0010 \\ 0.8138 & 0.3225 & -0.1959 & 0.4343 & 0.0820 & 0.0020 \end{bmatrix}$$

by computing the SVD of $\mathbf{X}_{\text{snap}}$. The singular values of $\mathbf{X}_{\text{snap}}$ and their associated $1-P_n$ (see equation (7)) values are : $\sigma_1 = 12.7308$ $(1-P_1 = 5.21\cdot 10^{-2})$, $\sigma_2 = 2.9377$ $(1-P_2 = 1.6\cdot 10^{-3})$, $\sigma_3 = 0.4285$ $(1-P_3 = 5.49\cdot 10^{-4})$, $\sigma_4 = 0.3059$ $(1-P_4 = 1.858\cdot 10^{-6})$, $\sigma_5 = 0.0178$ $(1-P_5 = 4.878\cdot 10^{-11})$, and $\sigma_6 = 9.133\cdot 10^{-5}$ $(1-P_6 = 0)$.

From the previous values, it is clear that the subspace spanned by the first 3 or 4 basis vectors contains most of the "energy" in the data. Therefore, we derived POD models of orders 3 and 4 by solving the optimization problems (26), (33) and (34). The POD model without stability constraint obtained by solving (26) or by using Galerkin will be referred to as P-POD-NC. The POD models that include the SDP and NSDP stability constraints will be referred to as P-POD-SDP and P-POD-NSDP respectively.

Table 1 presents the eigenvalues of the state matrix of each POD model as well as the cost function value at the solution of the associated optimization problem. In addition, Figures 3 and 4 show the response of the POD models of order 3 and 4 to the initial conditions $\mathbf{x}(t=0) = [-0.5, 1, 0.5, 1, 0.5, 1]^T$ respectively. Note that for generating the time response of the models we are using the same initial conditions that the ones used in the generation of the snapshot matrix. It should be clear that we could use different initial conditions, but in that case we can not expect accurate predictions from the POD models since their derivation is based on data that only consider the trajectory of the states when the initial conditions are the mentioned ones (the data is not rich enough).

For the case of $n = 3$, only the P-POD-NC model is unstable due to its eigenvalue at 0.1010 that is outside the stability region. By comparing the

eigenvalues of the P-POD-NC and P-POD-NSDP models, we can see that the unstable pole 0.1010 of P-POD-NC is replaced by an stable one (-0.0046) in the P-POD-NSDP case (the other eigenvalues are practically the same). However, this pole makes the model dynamics very slow and this condition could lead to a misinterpretation of Figure 3, since the steady state values of the states in the P-POD-NSDP case (dash-dotted line), seem to be different from zero. In reality the states go very slowly towards 0 in approximately 1000 s. Notice also that the cost function value in the P-POD-NSDP case is smaller than in the POD-SDP case, but from Figure 3 it is evident that the P-POD-SDP model offers a better approximation of the original system. These observations might look contradictory, but we have to bear in mind that the cost function value only indicates how well the polynomials fit the data, and not how accurate the POD model will be.

From Table 1 it is clear that the cost function value in the P-POD-SDP case is larger than in the P-POD-NC and P-POD-NSDP cases. Furthermore from Figure 4 we can see some small differences in the responses of P-POD-SDP and (35). These differences are not observed in the P-POD-NC and P-POD-NSDP cases where the response of both these models and the original system (35) are practically overlapping. All these results are due to the restrictive nature of the SDP constraint. Nevertheless, it is important to keep in mind that this constraint is only imposed on the linear terms. Therefore if we are dealing with higher degree P-POD models, the limitation in the search space would be less significant, since the optimization algorithm would have several unconstrained optimization variables to play with.

Although the NSDP constraint is non restrictive, its drawback is to lead to a non-convex optimization where we have to deal with the local minima problem. When $n = 4$ we can see that the P-POD-NSDP and P-POD-NC models are identical. However, in general this is not always the case, and we can end up with different models since the optimization with the NSDP constraint might be trapped in a local minimum.

The approaches considered in this section for guaranteeing the local stability

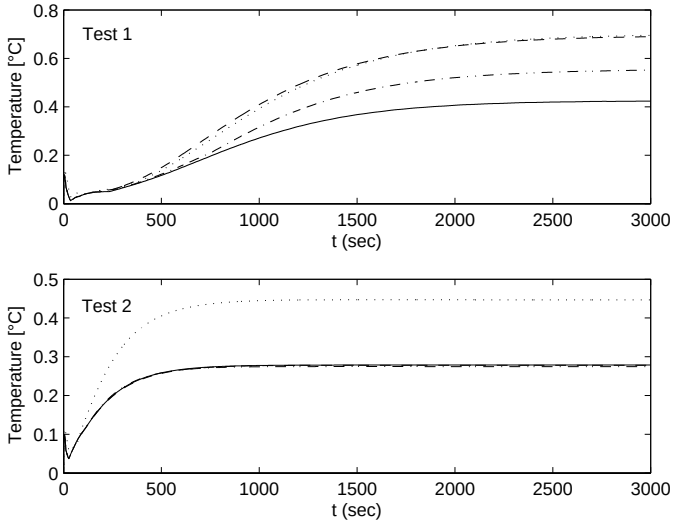

Figure 5: Maximum temperature deviation of the POD models with respect to the full order model along Tests 1 and 2. Solid line - POD model and P-POD model. Dashed line - Neural-POD model. Dash-dotted line - P-POD model with a reduced set of monomials. Dotted line - MPE-POD model.

of polynomial POD models, demand much more computational burden than the calculation of polynomial POD models by using least squares. Particularly the calculation of the P-POD model with the NSDP constraint is the most demanding case, and this fact might limit its practical applicability when the number of monomials increases as a consequence of augmenting either the number of selected POD basis vectors or the degree of the polynomials.

6. Validation and Simulation Results

In this section we will validate and compare the reduced order models of the bar that were presented in Sections 3, and 4, as well as the POD model based on an MLP proposed in [4], and a POD model derived using the Missing Point Estimation (MPE) technique. This technique, which is introduced in [2, 1], is used for accelerating nonlinear and linear time variant POD models.

The MLP devised in [4] for approximating the vector function $\mathbf{f}(\mathbf{a}(t))$ has 1 hidden layer with 10 hidden neurons. The input and output layers have 6

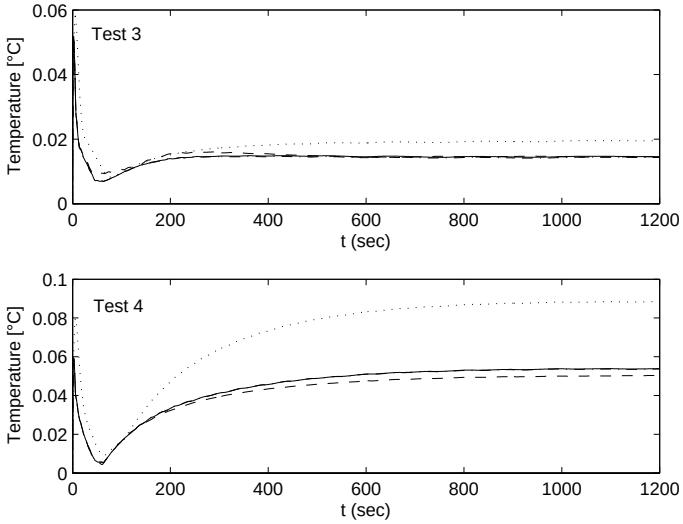

Figure 6: Maximum temperature deviation of the POD models with respect to the full order model along Tests 3 and 4. Solid line - POD model and P-POD model. Dashed line - Neural-POD model. Dash-dotted line - P-POD model with a reduced set of constraints. Dotted line - MPE-POD model.

neurons each. The hidden neurons are set with hyperbolic tangent activation functions whereas the output neurons use linear activation functions. From now on, the POD model based on this MLP will be referred to as Neural-POD model.

In the MPE method, the Galerkin projection is conducted only on some pre-selected state variables or points of the spatial domain, instead of the entire set. The remaining state variables are estimated by means of the POD basis vectors. The fact of using a subset of points of the spatial domain leads to a reduction of the time required for evaluating the POD model. In this study, we found a POD model for the bar using this technique, and such a model will be referred to as MPE-POD model. The selection of points was done by using the second screening criterion and the greedy algorithm described in [2, 1]. Since the boundary conditions must be satisfied by the reduced order model, the points adjacent to them have to be included. A total of 182 points from 499 were selected, including the 2 points adjacent to the ends of the bar. This number of points offers a good compromise between the accuracy of the model and its

computational gain.

In order to validate and evaluate the different POD models of the bar, the following tests were carried out:

- Test 1: A step of magnitude $1200 \cdot 10^3$ W $\cdot$ m^{-3} is applied to $u(t)$ and a step of magnitude $500 \cdot 10^3$ W $\cdot$ m^{-3} is applied to $d(t)$.
- Test 2: Steps of magnitude $-1100 \cdot 10^3$ W $\cdot$ m^{-3} and $-400 \cdot 10^3$ W $\cdot$ m^{-3} are applied to $u(t)$ and $d(t)$ respectively.
- Test 3: A step of magnitude $500 \cdot 10^3$ W $\cdot$ m^{-3} is applied to $u(t)$ and a step of magnitude $-200 \cdot 10^3$ W $\cdot$ m^{-3} is applied to $d(t)$.
- Test 4: Steps of magnitude $-400 \cdot 10^3$ W $\cdot$ m^{-3} and $300 \cdot 10^3$ W $\cdot$ m^{-3} are applied to $u(t)$ and $d(t)$ respectively.

Given that the discretized model of the bar (4) consists of a set of stiff differential equations, an ODE solver that can deal with this condition has to be used. Hence, we used the function *ode23tb* of Matlab which implements TR-BDF2, an implicit Runge-Kutta formula with a first stage that is a trapezoidal rule step and a second stage that is a backward differentiation formula of order two [28]. Along this work, not only (4) was solved with *ode23tb*, but also the ODEs describing each of the reduced order models. The solver was configured with a variable integration step and with a relative tolerance of 10^{-5} in all the cases. The initial conditions for the POD models were given by $\mathbf{a}(0) = \mathbf{\Phi}_n^T \mathbf{T}_n^{\Delta}(0) = \mathbf{0}$.

In Table 2 we present the computational gain of the POD models with respect to the full order model (4), and we also include a measure of their accuracy. In this table, $\Delta T_{\max}$ is the largest temperature deviation (error) of the POD models regarding the high-dimensional model (4) along the entire test, and G_{d} and G_{s} quantify the computational gain of the POD models with respect to the full order model. They are defined as follows:

$$G_{\mathrm{d}} = \tilde{t}_{\mathrm{fom}}/\tilde{t}_{\mathrm{pod}}, \quad G_{\mathrm{s}} = t_{\mathrm{fom}}/t_{\mathrm{pod}}$$

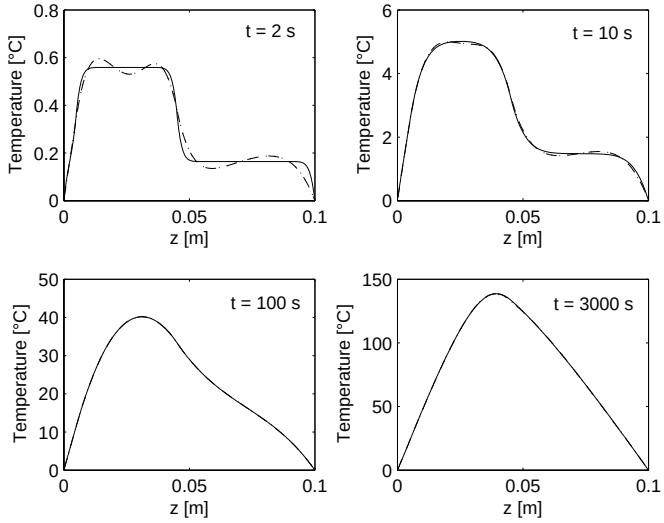

Figure 7: Temperature profile at different time steps during Test 1. Solid line - Full order model (4). Dashed line - POD model and P-POD model. Dotted line - Neural-POD model. The temperatures are relative to the ambient temperature (25°C).

where t_{fom} and t_{pod} are the times spent by the ODE solver for simulating the full order model and the POD model respectively, and $\tilde{t}_{\text{fom}}$ and $\tilde{t}_{\text{pod}}$ are the average times for calculating the derivatives along the test of the full order model and the POD model in that order. The values of G_{d} and G_{s} in Table 2 are average values found after 15000 and 1500 runs respectively. All the simulations were carried out in a PC with a dual core of 3 Ghz and a RAM memory of 2 GB.

Figures 5 and 6 show the maximum temperature deviation of the different POD models with respect to the full order model along the tests, and Figures 7 and 8 depict the evolution of the temperature profile of the bar during Test 1, the most severe test. In general, the accuracy of the POD models is good in spite of the big model-order reduction. Notice however that the MPE-POD model tends to be the less accurate model. We might select more points to improve its accuracy but this would compromise its computational gain. From Figures 7 and 8 we can observe how the temperature profiles of the POD models practically overlap each other.

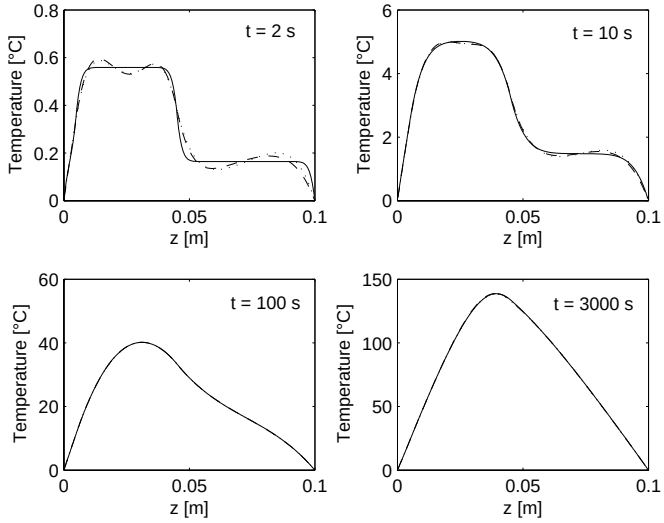

Figure 8: Temperature profile at different time steps during Test 1. Solid line - Full order model (4). Dashed line - P-POD model with a reduced set of monomials. Dotted line - MPE-POD model. The temperatures are relative to the ambient temperature (25°C).

From Table 2 we can notice that the derivatives of the P-POD model are calculated about 5.7 times faster than in the POD case. This computational gain has been achieved by exploiting the polynomial nature of the vector function $\mathbf{f}(\mathbf{a}(t))$ of (12) that allow us to write $\mathbf{f}(\mathbf{a}(t))$ in an efficient and compact form. This gain makes the simulation of the P-POD model about 2.4 times faster than the simulation of the POD model. Given that both models are equivalent, they have the same accuracy. Regarding the P-POD-RS model, the calculation of the derivatives is performed about 13.3 times faster than in the POD case and 2.3 times faster than in the P-POD case. This significant computational gain has been obtained by properly selecting a subset of monomials of the P-POD model. The impact of this gain on the simulation time is also remarkable. The simulation of P-POD-RS requires about 3.6 times less time than the simulation of the POD model. In addition, the difference in their responses is very small. The largest difference occurs during Test 1, and this difference is only of 0.13°C.

According to the results presented in Table 2, the P-POD-RS and Neural-

POD models are in that order the fastest reduced order models of the bar. Although the MPE-POD model was faster than the POD model, it was not neither as fast as the P-POD-RS, Neural-POD and P-POD models nor as accurate as them. At this point it is important to emphasize that the MPE method is a general technique that does not exploit the particular characteristics of the kind of problems treated in this chapter, and therefore the results obtained were expected. It is undoubted that the incorporation of prior knowledge about a problem in a given algorithm improves notably its results.

7. Conclusions

In this paper we have presented a way of accelerating the evaluation of nonlinear POD models derived from input-affine high-dimensional systems with polynomial nonlinearities. We have shown that by exploiting their polynomial nature, we can generate compact and efficient representations (P-POD models) which can be evaluated much faster than the POD models whose vector functions require the full spatial information of the original model. In addition, we have discussed how the sequential feature selection algorithms can help us to accelerate even more the evaluation of these P-POD models by selecting their most relevant monomials (suboptimal solution). Note that this approach might be also employed in the case of input-affine high-dimensional systems with non-polynomial nonlinearities, as long as these nonlinearities can be approximated by low degree polynomials. This condition is advised in order to keep the number of monomials as small as possible. A notable advantage of this approach with respect to the method based on neural networks is that we do not have the local minima problem, since we solve a convex optimization problem. However its drawback is that the number of monomials can be very large.

Furthermore in this paper, we have discussed two ways of guaranteeing the local stability of POD models with polynomial nonlinearities. In our first approach we end up with an SDP optimization problem whose constraint provides a sufficient condition for the local stability of the models. The problem of this

method is that the constraint can be a little restrictive. In the second approach we finish up with an NSDP optimization problem whose constraints unlike the first approach, give necessary and sufficient conditions for the stability of the POD models. The drawback of this formulation is that we have to deal with a non-convex optimization problem. In both methods the local stability of the POD models is guaranteed independently of the data used. However the quality of the reduced order models still depends on the quality of data employed in the model reduction process.

The simulation results showed that the P-POD model was as accurate as the POD model, but much faster. The P-POD model with a reduced set of monomials (P-POD-RS) provided a remarkable computational gain just by sacrificing a little bit its accuracy. This model together with the neural-POD model were the ones that offered the largest computational gains.

Further research is necessary in order to evaluate the application of the approach proposed in this paper on dynamic systems with harder nonlinearities.

8. ACKNOWLEDGMENTS

This research was supported by: • Research Council KUL: GOA AMBioRICS, CoE EF/05/006 Optimization in Engineering (OPTEC), IOF-SCORES4CHEM, several PhD/postdoc & fellow grants; • Flemish Government: ○ FWO: PhD/postdoc grants, projects G.0452.04 (new quantum algorithms), projects G.0499.04 (Statistics), G.0211.05 (Nonlinear), G.0226.06 (cooperative systems and optimization), G.0321.06 (Tensors), G.0302.07 (SVM/Kernel), G.0320.08 (convex MPC), G.0558.08 (Robust MHE), G.0557.08 (Glycemia2), research communities (ICCoS, ANMMM, MLDM); ○IWT: PhD Grants, McKnow-E, Eureka-Flite+; ○Helmholtz: viCERP; • Belgian Federal Science Policy Office: IUAP P6/04 (DYSCO, Dynamical systems, control and optimization, 2007-2011); • EU: ERNSI; FP7-HD-MPC (Collaborative Project STREP-grantnr. 223854) • Contract Research: AMINAL. Dr. Bart De Moor is a full professor at the Katholieke Universiteit Leuven, Belgium.

References

[1] P. Astrid, Reduction of process simulation models: a proper orthogonal decomposition approach, Ph.D. thesis, Technishche Universiteit Eindhoven, Eindhoven (Netherlands) (November 2004).

[2] P. Astrid, S. Weiland, K. Willcox, T. Backs, Missing point estimation in models described by proper orthogonal decomposition, IEEE Transactions on Automatic Control 53 (10) (2008) 2237–2251.

[3] P. Astrid, Fast reduced order modeling technique for large scale LTV systems, in: Proceedings of American Control Conference 2004, Vol. 1, 2004, pp. 762– 767.

[4] O. M. Agudelo, J. J. Espinosa, B. DeMoor, Acceleration of nonlinear POD models: A neural network approach, Accepted for publication in proceedings of the European Control Conference 2009 (ECC'09).

[5] S. Prajna, POD model reduction with stability guarantee, in: Proceedings of 42nd IEEE Conference on Decision and Control (CDC 2003), Hawaii, USA, 2003.

[6] A. Yousefi, B. Lohmann, J. Lienemann, J. G. Korvink, Nonlinear heat tranfer modelling and reduction, in: Proceedings of the 12th IEEE Mediterranean Conference on Control and Automation (MED '04), Kusadasi, Turkey, 2004.

[7] A. Chatterjee, An introduction to the proper orthogonal decomposition, Current Science 78 (7) (2000) 808–817.

[8] Y. C. Liang, H. P. Lee, S. P. Lim, W. Z. Lin, K. H. Lee, C. G. Wu, Proper orthogonal decomposition and its applications–Part I: Theory, Journal of Sound and Vibration 252 (3) (2002) 527–544.

[9] J. Lumley, P. Blossey, Control of turbulence, Annual Review of Fluid Mechanics 30 (1998) 311–327.

[10] K. Kunisch, S. Volkwein, Control of the burgers equation by a reduced–order approach using proper orthogonal decomposition, Journal of Optimization Theory and Applications 102 (2) (1999) 345–371.

[11] S. S. Ravindran, A reduced–order approach for optimal control of fluids using proper orthogonal decomposition, International Journal for Numerical Methods in Fluids 34 (5) (2000) 425–448.

[12] P. Beyer, S. Benkadda, X. Garbet, Proper orthogonal decomposition and Galerkin projection for a three–dimensional plasma dynamical system, Physical Review E 61 (1) (2000) 813–823. doi:10.1103/PhysRevE.61.813.

[13] A. Iollo, S. Lanteri, J. A. Désidéri, Stability properties of POD–Galerkin approximations for the compressible Navier–Stokes equations, Theoretical and Computational Fluid Dynamics 13 (6) (2000) 377–396.

[14] L. Huisman, Control of glass melting processes based on reduced CFD models, Ph.D. thesis, Technishche Universiteit Eindhoven, Eindhoven (Netherlands) (March 2005).

[15] L. Huisman, S. Weiland, Identification and model predictive control of an industrial glass feeder, in: Proceedings of the 13th IFAC Symposium on System Identification (SYSID–2003), Rotterdam, The Netherlands, 2003, pp. 1685–1689.

[16] P. Holmes, J. L. Lumley, G. Berkooz, Turbulence, Coherence Structure, Dynamical Systems and Symmetry, Cambridge University Press, Cambridge, 1996.

[17] M. M. Gupta, L. Jin, N. Homma, Static and Dynamic Neural Networks: From Fundamentals to Advanced Theory, Wiley–IEEE press, 2003.

[18] A. Webb, Statistical Pattern Recognition, 2nd Edition, John Wiley & Sons, LTD, 2002.

[19] R. Kohavi, H. John, Wrappers for feature subset selection, Artificial Intelligence 97 (1–2) (1997) 273–324.

[20] I. Guyon, A. Elisseeff, An introduction to variable and feature selection, Journal of Machine Learning Research 3 (2003) 1157–1182.

[21] A. Miller, Subset Selection in Regression. Monographs on Statistics and Applied Probability 95, 2nd Edition, Chapman & Hall/CRC, Boca Raton, FL, 2002.

[22] U. T. Ringertz, Eigenvalues in optimal structural design, in: A. Conn, L. T. Biegler, T. F. Coleman, F. Santosa (Eds.), Proceedings of an IMA Workshop on Large–Scale Optimizatioin, 1997, pp. 135–149.

[23] A. Blanco, J. Bandoni, Optimal design of stable process, Lat. Am. Appl. Res. 33 (2) (2003) 123–128.

[24] J. Sturm, Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones, Optimization Methods and Software 11 (1999) 625–653.

[25] M. Kočvara, M. Stingl, PENNON – a code for convex nonlinear and semidefinite programming, Opt. Methods and Software 20 (3) (2003) 317–333.

[26] A. Ben-Tal, M. Zibulevsky, Penalty–barrier multiplier methods for convex programming problems, SIAM J. Optim. 7 (1997) 347–366.

[27] D. Henrion, J. Löfberg, M. Kočvara, M. Stingl, Solving polynomial static output feedback problems with PENBMI, in: Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference 2005, Seville, Spain, 2005, pp. 7581–7586.

[28] L. F. Shampine, M. E. Hosea, Analysis and implementation of TR–BDF2, Applied Numerical Mathematics 20 (1996) 21–37.

Table 2: Performance of the POD models

Test	POD model			Neural-POD model			MPE-POD model (182)			P-POD model			P-POD-RS model		
	G_d	G_s	$\Delta T_{\max}$	G_d	G_s	$\Delta T_{\max}$	G_d	G_s	$\Delta T_{\max}$	G_d	G_s	$\Delta T_{\max}$	G_d	G_s	$\Delta T_{\max}$
1	0.97	2.02	0.423	9.66	8.153	0.689	2.08	3.75	0.695	5.61	5.00	0.423	12.89	8.22	0.551
2	0.97	3.61	0.279	9.32	11.69	0.278	2.08	5.76	0.45	5.47	8.80	0.279	12.88	12.48	0.275
3	0.97	4.52	0.0518	9.32	14.03	0.0519	2.10	7.45	0.0581	5.48	9.83	0.0518	12.91	13.85	0.0518
4	0.97	3.52	0.0594	9.30	11.43	0.0593	2.09	5.58	0.088	5.48	8.34	0.0594	12.92	12.72	0.0594

G_d is the computational gain in the calculation of the derivatives.
G_s is the computational gain in the simulation of the model.
$\Delta T_{\max}$ is the largest temperature deviation (error) in degrees celsius of the POD model.